Development of a multimedia service on a TINA platform
the DeskTop presentation service

Kijftenbelt, T.T.

Award date:
1998
Master's Thesis:

Development of a Multimedia Service on a TINA Platform

T.T. Kijftenbelt

Coach: dipl.Ing. L. Lehmann (Swisscom Corporate Technology)
dipl.Ing. C.E. Würgler (Swisscom Corporate Technology)
Supervisor: Prof.ir. J. de Stigter
Examiner: Dr.ir. A.C. Verschueren
Period: April - December 1997
Development of a Multimedia Service on a TINA Platform

The DeskTop Presentation Service

By: Thomas T. Kijftenbelt
Date: April 1997 - December 1997
Place: Bern (Switzerland)
Tutors: Prof. Ir. J. de Stigter
dipl.Ing. L. Lehmann
dipl.Ing. C.E. Würgler
Examiner: Dr. Ir. A.C. Verschueren

Swisscom Corporate Technology

Eindhoven University of Technology
Department of Electrical Engineering
ABSTRACT

In today’s open telecommunication market, rapid service development is of the utmost importance. To do so, the reuse of software components is inevitable. Furthermore also third party service providers have to be able to develop services. To meet these requirements a new telecommunications architecture has been defined: TINA.

Together with Alcatel and Ericsson, the Unisource partners have developed a platform compliant to the TINA architecture in order to demonstrate the feasibility of TINA. On top of this platform several high quality multimedia services have been developed. This report describes a development of the DeskTop Presentation service.

Besides rapid service development and the integration on the platform, the aim of developing a DeskTop Presentation service is to show the possibilities of encapsulation of existing software, and study the application of Internet technologies in TINA services.

The project was divided into two phases. During the first phase a DeskTop Presentation service was integrated into an, earlier developed, Desktop Video Conference, which takes care of the session management. This was done, as a first version of the DeskTop Presentation service had to be demonstrated to the potential customers in an early stage. In the second phase of the project the DeskTop Presentation service was implemented as a stand-alone service, which was necessary to show that TINA allows for rapid service development and to show that several services can be integrated on the same platform.

Based on the experience, gained during the project, it can be concluded that TINA allows for rapid service development, and that the TINA platform makes it possible for third party service providers to develop services, and integrate these on the TINA platform. Encapsulation of existing applications also allows for rapid service development, but its possibilities depend on the APIs of the existing application.
CONTENTS

1. INTRODUCTION .................................................................................................................. 1

2. OVERVIEW OF TINA AND CORBA ................................................................................. 3

2.1 TINA .................................................................................................................................... 3
   2.1.1 The Telecommunications Information Networking Architecture ................. 3
   2.1.2 The Computing Architecture .......................................................................... 4
   2.1.3 The Service Architecture ............................................................................... 6
   2.1.4 The Network Architecture .............................................................................. 6
   2.1.5 The Management Architecture ...................................................................... 7

2.2 CORBA .............................................................................................................................. 8
   2.2.1 The Object Management Architecture .......................................................... 8
   2.2.2 The Common Object Request Broker Architecture ....................................... 9

3. A DESKTOP PRESENTATION SERVICE ........................................................................ 13

3.1 AN EXAMPLE: THE POWERPOINT PRESENTATION CONFERENCE ...................... 13

3.2 POSSIBLE FEATURES .................................................................................................... 13

3.3 SERVICE ARCHITECTURE ............................................................................................. 15
   3.3.1 Overview of the Service Architecture ............................................................. 16
   3.3.2 DVC Architecture ......................................................................................... 17
   3.3.3 DTP Architecture ......................................................................................... 18

3.4 CONCLUSION .................................................................................................................... 18

4. BASIC DTP SERVICE ......................................................................................................... 19

4.1 SERVICE OVERVIEW ....................................................................................................... 19
   4.1.1 Description ........................................................................................................ 19
   4.1.2 Comparing Different Presentation Packages ................................................ 20
   4.1.3 Graphical User Interface .................................................................................. 21

4.2 SERVICE ARCHITECTURE .............................................................................................. 24
   4.2.1 Terminal Domain .............................................................................................. 25
   4.2.2 Network Domain .............................................................................................. 28
   4.2.3 Service Scenarios ............................................................................................. 29

4.3 DEVELOPMENT OF THE BASIC DTP SERVICE ....................................................... 30
   4.3.1 Service Development ....................................................................................... 30
   4.3.2 Encountered Problems ..................................................................................... 31

4.4 CONCLUSION .................................................................................................................... 31

5. INTERNET TECHNOLOGIES ........................................................................................... 33

5.1 BENEFITS OF INTERNET TECHNOLOGIES ................................................................. 33

5.2 POSSIBLE APPLICATIONS OF INTERNET TECHNOLOGIES .................................... 33
   5.2.1 Storing a Presentation on a WWW Server ..................................................... 33
   5.2.2 Downloading a Presentation from a WWW Server ......................................... 34
   5.2.3 Loading a DTP Applet .................................................................................... 36

5.3 CONCLUSION .................................................................................................................... 37
6. STAND-ALONE DTP SERVICE

6.1 BENEFITS OF A STAND-ALONE DTP SERVICE

6.2 SERVICE OVERVIEW

6.2.1 Description

6.2.2 Graphical User Interface

6.3 SERVICE ARCHITECTURE

6.3.1 Terminal Domain

6.3.2 Network Domain

6.4 DEVELOPMENT OF THE STAND-ALONE DTP SERVICE

6.4.1 Service Development

6.4.2 Encountered Problems

6.5 CONCLUSION

7. FILE TRANSFER FEATURE

7.1 BENEFITS OF A FILE TRANSFER FEATURE

7.2 POSSIBLE FILE TRANSFER MECHANISMS

7.2.1 File Transfer Directly over ATM

7.2.2 File Transfer Using IP over ATM

7.3 ARCHITECTURE OF THE FILE TRANSFER FEATURE

7.3.1 Terminal Domain

7.3.2 Network Domain

7.3.3 Feature Scenario

7.4 DEVELOPMENT OF THE FILE TRANSFER FEATURE

7.4.1 Feature Development

7.4.2 Encountered Problems

7.5 CONCLUSION

8. CONCLUSIONS AND RECOMMENDATIONS

ACKNOWLEDGEMENTS

ABBREVIATIONS

REFERENCES

APPENDICES

A. EXTERNAL SERVICE SPECIFICATIONS - DTP SERVICE

B. SERVICE PLATFORM DESIGN - DTP SERVICE
1. INTRODUCTION

The final step before concluding the study of electrical engineering is the diploma work. This report is the result of a diploma work at Swisscom Corporate Technology. Together with Alcatel and Ericsson the research departments of the Unisource partners (KPN, Telia and Swisscom) were involved in the SPOT project (Service Pilot On TINA). The goal of this project was to demonstrate the feasibility of TINA (Telecommunications Information Networking Architecture) and study its advantages and bottlenecks, with the main focus on service development and integration, and platform interworking. To do so, a platform compliant to TINA and several high quality multimedia services (Desktop Video Conference (DVC), DeskTop Presentation service (DTP), Chat service and Administrative services), which run on top of this TINA platform, were developed.

The assignment was to develop the DeskTop Presentation service on the SPOT platform. Besides focusing on the possibilities of service development and integration, the goal of developing the DeskTop Presentation service was to prove that it is possible to encapsulate an existing software application within a TINA service. To be able to demonstrate the DeskTop Presentation service to the customers in an early stage, the development of the service was divided into two phases. The first version was an integrated part of the Desktop Video Conference, which took care of the session management. The second version of the DeskTop Presentation service was stand-alone.

To reach the described goals, the following questions were kept in mind during the development of the DeskTop Presentation service:

- **Does the TINA platform allow for rapid service deployment?**
  In the open telecommunication market it is very important to be able to offer customers the services they need. To do so, a short time-to-market is a necessity.

- **Is it possible to encapsulate existing software products within TINA services?**
  The encapsulation of existing software products within telecommunication services reduces the development time of services incredibly and offers enormous possibilities for the creation of new services.

- **Does TINA allow various services to be integrated on the same platform?**
  TINA services are built using the generic service components, offered by the TINA platform. In the future, services will also be developed by third party service providers, therefore it must be possible to integrate all these services on the same platform.

- **Can Internet technologies be used to enhance TINA services?**
  Internet technologies offer interesting possibilities, because of their platform independence and uniform user interface. It would be interesting when TINA services could be enhanced using Internet technologies.

Answers to these questions can be found in this report.
The next chapter in this report provides some background information on TINA and CORBA. Chapter three focuses on a general example of a DTP service, its possible features and its architecture. In chapter four the development of the integrated version of the DTP service is described, and the next chapter gives insight in the possible applications and limitations of Internet technologies in the DTP service. Chapter six describes the complete development track of the stand-alone DTP service. The next chapter details a file transfer and is followed by the last chapter summarizing the conclusions and recommendations of this project.
2. OVERVIEW OF TINA AND CORBA

In this chapter some background information on TINA and CORBA, the architectures used within the SPOT project, is given. The aim of this chapter is to provide a basic understanding of both architectures, needed to understand the following chapters. More detailed information can be found in the references.

2.1 TINA

A worldwide consortium formed by network operators, telecommunication and computer equipment suppliers has been defining and validating an architecture for telecommunication services: TINA (Telecommunications Information Networking Architecture). The goal of this architecture is to separate service- and network management from switching functions. Service creation, testing and deployment have been made easier by creating reusable software components, supporting software interoperability and hiding the underlying technologies. More detailed information about TINA can be found in [BAR93], [DUP95], [IFIP95] and in the deliverables on [TINA].

2.1.1 The Telecommunications Information Networking Architecture

In TINA, telecommunication services are considered to be software-based applications. Services are constructed using so-called service components, these service components are built according to the Universal Service Component Model (USCM). Each service component consists of the following four parts (Figure 2-1):

- Core: This part describes the function of the service component.
- Usage: This part describes the client's point of view.
- Management: This part describes the operational, maintenance and management functions.
- Substance: This part describes the dependence on other service components.

![Figure 2-1: The Universal Service Component Model](image)

Each service component, on its turn, is constructed from one or more service components.
The various computational objects interact via the Distributed Processing Environment (DPE). The DPE hides the complexity of the underlying technology and the problems concerning the distributed nature of the application (see Figure 2-2).

![Overall Architecture](image)

Figure 2-2: Overall Architecture

In TINA the service layer and the transport network are separated, this is also displayed in Figure 2-2.

The TINA Overall architecture can be divided into four sub-architectures:

- Computing architecture: describes the computational objects and their interaction via the DPE (§2.1.2).
- Service architecture: describes the service components, how computational objects can be used to specify, create and operate telecommunication services and how computational objects can be reused (§2.1.3).
- Network architecture: describes the network components and provides an abstraction of the resources which are used by the various services (§2.1.4).
- Management architecture: describes a set of rules for the management of the three other architectures (§2.1.5).

### 2.1.2 The Computing Architecture

The computing architecture defines the modeling concepts which are used to specify the object oriented software in TINA. It also defines the distributed processing environment (DPE), which is responsible for the location of, and the interaction between computational objects.

As a telecommunication system is very complex, the system can be specified using five viewpoints (see Figure 2-3). Each viewpoint focuses on a sub-set of the characteristics of the system. The complete description of the system can be obtained
by examining the specifications of all viewpoints. For each viewpoint but the technology viewpoint, modeling concepts have been defined.

The Enterprise modeling concepts provide a framework for building enterprise specifications or models. An enterprise model describes a system from the perspective of the organizations and people that will use or operate the system.

The computational modeling concepts describe how computational objects interact via specified interfaces (see Figure 2-4). These interfaces can be divided into two groups:

- **Operational interfaces**: Interfaces with predefined operations, which are used for the transmission of functions.
- **Stream interfaces**: Interfaces without predefined operations, which are used for the transmission of data flows.

The interfaces of the computational objects are defined using the TINA Object Definition Language (TINA-ODL).

The computational modeling concepts, described above, are only used to describe how a service is divided into several computational objects, and how these objects interact. The functionality of the service is described using the information modeling concepts.

From the computational viewpoint, the DPE is considered as one homogenous platform, in reality the DPE is a distributed platform. The engineering modeling concepts describe how the DPE is distributed over various kernels, and how these kernels interact via the kernel Transport Network (kTN).
2.1.3 The Service Architecture

The service architecture provides means to build services and a service support environment. It describes the functionality of the service, how the service is managed and how terminal/user mobility is achieved.

In the service architecture, two main separation principles are defined:
- Separation of call and connection control: This separation is used to obtain a more flexible way to handle services.
- Separation of user and terminal related issues: This separation is used to improve the mobility.

Within the service architecture various reusable components are identified: user agents (represent the users in the network), terminal agents (represent the terminals attached to the network), service session manager (represents an instance of a running service), subscription manager (contains information related to customers and their subscriptions) and communication session manager (used to setup a connection). In Figure 2-5 some of these components are identified.

![Figure 2-5: Reusable Service Components](image)

2.1.4 The Network Architecture

The components on top of the DPE can be divided into three groups:
- Service components: These components contain core service functions.
- Resource components: Resource components are abstract representations of elements.
- Elements: These components contain software representations of individual resources. Elements are responsible for the communication with an actual device.
The network architecture describes the transport network in a technology independent way. It provides mechanisms for the establishment, modification and release of network connections. The network architecture defines a set of abstractions that the resources can work with. At one end it provides a high level overview of network connections to services, and at the other end it provides generic descriptions of elements, which can be specialized to particular technologies and products.

The network architecture is described using the Network Resource Information Model (NRIM). This is an informal specification of transmission and switch technologies, from which technical aspects have been extracted. It describes how individual elements are related and configured to provide and maintain end-to-end connectivity.

To provide a service oriented view of the connections, so-called connection graphs are used.

### 2.1.5 The Management Architecture

The TINA management architecture provides the concepts and principles to build management systems that can manage the entities in TINA systems. As with the service architecture, the computing architecture is used to define object types and interfaces that should be used to manage TINA services, resources and infrastructures.

There are two basic types of management (see Figure 2-6):

- **Computing management**: This type of management involves the management of the computers (NCCE), DPE and of software that runs on the DPE.
- **Telecommunications management**: This type of management involves the management switching and transmission networks and the management of services.

The main concern of the management is the installation and deployment of software, and load balancing.

![Diagram of Two Basic Types of Management](image)
The two types of management mentioned above, can be broken down into several smaller parts. Computing management deals with generic software, and the management of the DPE and the computer environment. Telecommunications management deals with service, network and element management.

2.2 CORBA

The Object Management Group (OMG) has developed a standard for distributed computing on heterogeneous environments: the Object Management Architecture (OMA). The key component of this standard is the Common Object Request Broker Architecture (CORBA). First an overview of OMA will be given, followed by a description of CORBA. For further information on CORBA see [VIN97], and the deliverables on [OMG] and [IONA].

2.2.1 The Object Management Architecture

The Object Management Architecture consists of two parts:

- **Object model:** This model defines how objects, which are distributed across a heterogeneous network, can be described. Objects are considered to be well defined entities with predefined interfaces. The location and implementation of the objects is hidden for the client.
- **Reference model:** This model describes how objects interact. The communication between objects is handled by the Object Request Broker (ORB).

The OMA reference model consists of the following components: Object Request Broker, Object Services, Common Facilities, Domain Interfaces, and Application Interfaces (see Figure 2-7).

![OMA Reference Model Architecture](image)

Figure 2-7: OMA Reference Model Architecture

The components of the OMA reference model have the following function:

- **The Object Request Broker** provides an infrastructure allowing objects to communicate, independent of specific platforms and techniques used to implement the objects.
- **Object Services** standardize the life-cycle management of objects.
- **Common Facilities** provide a set of generic application functions that can be configured to the specific requirements of a particular configuration.
- **Domain Interfaces** provide functionality of direct interest to end-users in a particular application domain.
- **Application Interfaces** represent component-based applications performing particular tasks for a user.

### 2.2.2 The Common Object Request Broker Architecture

The CORBA specification defines the interfaces and characteristics of the ORB component of the OMA model. Below, the various features of CORBA 2.0 will be described (most of these features are displayed in Figure 2-8).

![Figure 2-8: Common Object Request Broker Architecture](image)

- **ORB Core** handles the communication between the clients and the objects. It hides the following properties for the client:
  - Object location: A client does not know where the object, it wants to communicate with, resides.
  - Object implementation: A client does not know how the object was implemented, in what language or on what platform.
  - Execution state: A client does not know whether the object is already activated and ready to accept requests, or that the object has to be started first.
  - Object communication mechanisms: A client does not know what communication method is used for the communication with the object.

To make a request, the client specifies the target object by using an object reference. When a CORBA object is created, an object reference for it is automatically created.

The client issues requests to an object via predefined interfaces. The interface specifies the operations and types that are supported by the object. Interfaces for objects are defined using the OMG Interface Definition Language (OMG IDL).
OMG IDL is a declarative language, not a programming language. OMG IDL is programming language independent. Interfaces are defined, independent of the object implementations, so objects can be implemented using different programming languages and still communicate with each other.

The language mappings describe how OMG IDL interfaces are mapped onto the facilities of a given programming language. For example: OMG IDL interfaces map to C++ classes, with operations mapping to member functions of these classes.

The ORB Core passes requests from the client to the object implementation. In order to make a request, the client communicates with the ORB Core through the IDL stub or through the Dynamic Invocation Interface. The ORB Core transfers the request to the object implementation, which receives the request through the IDL skeleton or through the Dynamic Skeleton Interface.

The IDL stub and IDL skeleton are static interfaces, they are generated by the OMG IDL compiler at compile time.

The Dynamic Invocation Interface (DII) allows a client to issue requests to objects whose definition and interface are unknown at the client’s compile time. In order to use the DII, the client has to compose a request including the object reference, the operation and a list of parameters. The information can be retrieved from the interface repository, a database which contains information about object interface definitions.

The server side version of the DII is the Dynamic Skeleton Interface (DSI). Using this interface the object implementation is no longer accessed through a static skeleton, generated from an IDL interface specification, instead it is reached through an interface that provides access to the operation name and parameters (as in the DII the information can be retrieved from the interface repository). Thus DSI is a way to deliver requests from the ORB to an object implementation that does not have compile-time knowledge of the object it is implementing.

Object Adapters adapt the interface of the called object to the interface expected by the client. It allows a client to make requests to an object without knowing the true interface of the object (see Figure 2-9).

![Figure 2-9: Role of an Object Adapter](image-url)
The functions of an Object Adapter are:

- **Object registration**: Object Adapters provide operations which allow programming language entities to be registered as implementations for CORBA objects.
- **Object reference generation**: Object Adapters generate object references for CORBA objects.
- **Server process activation**: If necessary, the Object Adapter starts a server process in which objects can be activated.
- **Object activation**: When a request arrives for an object, the Object Adapter activates the called object if it is not already active.
- **Request demultiplexing**: The Object Adapter must ensure, in cooperation with the ORB, that requests can be received.
- **Object upcalls**: The Object Adapter dispatches request to the registered objects.

CORBA 2.0 introduced a general ORB interoperability architecture that provides for direct ORB-to-ORB interoperability and bridge-based interoperability. Direct interoperability is possible when two ORBs reside in the same domain (they understand the same object references and OMG IDL type system, and perhaps share the same security information). Bridge-based interoperability is necessary when ORBs from separate domains must communicate. The role of the bridge is to map ORB-specific information from one ORB domain to the other.
3. A DESKTOP PRESENTATION SERVICE

A DeskTop Presentation service (DTP service) allows a presenter to present his slides to other users (called conferees), using a presentation package (e.g. PowerPoint or Harvard Graphics). In this chapter an example of a DeskTop Presentation service is given, followed by a list of possible features. Then the architecture of such a service is briefly described.

3.1 An Example: The PowerPoint Presentation Conference

In this paragraph an example of a DTP service is given: the presentation conference option of PowerPoint. This example is given to provide a better understanding of the kind of service which has to be developed.

The presentation conference allows an user to show a presentation to other users, interconnected via a Local Area Network. To do so, the presenter must enter the node names of the conferees he wants to invite in the presentation session. The conferees must have started the presentation conference, so they can be invited.

The user interface of the PowerPoint presentation conference is shown in Figure 3-1.

![Figure 3-1: User Interface of the PowerPoint Presentation Conference](image)

The user interface gives the presenter the possibility to control the sequence of the slides at the conferees’ terminals.

3.2 Possible Features

Besides the basic control functions (slide forward, slide backward, go to first slide, go to last slide and go to a specific slide), several other features can be implemented. In Table 3.1 a list of possible features for a DeskTop Presentation service is given.
### Table 3.1: Possible Features for a DeskTop Presentation Service

<table>
<thead>
<tr>
<th>Feature</th>
<th>Comments</th>
<th>Available to</th>
<th>Priority level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet technologies</td>
<td>The application can be loaded dynamically from a WWW server, and the Graphical User Interface runs in a Web browser</td>
<td>All</td>
<td>High</td>
</tr>
<tr>
<td>Stand-alone DTP service</td>
<td>Possibility of executing the DTP service as an independent stand-alone service that eventually recuperates the session characteristics of an on-going DVC session</td>
<td>All</td>
<td>High</td>
</tr>
<tr>
<td>File transfer of the presentation to each participant</td>
<td>File transfer could be done using an existing application (ftp, Netscape) or developing a simple tool that sends the file over ATM.</td>
<td>Presenter</td>
<td>High</td>
</tr>
<tr>
<td>Advanced charging schemes</td>
<td>The DTP service can be paid for by the presenter, by the Desktop Video Conference (DVC) initiator or by all participants.</td>
<td>Presenter (with confirmation of the involved parties)</td>
<td>Medium/High</td>
</tr>
<tr>
<td>Remote animated pointer</td>
<td>The presenter can point at specific details (depends on the external control possibilities of the presentation package).</td>
<td>Presenter</td>
<td>Medium</td>
</tr>
<tr>
<td>Drawing functions</td>
<td>The presenter can draw in the slides he presents (depends on the external control possibilities of the presentation package)</td>
<td>Presenter</td>
<td>Medium/Low</td>
</tr>
<tr>
<td>List with slide names and numbers</td>
<td>The presenter can select a slide from a list with all slide names and numbers</td>
<td>Presenter</td>
<td>Low</td>
</tr>
<tr>
<td>Use of Slide Show Viewer</td>
<td>Use an executable slide show viewer instead of a complete presentation package. Eventually loaded dynamically on terminals where it is not installed.</td>
<td>All</td>
<td>Low</td>
</tr>
<tr>
<td>Dispatch of file updates or modifications of the presentation</td>
<td>When the presentation is modified, the presentation at the conferees’ terminals is automatically updated (depends on the external control possibilities of the presentation package)</td>
<td>Presenter only or All</td>
<td>Low</td>
</tr>
<tr>
<td>Integration of other applications (Word, Netscape, ...) to implement other facets of CSCW (Computer Supported Common Work)</td>
<td>Applications that support OLE Automation can be encapsulated in the same way as the DTP is encapsulated (i.e. display and modification of a Word document can be achieved by sending OLE automation commands to all participants)</td>
<td>All</td>
<td>Low</td>
</tr>
</tbody>
</table>

For reasons of clarity the basic control functions (e.g. slide forward/slide backward) have been omitted.
To all features a priority level has been assigned. The decision about the assigned priorities is based on:

- the added value for the SPOT project
- the actual usefulness for the DTP service
- the feasibility (mainly based on the APIs (Application Programming Interface) of the presentation package.

It was decided that in the basic version of the DTP service only the basic control functions had to be implemented: slide forward, slide backward, go to the first slide, go to the last slide and go to a specific slide. This decision was made as a basic version of the DeskTop Presentation service had to be demonstrated to the customers in an early stage. The basic version of the DeskTop Presentation service is described in Chapter 4.

In the final version of the DTP service some of the additional features had to be implemented (see Table 3.1). The feature selection was mainly based on the assigned priorities. The following features were chosen to be implemented in the final version of the DTP service:

- Internet Technologies: It had to be studied how Internet technologies could be used to enhance the basic DTP service. An example of such an enhancement is loading the application as an applet. More details about the application of Internet technologies can be found in Chapter 5.
- Stand-alone DTP service: The first version of the DeskTop Presentation service was integrated in the Desktop Video Conference, which took care of the session management. To prove that several services could be integrated on the same platform, the DTP service had to be implemented as a stand-alone service. The stand-alone DTP service is described in detail in Chapter 6.
- File transfer of the presentation to all conferees: In the first version of the DTP service the file which contains the presentation had to be present on all conferees' terminals (in a specified directory). In the final version the file containing the presentation should be sent automatically to all conferees when the DTP service is started. This feature is described in detail in Chapter 7.

3.3 Service Architecture

The basic version of the DeskTop Presentation service is integrated in the Desktop Video Conference. This means that the architecture of the DTP service is the same as the architecture of the DVC service except for some additional operations. So after an overview of the service architecture is given, the service-specific computational objects of the Desktop Video Conference are described, followed by a description of the additional operations which were needed for the implementation of the DTP service.
3.3.1 Overview of the Service Architecture

The DeskTop Presentation service allows a presenter to show his slides to other users. The first version of the DTP service is an integrated part of the Desktop Video Conference. In a DVC session, a video conference can be set up, with up to four participants. More information concerning the DVC service can be found in [XDVC97] and [DVC96]. A general description of the SPOT platform can be found in [BEN97] and [LEH97].

Within a running DVC session, the DTP service can be started. Before this can be done, it has to be checked whether the DTP service is not used by another party in the same session (there can only be one presentation in a DVC session at a time). When the DTP service is started, a presentation package is automatically opened on each participant's terminal. Specific slide numbers are broadcast to all conferees, forcing their presentation packages to display the specified slide.

Figure 3-2 shows a general overview of the SPOT platform.

As can be seen in Figure 3-2, TINA provides a clear separation into different architectural levels, thus decoupling the services from the connection management and allowing different services to make use of the generic service components provided by the platform.

The services use different capabilities offered by the underlying service platform and by other services. This results in an optimal reuse of generic service components and keeps the service-specific components as simple as possible, which results in a short development time.
3.3.2 DVC Architecture

As depicted in Figure 3-3, the Desktop Video Conference consists of six service-specific computational objects.

These computational objects have the following functionality:

- **DVC_GUI (Graphical User Interface)**: The DVC_GUI is implemented in MS Visual Basic. It translates the end-user’s actions into invocations on the DVC_UAP interface.

- **DVC_GUI_STUB**: The DVC_GUI_STUB is an intermediate object which eases the communication between the DVC_UAP and the DVC_GUI. It acts as the gateway from the CORBA world towards the non-CORBA world. When the DVC_UAP wants to invoke an operation on the DVC_GUI, it has to invoke this operation on the DVC_GUI_STUB, which forwards it to the DVC_GUI.

- **DVC_UAP (User Application)**: The DVC_UAP implements the DVC specific code in the terminal which is not related to the Graphical User Interface.

- **DVC_USS (User Service Segment)**: The DVC_USS enables the interaction between the DVC_UAP and the DVC_GSS.

- **DVC_GSS (Global Service Segment)**: The DVC_GSS provides centralized management for all active DVC sessions.

- **DVC_FAC (Factory)**: The DVC_FAC enables the DVC_UAP to create and delete DVC_USS computational objects and passes the DVC_USS interface reference to the DVC_GSS.

For a complete description of the DVC architecture see [DVC96].
3.3.3 DTP Architecture

The basic version of the DTP service uses the same computational objects as the DVC service. The difference is that some additional operations had to be added to the interfaces of the service-specific computational objects of the Desktop Video Conference.

The following functionality must be added:

- Reserve the DTP service: A participant can reserve the use of the DeskTop Presentation service.
- Start the DTP service: When the presenter starts a presentation session, the presentation package is automatically opened on each participant's terminal.
- Go to a specific slide: A specific slide number can be sent to all conferees, forcing their presentation packages to go to the specified slide.
- Close the DTP service: When the presenter ends his presentation, the presentation package is automatically closed on each participant's terminal.
- Free the DTP service: When the presenter has finished his presentation, he can free the reservation he has made, so the DTP service is available to other participants, who may want to use it.

When the presenter issues a command, it is transferred from the DVC_GUI to the DVC_GSS (via the DVC_UAP and the DVC_USS). In the DVC_GSS the command is interpreted, and an appropriate command is sent to the DVC.GUI_STUB component at the conferees' terminals (see Figure 3-4).

![Figure 3-4: Command flow for the DTP service](image)

3.4 Conclusion

The structure of the SPOT platform (as described in §3.3.1) allows for rapid service development as the service-specific components are kept as simple as possible. However to prove that various services can be integrated on this platform, it is not sufficient to have a DeskTop Presentation service which is an integrated part of the Desktop Video Conference, therefore the second version of the DTP service is stand-alone. It can already be noted that encapsulation of existing software applications within TINA services is dependent on the availability of the APIs.
4. BASIC DTP SERVICE

This chapter describes the basic version of the DeskTop Presentation service. First an overview of the service is given, followed by a brief description of the Graphical User Interface, and a description of the architecture. Finally the development of the service is described, followed by some conclusions.

4.1 Service Overview

In this paragraph the basic version of the DTP service is briefly described. First an overview of the service is given, followed by a comparison of various presentation packages. Finally a brief description of the Graphical User Interface is given.

4.1.1 Description

In the DTP service a presentation package is used to show a presentation to other users. The presenter controls the presentation using a Graphical User Interface (GUI). Commands from the GUI are sent to:

- The presentation package at the presenter’s terminal (via OLE Automation)
- The GUL_STUB component at the conferees’ terminals (via CORBA)

The GUL_STUB component forwards the commands to the presentation package at its terminal. The GUL_STUB component is used as an interface between the CORBA environment and the OLE environment, as it is not possible to send commands directly from CORBA to OLE.

The command flow, described above, is shown in Figure 4-1.

![Diagram of Command Flow of the DTP Service](image)

Figure 4-1: Command Flow of the DTP Service
The basic version of the DTP service will only contain the basic control functions. In the final version some of the features, listed in Table 3.1, will be added (e.g. file transfer of the PowerPoint file to all conferees). The basic version of the DTP service will be an integrated part of the Desktop Video Conference (DVC), this means the DTP service can only be started from an ongoing DVC session. This is to avoid having to manage service interactions between the DTP and DVC services.

4.1.2 Comparing Different Presentation Packages

In order to find a suitable application for showing a presentation in the DTP service, some presentation packages have been compared. The most important aspect of the presentation package was the possibility to control the application externally (for example with the use of OLE automation functions). The following products have been examined shortly (by studying their corresponding World Wide Web pages):

- COREL Presentations 7
- SPC Harvard Graphics 4.0
- Lotus Freelance Graphics 97
- Microsoft PowerPoint 7.0

On the WWW pages of “COREL Presentations” and “SPC Harvard Graphics”, no information about OLE automation or other possibilities for external control of the application was found, which made these applications unsuitable for the DeskTop Presentation service.

On the WWW pages of “Lotus Freelance Graphics” and “Microsoft PowerPoint”, some information about OLE automation was provided. Freelance Graphics can be controlled using “Lotus Script”, and PowerPoint can be controlled using “Microsoft Visual Basic”. At first sight it seemed that Visual Basic offers more extensive OLE automation possibilities than Lotus Script.

For the following reasons PowerPoint was chosen as the application for the DeskTop Presentation service:

- PowerPoint offers more extensive OLE automation possibilities than the other applications.
- There is more information available about the OLE automation functions of PowerPoint than about the external control of the other applications.
- Microsoft PowerPoint is already in use at Swisscom.
- The Graphical User Interface of the Desktop Video Conference has been made in Visual Basic.
- The other SPOT partners use Microsoft Office (which contains PowerPoint).
- Most of the potential customers use Microsoft PowerPoint.

Concerning the choice of the presentation package it has to be noted that the purpose of the comparison, made in this paragraph, has not been to fully investigate all possibilities of all presentation products available nowadays. Comparing all available
products would have taken too much time, so only four major presentation products have been compared, and they were only examined briefly.

Some of the extensions listed in Table 3.1 cannot be implemented as there is only a limited number of OLE automation commands for the external control of PowerPoint. For example: In order to implement the remote animated pointer, it should be possible to send the coordinates of the mouse pointer to all parties involved in the DTP service. However, the OLE automation functions of PowerPoint do not offer functions to obtain the current position of the mouse pointer in the slide show window, and it is therefore not possible to implement the remote animated pointer.

4.1.3 Graphical User Interface

In this subparagraph the Graphical User Interface (GUI) of the basic DTP service is described. The most important properties of the GUI are that all functions can be accessed and that the GUI is user friendly.

The DTP service can be activated from the DVC Other Services menu by choosing the DeskTop Presentation item. This item can be selected only if there is an ongoing video and/or audio session, otherwise the item will be grayed, which means it is inactive (see [XDVC97]).

Starting the DTP service will open the “DTP Startup Window” (see Figure 4-2). Any party can activate this service but only one at a time. The mechanism used for this is explained in the next paragraph. If the DTP service has already been activated by someone else, a message will appear to inform the user the DTP service is already in use.

![Figure 4-2: DTP Startup Window](image)

All windows, which are described in this subparagraph, are only available to the initiator of the DTP service (called the presenter). The windows can be minimized, maximized and closed using the standard Windows controls, which can be found in
the upper right corner. The title bars of the different windows all contain the service name. The menus in the command bars of the different windows are the same, however in a specific window the presenter can only use the appropriate menu options (the other menu options are grayed).

In all windows the Help menu is available. This menu consists of two options: ToolTips and About. The ToolTips option is a check box. When this option is checked, the function of a control button is displayed when the mouse pointer is on that button. The About option displays some general information about the DTP Graphical User Interface.

Now the presenter has to select a PowerPoint presentation. This can be done in two ways: the presenter can press the Select button, or he can select the Select option in the File menu. All other menu options, except the options in the Help menu, are grayed.

When the presenter presses the Select button (or selects the Select option in the File menu), a standard File Open dialog will be opened, and the user can select the PowerPoint presentation he wishes to present. All files, but those with the extension .PPT are filtered. When a presentation has been selected the “DTP Begin Presentation Window” appears (see Figure 4-3).

![Figure 4-3: DTP Begin Presentation Window](image)

The Text Display Box contains the name of the selected PowerPoint presentation file. It is read only. Now the presenter can press the Begin button, or select the Begin option from the Presentation menu. Further the presenter can use the Send option from the File menu to send the PowerPoint presentation to all other parties involved in the session (called conferees). The other options, except the options in the Help menu and the Select option in the File menu are grayed.

The Send option will send the PowerPoint presentation file to all conferees and will place it in a temporary file directory (e.g. “C:\TEMP”). This option was not implemented in the basic version of the DTP service, it is one of the extensions, which
has been implemented in the final version of the DTP service (see Chapter 7). In the basic version it is assumed that the file is already present at all terminals, in the same directory.

When the **Begin** button is pressed (or the **Begin** option in the **Presentation** menu is selected), the PowerPoint application, on each terminal involved in the session, is launched/activated and the presentation file found in the temporary directory (e.g. “C:\TEMP”) is opened. If PowerPoint is already running then it will just be activated before opening the file. The PowerPoint slide show window will open, displaying the first slide (in reduced size) so that the other windows will not be hidden completely and at least partially visible.

Then the “DTP Main Control Window” appears to the presenter (see Figure 4-4).

![DTP Main Control Window](image)

The presenter can now use the **End** option in the **Presentation** menu to close the presentation on all terminals. Then the PowerPoint file (if it is open) will be closed and the DTP service will be quitted. A warning will eventually be issued by PowerPoint if the file has been modified and has not been saved previously. The “DTP Main Control Window” will also be closed and the service will be freed, so the service can be used by another end-user.

The presentation can be controlled using the control buttons (listed below).

- **Go To First Slide** button: will force all ongoing presentations to display the first slide.
- **Go To Previous Slide** button: will force all ongoing presentations to display the previous slide (if possible).
- **Go To Next Slide** button: will force all ongoing presentations to display the next slide (if possible).
- **Go To Last Slide** button: will force all ongoing presentations to display the last slide.
- **Go To** button: will force all ongoing presentations to display the slide, specified in the text box by the presenter (if possible).
Development of a Multimedia Service on a TINA Platform

Note: Impossible actions, like Go To a slide that does not exist will be handled by the GUI. These actions have no effect on the presentation.

4.2 Service Architecture

This paragraph describes the modifications to the Desktop Video Conference architecture, which are necessary to integrate the DeskTop Presentation service into the Desktop Video Conference. A brief description of the DVC architecture can be found in §3.3.2. A more detailed description of the DVC architecture can be found in [DVC96].

The following functions are provided by the DeskTop Presentation service:

- A participant in a DVC session can reserve the DTP service.
- A presenter who has finished his presentation can free the DTP service, so it is available to other end-users.
- When the presenter starts (or ends) his presentation, a PowerPoint session, with his presentation, is automatically opened (or closed) on each participant's terminal.
- A specific slide number can be broadcast to all conferees, forcing their PowerPoint session to go to the specified slide.

To implement the described functionality, the functions required by the DTP service had to be added to the interfaces of the DVC service-specific components. In Figure 4-5 an overview of the modified DVC components is given. The interfaces, which had to be modified, are represented by light gray rectangles.

![Figure 4-5: Modified DVC Service-Specific Components](image-url)
The user will be able to broadcast commands to other participants by issuing invocations on the DVC_UAP Session REQ interface. The “broadcast” commands will be received by each participant and forwarded to their PowerPoint presentation. The PowerPoint presentation at the presenter’s terminal is controlled directly from the DTP_GUI.

### 4.2.1 Terminal Domain

In this subparagraph the computational objects in the terminal domain are described. The description is followed by an overview of the functions needed for the implementation of the DTP service. These functions had to be added to the interfaces of the computational objects. For a full description of the computational objects in the terminal domain and their interfaces, see [DVC96].

**DVC_GUI Component**

**General Description:**

The DTP_GUI is integrated in the DVC_GUI. The user can control the desktop presentation by using the controls offered by the DTP_GUI. The DTP_GUI is described in detail in §4.1.3. The DTP_GUI translates the end-user’s actions into invocations on the DVC_UAP Session REQ interface in the same way as the DVC_GUI does (see [DVC96]).

The information and request signals, which are the results of actions in the DTP_GUI, follow the path described in Figure 4-6.

![Figure 4-6: DTP Requests and Information Flows](image)

Possible requests (issued by the presenter), which occur at the REQ interfaces of DVC_UAP_SESSION (1), DVC_USS (2) and DVC_GSS (3), are:

- reserve_dtp
- broadcast_start_dtp
- broadcast_goto_slide
- broadcast_close_dtp
- free_dtp
Possible information messages (transmitted to the terminal), which occur at the INFO interfaces of DVC_USS (4), DVC_UAP_SESSION (5) and DVC_GUI_STUB (6), are:

- start_dtp
- goto_slide
- close_dtp
- reserved_dtp
- freed_dtp

The DVC_GUI_STUB component forwards these commands from its INFO interface via OLE Automation to PowerPoint (7).

**DVC_GUI_STUB Component**

**General Description:**
The DVC_GUI_STUB is an intermediate object which handles the communication between CORBA and OLE. It is namely not possible to communicate directly from CORBA to OLE. As the DTP_GUI is implemented in Visual Basic, it can act only as a CORBA client (using the Orbix OLE integration) *not as a CORBA server*. The DTP_GUI_STUB acts as the CORBA server *on behalf of* the DTP_GUI.

The DVC_GUI_STUB was changed in order to make it possible to transmit commands, received from the DVC_UAP, to the PowerPoint application. The following operations were added:

- start_dtp: start PowerPoint, and open a specific PowerPoint file.
- goto_slide: display a specific slide
- close_dtp: close PowerPoint

Also some operations were added which make the DVC_GUI change its state:

- reserved_dtp: the DTP has been reserved by someone else, so it is not possible for the user to start a presentation (menu item Desktop Presentation is grayed)
- freed_dtp: the DTP has been freed, so it can be used by another user (menu item Desktop Presentation is not grayed anymore)

The operations on the INFO interface do not require any confirmation.

**Interface Description:**

- **IND(ication) interface**
  No changes.

- **INFO interface**
  The following functions were added (function, parameter, parameter type):
  
<table>
<thead>
<tr>
<th>Function</th>
<th>Parameter 1</th>
<th>Parameter 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_dtp</td>
<td>filename in</td>
<td></td>
</tr>
<tr>
<td>goto_slide</td>
<td>slidenumber in</td>
<td></td>
</tr>
<tr>
<td>close_dtp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reserved_dtp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>freed_dtp</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DVC_UAP Component

General Description:
The DVC_UAP implements the DVC specific code in the terminal, which is not related to the Graphical User Interface. It allows the DTP_GUI to issue requests to control the DTP presentation.

The functionality that must be added, corresponds to actions performed by the user who presents his slides:
- reserve_dtp: reserve the DTP service, so only one user can give a presentation at a time
- broadcast_start_dtp: send a command to all terminals in the DVC session to open the PowerPoint application with the given PowerPoint presentation
- broadcast_goto_slide: send a command to all terminals in the DVC session to go to a specific slide
- broadcast_close_dtp: send a command to all terminals in the DVC session to close the PowerPoint application
- free_dtp: free the DTP service so that it can be used by somebody else

Interface Description:

- Access Interface
  No changes.
- Fac(tory) Interface
  No changes.
- Session REQ(uest) Interface
  The following functions must be added (function, parameter, type):
  - reserve_dtp reserved/freed out
  - broadcast_start_dtp filename in
  - broadcast_goto_slide slidenum in
  - broadcast_close_dtp - -
  - free_dtp - -

- Session LIST Interface
  No changes.
- Session INFO Interface
  No changes.
- Session IND(ication) Interface
  No changes.
- Session Ret_INFO Interface
  The Ret_INFO interface enables the DVC_USS to indicate a DTP action (e.g. start, go to, close) or status change (e.g. reserved, free) has to be performed on the terminal. The operations on this interface are identical to the DVC_GUI_STUB INFO interface operations.
4.2.2 Network Domain

In this paragraph the computational objects in the network domain are described. The description is followed by an overview of the functions needed for the implementation of the DTP service. These functions have to be added to the interfaces of the computational objects of the DVC service. For a full description of the service-specific computational objects in the network domain see [DVC96].

DVC_USS Component

General Description:
The DVC_USS enables the interaction between the DVC_UAP and the DVC_GSS. It transmits information, coming from the DVC_GSS through the INFO interface, to the DVC_UAP through the RecINFO interface. It also transmits requests, coming from the DVC_UAP through the REQ interface, to the DVC_GSS through the REQ interface.

Interface Description:
- **Session REQ(uest) Interface**
The REQ interface enables the DVC_UAP to indicate that a DTP action performed by the presenter (e.g. start, go to, close) or a DTP request (e.g. reserve, free) has to be broadcast to all terminals.
The DTP operations on this interface are identical to the DVC_UAP REQ interface DTP operations.
- **Session LIST Interface**
No changes.
- **Session INFO Interface**
The INFO interface enables the DVC_GSS to indicate that a DTP action (e.g. start, goto, close) or a DTP status change (e.g. reserved, freed) has to be performed on the terminal. The operations on this interface are identical to the DVC_GUI_STUB INFO interface operations.
- **Fac(tory) Interface**
No changes.

DVC_GSS Component

General Description:
The DVC_GSS provides centralized management of the active DVC sessions. The DVC_GSS keeps track of the following information, concerning the DTP service:
- DVC Session Description (DTP service is reserved or free)
- DVC Parties List (for each party the following information is stored: DTP Role is presenter or audience)

The use of the DTP service should have impact on the On-line Charge Counter. One tick represents a fixed amount of time but a variable cost. The DVC_GSS must calculate the cost per tick by taking in consideration whether the DTP service is used during a DVC session or not.
Interface Description:

- **REQ(uest) Interface**
  The REQ interface allows the DVC_USS to issue requests to manipulate the session information list. This interface is similar to the DVC_USS REQ interface except that for every operation, the sessionId and userId of the invoking user has to be specified as "in parameter".

- **LIST Interface**
  No changes.

- **Access Interface**
  No changes.

DVC Factory Component

*General Description:*
The DVC_FACTORY component enables the DVC_UAP to create and delete DVC_USS sessions. It passes the DVC_USS Session Info interface reference to the DVC_GSS.

*Interface Description:*

- **Usage Interface**
  No changes.

4.2.3 Service Scenarios

In this subparagraph the command flow of the basic control functions is described.

Figure 4-7 illustrates the reservation of the DTP service by one of the participants in a DVC Session and the liberation of the DTP service.

![Figure 4-7: Reserve/Free DTP Service](image-url)
When the DVC_GSS is informed about the reservation/liberation of the DTP service, it automatically distributes this information to the parties involved in the DVC session. It sends a message to all conferees, this message disables/enables the possibility of starting the DTP service on their terminals.

Figure 4-8 illustrates the starting/closing of the DTP service and the control sequence of the slides.

When the DVC_GSS is informed about an action performed by the presenter, it automatically distributes this information to all conferees. These messages trigger the corresponding PowerPoint OLE-Automation function.

4.3 Development of the Basic DTP Service

One of the goals of the project was to show that TINA allows for the quick deployment of new services. In this paragraph the manpower needed to develop and integrate the basic version of the DTP service is listed. Although the developed service is a prototype, the figures show that it is indeed possible to deploy a service within a relative short amount of time. Further an overview of the problems, encountered during the development of the basic DTP service, is given.

4.3.1 Service Development

The development of the basic version of the DTP service required approximately four person-months (PM):

- Describing the functions of the DTP Graphical User Interface took about ½ PM.
- The specification of the service platform design (i.e. IDLs, operations, etc.) took about ½ PM.
- The implementation of the basic version of the DTP service took about 1½ PM.
- The integration and testing of the DTP service took about 1½ PM.
The main reasons for the relatively short development time of the basic version of the DTP service are:

- The fact that the basic version of the DTP service is an integrated part of the Desktop Video Conference, which takes care of the session management.
- The fact that an existing software package is encapsulated within the DTP service.
- The fact that the service-specific components could be kept relatively simple because of the underlying platform.

### 4.3.2 Encountered Problems

The problems encountered during implementation and integration of the DTP service can be divided into three categories:

- Delays due to problems with the DVC service:
  The basic version of the DTP service is an integrated part of the DVC service, therefore some delay was caused by assisting to solve problems with the DVC service. This resulted in a better understanding of Orbix and the SPOT platform.
- Technical problems:
  Various software products were used in this project from which first a sufficient know-how had to be built.
- Gaps and bugs in the used software products:
  Various problems were due to either missing functionality or bugs in the used software products.
  Many problems occurred (as expected) with the Orbix-OLE interface, due to lack of compatibility. One example is the type “sequence”, which represents an unbounded list of any type of elements in CORBA and is particularly difficult to represent in Visual Basic.
  Moreover an Orbix server can be multithreaded, but an OLE automation server can not. This leads to many problems when operations are invoked asynchronously.
  Furthermore the Microsoft C++ Compiler was not capable of handling strings longer than 2048 characters. This lead to problems when IDLs with interfaces offering many operations were defined. This problem was solved with a workaround by inheriting a part of the operations from a subclass.

### 4.4 Conclusion

The development of the basic version of the DeskTop Presentation service proved that it is possible to deploy a TINA service within a short amount of time. This short development time was a result of the fact that the service-specific components in TINA are kept as simple as possible, and the fact that an existing presentation package was used to show the presentation (encapsulation).
The main problem is that the short development time was also a result of the fact that the DTP service was integrated into the DVC service, which takes care of the session management. So in order to prove that TINA allows for the rapid deployment of new services, and to prove that several services can be integrated on the same platform, the DTP service had to be implemented as a stand-alone service (see Chapter 6).

Encapsulation has a positive effect on the development time, but the main problem with the encapsulation of existing software is that not all functions of the encapsulated package can be controlled using the APIs. Therefore not all features can be implemented (e.g. remote animated pointer).

Due to the gained experience and various software updates, most of the encountered problems could be avoided during the second phase of the project.
5. INTERNET TECHNOLOGIES

In this chapter it is described how Internet technologies can be used to enhance the DTP service. First the reasons for studying the applications of Internet technologies are given, followed by a review of several enhancements, and finally some conclusions are drawn.

5.1 Benefits of Internet Technologies

The two main reasons for studying possible applications of Internet technologies are the following:

- More and more attention is given to the use of lightweight terminals and network computers. The graphical user interface and applications are dynamically downloaded from the network, minimizing the installation and management effort for the end-user.
- The Web technology is now generally accepted as a de facto standard, also in the telecommunication environment, and customers are used to the look and feel of a Web browser.

5.2 Possible Applications of Internet Technologies

In this paragraph three possible enhancements, using Internet technologies, are reviewed. For each a list of advantages and disadvantages is given.

5.2.1 Storing a Presentation on a WWW Server

To avoid having to transfer the presentation from the presenter to all conferees, it is possible to store the presentation on a WWW (World Wide Web) server. The presenter only has to transfer the URL (Uniform Resource Locator), via CORBA, to all conferees. The URL is the Internet address of a slide of the presentation. For each slide the presenter has to transfer a new URL (each slide has to be on a distinct page). When the conferees receive the URL, they start a Web browser to load the specified page (see Figure 5-1).

![Figure 5-1: Storing the Presentation on a WWW Server](image)
Starting the DTP service, the following steps can be determined:

1. The presenter sends a command to all conferees (via CORBA), which automatically starts a Web browser at their terminals.
2. The presenter sends an URL to all conferees (via CORBA).
3. At the conferee’s terminal the URL is transferred from the GUI_STUB component to the Web browser.
4. The specified WWW page is loaded into the Web browser at the conferee’s terminal.
5. At the presenter’s terminal the URL is transferred from the GUI component to the Web browser.
6. The specified WWW page is loaded into the Web browser at the presenter’s terminal.

The advantages of this extension are:
- The presenter does not have to transfer the presentation to all conferees.
- No additional disk space for storing the presentation file is needed at the conferees’ terminals.

The disadvantages of this extension are:
- All parties have to have access to Internet during the entire presentation.
- No guarantees about the load time of the pages can be given (i.e. it is not possible to specify a Quality of Service).
- When the presenter wants to go to another slide he has to transfer another URL.
- The presenter has to have access to a WWW server to store his presentation.
- The presenter has to create a separate WWW page, else he cannot control the presentation by simply sending an URL to the conferees.
- The extension has practically no added value for the SPOT project or for the application, as it is not more than an externally controlled Web browser.

5.2.2 Downloading a Presentation from a WWW Server

Some of the disadvantages of the former extension can be avoided by downloading the presentation from a WWW server. The presenter transfers an URL to all conferees via CORBA. The URL is the address of the WWW page which contains the PowerPoint presentation file. Every party downloads the presentation file from the specified WWW page using a Web browser. Now the presentation can be controlled by the presenter as before; the presenter sends the slide numbers to the GUI_STUB component at the conferee’s terminal, which forwards the slide numbers to PowerPoint. Afterwards the slide number is forwarded to the PowerPoint package at the presenter’s terminal (see Figure 5-2).
Starting the DTP service, the following steps can be determined:

1. The presenter sends a command to all conferees (via CORBA), which automatically starts a Web browser and PowerPoint at their terminals.
2. The presenter sends an URL to all conferees (via CORBA).
3. At the conferee’s terminal the URL is transferred from the GUI_STUB component to the Web browser.
4. The specified WWW page is loaded into the Web browser at the conferee’s terminal.
5. The presentation is downloaded to the conferee’s terminal.
6. At the presenter’s terminal the URL is transferred from the GUI component to the Web browser.
7. The specified WWW page is loaded into the Web browser at the presenter’s terminal.
8. The presentation is downloaded to the presenter’s terminal.

Now the presenter can control the presentation by sending slide numbers, via CORBA, to the conferees as before.

The advantages of this extension are:

- The presenter does not have to transfer the presentation to all conferees.
- The entire presentation can be stored on one WWW page.

The disadvantages of this extension are:

- When the DTP service is started all parties have to download the presentation (which is often a rather big file) from a WWW page via Internet. As there is no guaranteed throughput, it may take a while before all parties have the presentation at their terminal (i.e. it is not possible to specify a Quality of Service).
- The presenter has to have access to a WWW server to store his presentation.
- The extension has practically no added value for the SPOT project or for the application, as it is not more than downloading a file using a WWW browser.
5.2.3 Loading a DTP Applet

In the former extensions it was assumed that the components which were necessary to run the DTP service were already present at the end-user’s terminal. It is however also possible to go to a specific WWW page and download an applet (an applet is a program that runs in a Web browser). The main advantage of this approach is flexibility. When an end-user needs the DTP service he just downloads the applet.

After the presenter has sent a command to all conferees, which issues them to start the DTP service, they start a Web browser to download the DTP applet. Then the DTP applet is downloaded at the presenter’s terminal. The two applets do not have to be the same. The presenter needs a Graphical User Interface to send commands to the conferees. The conferees however only need a “DTP_GUI_STUB” component, which transfers the commands from CORBA to PowerPoint. Using this approach, the data size of the applet which has to be loaded at the conferee’s terminal can be strongly decreased. After the applets have been loaded on both the presenter’s terminal and the conferees’ terminals, the presenter can transfer the presentation file using a ATM connection. Now the presentation can be controlled by the presenter as before (see Figure 5-3).

![Diagram of loading a DTP applet](image)

**Figure 5-3: Loading a DTP Applet**

Starting the DTP service, the following steps can be determined:

1. The WWW page containing the DTP applet is loaded into the Web browser at the presenter’s terminal.
2. The DTP applet is started at the presenter’s terminal.
3. A command is sent to all conferees to start a Web browser and PowerPoint.
4. The WWW page containing the DTP applet (a small STUB version) is loaded into the Web browser at the conferee’s terminal.
5. The DTP applet is started at the conferee’s terminal.
6. The PowerPoint presentation file is transferred from the presenter to all conferees over an ATM connection.
7. Now the presenter can control the presentation by transferring slide numbers, via CORBA, to all conferees as before.
(8) At the conferee's terminal, the DTP applet forwards the slide number via OLE automation to PowerPoint.
(9) Then the DTP applet at the presenter's terminal forwards the slide number, via OLE automation, to PowerPoint.

The advantages of this extension are:
- The service is only present at the user's terminal when it is used. So it does not unnecessarily use system resources.
- At the conferee's terminal only a small DTP applet has to be loaded instead of the entire DTP_GUI, which results in a better performance.
- Only a service provider has to have access to a Web Server (to store the DTP applets on a WWW page).
- The DTP software does not have to be installed anymore. The applet is loaded when it is needed, and afterwards it is discarded. Therefore an end-user always has the latest version at his disposal.
- There is a considerable added value for the SPOT project, and the application. When it is possible to load TINA services dynamically, using a simple Web browser, the services can (and will) be used by more customers. Due to the standard Web browsing user interface they will be easier to use.

The disadvantages of this extensions are:
- It can not be guaranteed that the applet can be downloaded within a short amount of time. Due to network congestion the download time can be rather long (i.e. it is not possible to specify a Quality of Service).
- PowerPoint has to be controlled externally using Visual J++.

5.3 Conclusion

The main conclusion of this chapter is that Internet technologies offer lots of possibilities to enhance TINA services. Most of all the possibility of dynamically loading a TINA service applet in a Web browser is a promising enhancement: in that way end-users always have the latest version of the service at their disposal, and the standard Web browsing user interface makes services easier to use.

As no information about the external control of PowerPoint using Visual J++ could be found, the DTP applet could not be implemented. This shows the main problem with the encapsulation of existing software within TINA services: the external control of the existing software. On one hand encapsulation offers a lot of possibilities for rapid service development, but on the other hand encapsulation is limited due to the APIs of the existing software application.

The other enhancements were not implemented as they did not offer sufficient added value to the SPOT project. Therefore it was decided not to use Internet technologies to enhance the DTP service.
6. STAND-ALONE DTP SERVICE

This chapter describes the stand-alone DTP service. First the benefits of a stand-alone version of the DTP service are listed, then an overview of the service is presented, followed by a brief description of the service architecture. Finally some conclusions are drawn.

6.1 Benefits of a Stand-alone DTP Service

One of the objectives of the project was to prove that various services, created by several service providers, can be integrated on the same TINA platform, all using the generic service components provided by the platform. The use of generic service components should keep the service-specific components as simple as possible, which results in a relatively short development time.

The advantages of a stand-alone version of the DTP service are that:

- It can be proved that the TINA platform allows several services to be integrated on the same platform, all using the same generic service components.
- It can be proved that the TINA platform allows for rapid service development (as a result of both software reuse and encapsulation).

A disadvantage of the stand-alone version of the DTP service is that:

- No new functionality is created.

As the stand-alone DTP service can be used to prove the advantages of the TINA platform, it was decided to implement the stand-alone DPT service in the second phase of the project.

6.2 Service Overview

In this paragraph the stand-alone DTP service is briefly described. First a general overview of the service is given, followed by a brief description of the DTP Graphical User Interface. A more detailed description of the DTP Graphical User Interface can be found in [XDTP97] (see also Appendix A).

6.2.1 Description

Using the DeskTop Presentation (DTP) Service, an end-user can present a slide-show to other end-users using Microsoft PowerPoint (see Figure 6-1).
To do so the user who wants to show a presentation has to start a DTP session and invite other users into the session. In this version of the DTP service, the file containing the presentation has to be present on all terminals before starting the DTP service. After a session is started, the presentation is automatically loaded by Microsoft PowerPoint. Now the presenter is able to control the sequence of the slides at all conferees' terminals using the controls provided by the DTP Graphical User Interface.

6.2.2 Graphical User Interface

The DTP Graphical User Interface (DTP_GUI) can be started when the user is logged-in into the SPOT-A retailer domain. The DTP_GUI offers an end-user the possibility to start a DTP session with other end-users. Further the end-user can be invited into a session by another end-user.

To start a DTP session, the user has to setup the session parameters, using the session setup window (see Figure 6-2). A list with users, who are subscribed to the service, is automatically loaded.

Figure 6-2: Session Setup Window (Left: Setup Panel; Right: Parties Panel)
When the user has started a session successfully, the DTP workspace appears (see Figure 6-3). The workspace contains a frame with the various control buttons (only available to the presenter), a frame which contains the presentation filename and a frame which contains a list with selected conferees. Some session information is displayed in the status bar (e.g. the charge counter).

Other services (e.g. the Address Book) can be started from the Other Services menu. A detailed description from the Graphical User Interface can be found in [XDTP97].

6.3 Service Architecture

In this paragraph the architecture of the stand-alone DTP service is briefly described. The description is limited to an overview of the interfaces of the DTP service-specific components (see Figure 6-4). A description of the SPOT generic service components can be found in [SCC96]. A more detailed description of the service architecture, together with a description of the service scenarios, can be found in [DTP97] (see also Appendix B).

The DTP service-specific components implement the functionality which is required to offer the DTP service features to an end-user, and which is not provided directly by the generic service components. The components are divided into a terminal-part and a network-part.
The TINA generic service components offer generic capabilities to invoke new service sessions. The DTP service-specific components implement additional functionality, needed for the implementation of the DTP service, and not provided by the generic service components.

### 6.3.1 Terminal Domain

In this subparagraph the DTP service-specific computational objects in the terminal domain are briefly described. The following components can be distinguished in the terminal domain:

- **DTP_GUI**: The DTP GUI actually consists of two components: DTP_GUI and DTP_GUI_STUB. This decomposition is not really necessary conceptually, but is introduced for engineering reasons. The DTP_GUI is implemented in Visual Basic and can act only as a CORBA client (using the Orbix OLE integration) *not as a CORBA server*. The DTP_GUI_STUB acts as the CORBA server *on behalf of* the DTP_GUI.

- **DTP_UAP**: The DTP User Application (DTP_UAP) interacts with DTP service-specific components in the network domain for the management of the DTP session. It interacts with the GSEP and UAPAccessClient for the DTP service adaptation. It uses the Configuration interfaces of the Device Controller Objects (DCOs) to manage local devices (e.g. ATM board). Finally, it also interacts with the DTP_GUI and DTP_GUI_STUB for the interaction with the end-user.
DTP_GUI Component

*General Description:*
The DTP_GUI Computational Object offers the end-user the ability to use the features offered by the DTP Service. The GUI is described in detail in the SPOT-A deliverable [XDTP97].

The DTP_GUI component translates the end-user actions into invocations on the DTP_UAP Access and Session REQ interfaces.

Apart from handling the graphical aspects of the DTP, the following functionality is also provided by the DTP_GUI:

- Store the userld of the user logged in to the SPOT-A retailer domain on the current terminal.
- Check consistency of a number of end-user requests, such as:
  - do not use invalid userlds (userIds can only be provided by selecting them from a list of valid userIds)
  - do not add the same party twice to same session
  - do not add more users than the service defined maximum
- Disable presenter-restricted features for the conferees.

DTP_GUI_STUB Component

*General Description:*
The DTP_GUI_STUB is an intermediate object which eases the communication between the DTP_UAP and the DTP_GUI. It acts as the gateway from the CORBA world towards the non-CORBA world. When the DTP_UAP wants to invoke an operation on the DTP_GUI, it will invoke this operation on the DTP_GUI_STUB, whose responsibility it is to forward it to the DTP_GUI and correlate the corresponding answer.

*Interface Description:*
The DTP_GUI_STUB offers an INDication and an INFO interface towards the DTP_UAP (more information can be found in [DTP97]):

- **INDication interface**
  The INDication interface offers an operation which require an explicit answer.
  - `join_in_dtp_session`: Request from a presenter to join an existing session

- **INFO interface**
  The INFO interface offers operations which do not require any confirmation.
  - `add_dtp_party_info`: indicate that a party has been added to the session.
  - `delete_dtp_party_info`: indicate that a party has been deleted from the session.
  - `delete_dtp_session_info`: indicate that the session has been deleted.
  - `goto_slide`: indicate that the presenter has gone to the another slide.
  - `update_dtp_charge_counter_info`: update the value of the charge counter.
DTP_UAP Component

General Description:
The DTP_UAP implements the DTP-specific code which resides in the terminal and which is not related to the GUI aspects. It can be decomposed in two main parts:

- The first part is session independent and is launched before any DTP session is created. This part exports the Factory and Access interfaces.
- The second part will be launched only when a DTP session is launched. It exports session-specific REQ, INFO, IND and Ret_INFO interfaces.

Interface Description:
The Factory and Access interfaces are created prior to the creation of any DTP session. The REQ, Ret_INFO, INFO, and INDication interfaces are dynamically created and destructed during the lifetime of a DTP session (more information can be found in [DTP97]):

- **Access Interface**
  The Access interface enables the DTP_GUI to start a new DTP session.
  - `request_dtp_service`: session initiation by the presenter. A userlist is passed as inout parameter, because the operation will return the list of users successfully added to the session.

- **Factory Interface**
  The Factory interface enables the UAP Access Client to create a new DTP_UAP service session for both “incoming” (join_in_session) and “outgoing” (createUAPSessionClient) DTP sessions.
  - `createUAPSessionClient`: creates new INFO, IND, Ret_INFO, and REQ interfaces.
  - `join_in_session`: invitation to join a service session (and if accepted, create new INFO, IND, Ret_INFO, and REQ interfaces).
  - `deleteUAPSessionClient`: delete INFO, IND, Ret_INFO, andREQ interfaces for a certain DTP Session.

- **Session REQuest Interface**
  Allows the DTP_GUI to issue requests for the modification of the DTP session configuration.
  - `wait_dtp_join_in_session`: when a party has accepted an invitation to join in a session (issued by the presenter), its DTP_GUI should invoke this operation. It returns when the party has actually been added.
  - `delete_dtp_session`: the presenter can close a session. All parties are removed from session.
  - `exit_dtp_session`: only a conferee can exit a session.
  - `broadcast_goto_slide`: the presenter can broadcast a new slide number to all conferees.

- **Session INFO Interface**
  This interface is identical to the GSEP Session Control INFO interface. The operations on the INFO interface return immediately without passing any information to the GUI, because this information is received through the Ret_INFO interface in a DTP-specific format. The only exception is the `delete_session_info` operation.
• **Session IND(ication) Interface**
  This interface is identical to the GSEP Session Control Indication interface. The operations on the IND interface return immediately without requesting assistance from the end-user. The only exception is the `join_in_session_ind`.

• **Session Ret_INFO Interface**
  The Ret_INFO interface enables the DTP_USS to indicate a DTP session modification (e.g. cost update). The operations on this interface are identical to the DTP_GUI_STUB INFO interface operations.

6.3.2 **Network Domain**

In this subparagraph the DTP service-specific components in the network domain are briefly described. The DTP User Service Segment (DTP_USS) and the DTP Global Service Segment (DTP_GSS) Computational Objects map onto the TINA Service Segment. The DTP_Factory is responsible for managing the DTP_USS Session interface objects.

**DTP_USS Component**

*General Description*:

The DTP_USS enables the interaction between the DTP_UAP and the DTP_GSS. It has the following functions:

- Enable the DTP_UAP to request to modify the DTP session management information.
- Inform the DTP_UAP of a modification in the DTP session (via the DTP_UAP Session RecINFO interface). This includes the update of the billing information for the ongoing session.
- Enable the DTP_UAP to retrieve information about the DTP session.

*Interface Description*:

The DTP_USS offers DTP Session REQ, LIST and INFO interfaces which are created per DTP session. The REQ and LIST interfaces are used by the DTP_UAP, while the INFO interface is used by the DTP_GSS. The sessionId and userId are stored by the DTP_USS Session and are used to communicate with the DTP_GSS. The Factory interface is offered to the DTP_Factory to manage the life-cycle of the DTP_USS Session interfaces. A brief description of each interface and its operations is given below (more information can be found in [DTP97]):

- **Session REQuest Interface**
  This interface is offered towards the DTP_UAP.
  - `wait_dtp_join_in_session`: wait with join session until session status is unlocked.
  - `add_dtp_session`: report that a new DTP Session is created. This request is forwarded to the DTP_GSS which will create a new entry in the session list and returns a unique SessionName for this new session. The DTP_USS will store the SessionId.
**Development of a Multimedia Service on a TINA Platform**

- **delete_dtp_session**: report that a DTP Session is deleted. This request is forwarded to the DTP_GSS which will remove the entry in the session list.
- **add_dtp_party**: report that a new party is added to the session
- **delete_dtp_party**: report that a party is deleted from the session
- **broadcast_goto_slide**: report that the presenter has gone to another slide
- **set_dtp_status**: set the semaphore of a particular session to either *locked* to prevent simultaneous access to the session info or *unlocked* to free the semaphore. If set_status is used to lock a session, the function returns only when the semaphore of the session has been successfully locked

**Session LIST Interface**
This interface allows the DTP_UAP to retrieve information related to the DTP Session which is stored by the DTP_GSS.
- **list_dtp_session**: retrieve session information.

**Session INFO Interface**
This interface is the same as the DTP_UAP Session Ret_INFO

**Fac(tory) Interface**
This interface is offered towards the DTP_Factory
- **create_dtp_uss**: create new DTP_USS Session interfaces.
- **delete_dtp_uss**: delete DTP_USS Session interfaces.

**DTP_GSS Component**

**General Description:**
The DTP_GSS provides centralized management of the active DTP sessions. The DTP_GSS keeps track of the following information per DTP session (the \( ^T \) indicates that the information is also relevant to the generic service components):

- **SessionId\(^T\)**: the identifier of the session used by the generic service components.
- **DTP Session Name**: this name will be given by the DTP_GSS at start-up of the DTP Session.
- **DTP Session Description**: charging scheme and name of the presentation.
- **DTP Parties List**: UserId\(^T\), Role and PartyId\(^T\) (stored for each party)
- **Status**: a semaphore that indicates that the data related to this session can be modified or that it is already being accessed by another user. This prevents the simultaneous access of this data.
- **OwnershipID\(^T\)**: unique identifier within the service session graph of the session relationship which expresses the ownership of the session.
- **On-line Charge Counter**: the DTP_GSS computes the on-line billing information for every DTP session.

**Interface Description:**
The DTP_GSS offers three types of interfaces. The REQ and LIST interfaces are offered towards the DTP_USS, the Access interface is offered towards the DTP_Factory (more information can be found in [DTP97]):

---

46
• **REQ(uest) Interface**
The REQ interface allows the DTP_USS to issue requests to manipulate the session information list. This interface is similar to the DTP_USS REQ interface except that for every operation, the sessionId and userId of the invoking user has to be specified as “in parameter”.

• **LIST Interface**
The LIST interface allows the DTP_USS to extract data from the session information list. This interface is similar to the DTP_USS LIST interface except that for every operation, the sessionId and userId of the invoking user has to be specified as “in parameter”.

• **Access Interface**
The Access interface allows the DTP_Factory to report the creation of a new DTP_USS Session and pass on the relevant Session INFO interface.
  - `add_dtp_uss`: report creation of DTP_USS Session.

**DTP Factory Component**

**General Description**
The DTP_Factory CO enables the DTP_UAP to create and delete DTP_USS sessions. It passes the DTP_USS Session INFO interface reference to the DTP_GSS.

**Interface Description**
The DTP_Factory offers the Usage interface towards the DTP_UAP (more information can be found in [DTP97]):

- **Usage Interface**
  - `create_dtp_uss`: create DTP_USS session.
  - `delete_dtp_uss`: delete DTP_USS session.

### 6.4 Development of the Stand-alone DTP Service

To prove that TINA allows for rapid service deployment, an overview of the time, needed for the development of the stand-alone DTP service, is given. In this paragraph also an overview of the problems, encountered during the development of the stand-alone DTP service, is given.

#### 6.4.1 Service Development

The development of the stand-alone DTP service required approximately two person-months (PM):

- Describing the functions of the DTP Graphical User Interface took about ½ PM.
- The specification of the service platform design (i.e. IDLs, operations, etc) took about ½ PM.
- The implementation and integration of the stand-alone DTP service took about 1 PM.
The development time of the DeskTop Presentation service was relatively short due to the stable underlying platform, and the fact that an existing application was encapsulated.

6.4.2 Encountered Problems

The problems encountered during implementation and integration of the stand-alone DTP service were mainly caused by problems with the generic service components. It appeared that the platform had difficulties deploying more than one service at the time. Eventually an update, which solved this problem, was provided by Alcatel. However the file structure on the terminal had to be changed (the generic service components and the service specific components, which ran before in one process had to be separated.

Most of the problems which were encountered during the development of the basic DTP service (see §4.3.2), could be avoided during the development of the stand-alone DTP service. This because:

- Reasonable knowledge of the various software products had been obtained during the development of the basic DTP service.
- Various updates of the used software products resulted in a stable development environment.

6.5 Conclusion

The development of the stand-alone service showed that the SPOT platform allows for the rapid deployment of new services. The generic service components offer a stable platform, which takes care of all tasks concerning session and connection management. Therefore the service-specific components can be kept relatively simple, reducing the development time, and most of all the time needed to test the service. However it should be noted that the short development time is also caused by the fact that an existing application is encapsulated, and the fact that parts of the DVC code could be reused.

Extensions for the stand-alone DTP service can be divided into two categories:

- **Planned extension:**
  - Automatic File Transfer of the presentation to all conferees via ATM.

- **Possible extensions:**
  - Introduction of more elaborate charging schemes.
  - Introduction of charging related to usage of network resources.
  - Automatically launching the DTP service at a conferee’s terminal when this conferee is invited to join a DTP Session.
  - Conversion from charge units to normal currencies (currency will be user dependent).
7. FILE TRANSFER FEATURE

In this chapter is described how a file transfer feature can be used to enhance the DTP service. First an overview of the benefits of a file transfer feature is given, followed by a review of two possible file transfer mechanisms. Then one mechanism is specified in more detail and its development is described. Finally some conclusions are drawn.

7.1 Benefits of a File Transfer Feature

To use the DeskTop Presentation service all conferees must have access to the PowerPoint presentation, which the presenter wants to show. In the previous versions of the DeskTop Presentation service the presentation file was assumed to be present on all terminals before starting the service. As this is not very convenient, a file transfer feature was, already in a very early stage, mentioned as an important extension to basic DTP service (see Table 3.1). Therefore it was decided to extend the stand-alone DTP service with a file transfer.

7.2 Possible File Transfer Mechanisms

The end-users within the SPOT project are interconnected via ATM, therefore the file transfer has to be done via ATM. There are basically two solutions for a file transfer over ATM:

- The file can be transferred directly over ATM.
- The file can be transferred using IP (Internet Protocol) over ATM.

In order to decide which mechanism should be used for the implementation of the file transfer, the two possibilities are examined below.

7.2.1 File Transfer Directly over ATM

The transfer of the presentation file directly over ATM can be divided into two steps:

- **Setup of a connection:** The generic service components are used to setup a point-to-multipoint connection. Using this connection the presenter can send the data to all conferees at once. The generic service components are also used to setup a point-to-multipoint connection for the transfer of the audio- and video data.

- **Data transfer:** To transfer the file via the previously setup connection, the file has to be put into cells. These cells are sent to all conferees via the point-to-multipoint connection. At the conferees’ terminals, the cells are reconstructed to a file.
The main problem with this reconstruction is the order of the cells: the cells must be placed in the correct order. The ATM Adaptation Layer 3/4 (AAL 3/4) provides means to reconstruct the correct cell order, AAL 5 however does not (see [CUT93]). As the used ATM boards only support AAL 5, a protocol has to be written to ensure that the cells are retrieved in the correct order.

The advantages of a file transfer directly over ATM are:

- The generic service components can be used to setup a connection.
- Using the point-to-multipoint connection the file can be transferred from the presenter to all conferees at once.
- The kernel Transport Network (kTN) is not used for data transfer. The kernel Transport Network is a signaling network, which is used for the interconnection of the different TINA terminals and servers. Using a file transfer directly over ATM, a new connection is setup, which saves the capacity of the kTN for the transfer of control signals.

The disadvantage of a file transfer directly over ATM is:

- A protocol has to be written, to ensure that the cells are retrieved in the correct order.

It would be a good solution to transfer the file directly over ATM, using the generic service components to setup a channel for the data transfer. However a protocol has to be developed to ensure that the cells are retrieved in the correct order, as the used ATM boards do not support AAL 3/4.

It was decided not to implement the file transfer directly over ATM as:

- Developing a protocol to retrieve the cells in the correct order has practically no added value for the project
- The available version of the ATM driver did not run together with the other drivers.

7.2.2 File Transfer Using IP over ATM

The simplest solution to transfer the presentation from the presenter to all conferees, is using IP over ATM. In this case existing tools can be used for the transfer of the presentation.

IP over ATM uses a Virtual Channel (within a Virtual Path) for the data transfer between two terminals. The various locations within the SPOT project are interconnected by an ATM network, between these locations Virtual Channels have been defined for the kernel Transport Network.

The better solution would be to create new Virtual Channels between the various locations, but to do so, an additional IP address has to be assigned to each terminal. As this is not very convenient (especially not when more than a few users are involved), the existing Virtual Channels (used by the kernel Transport Network) are used for the transfer of the presentation.
The advantage of a file transfer using IP over ATM is:

- Using the kernel Transport Network for the transmission of data is a rather simple solution which takes just little time to specify and implement. The connections are already setup, and existing tools can be used for the actual data transfer.

The disadvantages of a file transfer using IP over ATM are:

- Using the kernel Transport Network for the transmission of data is not what it has been designed for. The kernel Transport Network is made for the transfer of signaling and control data between the various terminals/servers, not for the transfer of large amounts of data.
- The presenter has to transfer the file to all conferees individually. Using IP over ATM, the presenter can not setup a point-to-multipoint connection to transfer the presentation to all conferees at once.

Although it is not the most decent solution, the file transfer was implemented using IP over ATM. As the file transfer is only used when the DTP service is started, the load on the kernel Transport Network only has minor influence on the performance of the system. As existing tools can be used for the data transfer, the total development time is rather small, the added value for the project however is large.

### 7.3 Architecture of the File Transfer Feature

This paragraph describes the modifications to the DeskTop Presentation service architecture, which are necessary to integrate the file transfer feature into the DTP service. First the general concept of the file transfer feature is described, followed by an overview of the additional functions needed in the various computational objects in the terminal and network domain. Finally the file transfer scenario is described.

Transferring a file, the following steps can be determined:

**At the presenter's terminal**

1. Compress the presentation file.
2. UUencode the compressed presentation file.
3. Write the UUencoded file in a string.
4. Send the string to all conferees via CORBA.

**At all conferees' terminals**

5. Write the contents of the string into a file.
6. UUdecode this file.
7. Decompress the UUdecoded file in a temporary directory (e.g. “C:\TEMP”).
8. Change the path in the filename to that of the temporary directory (e.g. “C:\TEMP”).
Concerning the steps, mentioned above, the following remarks can be made:

- **UUencoding/UUdecoding** converts the contents of the file to ASCII characters. Although it enlarges the total amount of data to be transferred, it eases the transmission.
- The tool used for the compression/decompression of the presentation file puts a limitation on the length and appearance of the filename: the filename must be MS-DOS conform (this means that the filename may consist of up to eight characters, and the extension may consist of up to three characters).

In Figure 7-1 the components which had to be modified to implement the file transfer feature are shown. The interfaces, which had to be modified, are represented by light gray rectangles.

![Figure 7-1: Components Needed for the File Transfer](image)

The DTP_UAP at the presenter's terminal transmits the presentation via the REQ interfaces of the DTP_USS (1) and DTP_GSS (2) to the DTP_GSS. The additional operation needed at the REQ interfaces is:

- broadcast_file

The DTP_GSS distributes the information to all conferees via the INFO interface of the DTP_USS (3) and the Ret_INFO interface of the DTP_UAP (4). The additional operation needed at the INFO and Ret_INFO interface is:

- transfer_file

In the following subsections the various components are described in more detail.

### 7.3.1 Terminal Domain

In this subparagraph the computational objects in the terminal domain are described. In the terminal domain the following computational objects can be distinguished: DTP_GUI, DTP_GUI_STUB and DTP_UAP. The description in this paragraph is limited to the changes needed for the implementation of the file transfer. A more detailed description of the computational objects in the terminal domain can be found in: §6.3.1 and [DTP97].
DTP.GUI and DTP.GUI_STUB Component

General Description:
The file transfer is fully transparent for the end-user. This means that the presentation file, selected by the presenter, is automatically transferred to all conferees and placed in a temporary directory on their terminals (e.g. “C:TEMP”).

As the user has no influence at all on the transfer of the presentation file, the DTP.GUI and DTP.GUI_STUB Component do not have to be changed for the implementation of the file transfer feature.

Interface Description:
- INDication Interface
  No changes.
- INFO Interface
  No changes.

DTP.UAP Component

General Description:
The actual file transfer is done within the DTP.UAP component:

- When the DTP service is started, the presentation file is compressed, UUencoded, and written in a string on the presenter’s terminal. Then the broadcast_file procedure on the DTP.USS REQ interface is called, and the string containing the presentation file is passed as a parameter.
- At the conferee’s terminal, the string is received as a parameter of the transfer_file procedure on the DTP.UAP INFO interface. Then the string is written in a file, UUdecoded and decompressed in a temporary directory. Finally the path in the filename is changed to this temporary directory before the filename is sent to the DTP.GUI_STUB.

Interface Description:
- Access Interface
  No changes.
- Factory Interface
  No changes.
- Session REQuest Interface
  No changes.
- Session INFO Interface
  No changes.
- Session INDication Interface
  No changes.
- Session Ret_INFO Interface
  - transfer_file: to transfer the string, containing the presentation file, to the conferee.
7.3.2 Network Domain

In this subparagraph the computational objects in the network domain are described. In the network domain the following computational objects can be determined: DTP_USS, DTP_GSS and DTP_Factory. The description in this paragraph is limited to the changes needed for the implementation of the file transfer. A more detailed description of the computational objects in the network domain can be found in: §6.3.2 and [DTP97].

DTP_USS Component

*General Description:*  
The DTP_USS enables the interaction between the DTP_UAP and the DTP_GSS. It forwards the string containing the presentation via the broadcast_file procedure to the DTP_GSS. Further it forwards the string it receives from the DTP_GSS to the DTP_UAP via the transfer_file procedure.

*Interface Description:*
- **Session REQ(uest) Interface**  
  - broadcast_file: to send the string, containing the presentation file, to the DTP_GSS, which broadcasts it to all conferees in the session.
- **Session LIST Interface**  
  No changes.
- **Session INFO Interface**  
  - transfer_file: to transfer the string, containing the presentation file, to the conferee.
- **Fac(tory) Interface**  
  No changes.

DTP_GSS Component

*General Description:*  
The DTP_GSS receives a string from the DTP_USS. It forwards the string to all conferees in the session. To do so it invokes the transfer_file procedure on the INFO interface of the DTP_USS.

*Interface Description:*
- **REQ(uest) Interface**  
  - broadcast_file: to send the string, containing the presentation file, to the DTP_GSS, which broadcasts it to all conferees in the session.
- **LIST Interface**  
  No changes.
- **Access Interface**  
  No changes.

DTP_Factory Component

*General Description:*  
No changes to the DTP_Factory component have to be made to implement the file transfer.
Interface Description:

- Usage Interface
  No changes.

7.3.3 Feature Scenario

In Figure 7-2 the file transfer scenario is displayed.

![Diagram of file transfer scenario](image)

Figure 7-2: File Transfer Scenario

The filename and file are transferred from the presenter's terminal to the DTP_GSS, from there the file is distributed to all conferees' terminals.

7.4 Development of the File Transfer Feature

In this paragraph the development of the file transfer feature is described. As one of the goals of the project is to prove that TINA allows for rapid service deployment an overview of the time, needed to develop the feature, is presented. Further the problems encountered during the development of the file transfer feature are listed.

7.4.1 Feature Development

The development of the file transfer feature took about two Person Weeks (PW):

- The specification of the file transfer feature (i.e. IDLs, operations, etc.) took about one PW.
- The implementation of the file transfer feature and the integration into the DTP service took about one PW.

The development time of the file transfer feature was relatively short. This was mainly caused by the fact that the file transfer was implemented using IP over ATM: no additional connections had to be created and the file transfer could be done using an existing tool. The implementation of the file transfer directly over ATM would have taken much more time, as a protocol for retrieving cells in the correct order would have to be developed.
7.4.2 Encountered Problems

No problems were encountered during the specification and implementation of the file transfer feature.

7.5 Conclusion

The file transfer feature is a major enhancement for the DeskTop Presentation service, although it was decided to implement the file transfer using IP over ATM, and therefore using the kernel Transport Network for data transfer. A direct result of this decision is the short development time of the feature. It turned out that the actual load on the kernel Transport Network had no major influence on the system performance.

The following enhancements can be made to the file transfer feature:

- The file transfer can be done directly over ATM. As described in §7.2.1, this means developing a tool to retrieve the cells in the correct order.
- Another compression tool can be used, so there is no limitation to the length of the filename.
- The UUencoding/UUdecoding can be omitted, which will result in a faster transfer of the presentation file.
- Decompression is relatively slow, therefore it should be investigated whether there is an alternative for it.
- The presenter/conferees should be charged for the file transfer.

Note: Actually data transfer capabilities should be provided by the underlying platform. In that way the developed service is independent of the underlying transport network, as described in the TINA specifications. Further the data transfer capabilities can also be used by other services, which results in a decreased service development time.
8. CONCLUSIONS AND RECOMMENDATIONS

Based on the experiences, obtained during the development of the integrated Desktop Presentation service, the stand-alone Desktop Presentation service, and the file transfer feature, it can be concluded that a TINA platform allows for rapid service development. The generic service components provide a stable platform, and due to clear interface definitions the functionality provided by the generic service components could easily be integrated in the developed services. Therefore the service-specific components could be kept simple, which resulted in a major decrease of the development time.

Encapsulation also had its influence on the development time, but the encapsulation of existing software not only has a positive effect on the development time, it also prevents one from developing software that already exists, and the interface of commercial applications can be used instead of a self developed user interface. Encapsulation offers various possibilities for new services (e.g. shared document writing), but due to the limited APIs, there are limitations to the possibilities of encapsulation.

The integration of all services on the platform was rather easy due to the clear interface specifications. The interface definitions also prevented the implementations of the generic service components and the service-specific components to be dependent. This is very important as, in the future, services will also be developed by third party service providers.

Internet technologies offer lots of possibilities to enhance TINA services. Most of all the possibility of dynamically loading a TINA service applet in a Web browser is a promising enhancement: in that way end-users always have the latest version of the service at their disposal, and the standard Web browsing interface makes services easier to use. A problem occurred with the encapsulation of PowerPoint: the APIs of PowerPoint can only be controlled using Visual Basic, and not using Visual J++, therefore the DTP service was not implemented as an applet.

Based on experiences with the SPOT platform, it can be said that TINA is a promising architecture. As mentioned above, it allows for rapid service deployment, and service development by third party service providers. Internet front-ends can be used to enhance the developed services and will also increase the speed of service deployment as the development of the user interface will be much easier. Encapsulation also allows for rapid service development, but its possibilities depend on the APIs of the existing application.

Still unresolved problems are the complexity of the TINA architecture, and the required overhead: for each connection, a session has to be created, which requires a lot of control traffic. This is no problem for a high quality multimedia service, but it can be a problem for a simple voice connection between two parties. The behavior (and most of all the performance) of TINA in large networks still has to be proven.
Besides investigation of the behavior of TINA in large networks, other recommendations for future activities are: integration of data transfer capabilities in the generic service components, and the integration of an Internet front-end.

Data transfer capabilities are used by various applications, and should be provided by the generic service components. This not only because of the reduced development time, but also because data transfer involves the transport network, which should be separated from the services, especially as service developers have no a priori knowledge of the underlying transport network.

To be able to run the services on lightweight network terminals, the services should be equipped with an Internet front-end. This allows services to be loaded in a Web browser, and prevents an end-user from having to install the service on his terminal.
ACKNOWLEDGEMENTS

I would like to express my gratitude to Professor J. de Stigter, L. Lehmann and C. Würgler who all helped me bringing my work to a good end.

Bern, December 1997

Thomas T. Kijftenbelt
Gedempte Gracht 437
2512 AM Den Haag
The Netherlands

tel: +31-6-53 875 976
e-mail: t.t.kijftenbelt@stud.tue.nl
ABBREVIATIONS

AAL  ATM Adaptation Layer
API  Application Programming Interface
ATM  Asynchronous Transfer Mode
CO  Computational Object
CORBA  Common Object Request Broker Architecture
CSCW  Computer Supported Common Work
DCO  Device Controller Object
DII  Dynamic Invocation Interface
DPE  Distributed Processing Environment
DSI  Dynamic Skeleton Interface
DTP  DeskTop Presentation
DVC  Desktop Video Conference
FTP  File Transfer Protocol
GSEP  Generic Session EndPoint
GSS  Global Service Segment
GUI  Graphical User Interface
IDL  Interface Definition Language
IP  Internet Protocol
kTN  kernel Transport Network
NCCE  Native Computing Communication Environment
NRIM  Network Resource Information Model
ODL  Object Definition Language
OLE  Object Linking and Embedding
OMA  Object Management Architecture
OMG  Object Management Group
ORB  Object Request Broker
PM  Person Months
PW  Person Weeks
SPOT  Service Pilot On TINA
TINA  Telecommunications Information Networking Architecture
UAP  User Application
URL  Uniform Resource Locator
USCM  Universal Service Component Model
USS  User Service Segment
WWW  World Wide Web
REFERENCES

[BAR93] W.J. Barr; T. Boyd; Y. Inoue
The TINA Initiative
IEEE Communications Magazine, March 1993, p. 70-76

[BEN97] Proceedings of International Switching Symposium '97
Implementation of Services for Computer Supported Cooperative Work on TINA: The SPOT Project
J. Bengtsson; P. Hellemans; L. Lehmann; N. Mercouroff
Toronto, Canada, September 1997

[CUT93] L.G. Cuthbert; J.-C. Spanel
ATM: The broadband telecommunications solution
The Institution of Electrical Engineers, London, United Kingdom, 1993

[DTP97] Alcatel Telecom; Swisscom
SPOT: Service Platform Design - DeskTop Presentation Components
November 1997 (Version 3.1)

[DUP95] F. Dupuy; C. Nilsson; Y. Inoue
The TINA consortium: toward networking telecommunications information services
IEEE Communications Magazine, Issue 11, November 1995, p. 78-83

[DVC96] Alcatel Telecom; Swisscom
SPOT: Service Platform Design - Desktop Video Conference Components
November 1996 (Version 2.1)

[IFIP95] Proceedings of the IFIP TC6 Conference on Intelligent Networks and New Technologies
An overview of the telecommunications information networking architecture
Lyngby, Denmark, August 1995, p. 296-307

[IONA] IONA Home Page (Further reading and resources)
http://www.iona.com/Training/reading.html

[LEH97] Proceedings of TINA’97
Service Creation on a TINA Platform: an Experience Report
L. Lehmann; M. Cadorin; C.E. Würgler
Santiago de Chili, Chili, November 1997

[OMG] Object Management Group Home Page
http://www.omg.org
<table>
<thead>
<tr>
<th>Reference</th>
<th>Authors</th>
<th>Title</th>
<th>Date</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SCC96]</td>
<td>Alcatel Telecom; Swisscom</td>
<td><strong>SPOT: Service Platform Design - Service Common Components</strong></td>
<td>November 1996</td>
<td>Version 2.1</td>
</tr>
<tr>
<td>[XDTP97]</td>
<td>Alcatel Telecom; Swisscom</td>
<td><strong>SPOT: External Service Specifications - DeskTop Presentation Service</strong></td>
<td>November 1997</td>
<td>Version 3.2</td>
</tr>
<tr>
<td>[XDVC97]</td>
<td>Alcatel Telecom; Swisscom</td>
<td><strong>SPOT: External Service Specifications - Desktop Video Conference</strong></td>
<td>February 1997</td>
<td>Version 2.2</td>
</tr>
</tbody>
</table>
APPENDIX A

EXTERNAL SERVICE SPECIFICATIONS

- DTP SERVICE
Service Pilot On TINA (SPOT-A)

External Service Specifications

DeskTop Presentation Service

<table>
<thead>
<tr>
<th>History</th>
<th>Version</th>
<th>Description</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>28/08/1997</td>
<td>3.0</td>
<td>Stand-alone DTP service</td>
<td>Alcatel Telecom:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E. Montes de Oca</td>
</tr>
<tr>
<td>04/09/1997</td>
<td>3.1</td>
<td>Stand-alone DTP service, revised version</td>
<td>Swisscom:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>T.T. Kijftenbelt</td>
</tr>
<tr>
<td>10/11/1997</td>
<td>3.2</td>
<td>Stand-alone DTP service, integrated file transfer</td>
<td>L. Lehmann</td>
</tr>
</tbody>
</table>
Table Of Contents

1. SCOPE 4

2. OVERALL DESCRIPTION 5

3. GENERAL CONCEPTS 6
   3.1 Session Name 6
   3.2 Session Charging 6
   3.3 Maximum Number of Parties 6
   3.4 Service Start-up 6

4. DTP WORKSPACE 7
   4.1 Title Bar 7
   4.2 Menu Bar 7
   4.3 DTP Controls 13
   4.4 Status Bar 13

5. OVERVIEW OF DTP SERVICE FEATURES 14

6. OVERVIEW OF POSSIBLE FUTURE EXTENSIONS 15
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIGURE 4-1</td>
<td>DTP WORKSPACE</td>
<td>7</td>
</tr>
<tr>
<td>FIGURE 4-2</td>
<td>SESSION SETUP WINDOW (SETTINGS PANEL)</td>
<td>8</td>
</tr>
<tr>
<td>FIGURE 4-3</td>
<td>SESSION SETUP WINDOW (PARTIES PANEL)</td>
<td>9</td>
</tr>
<tr>
<td>FIGURE 4-4</td>
<td>DTP WORKSPACE WINDOW</td>
<td>10</td>
</tr>
<tr>
<td>FIGURE 4-5</td>
<td>INVITATION WINDOW</td>
<td>10</td>
</tr>
</tbody>
</table>
References

[ServiceOverview]  External Service Specifications - Overview
[AdminServices]    External Service Specifications - Administrative services
[DVC]              External Service Specifications - DVC service
[DTP]              External Service Specifications - DTP service
[Chat]             External Service Specifications - Chat service
1. Scope

The external specification of the services that will be developed within SPOT-A jointly by Alcatel, Swisscom and Telia are described in a series of documents ([AdminServices], [DVC], [DTP] and [Chat]). These documents elaborate the end-user’s view on these services. For an overview see [ServiceOverview].

This document presents the external service specifications of the DeskTop Presentation service.
2. Overall Description

The DTP service allows an end-user to present his slides to other end-users using Microsoft PowerPoint (version 7.0). An end-user involved in a DTP session is referred to as a party in the session. A DTP session is at all times owned by exactly one party. This owner is the party who requested the creation of the session. The owner of the session is referred to as the presenter, while the other parties are referred to as conferees.

The following limitations apply to this version of the stand-alone DTP service:

- A terminal can only be involved in one DTP session at a time.
- A DTP session consists of one presenter and a service-defined maximum of other participants. A DTP session can exist with just the presenter involved.
- The presenter can invite other end-users in the session, but this has to be done before the session is started. After the session has been started no additional conferees can be invited.
- The presenter can stop the session (and remove all parties from the session at once).
- During the session every party, except the presenter, can leave the session.
- Inviting a list of conferees to join the session will be handled as a list of independent requests. This implies that if the presenter creates a session with, for example, two conferees, the outcome of the addition of the first conferee has no influence at all on the outcome of the addition of second conferee (and vice versa). The creation of a new DTP session with a certain number of conferees is treated as the creation of an ‘empty’ DTP session - that is a session with only the presenter involved - followed by the addition of the conferees. Therefore, the DTP session creation can be successful even if none of the invited conferees accepts the invitation.
- The PowerPoint file is automatically sent to all conferees when a session is started. It is placed in the temporary directory (C:\TEMP), and is not deleted after the presentation is ended. Due to the utilities used for the file transfer, the filename of the presentation has to be MS-DOS conform (i.e. the filename can consist of maximal eight characters and the extension can consist of maximal three characters, both without blanks).
- It is assumed that all participants have Microsoft PowerPoint installed on their terminal.
3. General Concepts

3.1 Session Name
A name is assigned to each DTP session. The presenter can propose a name when he starts a new DTP session. This name is concatenated with the presenter's UserId to make up the complete session name. This complete name has to be unique, i.e. no two DTP sessions can simultaneously exist with the same name. When the name proposed by the presenter does not result in an unique session name the retailer system will add an unique identifier to the session name on behalf of the presenter.

3.2 Session Charging
A charging scheme is associated with each DTP session. Before starting the DTP service a charging scheme has to be selected. In this version of the stand-alone DTP service only two charging schemes are supported:
- 100% of the cost for the presenter
- The cost is equally split among the different parties in the session
More elaborate charging schemes are possible as future extensions (e.g. the session is paid for by one conferee or the session is paid by all conferees except the presenter).

The actual charge will be presented as a number of "charge units". A charge unit could equal, for example, 1.2 Dutch guilders. The way charge units are being accredited is explained in [AdminServices].

3.3 Maximum Number of Parties
No more parties than the service-defined maximum number of parties can be involved in a session.

3.4 Service Start-up
Start-up of the DTP service is achieved by double-clicking on the SPOT-A DeskTop Presentation Service icon in the "retailer domain program group". From this program group all SPOT-A services can be started (see [ServiceOverview]).

Before starting a SPOT-A service, the end-user has to be logged in. If the end-user has not performed the SPOT-A Login, the Login window is opened first. Before starting the service, the login has to be completed successfully (see [ServiceOverview]). After that the DTP workspace appears (see Figure 4-1).
4. DTP Workspace

![DTP Workspace](image)

Figure 4-1: DTP Workspace

Note: There can only be one DTP workspace on a terminal at a time.

4.1 Title Bar

Contains the name of the service, the UserId of the logged in end-user and the name of the session when a session is opened.

The minimize, maximize and close buttons for standard windows operations also appear (however the maximize button is disabled). Pressing the close button equals selecting the **Exit** command in the **Session** menu.

4.2 Menu Bar

Contains the different drop-down menus and their commands. Clicking a menu results in pulling down the command list. All menus and commands are explained in detail in the subsequent sections.

4.2.1 Session Menu

The Session menu contains the session control commands. The first command changes its function in a context sensitive way. The possible functions of this command are **Setup... session** (when the user is not in a DTP session) and **Stop/Leave** (when the user is in a DTP session as presenter respectively conferee). The **Exit** command closes the DTP GUI.
4.2.1.1  Setup... Command

When selecting this command, the Session Setup window is opened (see Figure 4-2). This window contains a text field to enter a session name, two panels—one to define settings and another to select participating parties—, Start and Cancel buttons, and a status bar.

Note: When an end-user selects Setup..., he will be the presenter of the new session.

Session name:

- Type in a session name in the text field. The limit is set to 70 characters. If no unique session name is entered the retailer system will add an unique identifier to the session name on behalf of the presenter.

Settings panel:

- Select a charging scheme for the new session: “Paid by presenter” or “Shared among all parties”.
- The presentation file dialog box displays the name of the selected file. A new (or another) file can be selected by clicking the Browse... button. Clicking this button provides a standard windows dialog box to select a file.

Figure 4-2: Session Setup Window (Settings Panel)
Parties panel:

- Define the source for the **conferees from** list. The address list can be supplied by a directory service which contains a list of all users in the Provider Domain, who are subscribed to the DTP service, or by the end-user’s personal address book (see [AdminServices]). If the presenter is currently participating in a DVC session, a list with the names of the other conferees in the DVC session can be loaded.
- The **selected conferees** list contains the names, i.e. UserIds, of the conferees who the presenter wants to invite to the session.
- The **presenter** field contains the name, i.e. UserId, of the presenter.
- The presenter can modify the **selected conferees** list by using the Add/Remove button (changes in a context sensitive way), or simply by double clicking the UserId of the user who has to be added or removed.

Start button:

When all items are defined, the session can be started using the **Start** button. If there are still some settings which are not defined, a message appears in the status bar. The status bar of the DTP workspace contains an overview of the defined settings.
When the session start fails, the presenter is informed by a message. The cause of the failure may be indicated. A non-exhaustive list of possible causes includes:
- The presenter is not subscribed to the DTP service.
- No response (i.e. a time-out has occurred).

When the session start succeeds, the DTP workspace is opened at the presenter’s terminal (see Figure 4-4).

![Figure 4-4: DTP Workspace Window](image)

Then the selected conferees are invited to join the DTP session (see Figure 4-5).

![Figure 4-5: Invitation Window](image)
The end-user can either accept or reject the invitation. He cannot change the session configuration, which is displayed only for information.

If the addition of a conferee fails, the presenter is informed by a message. A non-exhaustive list of possible causes includes:
- The invitation is rejected by the conferee.
- The conferee is not subscribed.
- The conferee is not logged in.
- The end-user is unknown.
- Terminal busy (i.e. the terminal is already involved in a DTP session).
- No response (i.e. a time-out has occurred).

If the end-user accepts the invitation but the addition fails, then this end-user will also get message indicating the reason of the failure.

When the conferee is added to the session, the DTP workspace window (without control functionality) is opened at his terminal. At all terminals the UserId of the new conferee will be added to the list with conferees. The file containing the presentation is automatically transferred to the conferee’s terminal and placed in the “C:TEMP” directory. Then PowerPoint is started, and the presentation is loaded.

**Cancel button:**
By clicking the **Cancel** button, the new session creation is aborted.

**4.2.1.2 Stop Command**
The presenter of a session can stop the complete session at any time by clicking on the session menu **Stop** command (or pressing the **Stop Session** button in the DTP workspace). If he does, a message box appears requesting an explicit confirmation. The conferees are informed of session termination by a message in the status bar. This message can also contain the cause of the session closure.
A non-exhaustive list of possible causes includes:
- Session closed by presenter.
- Network error.

After the termination of a session PowerPoint is closed at all terminals and the empty DTP workspace is displayed, so a new session can be started (see Figure 4-1).

**4.2.1.3 Leave Command**
A conferee can leave an on-going session at any time by clicking the session menu **Leave** command. (or pressing the **Leave Session** button in the DTP Workspace). Before leaving the session, a confirmation is requested. After having left a session, PowerPoint is closed and the empty DTP workspace is displayed, so a new session can be started (see Figure 4-1). All other parties are informed that a party has left the session. A message is displayed at their terminals and the corresponding UserId is removed from the list with conferees.
4.2.1.4 Exit Command
Clicking Exit will terminate the DTP service. If the end-user is participating in a session, he first has to leave/stop the ongoing session.

4.2.2 Other Services Menu
The menu Other Services distinguishes two groups of services. These are administrative services (Accounting and Address Book) and stand-alone services (Desktop Video Conference and Chat Tool).

4.2.2.1 Accounting Command
The Accounting option launches the associated service, i.e. the result is identical to double-clicking the Accounting icon in the SPOT-A program group (see [AdminServices]).

4.2.2.2 Address Book Command
The Address Book option launches the associated service, i.e. the result is identical to double-clicking the Address Book icon in the SPOT-A program group (see [AdminServices]).

4.2.2.3 Desktop Video Conference Command
The Desktop Video Conference option launches the associated service, i.e. the result is identical to double-clicking the DVC icon in the SPOT-A program group (see [DVC]). The party list of the DVC Service can be used to start a DTP session.

4.2.2.4 Chat Tool Command
The Chat Tool option launches the associated service, i.e. the result is identical to double-clicking the Chat icon in the SPOT-A program group (see [Chat]).

4.2.3 Help Menu
The ToolTips option is a check box. When this option is selected the function of a control button is displayed when the mouse pointer is on that button. The About... command displays a window with information regarding the application.
4.3 DTP Controls

The presenter of a session can control the sequence of the slides at the conferees’ terminals by using the control buttons in the workspace window:

- **Go To First** Slide button: will force all ongoing presentations to display the first slide.
- **Go To Previous** Slide button: will force all ongoing presentations to display the previous slide.
- **Go To Next** Slide button: will force all ongoing presentations to display the next slide.
- **Go To Last** Slide button: will force all ongoing presentations to display the last slide.
- **Go To** button: will force all ongoing presentations to display the slide, specified in the text box.

Note: Impossible actions, like **Go To** a slide that does not exist are handled by the Graphical User Interface, so the presenter is not disturbed by additional dialogues.

4.4 Status Bar

The status bar contains the following information:

- Cost of the session so far (in charge units) for the end-user of this terminal.
- The charging scheme of the session (“Paid by presenter” or “Shared among all parties”).
- The current time.
- The current and maximum number of parties in the session.
- The current slide number and the total number of slides.
- The latest information message received (e.g. information that a party has been added or removed).
- Some status indication (e.g. “Active”, “Creating”, ...).
5. Overview of DTP Service Features

The request by codes indicate which end-users can invoke the feature and are to be interpreted as follows:

- outside a service session:
  - su: end-user who is logged into the retailer domain and who has subscribed to the DTP service
  - all: any SPOT-A end-user
- inside a DTP service session:
  - pr: presenter
  - co: conferee (i.e. any party different from the presenter)
  - all: any party

The indication [impl] means that the request is generated implicitly, i.e. the end-user does not make the request explicitly through the GUI.

The confirm by indicates which end-users need to confirm, i.e. accept, the feature invocation.

A dash (-) indicates that the entry is not applicable.

<table>
<thead>
<tr>
<th>DTP Session Features</th>
<th>request by</th>
<th>confirm by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start a New DTP Session</td>
<td>su</td>
<td>-</td>
</tr>
<tr>
<td>Accept/Reject an Invitation to Join a DTP Session</td>
<td>su</td>
<td>-</td>
</tr>
<tr>
<td>Leave a DTP Session</td>
<td>co</td>
<td>-</td>
</tr>
<tr>
<td>Stop a DTP Session</td>
<td>pr</td>
<td>-</td>
</tr>
<tr>
<td>Select Charging Scheme at Session Creation Time</td>
<td>pr</td>
<td>-</td>
</tr>
<tr>
<td>On-line Update of Session Cost</td>
<td>all [impl]</td>
<td>-</td>
</tr>
<tr>
<td>Display of Incoming Information Messages</td>
<td>all [impl]</td>
<td>-</td>
</tr>
<tr>
<td>Display of Failure Diagnostics (detailed cause values)</td>
<td>all [impl]</td>
<td>-</td>
</tr>
</tbody>
</table>

1 An end-user can only be involved in 1 DTP session at a time.
2 Only 2 charging schemes supported: “Paid by presenter” and “Shared among all”.

<table>
<thead>
<tr>
<th>DTP Party Features</th>
<th>request by</th>
<th>confirm by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invite User to Join the Session</td>
<td>pr</td>
<td>invited user</td>
</tr>
<tr>
<td>Display Notification that a Party has joined the Session</td>
<td>all [impl]</td>
<td>-</td>
</tr>
<tr>
<td>Display Notification that a Party has left the Session</td>
<td>all [impl]</td>
<td>-</td>
</tr>
</tbody>
</table>

1 The presenter can only invite other users at session creation time.
6. Overview of Possible Future Extensions

- Introduction of more elaborate charging schemes.
- Introduction of charging related to usage of network resources.
- Automatically launching the DTP service at an end-user’s terminal when this end-user is invited to join a DTP Session.
- Conversion from charge units to normal currencies (currency will be user dependent).
APPENDIX B

SERVICE PLATFORM DESIGN

- DTP SERVICE
Service Pilot On TINA (SPOT-A)

Service Platform Design

DeskTop Presentation Components

<table>
<thead>
<tr>
<th>History</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>24/04/1997</td>
<td>Version 2.0 Revised version</td>
</tr>
<tr>
<td></td>
<td>Alcatel Telecom: E. Montes de Oca</td>
</tr>
<tr>
<td>15/09/1997</td>
<td>Version 3.0 Stand-alone DTP service</td>
</tr>
<tr>
<td></td>
<td>Swisscom: T.T. Kijftenbelt</td>
</tr>
<tr>
<td>11/11/1997</td>
<td>Version 3.1 Stand-alone DTP service, integrated file transfer</td>
</tr>
</tbody>
</table>
# Table Of Contents

1. SCOPE 1

2. DETAILED DESIGN - DTP-SPECIFIC COMPONENTS 2

2.1 General Description 2
   2.1.1 Service Adaptation 3
   2.1.2 Session Management 3

2.2 GUI Components 3
   2.2.1 General Description 3
   2.2.2 DTP_GUI Component 4
   2.2.3 DTP_GUI_STUB Component 4

2.3 TINA Components - Terminal Domain 5
   2.3.1 Computational Model Overview 5
   2.3.2 DTP_UAP Component 6

2.4 TINA Components - Network Domain 8
   2.4.1 Computational Model Overview 8
   2.4.2 DTP_USS Component 8
   2.4.3 DTP_GSS Component 10
   2.4.4 DTP Factory Component 12

3. DTP SERVICE ADAPTATION 13
Table of Figures

FIGURE 2-1: DTP SERVICE-SPECIFIC COMPONENTS - OVERVIEW  2
FIGURE 2-2: DTP SERVICE-SPECIFIC GUI COMPONENTS  3
FIGURE 2-3: DTP SERVICE-SPECIFIC COMPONENTS - TERMINAL DOMAIN  5
FIGURE 2-4: DTP_UAP INTERFACES  6
FIGURE 2-5: DTP SERVICE-SPECIFIC COMPONENTS - NETWORK DOMAIN  8
FIGURE 2-6: DTP_USS INTERFACES  9
FIGURE 2-7: DTP_GSS INTERFACES  11
FIGURE 2-8: DTP_FACTORY INTERFACE  12
FIGURE 3-1: START DTP SESSION - SERVICE SESSION CREATION  13
FIGURE 3-2: START DTP SESSION - ADD PARTIES  14
FIGURE 3-3: ACCEPT INVITATION TO JOIN SESSION  15
FIGURE 3-4: DTP_GSS INFORMATION DISTRIBUTION  15
FIGURE 3-5: LEAVE A DTP SESSION/DELETE A DTP SESSION  16
FIGURE 3-6: DELETE DTP SESSION  17
References

[ServiceOverview]  External Service Specifications - Overview
[AdminServices]  External Service Specifications - Administrative Services
[DVC]  External Service Specifications - DVC Service
[DTP]  External Service Specifications - DTP Service
[Chat]  External Service Specifications - Chat Service
[SPOverview]  Service Platform Design - Overview
[SPAdminServices]  Service Platform Design - Administrative Services Components
[SPCommon]  Service Platform Design - Service-Common Components
[SPDVC]  Service Platform Design - DVC Components
[SPDTP]  Service Platform Design - DTP Components
[SPChat]  Service Platform Design - Chat Service Components
1. SCOPE

This document presents the detailed design of the service platform components specific for the DeskTop Presentation service. The external specification of this service is detailed in [DTP]. An overview of the service platform design is presented in [SPOverview]. A detailed description of the service-common components is presented in [SPCommon].
2. Detailed Design - DTP-Specific Components

2.1 General Description

The DTP service-specific components (see Figure 2-1) implement the functionality which is required to offer the DTP service features to the end-user, and which is not provided directly by the TINA service-common components. The components can be divided into a terminal-part and a network-part. In this version of the stand-alone DTP service, the terminal code is assumed to be installed on all SPOT-A Terminals. In a future implementation, it might be possible to download the terminal code dynamically using an applet.

![Figure 2-1: DTP Service-Specific Components - Overview](image)

The TINA service-common components offer generic capabilities to invoke new service sessions.

The DTP service-specific components implement additional functions which can be categorized as Service Adaptation and Session Management. Each of those is elaborated in following sub-sections.
2.1.1 Service Adaptation

Most of the end-user service features offered by the DTP service can be realized using the TINA component interface operations. There is an one-to-one mapping between a DTP Session and a TINA service session. However, in most cases there is no direct one-to-one mapping between an end-user action in the DTP GUI and a TINA session control interface operation invocation. An example is the start of a new session. For the end-user this a simple action, but it requires a lot of extra work from the service-specific components.

The DTP service adaptation is described in more detail in section [3 DTP Service Adaptation].

2.1.2 Session Management

Session Management can be roughly decomposed into the following tasks:
- Keep track of all ongoing SPOT-A DTP sessions and their configurations.
- Pass the DTP session information peer to peer, i.e. between the DTP specific components in different end-user terminals.
- Determine the cost of each ongoing DTP Session.

2.2 GUI Components

2.2.1 General Description

Figure 2-2 highlights the GUI-related DTP service-specific components. The DTP Application GUI actually consists of two components: DTP_GUI and DTP_GUI_STUB.

![Diagram of GUI Components]

Figure 2-2: DTP Service-Specific GUI Components

This decomposition is not really necessary conceptually -i.e. from a computational viewpoint- but it is introduced for engineering reasons. The DTP_GUI is implemented in Visual Basic and can act only as a CORBA client (using the Orbix OLE integration) not as a CORBA server. The DTP_GUI_STUB acts as a CORBA server on behalf of the DTP_GUI.
2.2.2 DTP GUI Component

2.2.2.1 General Description

The DTP.GUI CO offers the end-user the ability to use the features offered by the DTP Service. The GUI is described in detail in the SPOT-A deliverable [DTP].

The DTP_GUI component translates the end-user’s actions into invocations on the DTP_UAP Access and Session REQ interfaces.

Apart from handling the graphical aspects of the DTP service, the following functionality is also provided by the DTP_GUI:

- Store the userld of the user logged in into the SPOT-A retailer domain on the current terminal.
- Check the consistency of a number of end-user’s requests, such as:
  - don’t use invalid userlds (userlds can only be provided by selecting them from a list of valid userlds)
  - don’t add the same party twice to same session
  - don’t add more users than the predefined maximum
- Disable presenter-restricted features for the conferees.

2.2.3 DTP_GUI_STUB Component

2.2.3.1 General Description

The DTP_GUI_STUB is an intermediate object which eases the communication between the DTP_UAP and the DTP_GUI. It acts as the gateway from the CORBA world towards the non-CORBA world. When the DTP_UAP wants to invoke an operation on the DTP_GUI, it invokes this operation on the DTP_GUI_STUB. It is the responsibility of the DTP_GUI_STUB to forward it to the DTP_GUI and correlate the corresponding answer.

2.2.3.2 Interface Description

The DTP_GUI_STUB offers an IND(ication) and an INFO interface towards the DTP_UAP.

2.2.3.2.1 IND(ication) Interface

The INDication interface offers an operation which requires an explicit answer.

\[
\text{join_in_dtp_session} \\
\text{Request from a presenter to join an existing session.} \\
in : \text{dtpGUISessionInfo} \\
in : \text{DTP_UAP Session REQ interface reference}
\]

2.2.3.2.2 INFO Interface

The INFO interface offers operations which don’t require any confirmation.

\[
\text{add_dtp_party_info} \\
\text{Indicate that a party has been added to the session.} \\
in : \text{userld} \\
\text{delete_dtp_party_info} \\
\text{Indicate that a party has been deleted from the session.} \\
in : \text{userld}
\]
2.3 TINA Components - Terminal Domain

2.3.1 Computational Model Overview

Figure 2-3 highlights the DTP service-specific components which reside in the terminal domain.

The only component identified is the DTP_UAP. The DTP_UAP interacts with DTP service-specific components in the network domain for the management of the DTP session. It interacts with the GSEP and UAPAccessClient for the DTP service adaptation (this is elaborated more in detail in section 3 DTP Service Adaptation). It uses the Configuration interfaces of the DCOs to manage local devices (e.g. the ATM board). Finally, it also interacts with the DTP_GUI and DTP_GUI_STUB for the interaction with the end-user.
2.3.2 DTP_UAP Component

2.3.2.1 General Description

The DTP_UAP implements the DTP service-specific code which resides in the terminal and which is not related to the GUI. It can be decomposed into two main parts:

• The first one is session independent and is launched before any DTP session is created. This part exports the Factory and Access interfaces.

• The second part will be launched only when a DTP session is launched. It exports session-specific REQ, INFO, IND and Ret_INFO interfaces.

2.3.2.2 Interface Description

Figure 2-4 presents an overview of the DTP_UAP interfaces. The Factory and Access interfaces are created prior to the creation of any DTP session. The REQ, Ret_INFO, INFO, and IND(ication) interfaces are dynamically created and destructed during the lifetime of a DTP session.

![DTP_UAP Interfaces](image)

Figure 2-4: DTP_UAP Interfaces

2.3.2.2.1 Access Interface

The Access interface enables the DTP_GUI to start a new DTP session, and to obtain a list of users logged in into the SPOT-A retailer domain.

Operations offered to the DTP_GUI:

**request_dtp_service**

Session initiation by the chairman. UserIdList is passed as inout parameter, because the operation will return the list of users successfully added to the session.

- **inout**: dtpSessionName
- **in**: userId (myself)
- **in**: dtpSessionDescription
- **inout**: UserIdList (list of users to add to the session)
- **out**: DTP_UAP Session REQ Itf Reference

**list_of_users**

This operation gets a list of users logged in into the SPOT-A retailer domain.

- **out**: userList

2.3.2.2.2 Factory Interface

The Factory interface enables the UAP Access Client to create a new DTP_UAP service session for both “incoming” (join_in_session) and “outgoing”
(createUAPSessionClient) DTP sessions. Further it provides an operation to delete a DTP_UAP service session.

**Operations offered to the UAP Access Client.**

**createUAPSessionClient**
Create new INFO, IND, Ret_INFO and REQ interfaces.

- **in** : serviceld
- **in** : sessionld
- **in** : GSEP REQ, ReqAPI and LIST interface references
- **out** : UAPSessionClient IND and INFO interface references

**join_in_session**
Invitation to join a service session (and if accepted, create new INFO, IND, Ret_INFO and REQ interfaces).

- **in** : userld and partyld of invited user
- **in** : userld and partyld of inviting user
- **in** : sessionld
- **in** : sessionDescription
- **in** : GSEP ReqAPI, REQ and LIST interface references
- **out** : UAPSessionClient IND and INFO interface references

**deleteUAPSessionClient**
Delete INFO, IND, Ret_INFO and REQ interfaces for a certain DTP Session.

- **in** : sessionld

**2.3.2.2.3 Session REQ(uest) Interface**

Allows the DTP_GUI to issue requests to modify the DTP session configuration.

**wait_dtp_join_in_session**
When a party has accepted an invitation to join in a session (issued by the chairman), its DTP_GUI should invoke this operation. It returns at the moment the party has actually been added.

- **out** : dtpGUISessionInfo

**delete_dtp_session**
The presenter can close the whole session. All parties are removed from session.

**exit_dtp_session**
A conferee can exit a session.

**broadcast_goto_slide**
The presenter can broadcast a new slide number to all conferees.

- **in** : slidenumber

**2.3.2.2.4 Session INFO Interface**

This interface is identical to the GSEP Session Control INFO interface. The operations in the INFO interface return immediately without passing any information to the GUI, because this information is received through the Ret_INFO interface in a DTP service-specific format. The only exception is the delete_session_info operation (see DTP service adaptation scenario’s).

**2.3.2.2.5 Session IND(ication) Interface**

This interface is identical to the GSEP Session Control Indication interface. The operations in the IND interface return immediately without requesting assistance from the end-user. The only exception is the join_in_session_ind (see DTP service adaptation scenario’s).
2.3.2.2.6 Session Ret_INFO Interface

The Ret_INFO interface enables the DTP_USS to indicate a DTP session modification (e.g. cost updated). The operations on this interface are identical to the DTP_GUI_STUB INFO interface operations. There is one additional operation, needed for the file transfer of the presentation:

**transfer_file**

The parameters contain the name of the presentation file and the compressed presentation file in string format.

```plaintext
in : filename
in : file
```

2.4 TINA Components - Network Domain

2.4.1 Computational Model Overview

Figure 2-5 highlights the DTP service-specific TINA Components in the network domain. The DTP User Service Segment (DTP_USS) and the DTP Global Service Segment (DTP_GSS) COs map onto the TINA Service Segment. The DTP_Factory is responsible for managing the DTP_USS Session interface objects.

![Figure 2-5: DTP Service-Specific Components - Network Domain](image)

2.4.2 DTP_USS Component

2.4.2.1 General Description

The DTP_USS enables the interaction between the DTP_UAP and the DTP_GSS. It has the following functions:

- Enable the DTP_UAP to request a modification of the DTP session management information.
- Inform the DTP_UAP of a modification of the DTP session (via the DTP_UAP Session Ret_INFO interface). This includes the update of the billing information for the ongoing session.
- Enable the DTP_UAP to retrieve information about the DTP session.
2.4.2.2 Interface Description

The DTP_USS (see Figure 2-6) offers DTP Session REQ, LIST and INFO interfaces which are created per DTP session. The REQ and LIST interfaces are used by the DTP_UAP, while the INFO interface is used by the DTP_GSS. The sessionId and userId are stored by the DTP_USS Session and are used to communicate with the DTP_GSS. The Factory interface is offered to the DTP_Factory to manage the life-cycle of the DTP USS Session interfaces.

![Figure 2-6: DTP_USS Interfaces](image)

2.4.2.2.1 Session REQuest Interface

This interface is offered towards the DTP_UAP.

**wait_dtp_join_in_session**

A new user has to wait, till he's actually added to the session.

```
wait_dtp_join_in_session
```

```
out  : dtpSessionInfo
```

**add_dtp_session**

Report that a new DTP Session is created. This request is forwarded to the DTP_GSS which will create a new entry in the session list and returns a unique SessionName for this new session. The DTP_USS will store the SessionId.

```
inout  : dtpSessionName

in  : dtpSessionDescription

in  : userId & partyId (presenter)

in  : ownershipId (of session ownership relation)
```

**delete_dtp_session**

Report that a DTP Session is deleted. This request is forwarded to the DTP_GSS which will remove the entry in the session list.

```
add_dtp_party
```

```
in  : partyId

in  : userId
```

**delete_dtp_party**

Report that a party is deleted from the session.

```
broadcast_goto_slide
```

```
in  : slidenumber
```
**set_dtp_status**
Set the semaphore of a particular session to either **locked** to prevent simultaneous access to the session info or **unlocked** to free the semaphore. If set_status is used to lock a session, the function returns only when the semaphore of the session has been successfully locked.

```
in  :  dtpStatus (locked or unlocked)
```

**broadcast_file**
The presenter broadcasts the compressed presentation file as a string to all conferees in the session.

```
in  :  filename
in  :  file
```

### 2.4.2.2 Session LIST Interface
This interface allows the DTP_UAP to retrieve information related to the DTP Session which is stored by the DTP_GSS.

**list_dtp_session**
Retrieve the session information.

```
out  :  dtpSessionInfo
```

### 2.4.2.3 Session INFO Interface
This interface is the same as the DTP_UAP Session Ret_INFO (see section [2.3.2.2.6 Session Ret_INFO Interface]).

### 2.4.2.4 Fac(tory) Interface
This interface is offered towards the DTP_Factory

**create_dtp_uss**
Create new DTP_USS Session interfaces.

```
in  :  userId
in  :  sessionId
in  :  DTP_UAP Ret_INFO interface reference
out  :  DTP_USS REQ, LIST and INFO interface references
```

**delete_dtp_uss**
Delete DTP_USS Session interfaces.

```
in  :  userId
in  :  sessionId
```

### 2.4.3 DTP_GSS Component

#### 2.4.3.1 General Description
The DTP_GSS provides centralized management of the active DTP sessions. The DTP_GSS keeps track of the following information per DTP session (the * indicates that the information is also relevant to the TINA service-common components)

- **SessionId**: the identifier of the session used by the TINA service-common part of the architecture.
- **DTP Session Name**: this name will be given by the DTP_GSS at start-up of the DTP Session.
- **DTP Session Description**:
  - Charging Scheme.
  - Filename of the presentation
• **DTP Parties List**: for each party the following information is stored
  - **UserId**: unique identifier of the party object in the service session graph.
  - **Role**: presenter or conferee
  - **PartyId**: unique identifier of the party object in the service session graph.
• **Status**: a semaphore that indicates if the data related to this session can be modified or that it is already being accessed by another user. This prevents the simultaneous access of this data. The Status can be either *locked* or *unlocked*.
• **OwnershipId**: unique identifier within the service session graph of the session relationship which expresses the ownership of the session.
• **On-line Charge Counter**: The DTP_GSS computes the on-line billing information for every DTP session. When a DTP session is started, it retrieves the currently valid tariffing information from the ACC_GSS (see [SPAdminServices]). One tick has a fixed time but a variable cost. The DTP_GSS calculates the cost per tick based on the base cost per charge unit, the number of parties in the session and the time of the day.

### 2.4.3.2 Interface Description

The DTP_GSS offers three types of interfaces (see Figure 2-7). The REQ and LIST interfaces are offered towards the DTP_USS, the Access interface is offered towards the DTP_Factory.

![Figure 2-7: DTP_GSS Interfaces](image)

#### 2.4.3.2.1 REQ(uest) Interface

The REQ interface allows the DTP_USS to issue requests to manipulate the session information list. This interface is similar to the DTP_USS REQ interface except that for every operation, the sessionId and userId of the invoking user has to be specified as an in parameter.

#### 2.4.3.2.2 LIST Interface

The LIST interface allows the DTP_USS to extract data from the session information list. This interface is similar to the DTP_USS LIST interface except that for every operation, the sessionId and userId of the invoking user has to be specified as in parameter.

#### 2.4.3.2.3 Access Interface

The Access interface allows the DTP_Factory to report the creation of a new DTP_USS Session and pass on the relevant Session INFO interface.
add_dtp_uss
Report the creation of a new DTP_USS Session.
in : userId
in : sessionId
in : DTP_USS Session INFO interface reference

delete_dtp_uss
Report that the DTP_USS Session has been deleted.
in : userId
in : sessionId

2.4.4 DTP Factory Component

2.4.4.1 General Description
The DTP_FACTORY CO enables the DTP_UAP to create and delete DTP_USS sessions. It passes the DTP_USS Session INFO interface reference to the DTP_GSS.

2.4.4.2 Interface Description
The DTP_FACTORY offers the Usage interface towards the DTP_UAP (see Figure 2-8).

Figure 2-8: DTP_FACTORY Interface

2.4.4.2.1 Usage Interface
The Usage interface allows the DTP_UAP to order the creation of a DTP_USS Session and pass on the relevant Session REQ and LIST interfaces. Further the DTP_UAP can order the removal of a DTP_USS Session.

create_dtp_uss
Create a new DTP_USS Session.
in : userId
in : sessionId
in : DTP_UAP Session Ret_INFO interface reference
out : DTP_USS Session REQ and LIST interface references

delete_dtp_uss
Delete a DTP_USS Session.
in : userId
in : sessionId
3. DTP Service Adaptation

Figure 3-1 illustrates the creation of the TINA service session without any parties involved, except the presenter.

Note the following issues:

- The **request_dtp_service** invocation contains two inout parameters: **dtpSessionName** and **userIdList**. The initiator of the DTP session can propose a **dtpSessionName**, but the final decision on the name will be made by the DTP_GSS (which can check the uniqueness of the name). When **request_dtp_service** returns, the **userIdList** will contain the UserIds of all users who have been added successfully to the DTP session.

- Before forwarding a **request_service** to the UAPAccessClient, the DTP_UAP can optionally issue a **list_of_services** to check whether **myUserId** is actually subscribed to the DTP service.

- The user who creates the DTP session assumes ownership of the TINA service session graph.

- When the TINA Session has been created, the DTP_UAP requests for the creation of a new DTP_USS session and informs the DTP_USS/DTP_GSS about the details of the newly created DTP session. The Session Status is initialized to locked.
Figure 3-2 presents the scenario when the `userIdList` contains two users that need to be added to the DTP session.

- The `add_party` will result in an incoming `join_in_session` at the terminal where the invited user is logged in (see also Figure 3-3).
- When the user accepts the invitation, the DTP_USS/DTP_GSS is informed that a party has been added to the DTP session.
- When the second conferee `-u2UserID-` has also accepted the invitation to join the DTP session, the DTP_USS/DTP_GSS is again informed.
- When both conferees have been added, the DTP status is unlocked and the `dtpSessionName` (determined by DTP_GSS) and `userIdList` (containing both `u1UserID` and `u2UserID`) are returned as out parameters of the `request_dtp_service` invocation.

Figure 3-3 presents the scenario where a user is invited to join a DTP session. As a result of an `add_party` issued by the presenter (see also Figure 3-2), the invited party receives a `join_in_session` invitation.

- When the DTP_UAP receives the invitation, it must first create a DTP_USS session and retrieve the complete description of the DTP session from the DTP_USS/DTP_GSS.
- When the invited user accepts the invitation, the `join_in_dtp_session` returns (in case of rejection, an exception is thrown by the DTP_GUI_STUB).
- As a result of accepting the invitation, the `add_party` at the presenter's side returns and the DTP_USS/DTP_GSS is informed.
- In order to know the final outcome of the session invitation, the DTP_GUI should therefore invoke a `wait_dtp_join_in_session`, which is forwarded towards the DTP_USS/DTP_GSS. This function will only return when the presenter has unlocked the DTP status.
When the DTP_GSS is informed about the addition/deletion of a party, it automatically distributes this information to all parties involved in the DTP session. It sends _info messages to all parties, except for the party who is reporting the modification. The add_party_info is also not sent to the party that has just been added or deleted. Notice that the invocations on the DTP_GSS Request interface always have to be labelled by the sessionId and the userId of the party who is doing the reporting. Also the presentation file and a command, indicating the presenter has gone to another slide, are distributed in this way. This is illustrated Figure 3-4.
The two scenarios in Figure 3-5 illustrate a conferee who is leaving a session on its own request and the deletion of the complete session by the presenter.

Figure 3-5: Leave a DTP Session/Delete a DTP Session

When a conferee decides to leave a DTP Session:
- His DTP_GUI invokes an `exit_dtp_session`.
- The DTP_UAP forwards this as an `exit_session` to the GSEP.
- The DTP_UAP issues a `delete_service_session` to the UAP Access Client to initiate the cleanup of the relevant session interface objects in the service-common components.
- The DTP_UAP informs the DTP_USS/DTP_GSS that it is deleted from the DTP session and requests the DTP_Factory to delete the DTP_USS session. The `delete_dtp_party` is forwarded to all other parties (see Figure 3-4).
- The DTP_UAP destructs the relevant DTP_UAP session interface objects.

A presenter can't leave a session, but he can delete the complete session:
- His DTP_GUI invokes a `delete_dtp_session`.
- The DTP_UAP forwards this as a `delete_session` to the GSEP.
- The DTP_UAP issues a `delete_service_session` to the UAP Access Client to initiate the cleanup of the relevant session interface objects in the service-common components.
- The DTP_UAP informs the DTP_USS/DTP_GSS that the session is deleted and requests the DTP_Factory to delete the DTP_USS session.
- The DTP_UAP destructs the relevant DTP_UAP session interface objects.
All conferees are informed of the session deletion by a `delete_session_info` invoked by the GSEP on the DTP_UAP Session INFO interface (see Figure 3-6).

Figure 3-6: Delete DTP Session