Intelligent Network (IN) Service Creation

Definition

Using the intelligent network (IN) approach for supporting telecommunications services enables the use of productivity-enhancing techniques for the creation of new services. Rather than creating new services by cutting fresh software code, modules of service-independent logic can be arranged using graphical user interfaces. As a result, advanced telecommunications services, which are independent of the central switch, can be created more quickly than ever.

Overview

This tutorial will explain improvements to IN techniques for telecommunications service creation. First, the nature of the telecommunications industry and how this nature influences the selection of a toolset to support service creation is discussed. Next, the tutorial explores the service creation lifecycle and methods for supporting rapid service implementation. Then, model architecture and the building blocks that are used to construct services are examined, along with a discussion of a complementary execution and off-line testing environment. Finally, the tutorial explores how the supporting technology will evolve in the future.

Topics

1. Issues in Delivering Rapid Service Creation
2. An Approach for Rapidity
3. A Model Architecture for Rapid Service Creation
4. Building Blocks
5. Supporting Infrastructure
6. Service Data
7. Rapid Testing
8. Future Evolution

1. Issues in Delivering Rapid Service Creation

What Is a Service?

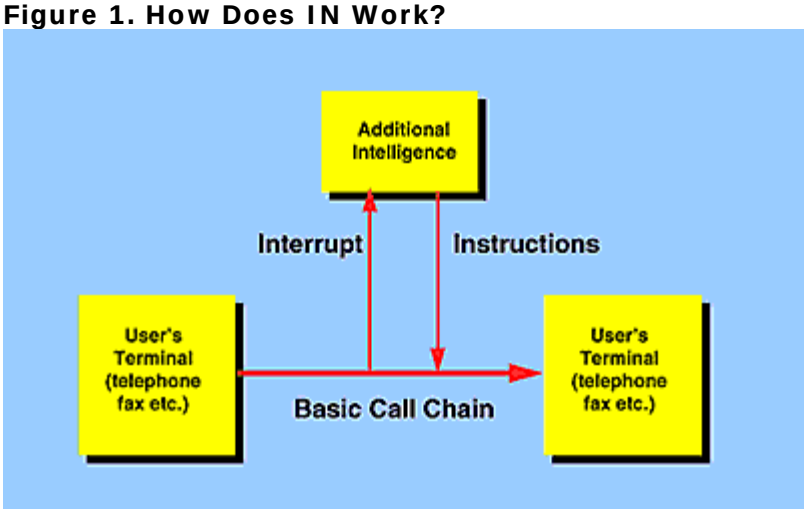
This is a question that has many different answers, depending on one's perspective.

- From a user perspective, a service is a black box with external stimuli triggering the appropriate responses. This is an appropriate view to take for capturing requirements, but it introduces difficulties when the implementation of a service is attempted.

- An alternative viewpoint may be termed a "platform perspective." This treats the service as a white box, exposing its internal operations.

For the purposes of developing a service creation environment, it is preferable to take a platform-centered view. By choosing this approach one ensures that the infrastructure and service-creation tools will support a rich set of capabilities that are substantially independent of network capabilities and network users.

Figure 1. How Does IN Work?



This leads to the following definition:

A service is an application on the IN platform that provides a defined set of functions that interact with the network (and therefore users) and a set of service data.

The set of services to be supported for any operator is not fixed over time. Indeed, at the outset, the eventual service set may not be known at all. In addition, new services are always being devised to add to the existing set. These new services may require functions that do not yet exist in the IN node but need to be introduced specifically for these services. Because of these effects, adding a service is understood to mean that the network's functionality is increased, rather than changed by means of altering data. Having defined "service," "service creation" can now be defined.

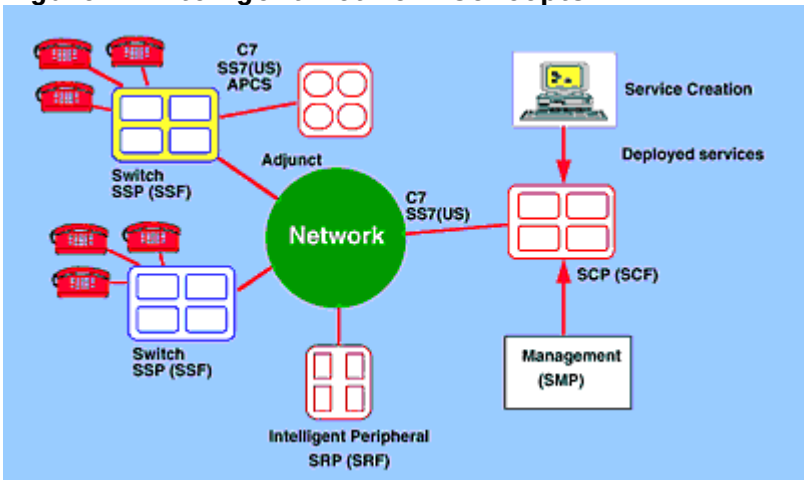
Service creation consists of a number of activities. The first is the creation of the service logic to perform the functions required. As well as defining the functionality of the service, the data model—which defines the type and scope of data used by a service—also needs to be specified. These two activities are carried out together by the service creator. Finally, there is a need to produce data management screens to manage the service and subscribers to that service. This activity follows from defining the service logic, because for any given service there may be several management screens required.

This leads to the requirement that a service creation system must be very flexible and able to support the creation of many types of applications and new forms of interaction and data as they are added to the IN network.

2. An Approach for Rapidity

With the earlier definition of a service in mind, the philosophy behind the choice of a service creation method can be explored. The primary need is to allow service creators to express the service in an abstract way. The most natural method is to employ a pictorial representation using graphical techniques. The technique chosen was to provide a set of iconic images that can be linked to form a pictorial representation of the service. Each image represents a building block. The types of images were chosen to express what a service does rather than how it operates in detail. This expression is encouraged by the selection of verbs or actions as the building blocks. The concept of building blocks removes the need for service creators to have a good grasp of the platform computer programming language, such as C and C++. Another related benefit is the reduction of programming errors. Using a high level of abstraction and integrating all the tools into a single environment encourages services to be developed using iterative methods, rather than a rigid analyze-design-code-test process. The iterative methods allow external behavior to be decided with the users early on and the detailed work to be done once the usability aspects have been finalized.

Figure 2. Intelligent Network Concepts

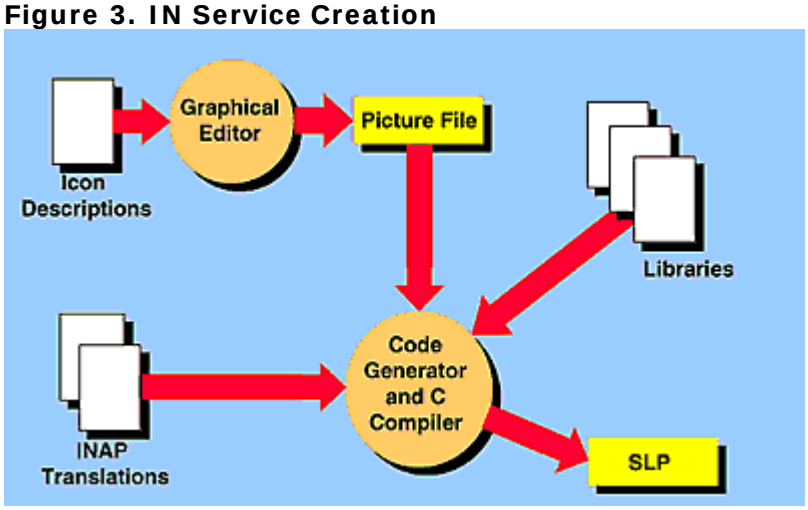


Switch independence issues must also be addressed. This can be achieved by removing the network-specific information as far as possible from the service creator's domain and placing it at a lower level that is amenable to automatic processing. Finally, it is critical to allow the system to be extended with the minimum of impact to the software tools. To this end, formally defined languages are used frequently in the internal operation of the tools. To varying degrees, commercially available toolsets attempt to satisfy these requirements, reflecting a universal agreement on the advantages of this philosophy.

3. A Model Architecture for Rapid Service Creation

Figure 3 shows the processes involved in the creation of a service. This is only a part of an SCE product and is intended to show how the loosely coupled architecture enables the goals of service creation to be realized.

Figure 3. IN Service Creation



First the building blocks are expressed in a formal language and can be quickly extended. As can be seen, the graphical editor is not aware of any details concerning the target application, so adding new capability means that this complex component is unaffected. The user uses the graphical editor to add building blocks to the picture and to link them together. Each building block has a set of attributes that may be changed using an associated properties box.

Once the picture is complete, it is saved to an intermediate picture file and then processed by the code generator. This uses the translation information, which is expressed as a formal language, and generates C code, conforming to the application-programming interface (API) supported by the IN platform. Finally, a set of platform- and network-specific libraries is linked to create the final service logic program. The complete service logic program is then ready for off-line testing and eventual deployment to an IN node. The translations can be changed easily to support new INAP variants and to add new network capabilities. Service logic programs may be separate UNIX processes and, therefore, adding new functionality to a live IN node does not impact existing services. This type of architecture lends itself to extension, for instance, to process the picture file in order to produce executables for other run-time infrastructures.

4. Building Blocks

Following the platform-centered view of services, the set of building blocks provides a set of intermediate to low-level operations. Consequently, this level of abstraction is pitched high enough to allow service ideas to be trialed as well as to provide enough power for the creation of robust, network-ready services. A central theme is that adding new building blocks, while simple, will not be necessary for the vast majority of cases.

Building blocks are loosely grouped into families:

- **logic control**—These building blocks are used to express the behavior of the service. This allows the creator to build control structures and manipulate the per-session data variables.

- **call control**—These enable the interface to the network by providing operations such as connect call, manipulating speech legs, and requesting that billing and charging information be generated.

- **resource control**—This family allows announcements to be played, digits to be collected, and other IP components to be manipulated.

- **data operations**—An architecture may support many types of service-related data, and these building blocks allow a service to gain access to this data. Data types include time- and data-related routing, number translation, call distribution, as well as a flexible, user-defined set of tables.

- **picture control**—These building blocks allow a service developer to group sets of building blocks together and create reusable macros that can be utilized in other services.

Some situations, of course, need the addition of a building block typically to support new network interactions or new data requirements. In these cases, new building blocks can be designed and implemented in a short time frame.

5. Supporting Infrastructure

As described earlier, the code generator is responsible for creating the service logic, which enables deployment to the run-time system. This module looks in more detail at the run-time platform.

Many IN platforms use the service logic execution environment (SLEE) concept, which involves employing a set of portable APIs and a supporting infrastructure. The infrastructure supports the deployment activities, allowing external management of the platform by using a set of managed objects. Using these managed objects, services can be added, activated, and monitored without interrupting the existing services running on that node.

Up until now, the terms "service" and "service logic program" (SLP) have been used interchangeably. In practice, a service is realized using one or more SLPs, with each SLP performing a specific set of functions. For instance, a personal number service may require two distinct functions:

- one to handle normal calls to a personal number (PN) subscriber, which routes a caller to the subscriber's current location

- a second to allow subscribers to change their current network destination address and perform other management tasks

This service can be implemented as two separate SLPs that are deployed together to form the service. The selection of the correct SLP is undertaken on the SLEE by a switch interface component using key service information provided by the SSP.

The principle vehicle for realizing service logic programs is a generic, multi-session state machine that can be set up to support any given service. This state machine can be implemented as a set of C functions and libraries that are compiled into a UNIX process. For the majority of cases, this gives a good balance between performance and resource utilization. For other applications, however, where an operator needs many small services, an alternative is to employ a service script interpreter. In this case, an alternative code generator is employed to create the script. The choice of running any given service as an SLP or as a script is made at service-deployment time. The service creator need not be aware of any differences between the two mechanisms.

6. Service Data

Most services require access to a managed database during their execution. This database is used to hold per-service information and per-subscriber information. In the personal number example, the database would hold the current network destination address of the subscriber and would be updated when requested by that subscriber.

An IN platform may define a set of managed objects that hold the service data. These are called service data tables or SDTs. These SDTs can be accessed in real time by the service logic programs and by data management tools located on a management system.

SDTs provide a layer of decoupling between SLPs and the data. This is important, as it gives a high level of data-location transparency and data-representation transparency. Both of these attributes allow systems to capitalize on changes in database technology without impacting the rest of the system.

SDTs can be defined as either private to a service (per-service data) or as shared by all services (global data). In addition, SDTs can also utilize a subscriber identity key, which allows common data to be shared by all subscribers without compromising the privacy of individual subscribers' own data.

SDTs and their behavior are defined during the service creation process, as is the setting up of template data for subscribers. Once the data model has been defined, management screens can also be developed to allow operators to perform service provisioning and subscription management tasks.

7. Rapid Testing

Traditional testing strategies for telecommunications products have relied heavily on expensive models with all of the associated problems of scheduling access and build configuration control. For many operators, of course, access to a model is impossible; therefore an alternative was necessary.

To address his situation, some suppliers have developed an off-line simulator based on SCP software that requires only standard workstations. This makes it feasible to give all service creators a complete creation and test environment on their desks.

The tester is provided with a set of graphical tools including simulated telephones (with tones and announcements), an SSP simulator, IPs, and billing analysis tools. This testing ensures that the service provided meets user standards. The fact that the off-line simulator uses the same software as the SCP or Adjunct further increases the user's confidence that the service will function correctly once it is deployed.

In practice, off-line testing covers all of the functional tests a service requires, relegating the use of models where available to final proofing. For many operators, a service is taken directly from the off-line test environment to full on-line trials.

The fact that service creators can rapidly create and test a service without leaving their desks further encourages iterative service development.

8. Future Evolution

Mobility

The use of the services provided by IN is becoming ubiquitous. Telecommunications subscribers are coming to take these services for granted. If subscribers move to a new network (which may be a mobile network), they miss services on which they may have grown to depend. The commercial benefit to operators of offering IN services has earned a good reputation. This has lead mobile operators to want to add IN functionality to their networks in order to be able to offer the same type of services and accrue the same commercial benefits as the fixed network operators. Operators are now providing seamless service offerings across fixed and mobile networks.

Broadband

The well-documented Internet phenomenon represents a great demand for multimedia information on-demand. There are international initiatives to deliver this class of information anywhere, anytime. This will result in the need for intelligent network technology to support much more richly featured and complex services.

Telecommunications Information Networking Architecture (TINA)

TINA is a concept that endeavors to employ the next generation of software technology to provide telecommunications services and improve the benefits delivered.

The TINA Consortium (TINA–C) is a research collaboration involving major players from the telecommunications and computing industries to reach a consensus. The technology exploited includes object orientation and distributed processing, which the proponents of these standards claim will raise the benchmark by which standards are set.

Self-Test

1. In this context, service is _____.

    a. an application on the IN platform that provides a defined set of functions that interact with the network (and therefore users) and a set of service data

    b. a black box, with external stimuli triggering appropriate responses

    c. an option offered to the user that exceeds the options provided by the traditional black box

    d. a fixed option offered by the service provider to an operator

    e. none of the above

2. Switch independence issues can be resolved by _____.

    a. placing network-specific information in the service creator's domain

    b. removing network-specific information from the service creator's domain

c. placing network-specific information at a level amenable to automatic processing

d. both a and b

e. both b and c

3. In rapid service creation _____.

a. the SLP precedes the picture file

b. the graphical editor aids in INAP translations

c. libraries contribute to the final code generation

d. none of the above

4. In service creation, the _____ is not aware of any details concerning the target application.

a. SLP

b. INAP translators

c. graphical editor

d. icon describer

5. Families of building blocks, such as _____, enable a set of intermediate to low-level operations.

a. logic control, call control, and resource control

b. picture control, call control, and data control

c. resource reserve, data operations, and logical infrastructure

d. all of the above

6. Most IN platforms do not use the service logic execution environment (SLEE) concept.

a. true

b. false

7. Service data tables (SDTs) are a set of managed objects that hold the service data in an IN platform

    a. true

    b. false

8. DTM is based on all but which of the following principles?

    a. true

    b. false

9. Telecommunications information networking architecture (TINA) is a concept that endeavors to employ the next generation of software technology to provide telecommunications services and improve the benefits delivered.

    a. true

    b. false

10. With INs, there is no need to program in C/C++.

    a. true

    b. false

Correct Answers

1. In this context, service is _____.

    a. an application on the IN platform that provides a defined set of functions that interact with the network (and therefore users) and a set of service data

    b. a black box, with external stimuli triggering appropriate responses

    c. an option offered to the user that exceeds the options provided by the traditional black box

    d. a fixed option offered by the service provider to an operator

    e. none of the above

    See Topic 1.

2. Switch independence issues can be resolved by _____.

    a. placing network-specific information in the service creator's domain

    b. removing network-specific information from the service creator's domain

    c. placing network-specific information at a level amenable to automatic processing

    d. both a and b

    e. both b and c

    See Topic 2.

3. In rapid service creation _____.

    a. the SLP precedes the picture file

    b. the graphical editor aids in INAP translations

    c. libraries contribute to the final code generation

    d. none of the above

    See Topic 3.

4. In service creation, the _____ is not aware of any details concerning the target application.

    a. SLP

    b. INAP translators

    c. graphical editor

    d. icon describer

    See Topic 3.

5. Families of building blocks, such as _____, enable a set of intermediate to low-level operations.

    a. logic control, call control, and resource control

    b. picture control, call control, and data control

c. resource reserve, data operations, and logical infrastructure

d. all of the above

See Topic 4.

6. Most IN platforms do not use the service logic execution environment (SLEE) concept.

a. true

b. false

See Topic 5.

7. Service data tables (SDTs) are a set of managed objects that hold the service data in an IN platform.

a. true

b. false

See Topic 6

8. DTM is based on all but which of the following principles?

a. true

b. false

See Topic 7.

9. Telecommunications information networking architecture (TINA) is a concept that endeavors to employ the next generation of software technology to provide telecommunications services and improve the benefits delivered.

a. true

b. false

See Topic 8.

10. With INs, there is no need to program in C/C++.

a. true

b. false

See Topic 2.

Glossary

API
application programming interface

CAMEL
customized applications for mobile networks enhanced logic

CSE
CAMEL service environment

ETSI
European Telecommunications Standards Institute

GSM
global system for mobile communications

IN
intelligent network

IP
intelligent peripheral

ITU
International Telecommunications Union

MSC
mobile switching center

PN
personal number

SCP
service control point

SDP
service data point

SDT
service data table

SIBB
service independent building block

SLEE
service logic execution environment

SRF
specifically routed frame

SSF
service switching function

SSP
service switching point

TINA
telecommunications information networking architecture