INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

FEATURE INTERACTIONS DETECTION IN INTELLIGENT NETWORKS

Azimeh Sefidcon

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

December 1999

ABSTRACT

Feature Interactions Detection In Intelligent Networks

Azimeh Sefidcon

Intelligent Networks (IN) have been introduced for rapid development and deployment of new services. However, these new services may interact with old ones in a negative and unexpected manner. This is known as feature interaction (FI) problem. In this thesis, a new pragmatic approach for feature interaction detection is proposed. A pragmatic method for FI detection is based on the gained experience from the study of the known FIs. In this approach, the causes of FIs given in Bellcore and European benchmarks serve as the starting point. The sufficient information for feature description is derived from these causes. This information is modelled using an object-oriented template. Features are described in terms of necessary resources and actions. Feature participants call model, their triggering constraints and operations describe the behaviour of the feature. The detection method takes these models as input and checks for interactions between features. The method consists of three parts: filtering algorithm, feature instantiation and detection algorithm. Using the filtering algorithm, all the possible interaction-prone scenarios are produced and the actual features are instantiated using the list of topologically different call combinations. The detection algorithm is run on these actual participants of the features in order to detect potential interactions. The approach has been implemented as a tool and applied successfully to the existing feature interaction benchmarks.

Acknowledgement

First of all, I would like to express my deepest respect and gratitude to my supervisor Dr. Ferhat Khendek, without whose expert guidance and encouragement the quality and the extent of this research would not have been possible. This work was partly supported by Prof. Khendek's NSERC research grant.

I also wish to express my sincere thanks to Dr. Al. Khalili for his guidance and great advice, which have led me through the entire program.

Finally, I would like to acknowledge the friendship, support and encouragement I have received from all my friends at Concordia University.

To my daughter Ghazal

Table of Contents

List of Figures

List of Tables

List of Acronyms

AC	Answer Call
ACB	Automatic Call Back
AD	Administrative domain
AJ	Adjunct
ANC	Area Number Calling
ARC	Automatic Recall
BCP	Basic Call Process
BCSM	Basic Call State Model
CCAF	Call Control Agent Function
CCF	Call Control Function
CCITT	International Telephone and Telegraph Consultative Committee
CCSN	Common Channel Signalling Network
CF	Call Forwarding
CND	Calling Number Delivery
CPE	Customer Premise Equipment
CW	Call Waiting
DFP	Distributed Functional Plane
DP	Detection Point
DSL	Distributed Service Logic
ETSI	European Telecommunication Standards Institute
FE	Functional Entities
FEA	Functional Entity Actions
FI	Feature Interaction
GFP	Global Functional Plane
GSL	Global Service Logic
IN	Intelligent Network
INCM	Intelligent Network Conceptual Model

IP	Intelligent Peripheral
ISDN	Integrated Service Digital Network
ITU-T	International Telecommunication Union Telecommunication sector
LDC	Long Distance Calls
LEC	Local Exchange Carrier
MRC	Message Rate Charge
NAP	Network Access Point
OCS	Originating Call Screening
OS	Operator Service
OSS	Operator Service System
PE	Physical Entities
PIC	Point In Call
PLC	Programmable Logic Control
POI	Point of Initiation
POR	Point of Return
PP	Physical plane
PSTN	Public Switched Telephone Network
RDBMS	Relational Database Management System
SCF	Service Control Function
SCEF	Service Creation Environment Function
SCE	Service Creation Environment
SCP	Service Control Points
SDF	Service Data Function
SF	Service Features
SIB	Service Independent Building blocks
SIHP	Service Interaction Handling Process
SMF	Service Management Function
SMAF	Service Management Access Function
SMS	Service Management System
SP	Service Plane

SPC	Stored Program Control
SRF	Specialised resource Function
SS7	Signalling System number 7
SSF	Service Switching Function
SSP	Service Switching Points
STP	Signalling Transfer Point
TCS	Terminating Call Screening
TWC	Three Way Calling
UN	Unlisted Number

Chapter 1

INTRODUCTION

1.1 EVOLUTION OF TELEPHONY SYSTEMS

The goal of communications is the transmission and sharing of information in an understandable manner. The communication industry is in a transition phase with many developments bringing new opportunities and challenges for both network operators and equipment suppliers. These developments include changes in market regulation, global companies, demands of subscribers, standardisation, mobility, and network diversity.

In the early days of telephony, to connect telephone lines, operators manually controlled the switching devices shown in Figure 1.1. Later, Programmable Logic Control (PLC) [1] units replaced these devices.

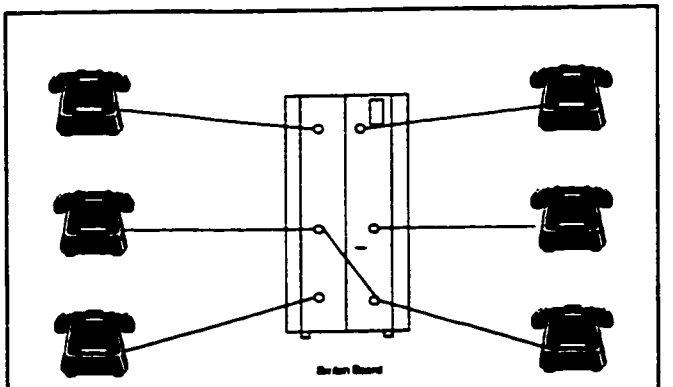


Figure 1.1: Early Switchboard

The first automatic communication network was the Public Switched Telephone Network (PSTN). The basic service offered by the PSTN is the transmission of voice and/or data information over telephone lines. In addition to the basic service, PSTN subscribers may also avail of supplementary subscriber services such as call transfer, conference call, wake up call and abbreviated dialling or value added services such as electronic mail and videotex.

Introduction of services into telephony networks has traditionally been achieved through a redundant distribution of service intelligence inside individual nodes in the networks. This redundancy made the deployment and the maintenance of the services difficult and very costly.

The introduction of the Intelligent Network (IN) has changed this focus by concentrating service intelligence at a central location within the network rather than in every network node. IN aims for rapid creation and deployment of communications services. IN is a service independent architecture. IN Service creation requires a structured and disciplined approach. It usually entails modelling, building generic blocks for simulation, and specifying the service through compiler-independent languages.

While the service creation is a goal on the surface and appears to be a simplex task, in fact it is difficult and complex to achieve [2]. A detailed overview of IN is given in Chapter 2.

1.2 FEATURE INTERACTION PROBLEM

IN aims to achieve the following goals:

- Provide timely creation of new services for the customer

- Support of a wide range of services (generic and tailored)
- Support of an efficient maintenance of IN systems
- Provision of a seamless environment between systems vendors
- Insofar as possible, automation of services that leads to operator intervention

However, the problem of service interactions may hinder the IN from reaching its goals. Non-associativity of feature compositions is considered as a Feature Interaction (FI) problem. In some situations, a new feature or a set of features may modify or influence negatively the behaviour of another feature, which always behaves correctly when it is isolated. These interactions will disturb the customers or users and may lead to an unpredictable usage of network resources.

An example of service interactions happens when Originating Call Screening (OCS) and Call Forwarding Unconditional (CFU) are combined. CFU allows incoming calls to be redirected to another directory number, while OCS aborts attempts to connect the subscriber to some other directory numbers. Assume subscriber1 has OCS in order to prevent connecting to the number C, and subscriber2 has CFU which is set to forward the incoming calls to number C. When subscriber1 dials subscriber2 number, his OCS service checks the destination number against the screening list and allows call to be completed. However, based on the CFU, the calls to subscriber2 are forwarded to the C line, which will result to connecting subscriber1 to C.

1.3 OUR CONTRIBUTIONS

We introduce a novel offline FI detection approach that is applicable in the IN context. Our method is pragmatic and based on the experience stemmed from existing feature

interactions, introduced in Bellcore [4] and European [5] benchmarks. The main steps of our study of feature interactions are as follows:

1) Determination of the causes of feature interactions
2) Definition of the necessary information for feature description
3) Modelling the feature description using an object-oriented template
4) Adapting the filtering method to derive the interaction prone call combinations.
5) Instantiating the features using the filtered combinations
6) Devising a novel detection algorithm based on sufficient conditions for feature interactions.
7) Implementing a tool to validate our approach.

1.4 ORGANISATION OF THE THESIS

The organisation of this thesis is as follows:

Chapter 2 provides an overview of Intelligent Network concepts and infrastructure. A brief history of the IN development, its components, its call model and capability sets are covered. IN planes and the way of introducing new features are described.

Chapter 3 focuses on the feature interaction problem in IN. Emphasis is put on different ways of tackling the feature interaction problem and detection techniques.

Chapter 4 gives an overview of our novel approach for FI detection. A classification of the possible causes of feature interaction is presented. For each category the known causes of feature interaction along with a real example are presented.

Chapter 5 describes the necessary information to describe features. Emphasis is put on the modelling of information. The related object-oriented aspects of our model are presented, and the proposed model is described. This chapter also includes the overall model, the relation between the components and the specification of each component.

Chapter 6 presents our detection technique. The different components of the detection technique, including filtering, feature instantiation, and detection algorithm, are described.

Chapter 7 provides the architecture of the feature interaction detection tool and its user interface. The results of applying the tool to the features introduced in the benchmarks are summarised. Furthermore, new feature interactions are presented.

Chapter 8 summarises our contributions as well as the advantages of our approach. Finally, plans for future work and extensions are presented.

Chapter 2

INTELLIGENT NETWORK CONCEPTS

2.1 INTRODUCTION

The provisioning of services within the communications networks has traditionally been focused on implementing service intelligence in the individual nodes within the network. Operators without an intelligent network capability have to deploy new services directly on to the switching nodes.

However, the IN services are created at a service management centre and can be easily adapted or changed as required without affecting the network switching nodes. The introduction of standardisation and deregulation in the communications market has resulted in increased competition between network operators.

Operators are now distinguished by the range of services they offer using the intelligent networks, since an intelligent network enables the rapid creation and supply of advanced supplementary services for customers [2]. The objectives of IN are:

- Rapid deployment of services in the network
- Vendor independent and standard interfaces
- Opportunities for third party and non operating companies to offer services

2.2 INTELLIGENT NETWORK DEVELOPMENT

Prior to the 60s, the service logic was hard-wired in switching systems. The network operators had to meet with the switch vendors, discuss the types of services the customers required, negotiate the switching features that provide the services, and then agree on a generic release date for the feature. This process was suffering from several weaknesses. For example modifying the implemented services was very difficult and services were not offered in the same time across an operator area [1]. In the traditional POTS, the switching system performed the basic call processing. In this system, the introduction of telecommunication services was a difficult procedure, which required the modification of the basic call process.

Around mid-60s, the idea of Stored Program Control (SPC) was a major step forward. It made the introduction of the new services easier. After that progress, the service provider oriented toward centralisation, since using the SPC, there was a dependency between the service and service-specific logic.

Another problem with the traditional service offering was the calls setup (the signalling and call supervision that takes place between switching systems and the actual call). When a call was setup, the signal and talk path used the same common trunk from the originating switching system to the terminating switching system. When the terminating call was busy, it was leading to a useless setup of the multiple offices involved in the call.

Introduction of the Common Channel Signalling Network (CCSN) was a step toward solving that problem. Signalling System number seven (SS7) is the protocol that runs over CCSN.

2.2.1 ISDN

A movement for standardisation started by the International Telephone and Telegraph Consultative Committee (CCITT) which is now called the ITU. The ITU-T is a standardisation sector of the ITU, which is responsible of developing the telecommunication standards for telephony. Beginning of the 60s, the telephone system gradually began converting its internal connections to a packet-based, digital switching system. Integrated Service Digital Network (ISDN) is a standard for the global digital communications system. It allows the complete integration of both voice and non-voice such as data, fax, and video transmission within a single system. ISDN call setup is similar to the current telephone system. To establish a connection the user performs some ISDN control functions [41]. The ITU-T is the developer of ISDN and IN. IN provides an open platform, which can be considered as an additional layer on top of any bearer network, such as ISDN, PSTN, or Public Land Mobile Network (PLMN) [6].

2.2.2 SIGNALLING SYSTEM NUMBER 7 (SS7)

Modern communications networks around the world require a signalling system that allows the different network applications to interwork successfully. SS7 provides the protocols and communication paths to fulfil these requirements as well as providing comprehensive signalling capabilities.

SS7 defines the procedures for the setting up and clearing of a call between telephone users. It performs these functions by exchanging control messages between the SS7 components, which are supporting the end user connections.

Figure 2.1 depicts a typical SS7 topology. The subscriber lines are connected to the SS7 network through the Service Switching Points (SSP). The SSPs receive the signals from the customer and perform call processing on behalf of the user.

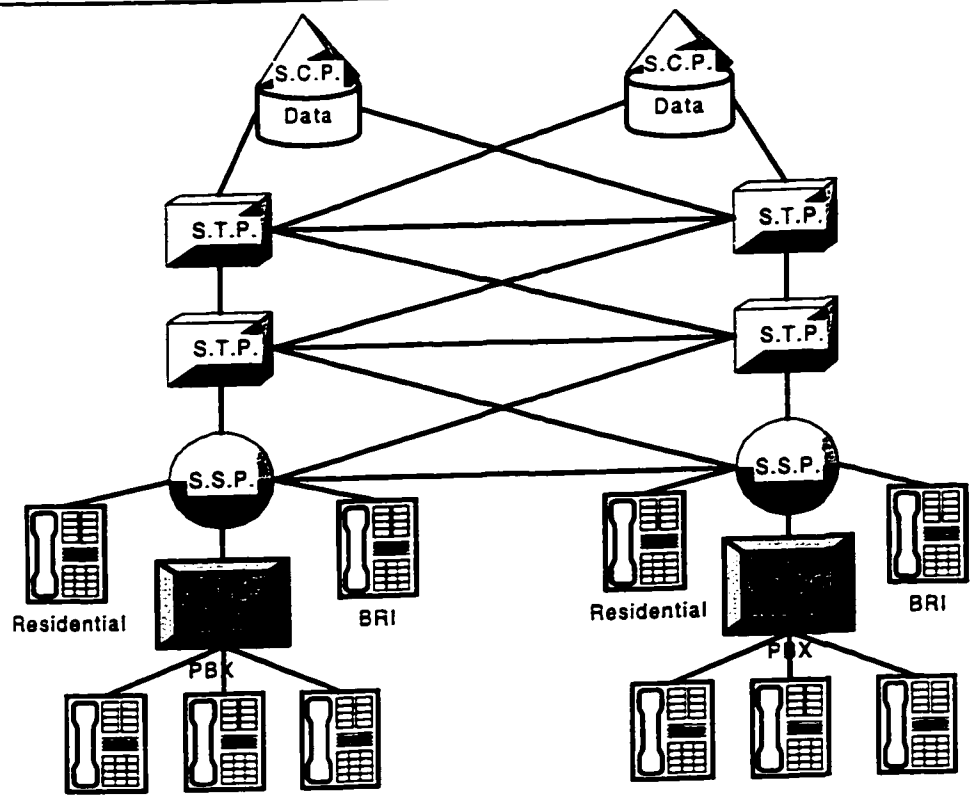


Figure 2.1:Typical SS7 topology

The SSP serves as the source and destination for the SS7 messages. Therefore, SSP initiates SS7 messages either to another SSP or to Signalling Transfer Point (STP). STP translates the SS7 messages and the routing of those messages between network nodes and databases. STPs are switches that relay messages between SSPs, STPs, and Service Control Points (SCP). A SCP contains software and databases for the management of the call.

SS7 network separates the call setup information and talk path from the common trunks that run between switching systems. The call setup information travels outside the common trunk path over the SS7 network. Therefore, the SS7 technology frees trunk circuits up between switching systems for the actual calls. SS7 is the backbone of IN[42].

2.2.3 OSS AND 800 SERVICE

The Bell operating companies described Operator Service System (OSS) to include non-automated and automated operator services. The OSS consists of a switching system that has special operator services. It can receive calls from other Local Exchange Carrier (LEC) and inter-exchange carrier, and end offices. The OSS database contains the

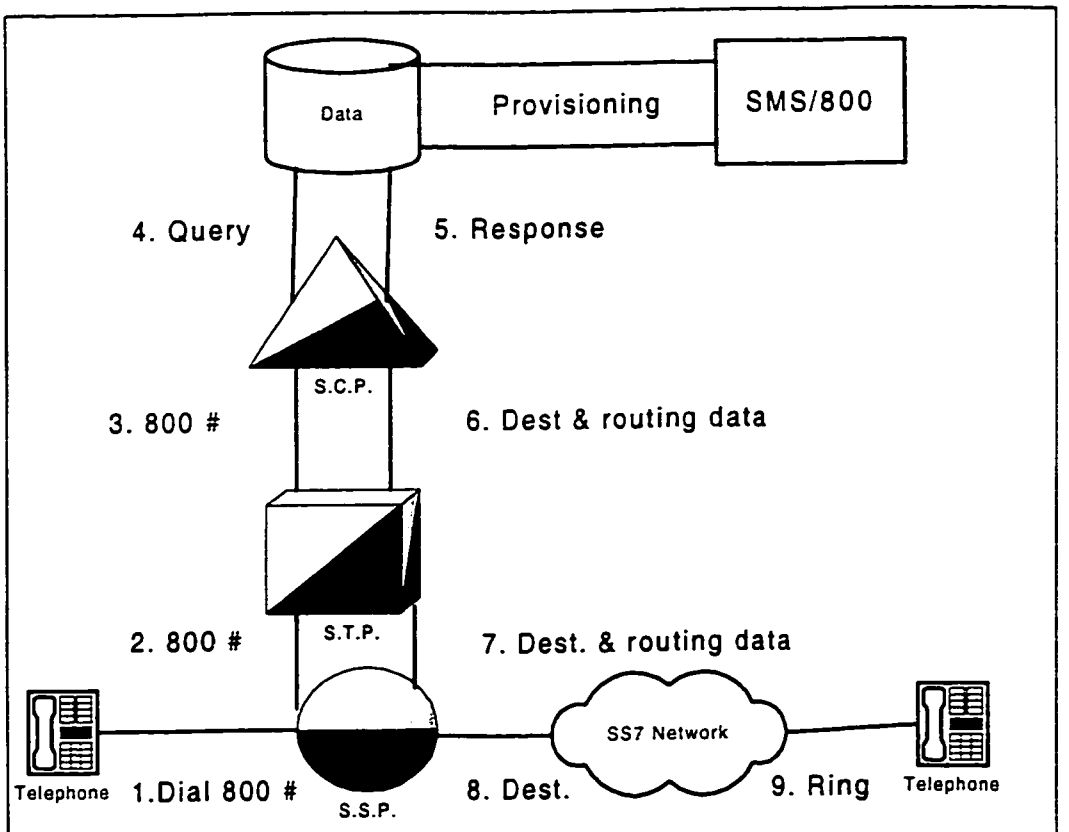


Figure 2.2: The 800 Service

required information for processing the calls and providing information services to the customers.

The 800 service has been in existence since 1976. When a SSP processes the calls, in the case of recognition of 800 number, it suspends normal call processing, forms an SS7 message, and sends the message via STP to a SCP. The query is sent to SCP 800 database, which contains a copy of the associated customer record. The record content defines how the call is to be handled. Figure 2.2 shows how this service works [2].

2.2.4 IN GOAL

In addition to the well-known market forces and the needs of telecommunications providers to offer more for less, time-based competition appears to be an essential driver for global IN. IN can reduce time intervals in three dimensions [17]:

- Circuit provisioning which is the time from when a customer places an order for a circuit to when it becomes available.
- Business service; The global corporation needs uniformity of services that are user-friendly and consistent when using telecommunication facilities in any part of the corporation in any part of the world.
- Service feature availability: The separation of functionality between basic and supplementary services, along with the use of standard control capabilities and protocols allows for independent development efforts.

Looking beyond the traditional "enhanced services," such as call waiting, service providers must move into new markets. Traditional telephone services are not growing

much, and to survive in the marketplace, many providers are looking for new services that will meet the customer needs and bring revenue [2]. This can be provided by IN.

2.3 INTELLIGENT NETWORK INFRASTRUCTURE

IN is a service-independent telecommunications network. That is, intelligence is taken out of the switch and placed in computer nodes that are distributed throughout the network. This idea provides the possibility of developing and controlling services more efficiently. The IN is an evolving concept based on OSS and 800 service and further it uses the technology of SS7 and adds its functionality at the application layer to achieve its goals.

2.3.1 IN COMPONENTS

The implementation of IN-specific components enhances and improves the network, mainly by aiding the rapid creation of services and their efficient maintenance. These components, which are called IN physical nodes, are as follows [8]:

- Service Creation Environment (SCE): It provides design and implementation tools to assist in creating and customising services in the SCP.

- Service Management System (SMS): It is a database management system and is used to control the IN customer services.

- Intelligent Peripheral (IP): It can connect to an IN call and provides the following services: tone generation, voice recognition, playback, compression, call control, and record.

- Adjunct (AD): It performs the same operations as a SCP, but it is configured for one service for a single switch.

- Network Access Point (NAP): It is a switch that has no IN function and is connected off an SSP and interfaces to trunks with SS7 messages or frequency tones.

- The IN STPs perform two functions beyond their usual operations. They can employ pseudo-addressing that enables them to balance the load between two or more SCPs. Furthermore, they can employ alternate routing in case of a problem in the network. Figure 2.3 shows the IN specific components.

Functional entities are the components of the physical nodes and are summarised as follows:

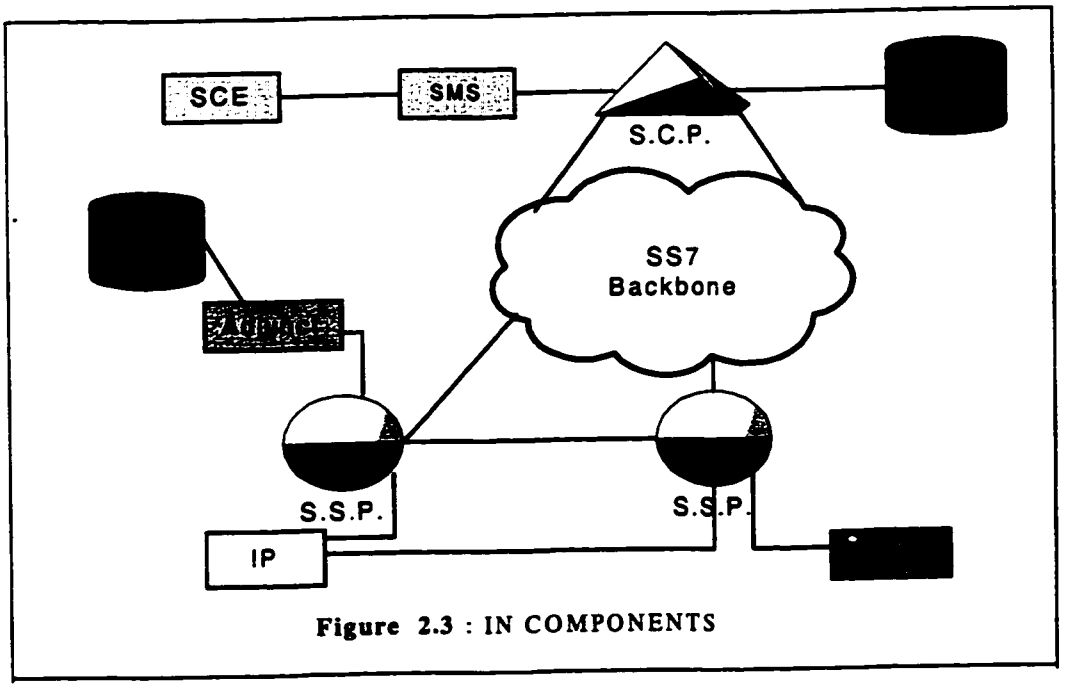


Figure 2.3 : IN COMPONENTS

- Service Switching Function (SSF): It provides the means to recognise the calls requiring IN service processing. It interacts with call processing and the service logic on behalf of those calls.

- Call Control Function (CCF): It provides the means for establishing and controlling bearer services on behalf of network users.

- Call Control Agent Function (CCAF): It provides users with access to the services and represents the users to call processing.

- Service Control Function (SCF): It contains IN service logic providing the logical control to be applied to a call involving an IN service.

- Service Data Function (SDF): It handles service-related and networks data. It provides the SCF with a logical view of the data.

- Service Management Function (SMF): It provides the service provisioning deployment, and management control.

- Specialised resource Function (SRF): It provides end-user interaction with the IN through control over resources such as DTMF receivers, voice recognition capabilities, protocol conversion, and announcements.

- Service Management Access Function (SMAF): It controls access to service management functions.

- Service Creation Environment Function (SCEF): It supports the creation, verification and testing of new IN services.

2.3.2 IN CALL MODEL

The call model is a representation of a sequence of procedures executed by an IN to set up, manage and clear an IN session between IN components. It defines the interfaces, states and events that are associated with each type of IN service [9].

IN call model consists of Point of Initiation (POI) and Point of Return (POR). Whenever the basic call processing is not able to provide a requested service, it invokes the global service logic. This invocation occurs in the basic call processing at the POI. The execution is turned over to Global Service Logic (GSL), which executes a chain of Service Independent Building blocks (SIB), after which it returns to the BCP at the POR.

Table 2.1 shows the POI and POR for the CS-1.

POI	POR
Call originated , Address collected, Address analysed , Call arrival , Busy , No answer , Call acceptance , Active state , End of call	Continue with existing data , Proceed with new data , Handle as transit , Clear call , Enable call party handling , Initiate call

Table 2.1: POIs and PORs in CS-1

2.3.3 IN CAPABILITY SETS

IN is just a standard and needs a series of compatible Capability Sets (CS) to become implementable. A CS defines a specific standard stage of IN evolution in terms of the services to be supported and the functional architecture supporting these services. In March 1993, the ITU-T approved a set of capabilities for intelligent networks called CS-1 [6]. However, in North America the tendency is not to follow the formal international

standards. Bellcore has published a set of specification known as Advanced Intelligent Network (AIN) releases 0.1, 0.2 and 0.X [2]. In this thesis, we use the concept of IN, however only for more clarification some of the given examples will refer to the AIN.

In 1996, Bellcore released another version of its AIN specification. In Europe, the European Telecommunication Standards Institute (ETSI) introduced Intelligent Network Application Protocol (INAP), which is another activity in this domain.

On the other hand, the ITU-T has provided series of CS. CS-1 is targeted to support single-ended and single-point-of-control services and does not support services where several IN subscribers may be associated with a single call. CS-2 provides a significant challenge to specify adequate details for multi-vendor compatibility.

2.4 INTELLIGENT NETWORK PLANES

In order to establish specific standards and requirements for the development of IN architectures, a reference model called INCM has been created. For development of IN architecture in the future, this model is to be followed carefully, to ensure the worldwide development conforms to the specified standards.

The INCM defines four specific planes. The lower two planes address the actual IN architecture, whereas the upper two planes focus on service creation and are closer to the user point of view. The four planes are as follows:

2.4.1 IN SERVICE PLANE

This plane is of primary interest to service users and providers. The services are introduced in this plane. Each service in this plane is a group of Service Features (SF).

Each SF is a description of a single set of GSL. Figure 2.4 shows the IN service Plane [10].

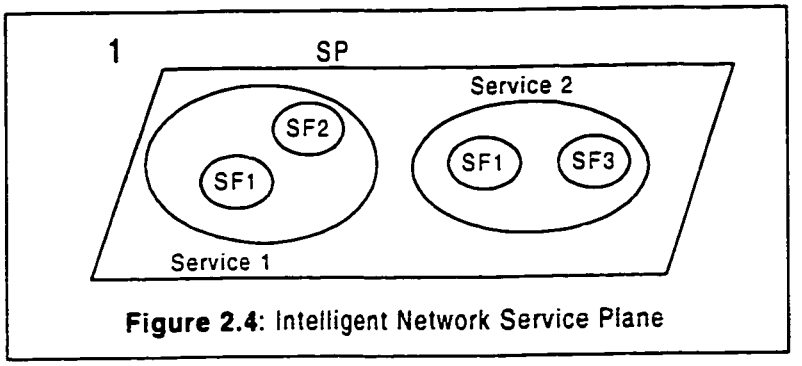


Figure 2.4: Intelligent Network Service Plane

2.4.2 IN GLOBAL FUNCTIONAL PLANE

This plane is of primary interest to service designers. It models the network from high-level perspective as a single entity, which is created from specific units of service functionality called SIBs. SIBs can be seen as functions in a programming language. In turn the GSL defines how the SIBs are combined to provide service features. Basic Call Process (BCP) is a special SIB and is present in this plane. This plane also describes the POI and POR between the BCP and one SIB or a sequence of SIBs. Figure 2.5 shows the IN Global Functional Plane [9].

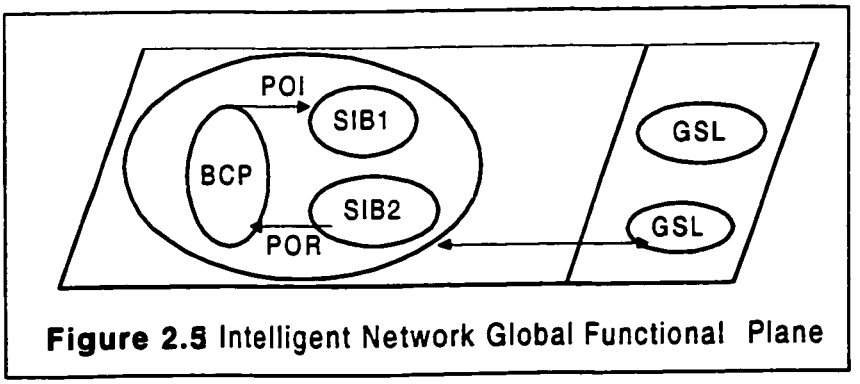


Figure 2.5 Intelligent Network Global Functional Plane

The following SIBs are defined in the ITU-T Q.1203 and Q.1213 (CS-1):

- Algorithm SIB: Simply increments or decrements a call specific variable.
- Authenticate SIB: Checks only a user name and password or more complex authentication.
- Charge SIB: Defines how a call is to be charged.
- Compare SIB: Compares an identifier with a reference value.
- Distribution SIB: Supports the distribution of calls to other ends.
- Limit SIB: Limits the number of calls that are processed in the IN.
- Log call information SIB: Logs information about calls such as call attempts, disconnect times, and dialled numbers.
- Queue SIB: Queues incoming calls when an agent is busy.
- Screen SIB: Compares a value to a list of stored values called a screen list.
- Service data management SIB: Modifies, stores, corrects and retrieves information about a customer.

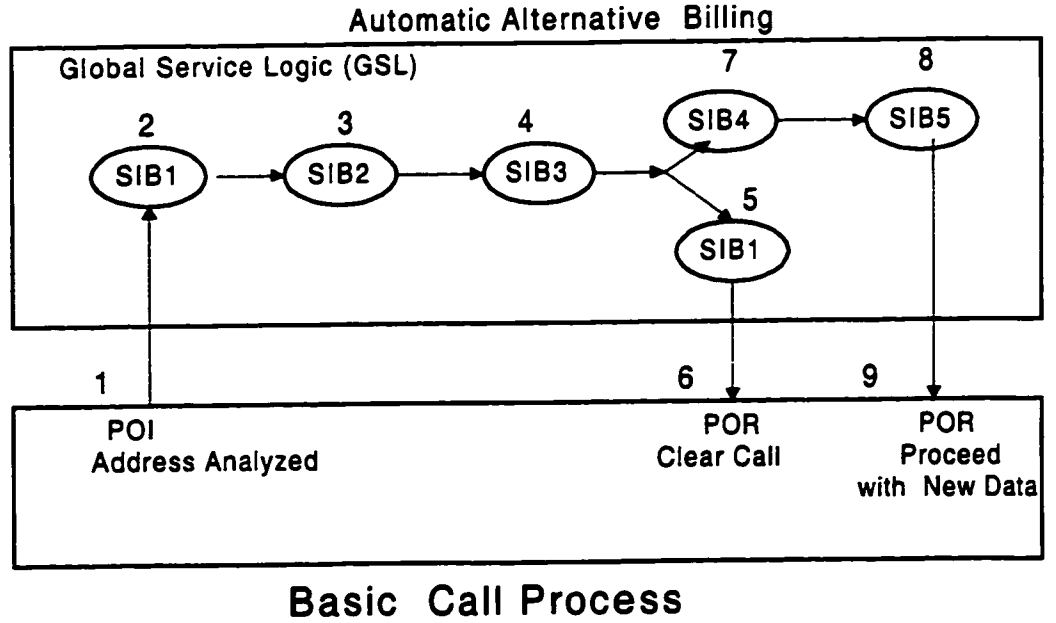


Figure 2.6: The relation between the BCP and SIBs for one Service

- Status notification SIB: It monitors network resources to determine their states or status.
- Translate SIB: It receives input information and translates them to different output information.
- Verify SIB: It checks the syntax of the information to determine it is correct or incorrect.
- Basic Call process SIB: It is responsible for basic call processing.

Figure 2.6 shows the relation between the BCP and the SIBs.

2.4.3 IN DISTRIBUTED FUNCTIONAL PLANE

This plane is of primary interest to network designers and providers. It defines the units of network functionality as Functional Entities (FEs). The SIBs in the GFP are realised in the DFP by a sequence of functional entity actions and the resulting information flows between them. In other words, a SIB is instantiated by FEs and their actions. The actual protocols in the INAP occur when SIBs need to send information between FEs, which in essence allow the SIB to be realised. Figure 2.7 shows the IN DFP [8].

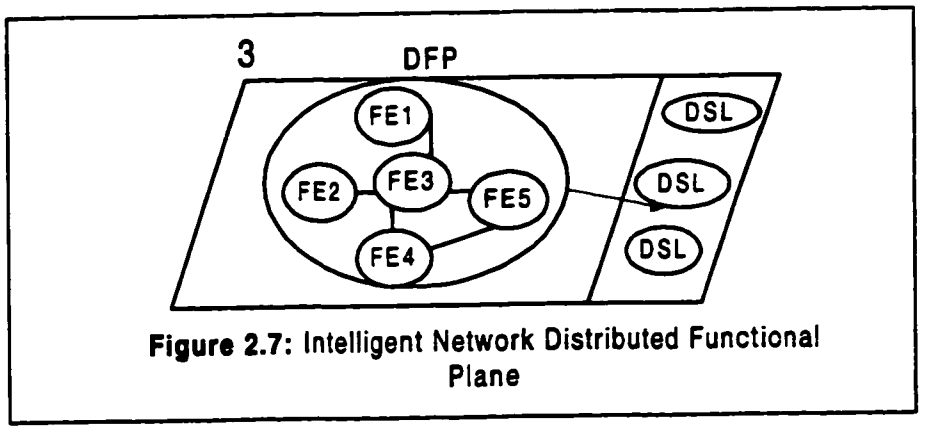


Figure 2.7: Intelligent Network Distributed Functional Plane

2.4.4 IN PHYSICAL PLANE

This plane is of primary interest to network operators and equipment providers and defines the 'real' physical IN architecture and indicates possible mappings of the functional entities in the DFP into Physical Entities (PEs).

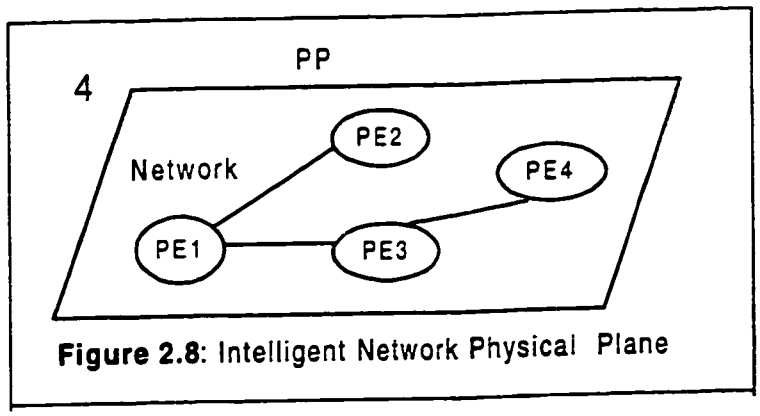


Figure 2.8: Intelligent Network Physical Plane

This plane also defines the required interfaces between the PEs. To summarise, each of the IN FEs in the DFP needs to map to PEs in the PP.

One way to view this model is to think of the two upper planes focussing on how services are created, and the two lower planes especially the PP, concerning with the specific networks and service provisioning. Figure 2.8 shows the IN PP.

2.5 SERVICE CREATION

Putting all the previous explanations together, the service creation is straightforward. Figure 2.9 shows the INCM planes and their relations for the service creation.

Each service in the Service plane of the INCM is a combination of some Service Features (SF). Each SF can be represented using several different types of GSL. There is one set of GSL per SF and it uses SIBs in GFP.

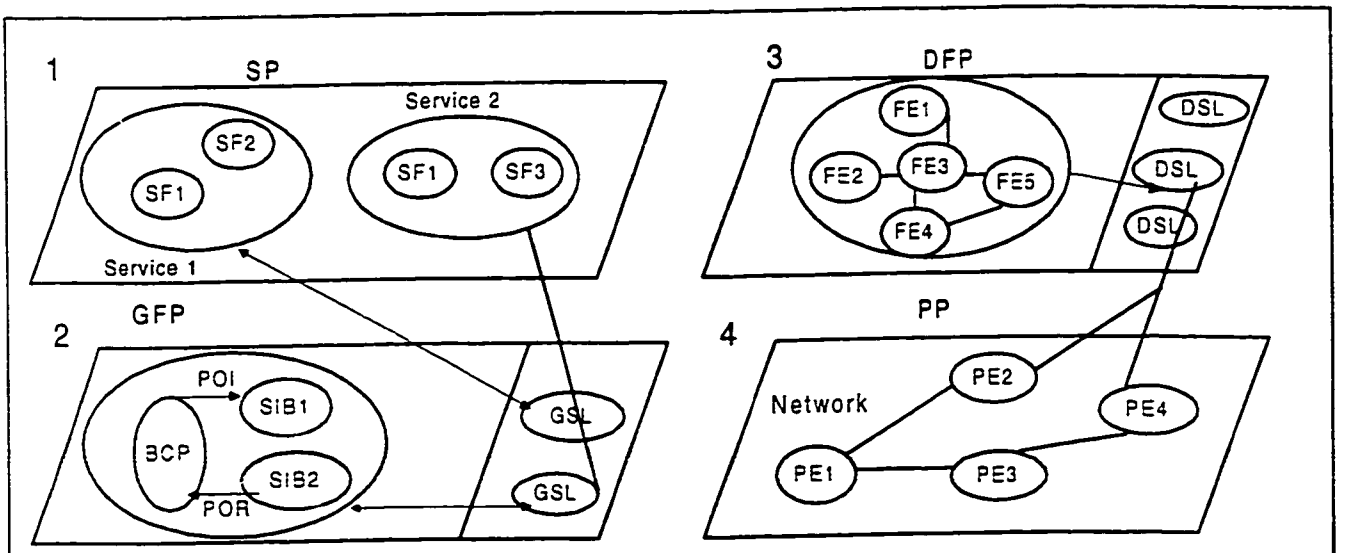


Figure 2.9: Intelligent Network Conceptual Model Planes and their relations

There is one set of Distributed Service Logic (DSL) per SIB and it uses Functional Entity Actions (FEA) and information flows in DFP. In PP the service logic programs may be installed into and executed by any physical entity. Service logic must be implemented in a service-independent way. This can best be done by means of SIB. These SIBs are the programs written in programming languages and are presented in GFP. SIBs are realised in the DFP by a sequence of particular FEA. FEAs are performed within various FEs, which are presented in DFP. The FEs are implemented in PE which are presented in the PP. PP models the physical aspects of IN-structured networks. The model identifies the different physical entities and protocols that may exist in real IN-structured networks.

Services consist of one or more SFs. A SF is the smallest part of a service that can be perceived by the service user. Each SF is a combination of one or more SIBs as described previously. All individual telecommunication services identified in the SP are described as seen from the user point of view. They are considered without any reference to how the services are implemented in the network

2.6 CHAPTER SUMMARY

In this chapter, the concept of IN was reviewed. Looking at history of IN, and its backbone, ISDN, SS7, two old services, OSS and 800 service were explained. For more clarification, the IN structure and its components including call model, which will be referred in the following chapters, were described. Then, the planes of IN and the way the services are created were explained.

The important point is that the IN will continue to provide customer-pleasing services well into the next century. When an IN is implemented, the problem of feature interaction can always affect the performance and hinder achieving the goals. Solving this problem is the fundamental reason of doing a lot of research in all aspects of feature interaction problem in telecommunication systems.

Chapter 3

FEATURE INTERACTION DETECTION

3.1 INTRODUCTION

The feature interaction problem is the main obstacle, which prevents the IN from easily achieving its goals. "How to solve this problem?" is the common goal of the researchers in this field, however, each one has seen the problem differently and tried to solve it with a different approach. Solving the interactions between two or more features involves the detection of the interactions. After finding where the interaction happens, it is possible to resolve it or provide a new architecture in order to avoid it completely.

3.2 FEATURE INTERACTION PROBLEM

Services are the communication capabilities made available to the customers by telecommunication providers. The words " feature" and "service" are often used interchangeably. A "service" carries the connotation of something bigger and more fundamental than a feature. A service feature is a specific aspect of a service that can also be used in conjunction with other services or service features as part of commercial offering. It is either a core part of a service or an optional part offered as an enhancement to a service [3]. These services drastically extend the scope of telephony.

The increased demand for new telecommunication services has led to a rapidly growing number of new services as well as the enhancement of existing services with new features. As previously described, in the IN architecture, service logic can be stored in the SCP while the switching functions are provided in the SSP. In addition, IN provides a set of well-defined interfaces between the SSP and the SCP to allow service logic programs in the SCP to be invoked by the SSP in order to influence the call processing. Thus, it eases the procedure of realising the new services [2]. As the number of features grows, the amount of work, which has to be invested into the treatment of interactions between them, explodes and the complete testing of the features in combination is almost unmanageable.

We use an example from the telephony features to illustrate the problem and demonstrate the character of real interactions. For example a flash hook signal generated by hanging up briefly or depressing a 'tap' button, issued by a busy subscriber, could have different meaning. It means to start adding third party to an established call for TWC or to accept a connection attempt from a new caller while putting the current conversation on hold for Call Waiting (CW). Figure 3.1 shows the interaction between CW and TWC.

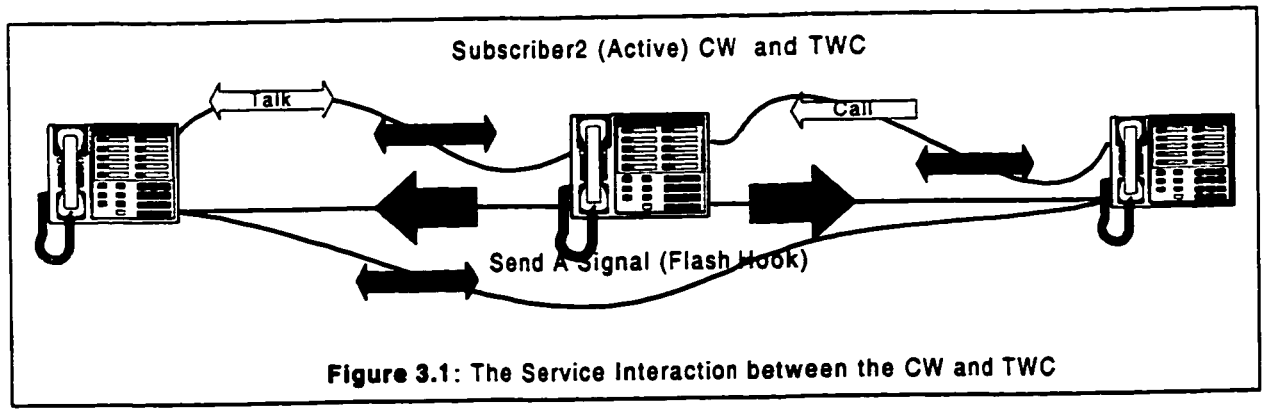


Figure 3.1: The Service Interaction between the CW and TWC

Suppose during a phone conversation between subscriber1 and subscriber2, an incoming call from subscriber3 has been activated at the switching element for the line of

subscriber2 and triggered CW feature to which subscriber2 has subscribed. However, before being alerted by the CW tone, subscriber2 has flashed the hook, intending to initiate a TWC. Should this flash hook be considered the response for CW, or an initiation signal for TWC? This ambiguous situation is called FI between CW and TWC.

As described by the ITU-T recommendation for CS-1, the service interactions are described from the user point of view. Service interaction applies to all interactions of the service being defined with the other services, which have already been defined. It can happen in different cases such as:

- Among different features associated with the same service;
- Between features associated with a service for a given service-user and features associated with the other services the same user may have requested or been assigned;
- Between features associated with a service for a given service-user and features associated with possible services related to the terminal or calling line that the user is currently using;

An IN structured network handles multiple services for the same call. The necessary interactions shall be defined for the processing of several services for the same call. When multiple services can be activated concurrently, some prioritisation of services will be necessary. User specific requests may take priority over a group service request. Additionally, certain services may deactivate other services.

The service interaction is part of the specification of services, and should be dealt with in the service plane modelling. Different examples of FIs are presented in [4] and [5]. Another simple example of service interaction happens between two services: Unlisted

Number (UN) and Calling Number Delivery (CND). Figure 3.2 shows the interaction between CND and UN. CND is a call-processing feature that delivers the directory number of the calling party to the customer premises equipment during the ringing cycle; this assumes that information such as the number of subscriber will be released. UN, on the other hand, is a directory-service feature designed to allow a subscriber to keep the number private. Conflicts between the goals of these two features arise when the customer A with UN places a call to another customer B with CND. If the network allows number of A to be delivered to B, then A loses privacy, if it does not, then B gets no information. This shows the interaction between CND and UN.

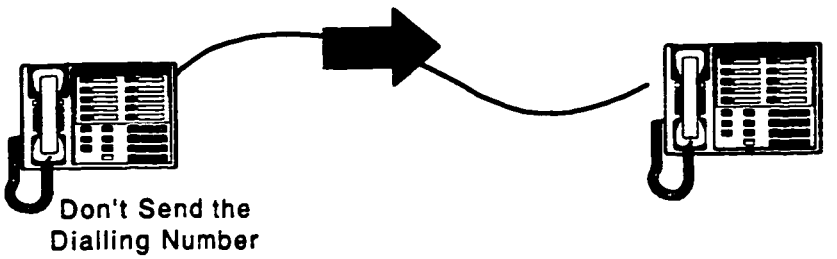


Figure 3.2: Service Interaction between CND and UN

3.3 FEATURE INTERACTION MANAGEMENT

So far, a lot of research for identification, nature, cause, management and resolution of feature interactions has been done. Some of these works have particularly been accomplished in order to organise the structure of the problem and classification of the other researches.

In [11], different branches of research in this domain are introduced. In [29], a good overview of the different ways of tackling feature interaction is presented. [4] and [5] are the useful benchmarks for the feature interaction problem and referring to the definition of the problem, they propose the structured ways of categorising the feature interactions.

For the different solutions proposed by the researchers, the generally accepted categorisation of approaches based on the terminology of avoidance, detection, and resolution has been introduced by [16]. It has been refined by [12] to the classification displayed in Figure 3.3. Keck and Kuehn [30] added the management category to the previous categories.

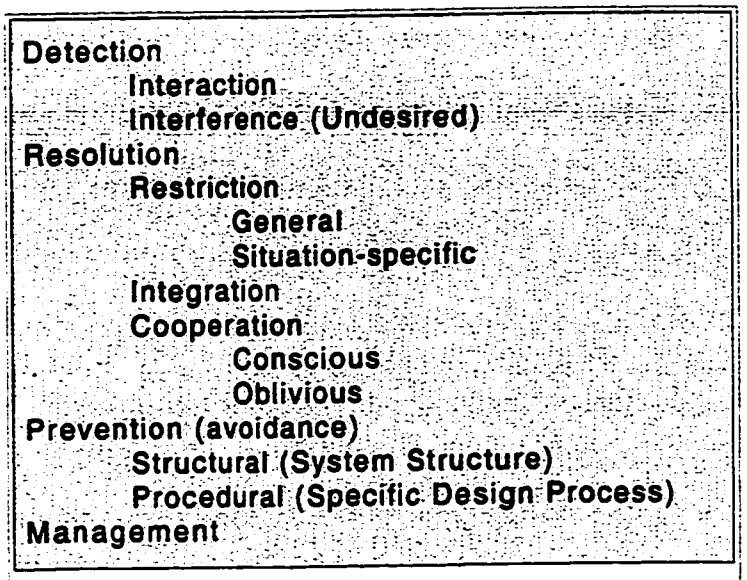


Figure 3.3: Classification of Approaches

3.3.1 DETECTION

The goal of the detection approaches is to find out whether and where the feature interaction exists between combination of two or more features. So, the existence of the interaction is searched. However the interactions are not always undesired (interference). The detection of interference situations needs the concept of desirableness, as only undesired interactions are identified and desired ones are ignored. Undesired interactions are ones, which lead the services to fail or provide subscriber unsatisfactory. It is worth to consider the interpretation and expectation of the subscriber from the services may vary

that affect the concept of interaction. For example, some may use CFU in order to be reachable at another location, while others as a form of "Do not Disturb".

The detection approaches can be implemented using two methods: Design-oriented methods and analytical methods. Analytical methods are applied in order to handle the feature interactions in existing systems or system models. The analytical methods are further divided into formal methods and pragmatic methods. The approaches can also be classified based on the stage of the system life cycle in which they are applied. This classification divides them mainly in two groups: On-line and Off-line.

The On-line approaches are applied after the system is implemented. They can be applied during the system test and before using the system, in the real environment. Also, On-line approaches embrace the ones that are applied after installing the system in its working environment. Some examples of these kinds are applied for the provisioning, registration, activation, and invocation of the services. These approaches mainly use the formal techniques. The Off-line approaches are the ones used in specification level, in the level of system requirements, or in the implementation stage. Offline approaches can be applied using different type of techniques: Formal techniques or Pragmatic techniques.

3.3.1.1 FORMAL DETECTION TECHNIQUES

In these methods, the features or services are described using formal description language, and then safety and special properties are described using another appropriate language. As an example, temporal logic language is used beside Promela to describe the properties. Detection algorithm consists mainly of running the features randomly and in any possible order while watching the properties to see whether they are satisfied in all situations or not. Whenever the properties are not satisfied, an interaction is detected.

The authors in [49] introduce a methodology for describing the services using the modelling languages Promela and Z, and the necessary properties in "Temporal Logic". Verifying the properties while running the feature models will detect potential FIs.

Bryce Kelly et al. have developed two abstract models to describe features and detect FIs using SDL [32]. In Centrex model, FIs are detected by observing the conflicts in the features behaviours inside the switching software of the "complete node". In service plane model, the potential FIs are detected by means of conflict "messages" sent to users from a set of active features as well as contradictory "feature return requests".

In [45], the authors proposed a way to detect and resolve FIs at different stages of feature development: specification, design, manufacturing and execution. FSM models were used to analyse the specifications and search for conflicts between features. FIs are detected by tracing for non-deterministic conflicts among the features.

L. Logrippo et al. from University of Ottawa have used LOTOS as a specification language in their study of FIs [20, 39, 40]. They have used a LOTOS model of a sample telephony system and applied the step-by-step simulation to detect FIs. They also developed a method based on feature composition and integration. They detect FIs by comparing features individual behaviours and their supposed combined behaviours. They present a detection method using backward reasoning technique in formally specified LOTOS-based IN model.

Many other formal detection methods can be found in the proceedings of the international Feature Interaction Workshops [21-25]. However, these methods have many drawbacks, which hinder their practical usage. There are some drawbacks in using these methods, which hinder their practical uses. Some of them are:

- Finding appropriate level of abstraction is difficult.

- Formal modelling and analysis is time consuming.
- In multiple vendor market, source specification, whether informal or formal one is not available.
- Most formal analysis can only consider features pair-wise.
- There is always the state explosion risk.

3.3.1.2 PRAGMATIC DETECTION TECHNIQUES

A pragmatic method is an off-line approach established based on the experience of the existing FIs and anticipating the coming feature interactions. In this kind of methods the causes of interactions between features are used to find the sufficient knowledge for describing the features. Then the information items are modelled using special template. The detection algorithm is basically finding the presence of causes of FIs. Previously, most of the pragmatic methods were established on the knowledge of an expert to run his or her algorithm manually and change the algorithm whenever it was necessary. A good pragmatic method must preferably exist in automatic version, in order to provide expert independent and user-friendly environment. It must be as generic as possible and extendable for the new features.

In [36, 37] a detection method which is based on describing a feature using two models is introduced. The two models are "call context model" and "feature context assumption". Analysing these two concepts for two or more given features, the existence or absence of the common call context can be derived. In the case of having common call context, the conflicting assumption is searched to identify the possible interaction between the features. This method is called Peng-Method and will be referred to, for the purpose of comparison in the next section.

In [33], a method to handle service interactions in IN at specification level is proposed. It describes features using a template that highlights those aspects of the service that are relevant for its interactions with other services. Then using a general detection procedure on those feature descriptions, feature interactions can be detected.

Some of the advantages of the pragmatic methods are as follows:

- The analysis algorithm is simple.
- The necessary information for describing the features is available.
- The features are described using the actual behaviour of the features not the written prescriptive specification.
- Abstraction of the other involved components is simple.
- The interactions between any number of the features are detectable.

Based on the given classification of research areas and possible approaches to solve the feature interaction problem, we chose the detection category since it is the first and main step of solving the problem. Furthermore, regarding the drawbacks of formal methods and the simplicity of pragmatic ones, we chose to follow a pragmatic approach.

3.3.2 RESOLUTION

Resolution comes after detecting the interaction. It is to solve the occurrence of the interactions. Restriction, integration and co-operation, are different ways of resolution. In these approaches, each feature does not modify or adapt its normal behaviour, but a set of rules disallows the activation of one feature if another feature is present. By restriction, the presence of two features in an interaction-prone situation is restricted. Co-operation is changing the behaviour of one or all the features that are to coexist in order to resolve

interactions. Integration of some features into one with larger functionality is another way of resolution.

3.3.3 PREVENTION

Prevention or avoidance techniques are used to avoid feature interactions completely. These avoidance approaches can be divided into two categories: Structural and Procedural. In structural approaches the interaction-prone situation is avoided by suitable system structure. The procedural approaches introduce new strategy for service creation or give a set of design guideline to avoid FIs.

3.3.4 MANAGEMENT

The feature interaction management refers to the management aspects in order to solve the problem. They try to see the problem as a whole. Some of them associate all the necessary components for FIs solution, prevention, detection with resolution, or any other possible combination. Also the approaches which address the management of one component of the system, can be put in this category. One example of this type is the management of a huge number of test cases which is called the filtering method. Considering the management aspects of the problem was an important issue during our study.

3.4 SURVEY AND COMPARISON OF DETECTION TECHNIQUES

To understand the detection approaches some of them were chosen for deep study. This study shows the existing formal methods and approaches based on the classification given in the previous sections require significant amount of work and thus are relatively

time-consuming. All the methods are found to be limited from different point of views. Some use the abstract concept and are not simply applicable to the actual environment and others not general enough to be applicable to many types of interactions or to many combinations of services and service features. To find out the best way to tackle the feature interaction problem a comparison of at least some of the existing methods and approaches is necessary. The metrics for this comparison are:

- Complexity (Does the method apply the complex computations)
- Generality (Does the method cover all kind of interactions)
- Reusability (Is the method applicable for the new features as well as the existent ones)

In selecting the methods for deep study, we tried to look at management of the problem, detection in general and different techniques including formal and pragmatic ones. Each method is described in terms of its goal, base, context, necessary information and algorithm. Then its strengths and weaknesses are highlighted.

3.4.1 KECK- METHOD

In [31], the idea of extracting the interaction-prone scenarios from the possible scenarios is introduced. It describes all the possible call combinations, which can happen between two features, then using a filtering algorithm it detects the interaction-prone scenarios to avoid further processing of the other scenarios. This approach addresses the management of test cases and can be used before any other detection approach.

It uses the description of each service, in order to create a large list of call scenarios for each combination of services. The length of the list of scenarios is reduced by topological

and non-topological criteria. The output reports the interaction-prone call scenarios to be investigated by detection method (e.g. formal method). It is applied for IN context. For each service, the name of the service, number of formal participants and the topology of the scenario are necessary. The algorithm is mainly the application of conditions to derive the interaction prone scenarios. The output is the number of topologically different call scenarios. The number of non-problematic call scenarios and the number of problematic call scenarios to be processed by another detection method.

COMPLEXITY: It uses the simple description for the features.

GENERALITY: It cannot cover the interactions that occur in the logical level of BCSM. Furthermore, it supports only the interactions between two services.

REUSABILITY: The language must be extended to embrace other characteristics of the services.

RESTRICTIONS: The conditions only support the identification of interaction-prone scenarios on the abstract architectural level. The interactions on the logical level are not covered

3.4.2 PENG-METHOD

Peng-method is a pragmatic method for feature interaction detection. It analyses various feature-using situations in which different feature subscribers play different roles. The method, which is applied for IN context relies on two kind of information:

- Call context model, which includes the name of the feature, subscriber, first call context, other call context. For each call context: one caller, one called.

- Feature context assumption, which includes the triggering condition, priority or cascading, data manipulation, resource requested, operation requested and signal claiming.

The detection algorithm combines two or more features by joining them at one or more common call context. The satisfaction of the requirement of each feature is checked. The checks can lead to two cases: If the requirements are satisfied, there is no feature interaction. Otherwise, there is potential feature interaction. Then different criteria including triggering conflict, cascading conflict, data manipulation conflict, resource requested conflict, call operation conflict, signal claiming conflict and basic assumption conflict are checked. The result is the detected feature interaction with respect to the feature using cases and conflict types.

COMPLEXITY: It is computationally simple and no state-explosion occurs.

GENERALITY: It can detect most known (Not all) feature interactions as well as new ones. This approach can be used at different levels of feature introduction, specification, design and implementation level.

REUSABILITY: It can be applied to combination of more than two features.

RESTRICTIONS: If there is no conflicts, we are not able to conclude that the features under study do not have undesirable interaction. A list of causes of feature interaction has to be completed. There is a lack of series of complete and standardised set to allow unique assumption specification.

3.4.3 FUCHUN-METHOD

In [35], the sources of feature interactions are assumed to be control and data sharing. The proposed methodology is based on having the input, process and an output. The

inputs are information modelling and features specifications. The process is an algorithm and the output will be an anomaly report. The information modelling is based on the relational database and the detection algorithm is based on data and control sharing.

This pragmatic method does part of filtering and part of detection and can thus, be considered as intermediate in nature. Once a feature is specified, its specification is validated for consistency with respect to the control and data. Interaction analysis is conducted for a set of features based on the sharing of call variables between the SSP and SCP to reveal the potential interaction. It is applied for AIN context. Information modelling includes encoding and organising the feature execution environment in a machine manipulable form, considering basic call model, call-related TCAP message and a set of call variables, which are exchanged between SSP and SCP. The minimum required feature information from the service providers described in two categories:

Data specification: How feature manipulates the data within the scope of its control.

Control specification: The control flow of feature with respect to the AIN calls model.

The detection algorithm has two phases: Message sequences are generated for each AIN feature and the features specifications are validated. Parsing the control specifications for AIN features and sequence generation. Computing cumulative call variable usage(s). Validating data specifications for AIN features. Formalising interaction concepts. A package of features is checked for interactions. Reducing the amount of analysis for feature package. The output is the anomalies discovered by the algorithm.

COMPLEXITY: The published information of the features is enough to apply this method, but knowledge of relational databases is necessary to model them.

GENERALITY: Two types of interaction can be detected: side effect and disabling interactions. The methodology is applicable for the AIN features.

REUSABILITY: The current algorithm must be extended in order to cover the scope of IN CS-1 and cannot be used directly.

RESTRICTIONS: Multi-party interactions are not covered in this methodology. Only interaction between AIN features and switch-based features and the interaction between AIN features are discussed. Interactions among the switch-based features are not discussed. In this contribution, the execution of an AIN feature is assumed atomic with respect to other features. In addition, the activation and termination of an AIN is assumed to be done at the same PIC in a BCM, which is not always the case.

3.4.4 LIN- METHOD

In [34], there is an approach to develop an environment based on formal methods to support service creation and feature interaction analysis and management for AIN services. To describe the framework, it describes its three aspects: Process, methodology and tools. This method is a formal one based on the modelling the feature logic and its operating environment as units of add-on functions or building blocks that can be combined for detecting incomplete or incorrect specification. Detecting interactions can be viewed as process of checking whether the actual behaviours of feature in the presence of other features are different from the intended one. It is applied for the IN context. The necessary information is: procedural specification and feature logic to represent the operations of the feature, corresponding context to represent the environment of the feature, and the behavioural specification as a set of temporal logic formulas, each showing a property that must be valid in all circumstances.

The detection algorithm consists of defining a basic call model, deriving a basic feature context from BCM, specifying the features in terms of FCs, composing the feature

specifications for detecting interactions, and finally modelling the interaction resolutions for verifying their correctness.

COMPLEXITY: The familiarity with the computer-aided tools and the related languages is necessary to do the tasks of modelling, configuring, analysing and storing the necessary information to deal with the feature interaction problem.

GENERALITY: It supports presently the interaction among the AIN features. Using the message passing mechanism instead of data sharing, the method can be applicable to the switch-based features as well. It must be adapted to the other networks.

REUSABILITY: This method can be used for current features as well as futures ones.

RESTRICTIONS: Like all the other formal methods, there is the state explosion possibility.

3.4.5 YASUKI- METHOD

In [44] three type of knowledge are used to detect the feature interaction.

- Knowledge of clarifying and completing the specification of each feature
- Knowledge for identifying the implicit relationships among the features
- Knowledge for detecting the interactions from the extended feature specification derived from the first two types of knowledge.

Yasuki-method applies to all type of networks including the IN. The detection algorithm consists of describing the specification of each feature, then, completing the feature specification using knowledge (1) and finally identifying the execution conditions of feature in terms of various parameters using knowledge (2). Knowledge of type (3) is applied to detect the interaction.

The output of the detection algorithm shows the features, which are involved in the interaction? What is the interaction? Where does the interaction take place? What condition causes the interaction to occur?

COMPLEXITY: It uses the simple algorithm but needs additional information for completing the features descriptions.

GENERALITY: It covers the interaction that occurs in all kind of network systems including IN.

REUSABILITY: The language must be extended to include other characteristics.

3.4.6 KUISCH-METHOD

In [28], a method to handle service interactions in the IN at specification level is proposed. It describes the features using a template to highlights those aspects of the service that are relevant to its interaction with other services. Then using general detection procedure on those features description, the feature interactions can be detected. The method is applied to IN context. The interaction template contains the following items:

- Introduction: The capabilities of the service and its requirements on the network that supports the service.
- Features: Features are described in terms of their co-operation. The role of features in normal and exceptional call processing. It includes time sequence diagram to describe the active features in each phase.
- Information flows and elements: The functional entities, information flows, information elements, and relation to the basic call state model.

The detection algorithm begins by producing the behaviour specification according to an interaction template for each service specification.

1. Comparing each new behaviour specification against the older one. If an older one has an identical interaction characteristic, the algorithm is continued from step 3, otherwise from step 2.
2. From all behaviour specifications, single interaction detection document is produced.
3. An interaction resolution document is produced from the interaction detection document.

COMPLEXITY: It does not have complex computation and the necessary information about service can simply be derived.

GENERALITY: It supports the logical interactions at the specification level.

REUSABILITY: The method needs to be completed concerning other resources,

RESTRICTIONS: The implementation-dependent interactions are not supported. The interactions among the IN services are only supported with this method.

3.4.7 SUMMARY OF THE COMPARISON

The Table 3.1 contains the summary of the methods in terms of goal, context, information, causes and proposed algorithm.

	KECK	PENG	FUCHS	LIN	YASUKI	KUNSCH
GOAL	Management of Test Cases	Pragmatic Detection	Filtering & Detection	Formal Detection	Detection	Detection
CONTEXT	IN	IN	AIN 0.1	IN	All Networks including IN	IN
INFORMATION	**SD:** Name of S # of FPs: Name of FP ID # of FP BCSMs: DPs:	**SD:** Name of F Subscriber Call Context(s) Caller Callee **SE:** T Condition Priority D Manipulation O Requested S Claiming BA: # of users # of lines	**SD:** D Specification C Specification **SE:** BCSM TCAP message Call Variables TCAP inter-relations	**PS:** **SD:** Operation **SE:** Environment **BS:** Properties or Conditions	**SD:** Required-Processing Execution-Condition Input/ Output of processing Entity MSC among Processing	**SD:** Functional Entities Information flows Information elements Relation to BCSM Resources Logical R usage
Cause Or (CONDITION)	T Collision D Conflict R Conflict	T Collision D Conflict R Conflict Call O Conflict S Conflict BA Conflict	D Sharing C Sharing	Changing the actual behaviour of the feature from the expected one (Any Reason)	Duplication Redundancy Incorrect Order Inconsistency Vagueness Looping	Interaction on BCSM Control Conflict Resource Conflict
ALGORITHM	Applying Condition on the feature description	Applying Conditions on the feature description	Checking the D & C sharing among the packages of the features	Verification of the service specification against the properties	Completing the feature specification & applying the detection conditions	Comparing the Behavioural Specification of the feature against the standard

Table 3.1: Summary of the detection methods

SD: Service Description
SE: Service Environment
T: Trigger
R: resource
D: Data
O: Operation
C: Controi
BA: Basic Assumption
PS: Procedural Specification
BS: Behavioural Specification

3.4.8 DISCUSSION

Looking at the description of the methods, the following results are gained:

The Keck and Fuchun methods support the idea of filtering. However, the others are pure detection methods. The Keck, Peng, Lin and Kuisch methods are applied in the IN context. The Keck, Peng, Fuchun, Yasuki and Kuisch have some similarities in terms of processed information for the features. The Keck, Peng, Fuchun and Yasuki methods are similar in the applied algorithm. All of them check the feature description against some conditions. They are common in the applied criteria, however they use different conditions.

The differences of the described methods are:

The Lin method is different from the others, because it follows a formal verification method. The Fuchun and Yasuki methods are different in terms of the context. The Lin method is different from the others in describing the feature. The Lin method is different in applying the detection condition because it is not restricted to special conditions.

Among the detection category, there is an overwhelming number of formal methods and approaches using all kind of methods, while there is very little support for real implementation, especially for dealing with legacy services. In the resolution area, there are promising approaches, but many of them rely on explicit knowledge of the services under process, which is clearly a problem in a multi-provider environment.

The prevention approaches are long term solutions that require a suitable network infrastructure. Here, almost no short-term perspective is seen, especially if legacy services have to be covered. Management systems are still of modest number, although the complexity and size of the problem makes the mastering in this area a critical issue.

One of the important components necessary for managing feature interaction in telecommunication networks is defining and managing the information about the features. Fortunately, the past decade has provided much of the information technology

needed to create and manage this kind of information [11]. However, as yet no significant progress has been made in using this technology to manage information about telecommunication features and services. The other important component is an automatic tool that can be scalable to the industrial usage.

3.5 CHAPTER SUMMARY

In this chapter different ways of tackling the feature interaction problem were described. Classification of the possible solutions along with a survey was given. Some chosen methods, which had been deeply studied, were compared. In each case the feature description, context in which the method is applied, and the algorithm is described. Then, these methods were compared in terms of their strengths and drawbacks.

Chapter 4

CAUSES OF FEATURE INTERACTIONS

4.1 INTRODUCTION

Detecting interactions between two or more features involves an environment in which the actions of the features are watched to find out where the interaction occurs. For any method, the features must be described in terms of procedural, behavioural and/or working environment characteristics. According to the applied methodology, the type of necessary information, the combination of information and the applied detection algorithms on the features specification can be different.

We based our proposal on the analysis of the whole problem and existing solutions. In our approach we tried to avoid the restrictions of the existing methods and use their strong points as much as possible.

Filtering is seen as a method to cut down on the number of cases that are supposed to be tested by detection methodology. According to EURESCOM project [5] that has introduced the concept of the Service Interaction Handling Process (SIHP), filtering sub-process is done before detection sub-process and it is a required component in SIHP.

Based on our analysis, the necessary ingredients of a detection method are as follows:

1) Analysing the possible causes of interactions

2) Information about feature:

- Feature logic
- Feature environment

3) A modelling method/tool to describe the features

4) An algorithm for analysing these specifications

- Producing sufficient test cases
- Filtering the interaction-prone test cases
- Clarifying the conditions in which the interactions occur

Each of these components has its own importance. Increasing the amount of information about features leads to a more general method. Furthermore, good modelling template simplifies the application of the method and its reusability. Wide view of the causes of interactions and the effects of each lead to an efficient algorithm. Finally, an effective tool, which implements these techniques is necessary. The method must implement the idea of filtering to reduce the number of the test cases for detection process. It must use sufficient information about a feature and its environment.

The first step of our study is the realisation of the possible causes of feature interactions. For this, existing detection methods and benchmarks were studied. The benchmarks present classifications and instances of the problem. The causes of feature interactions can be classified into three main categories: assumption, resources, and operations. The following causes are defined and covered by our method. In this chapter, each cause of feature interaction is followed by the way it can be detected, which leads us automatically to the required information for feature description.

4.2 ASSUMPTIONS

The correct behaviour of a feature is based on pre-defined assumptions. These assumptions may include for instance a particular call processing, a special billing system, or even how customers perceive the network operations. If one of the necessary assumptions of the feature is not satisfied, it simply can not behave correctly. Some of the causes of interactions, which are violations of assumptions, are described in this section.

4.2.1 NAMING ASSUMPTIONS

Originally, a telephone number uniquely represented a telephone line. Also, each customer associated with a specific line, was forming a single entity and did not need to be further distinguished. However, new features enabling re-routing or providing multiple persons with personal identification assigned to a specific line have made the old model obsolete. Features that use "aliases" or additional names for accessing the same line or the same entity will violate the naming assumptions for a party. The violation of the naming assumption by a feature will cause disabling of another feature, which in turn relies on the naming assumption.

EXAMPLE

Let us consider the Originating Call Screening (OCS) and Call Forwarding (CF) features, as shown in Figure 4.1. OCS blocks the calls based on the dialled number. So, calls to a particular line are blocked only if the dialled number associated with the line is on the outgoing call screening list. However, CF connects to a line other than the one associated with the dialled number. If user2 forwards all incoming calls to user3, when user1 calls user2, even if user1 has user3 on the outgoing call screening list connections from user1

to the line identified by user3 will be established. In this case, CF violates the naming assumption and the OCS fails.

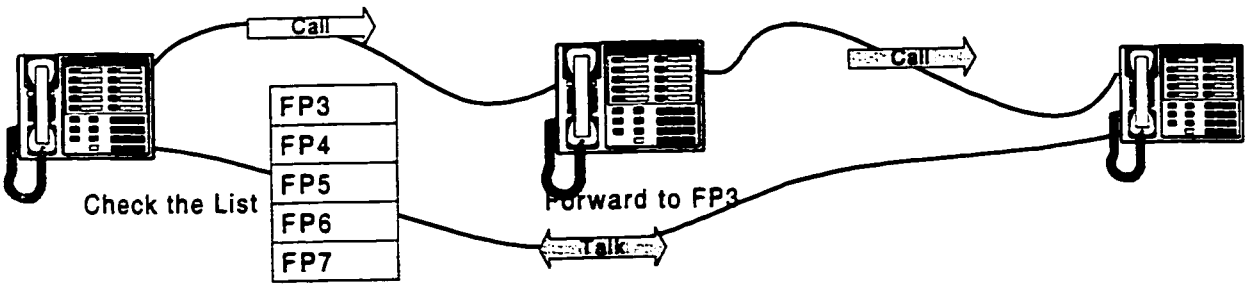


Figure 4.1: The Service Interaction between the CFU and OCS

HOW TO DETECT IT?

It is required to identify whether the feature is sensitive to naming assumption or not. If the naming assumption is important for this feature, it must be determined. Furthermore, it is necessary to know whether a line given to a subscriber relies basically on multiple number-one line assumption.

4.2.2 ADMINISTRATIVE DOMAIN

Administrative domain (AD) is a telephone network administered by a single organisation. A feature that is defined in one AD may not be usable from another AD.

EXAMPLE

Assume two features: Long Distance Calls (LDC) and Message Rate Charge (MRC) services, which are shown in Figure 4.2, have been put together. It is not clear whether a customer should be charged for the segments that have been successfully completed even if the call did not reach its final destination.

HOW TO DETECT IT?

The information about the administrative domain of the features is required. The compatibility of ADs of the features that are supposed to coexist must be checked.

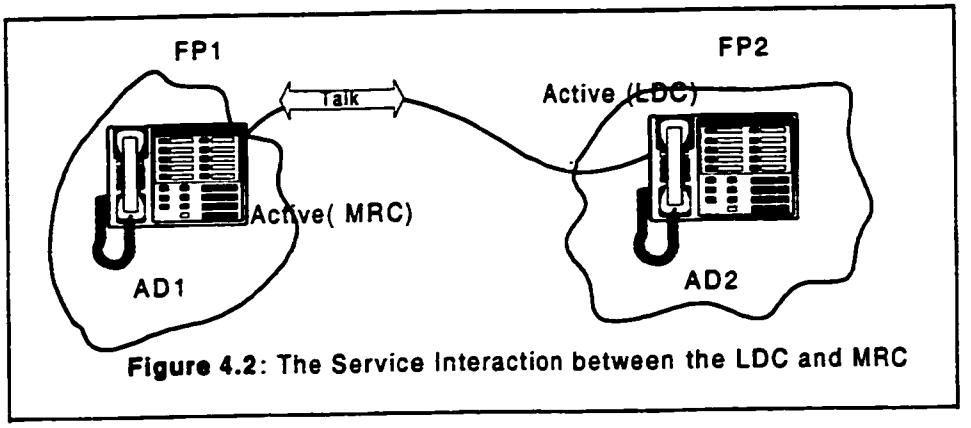


Figure 4.2: The Service Interaction between the LDC and MRC

4.2.3 DISTRIBUTED SUPPORT OF THE FEATURES

The IN distributes the support of the features. It means different features may be supported by different network components even when a single user uses the features. In this case, activating one feature in one component may prevent processing of the features in another component.

EXAMPLE

Let us consider two features: Operator Service (OS) and OCS, which are shown in Figure 4.3. If OS is handled in a remote switching, it does not have access to the feature subscription profile of every customer who wishes to use the OS. Therefore, a feature like OCS, which needs to check the outgoing call number, will fail since the OS does not have the screening list.

HOW TO DETECT IT?

The information about the supporting network components in which the feature resides is necessary. Using that information the compatibility of the supporting network components can be examined.

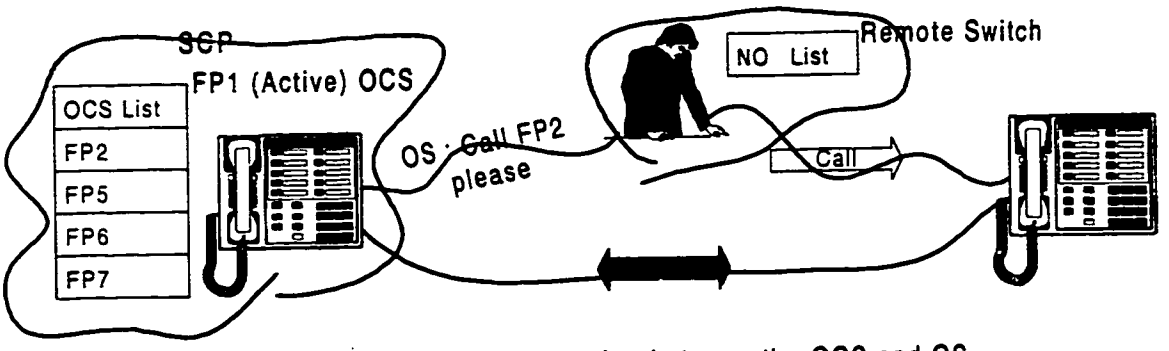


Figure 4.3: The Service Interaction between the OCS and OS

4.2.4 PERSONALISED INSTANTIATION

Many features allow the subscriber to assign values to some parameters before using the features. In many cases, a particular set of assignments can make two independent features collide with each other.

EXAMPLE

Let us assume two features Call Forwarding Unconditional (CFU) shown in Figure 4.4 have been activated for two subscribers FP2 and FP3. We know that a customer with CFU will redirect received calls to another number. The subscriber determines the desired destination number. Assume FP2 forwards all incoming calls to FP3. However, FP3 is also a subscriber of CFU and forwards all his calls to FP2. The result of any call from FP1 to FP2 will be an infinite loop between the FP2 and FP3.

NOTE: For a combination of some features more than one cause of interaction may exist. For example, CFU and CFU interact because of "Personal Instantiation" and also because of "Contradictory Operation" which will be described later in this chapter.

HOW TO DETECT IT?

It is possible to keep information about the feature, to show whether it has parameters to be instantiated by the user or not, and then checking the compatibility of this parameter for some features will detect the interaction of this type.

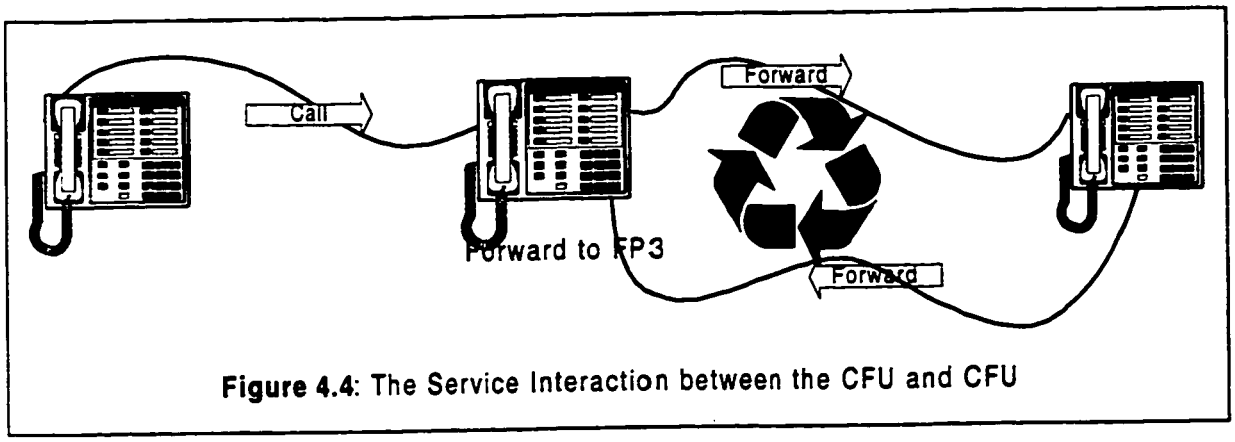


Figure 4.4: The Service Interaction between the CFU and CFU

4.2.5 SIGNALLING PROTOCOL

A feature can use the status of the line as an information item. For example, a busy signal means the destination line is in use, a ring-back means the line is available. Many features now can make a line appear to be available while it is in use. Interaction will occur when one feature needs to know the status of the other party, but from the received signal it can not exactly recognise what the status is.

EXAMPLE

Consider two features, Call Waiting (CW) and Automatic Call Back (ACB) which are shown in Figure 4.5. ACB is triggered if the called party is busy. Since a line with CW appears to be idle to a caller, even if it is actually busy, they will interact. Assume that FP2 has activated CW feature and FP3 has ACB feature. Will ACB work for FP3 if FP2 is busy talking to FP1 while showing the line is idle by his CW feature?

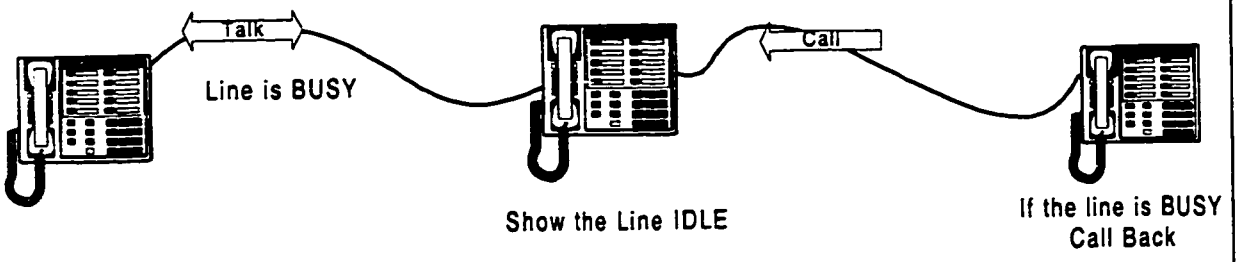


Figure 4.5: The Service Interaction between the CW and ACB

HOW TO DETECT IT?

Status of the line is an information the features may use. By having the information about whether a feature is going to change and/or use a special data and further checking it to see whether that data is common for both features, the interaction can be detected. In the above example, CW writes the data (Status of the line) and ACB reads the same data.

4.3 RESOURCES

The resource problem, which is typical in large distributed systems, is another cause of interactions in IN. Each feature needs some resources in order to perform its required operations. Resources can be of any kind. Data, physical resources, or any other kind of

resources, which are necessary for the feature operation are critical for the correct behaviour of the feature. The activation of one feature can make the resources of the other one unavailable. Some examples of this kind of resources are studied in this section.

4.3.1 LIMITED CPE SIGNALLING CAPABILITIES

The set of signals that may be sent from most current Customer Premise Equipment (CPE) to a switch is limited to *, #, the ten digits 0-9, flash-hook, and disconnect. Because of the limited number of signals, one signal may be used in different situation by different meanings. Interaction may arise when the interpretation of a signal is ambiguous.

EXAMPLE

Let us consider CW and TWC services, which are shown in Figure 4.6. A busy party issues a flash-hook signal that is generated by hanging up briefly or depressing a 'tap' button. That means the start of adding a third party to an established call in TWC feature. The same signal means to accept a connection attempt from a new caller while putting the current conversation on hold in CW feature.

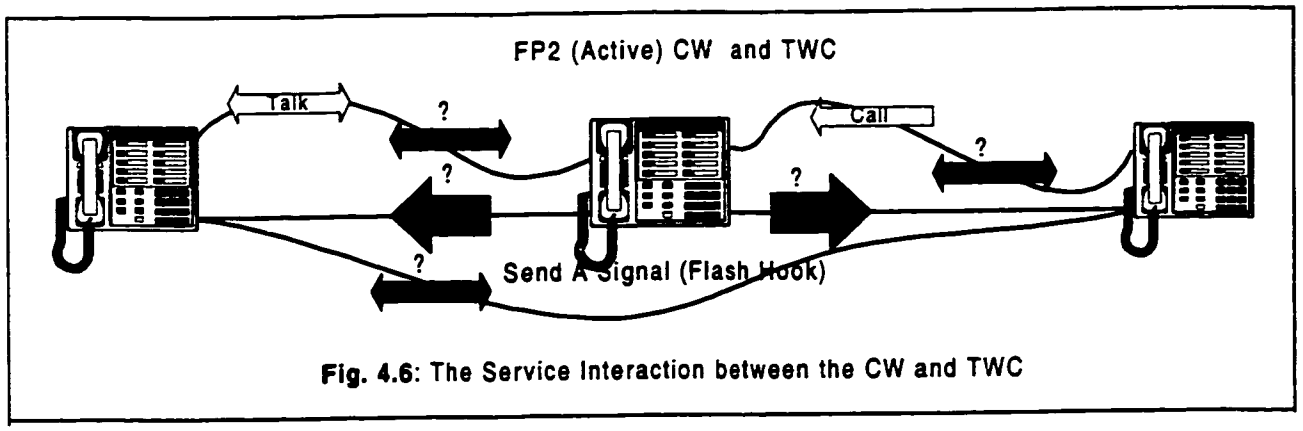


Fig. 4.6: The Service Interaction between the CW and TWC

HOW TO DETECT IT?

It can be detected by keeping the information about the number of CPE signalling that are supported by the subscriber equipment, and the number of required CPE signal by each feature and their type. In the case of activation of some features at the same time that use the same signal, the limit of using the signal must be verified. If the limit is violated, features will interact.

4.3.2 LIMITED FUNCTIONALITY FOR COMMUNICATION

One of the problems related to the limited functionality, is the limited information that a SCP can send to a switching element via a message. The protocol for transporting queries to SCPs supports a limited number of queries per call, but a collection of features may require more. This may be the cause of interaction of two or more features since a feature may need to launch a query while all the allowed queries are already launched for another feature.

EXAMPLE

Let us consider OCS and Area Number Calling (ANC) features that are shown in Figure 4.7. Both of these features need to send a query to the SCP. If the switching element imposes a restriction on the number of queries per call, after launching a query for the OCS feature, the component will not be able to launch another query for the ANC feature.

HOW TO DETECT IT?

It can be detected by keeping the information about the limitation of the switch on the number of queries per call and the number of necessary queries for each feature and

checking the limit in the case of having a combination of features that are requesting queries per call.

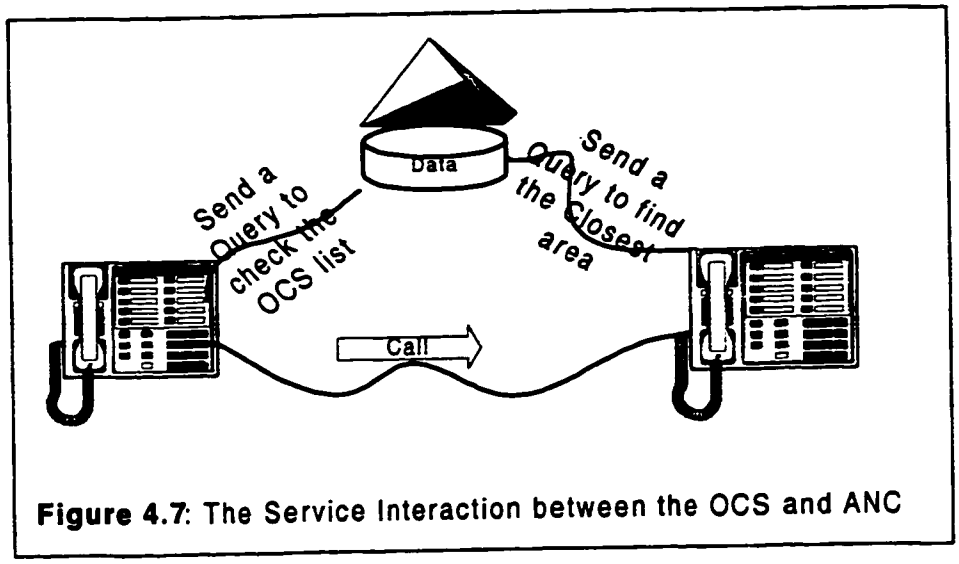


Figure 4.7: The Service Interaction between the OCS and ANC

4.3.3 DATA AVAILABILITY

Some features may assume that certain types of data are widely available, while other features such as Calling Number Delivery (CND) may use the same data as a private one. Features can not work properly unless they are able to obtain the data needed by their functionality.

EXAMPLE

Let us consider combination of the CND and Unlisted Number (UN) services that are shown in Figure 4.8. UN is a directory service feature designed to allow a subscriber to keep the number private. CND is a call-processing feature that delivers the directory number of the calling party to the customer premises during the ringing cycle.

When a customer with UN places a call to another customer B with CND, it will loose privacy or the other side will get no information.

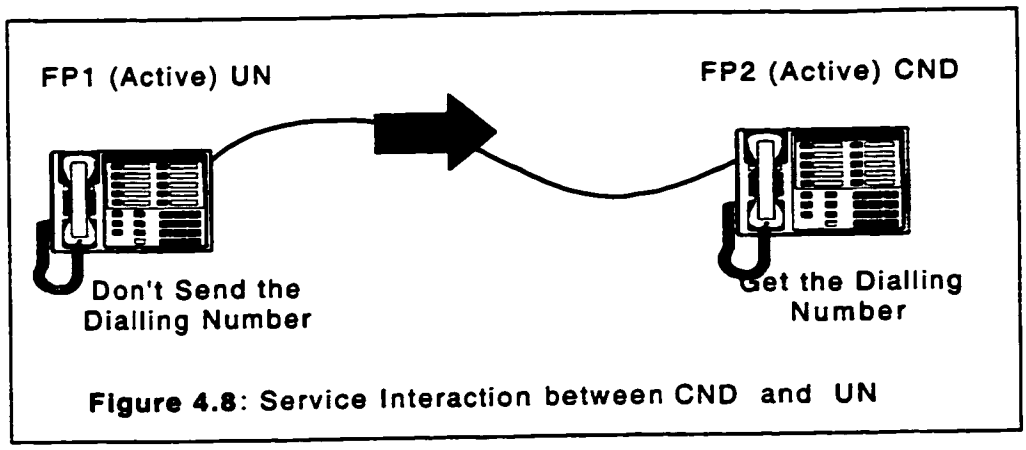


Figure 4.8: Service Interaction between CND and UN

HOW TO DETECT IT?

Each resource used by the feature operation can be used in one of two modes: public or private. Defining the operation of the feature, it is required to identify whether this resource is used in public mode or private mode. Having two activated features that are using the resources in two modes while at least one of them is in the private mode will certainly lead to an interaction.

4.4 OPERATIONS

Operations of the features can be the cause of interaction in different ways. They can simply be contradictory (like read and protect-from-read), lead to the race condition situation (when two features want to manipulate the same data) or violate the assumption of each other. Some of the causes of interactions, which are related to operations, are described in this section.

4.4.1 CALL CONTROL

Call control means the ability of a feature to manipulate (accept, refuse, terminate, or change the status of) a call. If two features want to control the call, they can not properly work together.

EXAMPLE

Automatic Recall (ARC) feature switches the role of a user from called to caller which prevents the user from activating the terminating features such as Terminating Call Screening (TCS). Another example is the combination of 911 and TWC, which is shown in Figure 4.9. Both 911 and TWC features want to control the call. TWC can put the called party on hold to call third party. If the called party is 911 which wants the absolute control of the call, TWC will not be able to put 911 on hold and will fail.

HOW TO DETECT IT?

In description of the feature, when it comes to the operation level, it is necessary to identify whether this operation wants the complete control of the call or not. Two operations are described as call-control and they are incompatible.

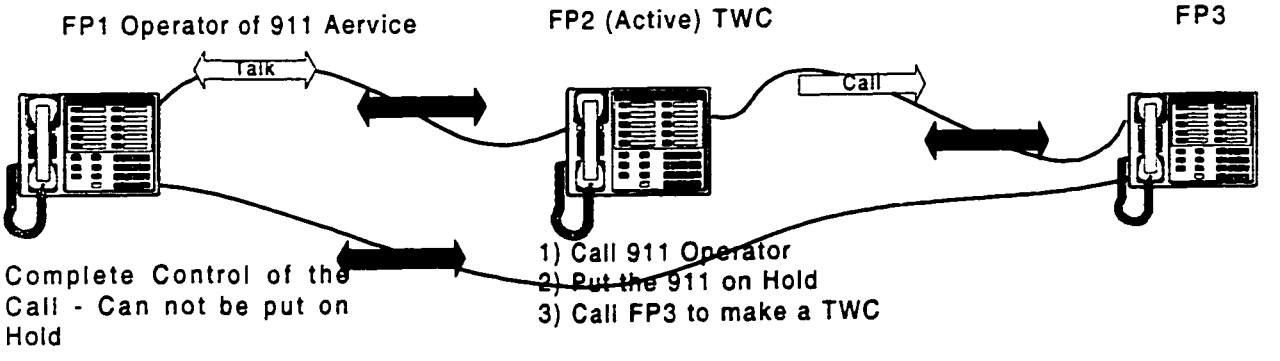


Figure 4.9: The Service Interaction between the 911and TWC

4.4.2 TRIGGERING COLLISION

A feature is activated under certain condition named triggering conditions. During the basic call processing procedure, when these conditions are satisfied, the system will invoke the feature. When two features have the same triggering conditions, this will result in an ambiguous situation.

EXAMPLE

Consider two features, CW and Answer Call (AC) that are shown in Figure 4.10. When the O-BCSM is in the O-Active and the TBCSM is in the called-Party-Busy, both features can be activated. Assume the situation a call attempts to reach a busy line, CW generates a call waiting tone to alert the called party, whereas AC connects the calling party to an answering service. Suppose that FP2 is a subscriber of both features. If FP2 is busy talking to FP1 when the FP3 calls him, should FP2 receive a call-waiting tone to be informed somebody is waiting for him or should the second call be directed to the answering service?

HOW TO DETECT IT?

By keeping the information about the triggering conditions of the features and check the conflict between them. For each detection point in basic call process, the information about the special detection point in which the feature will be activated is necessary. For the detection point an id and the situation of the other formal participant are required.

4.4.3 ACTIVATING PRIORITY

If an order is to be considered, when two features have the same triggering conditions, there is an activation priority. If there is no priority between two features or, their priority orders are not compatible, then an interaction will occur between them.

EXAMPLE

Consider the same example described for CW and AC that is shown in Figure 4.10. If the subscriber has assigned a priority for each feature, the problem may be solved. But if there is not any assigned priority, the result is an ambiguous situation in which one or both of the features may fail.

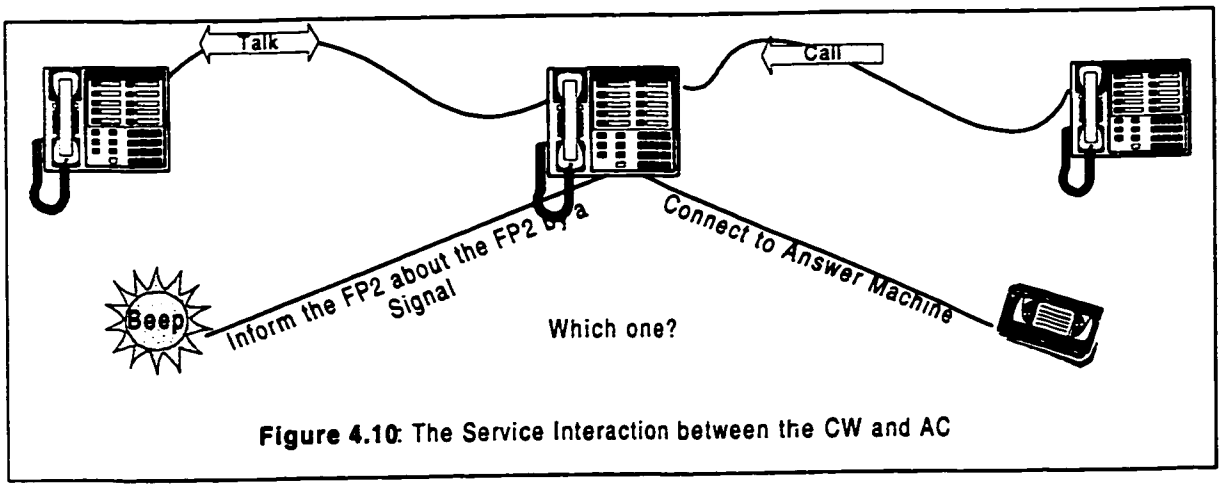


Figure 4.10: The Service Interaction between the CW and AC

HOW TO DETECT IT?

It is worth to keep the information about the activation priority for the features. In the case of having no priority, for the features, which are activated in the same detection point, there will be interaction.

4.4.4 NON-ATOMIC OPERATIONS

One of the consequences of the distributed service logic is that provisioning could be non-atomic operations. For example, AIN based 900 call screening gives additional features such as screening only during certain period and screening with PIN override. In this case, updates must be made to two network components, the switching elements and the SCP. Updating is a non-atomic operation and will prevent the processing of the other operations.

EXAMPLE

Let us consider AIN-based services and POTS. According to the previous explanation, the provisioning of the AIN-based services will disable the POTS.

HOW TO DETECT IT?

For generalisation we define two modes of operations – atomic and non-Atomic. In the feature description when the operation of the feature is defined, it is required to make clear whether the operation is atomic or non-atomic.

4.4.5 TIMING AND RACE CONDITION

Timing is always critical in distributed systems. The problem is even worse in telephone services because it deals with external operation, which comes from the human interactions. If the external events are to be taken into account, unfortunately there is no way to detect the interaction. If we neglect the human behaviour, the timing and race condition between two features may cause interaction where they have common operations or common objects under manipulation.

EXAMPLE

Let us consider two features: CW and TWC that are shown in Figure 4.11. Suppose that during a telephone conversation between FP2 and FP1, an incoming call from C has arrived at the switching element for line of FP2 and triggered the CW feature that FP2 has subscribed. However, before being alerted by the call-waiting tone, FP2 has flashed the hook, intending to initiate a TWC. It is not clear, whether the flash hook should be considered the response for CW or an initiation signal for TWC.

HOW TO DETECT IT?

Neglecting the human action in the case of common operation on common objects under manipulation the situation of race condition is evident.

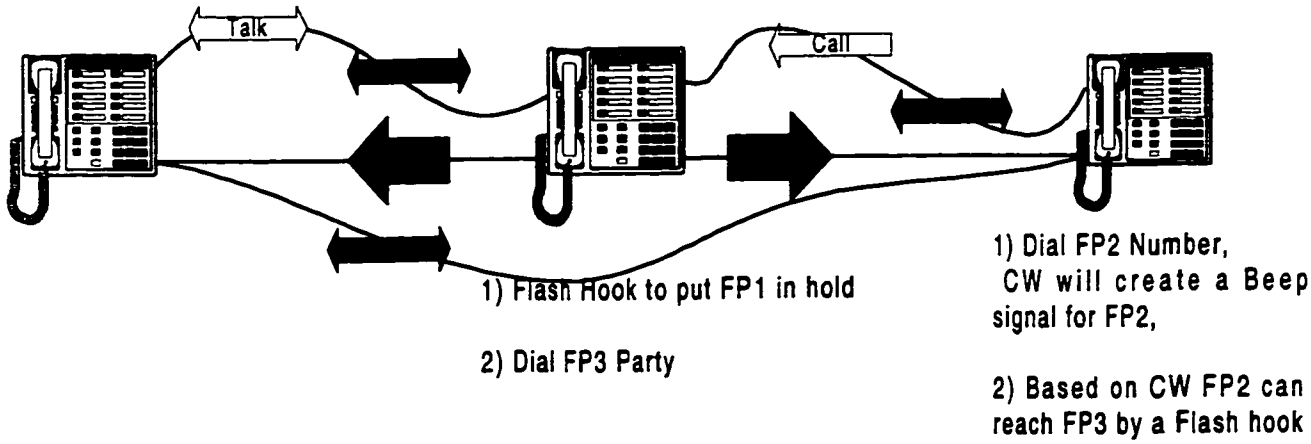


Figure 4.11: The Service Interaction between the CW and TWC

4.4.6 COMMON OBJECT MANIPULATION

European benchmark defines the notion of a feature affecting a telecommunication system as feature logic manipulating certain objects. The objects are entities and concepts

in a telecommunications infrastructure that can be manipulated by feature logic. Manipulation refers to various ways in which feature logic can control or influence an object.

In our design, the European concept and classification is used at a lower level, in the feature operation level to make abstraction of the feature action. During BCSM each feature needs a certain condition (Triggering Condition) in order to be activated. After activation whatever it does, can be considered as an operation (an Object Manipulation). The object and their possible manipulation are shown in Table 4.1.

When two features have common operation, they may interact. The operations of the features that are to coexist may be contradictory. If one of the features want to reserve a physical resource and another feature wants to protect the physical resource from being used; these features will not be able to work properly at the same time.

We have defined the notion of incompatible operations. Some of them are as follows:

"Control" # "Control"

"Read" # "Write"

"Read" # "Protect-From-Read"

"Write" # "Protect-From-Write"

"Use" # "Protect-From-Use"

"Release" # "Protect-From-Release"

"Forward" # "Forward"

"Reserve" # "Protect-From-Reserve"

"Ring" # "Protect-From-Ring"

OBJECT	MANIPULATION
Data Element	Read, Write, Protect-from-read, Protect-from-write
Signal/Event/Query	Send, Receive
Program	Control
Session	Create, Terminate, Add party, Drop Party, Add leg, Drop Leg, Join Leg, Split Leg, Change Owner
Physical Resource	Reserve, Use, Release, Protect-from-Reserve, Protect-from-Use, Protect-from- Release
Sequencing Structure	Progress, Alternative-next-step, Halt-progress
Branching Structure	Choice
Looping Structure	Termination

Table 4.1: Different Operations or Object Manipulation

Each object is used in one of the possible modes: public or private.

Each manipulation has one of the possible modes: atomic or non-atomic.

HOW TO DETECT IT?

This cause can be detected by keeping the information about the operations of the features, type of objects and type of manipulations. Referring to the table of incompatible operation to see whether the operations can co-exist the compatibility of the combinations of some features can be checked.

4.5 CHAPTER SUMMARY

In this chapter, our approach was introduced briefly. Then, the first step of our study, the causes of interactions, was explained. To find the causes of an interaction, some of the existing methods were studied, compared, and analysed in Chapter 3. One of the main goals of the comparison was the investigation of all the causes of the interactions considered in each method and the sufficient conditions for the detection algorithm in each case. For each case one example and the way of detecting that cause, were presented. These causes are used as the starting point to define the required information for describing the features.

Chapter 5

FEATURE DESCRIPTION

5.1 INTRODUCTION

In this chapter, using the studied causes in Chapter 4, the necessary information for the feature description will be described. To detect each cause of the interaction, a special technique must be used. To devise a technique for detecting each cause, some information about the feature is required. These information items make up the feature description. Using these descriptions a model will be devised. The goal of the modelling is to show the relations between the system components and systemise the application of filtering and detection algorithms. The relations of causes, feature descriptions, object-oriented model and the detection process are shown in Figure 5.1.

We choose the object-oriented paradigm for our model. Using the advantages of this paradigm, extending the description model as well as detection method according to the IN evolution will be straightforward.

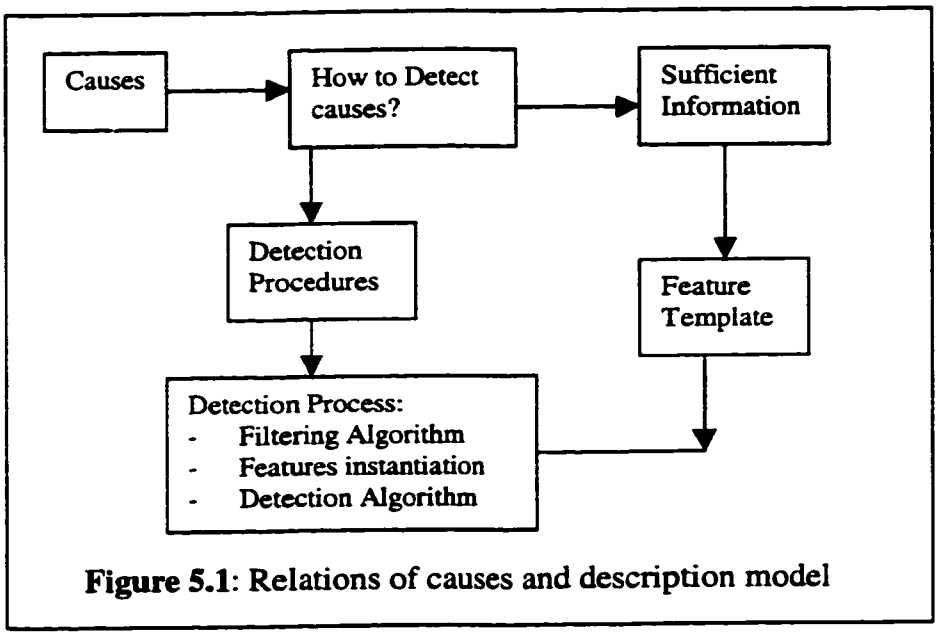


Figure 5.1: Relations of causes and description model

5.2 NECESSARY INFORMATION

In feature description, there are some assumptions about the subscriber line and the feature as the static specifications. The dynamic behaviour of the features is seen through its formal participants. For each formal participant, the special BCSM (Originating or Terminating) associated with the special detection point in which the feature begins its operation must be described. Furthermore, for each Detection Point (DP) the operation of the feature is described along with the required object and the type of activation.

The necessary information derived from the causes of feature interactions described in Chapter 4 is explained in this chapter. These information items are generalised to cover the coming features, coming feature interactions and new causes.

Basic Call #1

Subscriber name: A symbolic name

Number of available CPE signal: An integer number

Number of query per call: An integer number

Administrative domain: A symbolic name

Source address: An integer number

Destination address: An integer number

Paying party: An integer number

Feature Form #2

Name: A symbolic name

Activating priority: An Integer number

Supporting network component: Component name

Administrative domain: A symbolic name

Naming assumption: "1L-1N", "1L-MN" or any other possible types

Special variable: The variable name if applicable

Number of formal participant: An integer number

Formal participant list: List of FPs (Object #3)

Formal Participant Form #3

Formal participant ID#: An integer number
Type: "Active", "Passive"
User Status: "Caller", "Called"
Line status: "Busy", "Idle"
BCSM: Originating (Object #4)
BCSM: Terminating (Object #4)

BCSM Form #4

Type of BCSM: "Originating", "Terminating"
Detection point: (Object #5)
Operation: (Object #6)

Triggering Form #5

ID: A symbolic identifier
Triggering type: A string name
Triggering criteria: A list of the other FPs situations

Operation #6

Name: A symbolic name
Object: Can be chosen from a list
Manipulation: Can be chosen from a list

5.3 INTRODUCTION TO THE OBJECT ORIENTED PARADIGM

An object-oriented approach to programming was first proposed in 1967. However, it is only since the late 1980s that this paradigm been widely adopted in industry. To support object-oriented programming, an object-oriented development approach must be adopted. This means expressing the system requirements using an object model, designing using an object-oriented approach and developing the system in an object-oriented programming language such as C++ or JAVA.

Object models developed during requirements analysis may be used to represent both system data and its processing. They are useful for showing how entities in the system may be classified and composed of other entities. These objects should not include details of the individual objects in the system. This is a design consideration. Rather, they should model classes of objects representing real-world entities. An object class is an abstraction over a set of objects, which identifies common attributes (as in the semantic data model) and operations, which are provided by each object.

Various types of object models can be produced showing how object classes are related to each other, how objects are aggregated to form other objects, how objects use or inherit the services provided by the other objects, how one object is composed of some other objects and so on. For some classes of system, object models are the natural ways of reflecting the real-world entities that are manipulated by the system. This is particularly true where the system is concerned with the processing of information about concrete entities such as subscriber, features and its participants. An object class is similar to an abstract data type that can be used as a basis for creating objects, which in turn are executable entities with the attributes and services of the object class [42].

5.3.1 INHERITANCE MODEL

For each class some attributes and functions are defined. When objects are created, they inherit the attributes and operations of their class. Inheritance tree shows how objects inherit attributes and services from their super class. Assume two classes have been defined for Basic-Connection and Feature. There are some attributes and services, which are common to all the Basic-Connection including a connection controlled by a Feature.

In this case the supper class Basic-Connection is defined to keep all the attributes and services for basic call and subclass Feature inherits all the attributes and common services of basic call and then adds particular attributes specific for the programmers.

5.3.2 OBJECT AGGREGATION

After realising different components of the system as the objects, the relations between them must be considered. In practice, objects are organised into an aggregation structure that shows how one object is composed of a number of other objects. The aggregation relationship between objects is a static relationship. When implemented, objects, which are parts of another object may be implemented as sub-objects. Their definition may be included in the definition of the object of which they are a part. An example of aggregation is shown in Figure 5.2. Composition is the usage of the object classes as the type. Whenever the object is instantiated, the types (object classes) of its attributes are also instantiated.

Aggregation is a kind of composition in which the associated object classes considering the actual needs, are dynamically used. Assume the object classes are user of telephone line, BCSM, detection point and operation. Each user is defined by its associated

components. Simple definition says: "Each user HAS BCSM and each BCSM HAS detection point and operation.

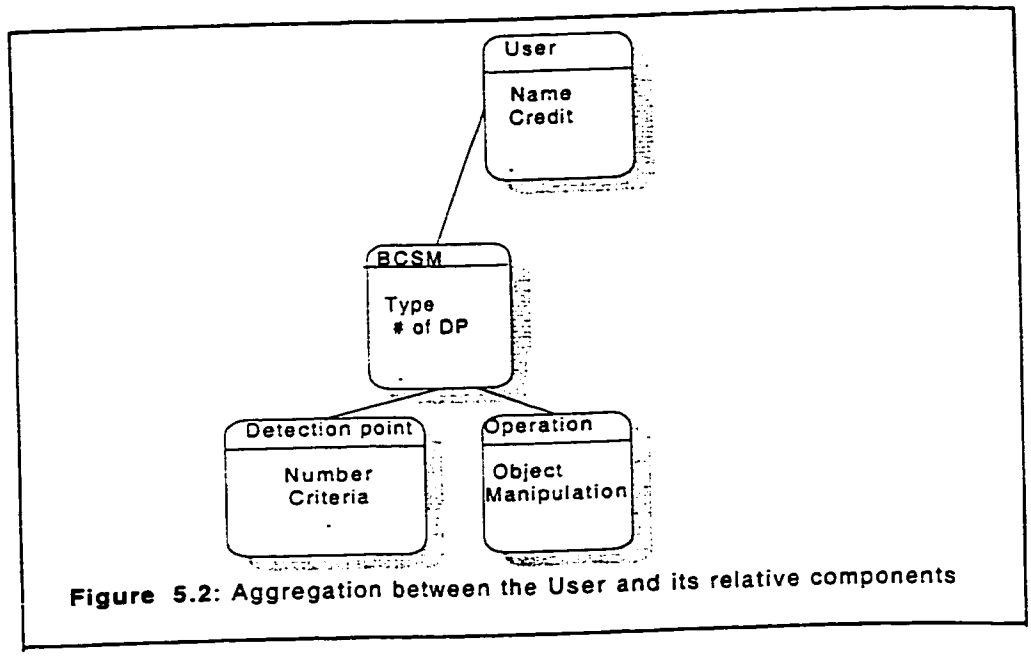


Figure 5.2: Aggregation between the User and its relative components

5.4 A TEMPLATE FOR FEATURE SPECIFICATION

The most important point in our model is the relation of the different components or object classes. These relations reflect the constraints and flow of control, which exist between these components in the actual environment. In this model each object consists of some attributes to show the characteristics of that object and its behaviour.

5.4.1 RELATION OF THE SUBSCRIBER and FEATURE CLASSES

The first object class of our model is the subscriber or the basic call for the user who may have subscribed for some features. The required information for this subscriber includes the attributes:

- Subscriber Name
- Source Address
- Destination Address
- Paying Party
- Number of CPE signal available
- Number of query per call
- Administrative Domain

The relations of the subscriber and feature classes are shown in Figure 5.3. A feature inherits the attributes of the subscriber since it belongs to that subscriber and it is going to work under its limitation. Therefore, the feature inherits all the variables of the basic call. The object class feature describes the feature and its information items. Each feature has some description parameters to be set and a list of formal participants to play the feature action including:

- Name
- Activating priority
- Supporting network component
- Administrative domain
- Naming assumption
- Special variables
- Number of formal participant
- Formal Participant List

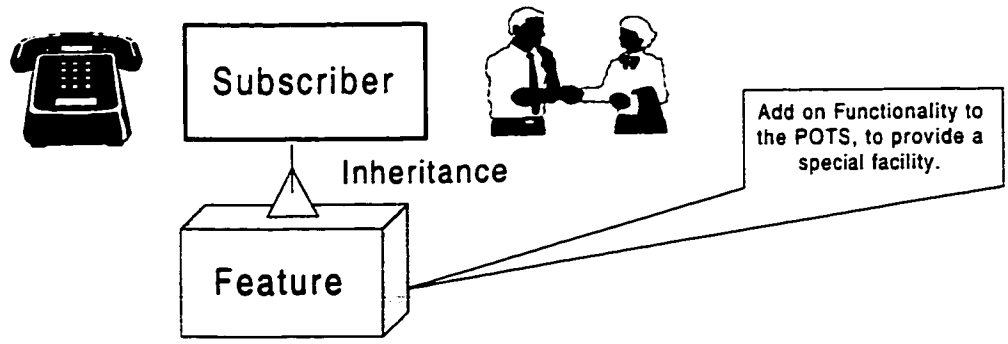


Figure 5.3: The relation of the Subscriber and the Feature Class

5.4.2 RELATION OF THE FEATURE and FORMAL PARTICIPANT CLASSES

The next object class belongs to the formal participant. These objects are the actors for the feature scenario. In other words, the behaviour of the feature is described through its formal participants. They are the people in the real life that do the operations defined in the feature scene. Considering the previous explanations about the aggregation relation between the object classes, we can simply observe that the feature class is related to the formal participant class using the aggregation mechanism. The relation between the feature and formal participant classes is shown in Figure 5.4. Depending on the feature description, the numbers of formal participants for the features are various.

For each formal participant besides the identification, the status of that formal participant and the status of the line in the time of feature activation are important information items.

Since the situation for the active feature is supposed to be precisely described, for each formal participant the BCSMs are described.

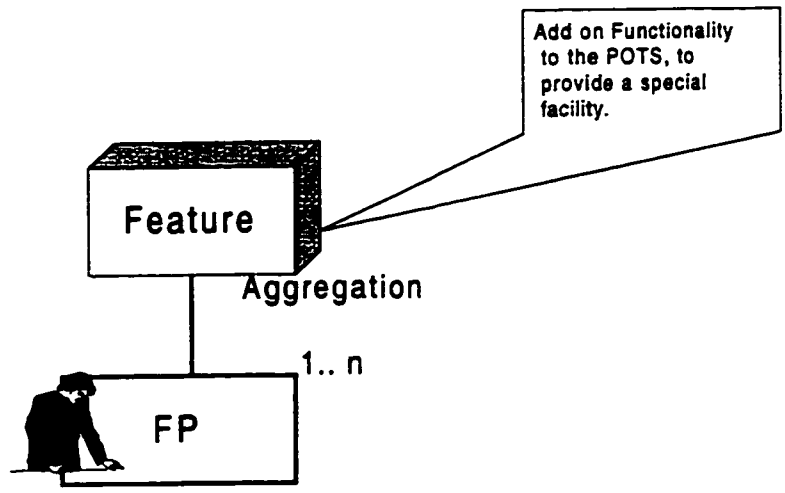


Figure 5.4: The relation of the Feature and Its Formal Participants

- Formal Participant ID
- Type: "Active party", "Passive"
- User Status: "Caller", "Called"
- Line Status: "Busy", "Idle"
- BCSM: Originating BCSM, Terminating BCSM, None
- BCSM: Originating BCSM, Terminating BCSM, None

5.4.3 RELATION OF THE FORMAL PARTICIPANT and BCSM CLASSES

Another object class is the BCSM class. As described in the second chapter, and based on the ITU-T recommendation set for the IN, BCSM is a finite state machine for presenting the possible states and flow of control for the duration of a call. Each BCSM contains the important points named Point in Call (PIC) or DP, which can be armed in order to notify

an IN service of reaching a special DP. When the DP is met, the operation of the feature is done. There are two kinds of BCSM: Originating BCSM, which deals with the actions and DPs for the caller side and Terminating BCSM, which deals with the actions and DPs for the called side.

Depending on each formal participant, the feature description may have Originating BCSM, Terminating BCSM or both sides. For example in the CFU feature, the first formal participant is the originating part, the second one is the terminating and originating part at the same time and the third participant is the terminating part.

Call Forwarding Unconditional		
FP1	FP2	FP3
OBCSM → TBCSM	OBCSM → TBCSM	

In each BCSM, one of the important points to be considered as an information item is the detection point in which the normal processing of the call is changed to the special operation/event. The other important part of the BCSM is the event or operation that is to be done at each detection point. Actually, the complete task of each feature is the collection of these operations. The relation of the FP and BCSM is shown in Figure 5.5.

- Type of BCSM: "Originating", "Terminating", "None"
- Detection point
- Operation

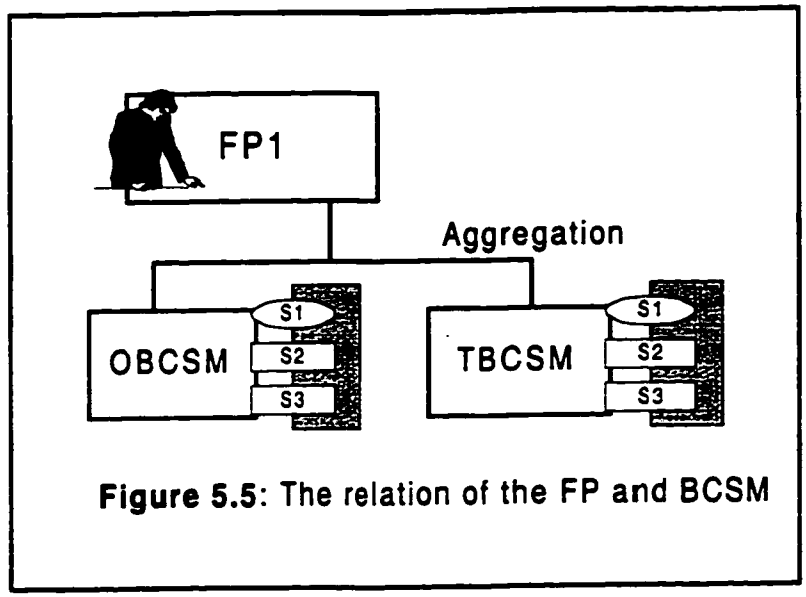


Figure 5.5: The relation of the FP and BCSM

5.4.4 RELATIONS OF the BCSM, DETECTION POINT AND OPERATION CLASSES

Another object class is the DP class. It is necessary to describe the detection point number, the ID of the DP, the triggering type and the triggering condition. Triggering criteria refer to the situation of the other formal participants during the feature activation.

- ID
- Triggering Type
- Triggering Criteria

To describe an operation, it is suitable to describe it as an object manipulation. Here the operations, objects, manipulations, and results are simply literal. In this case, there is no concern of the dynamic aspects of the running two features together. Therefore, the last object class in our mode has been defined to describe the operation along with its associated attributes. It is sufficient to describe the action of an operation/event in a special detection point.

Figure 5.6 shows the relation between the BCSM object and DP and OPERATION as the composition relation. Furthermore, the operation consists of the object classes OBJECT and MANIPULATION with the composition relationship. The attributes of the operation class are:

- Name
- Object
- Manipulation

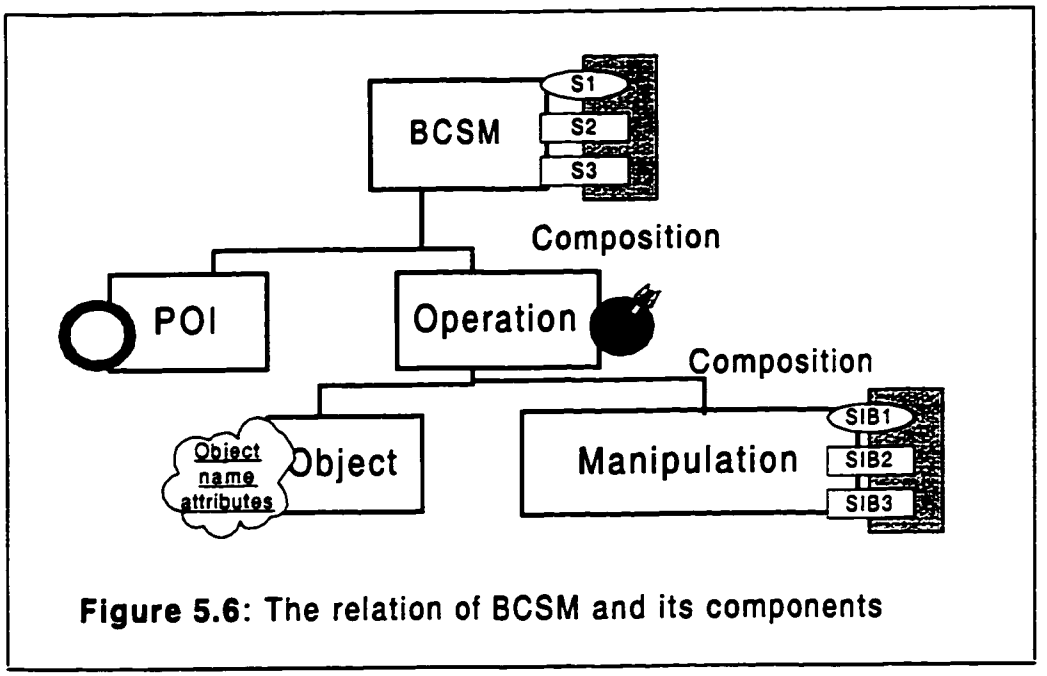


Figure 5.6: The relation of BCSM and its components

The attributes of the OBJECT class are as follows:

- Name
- Mode

And finally the attributes of the MANIPULATION Class are as follows:

- Name
- Mode

5.4.5 THE ENTIRE SYSTEM

The entire model consists of all the described object classes in the previous sections and their relations. The classes are "SUBSCRIBER", "FEATURE", "FORMAL PARTICIPANT", "BCSM", "DETECTION POINT" and "OPERATION". As the summary of previous explanations, "FEATURE" inherits the attributes of "SUBSCRIBER". "FEATURE" behaviour is described using its "FORMAL PARTICIPANTS". The role of each "FORMAL PARTICIPANT" is described using the necessary "BCSM". Finally in each "BCSM" the "DETECTION POINT" and "OPERATION" are considered. The entire model is shown in Figure 5.7.

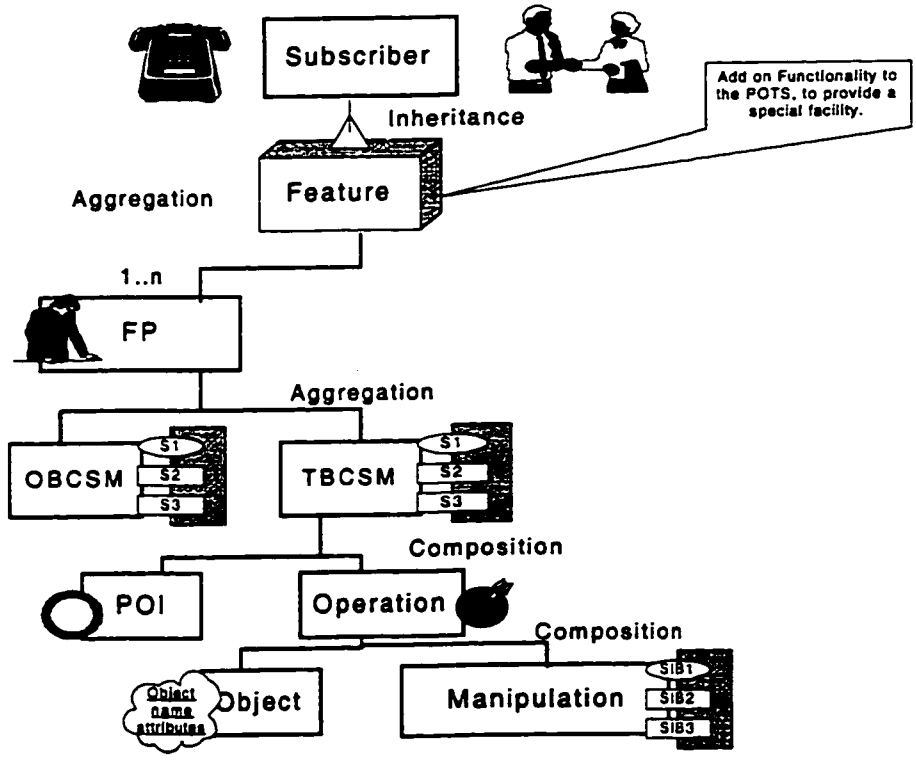


Figure 5.7: The entire system and its associated components

5.5 CFU - EXAMPLE

Call Forwarding Unconditional (CFU) is a feature that allows its subscriber to forward his/her calls to a predefined destination. Figure 5.8 depicts an instantiated feature for the CFU feature. The subscriber class has been described using its necessary attributes. The feature class has been described using its name, activating priority, supporting network, administrative domain, naming assumption and the number of its formal participant. Naming assumption for the CFU is one line-multiple number, since the logic of this feature is to identify a second number for the subscriber line and forward the incoming calls to that number. The first formal participant is a caller without any active feature, so its BCSM has zero detection point. The second formal participant is actually the subscriber of CFU. It has the called status and active mode. Its role is described using two BCSM. In the terminating BCSM it has one detection point in which it initiates another call to the third party. The third formal participant is the receiver of the call. It has the called status and is passive, so its BCSM has zero detection point.

5.6 CHAPTER SUMMARY

In this chapter the modelling of information about the features and subscriber has been introduced. The required information for feature specification was explained. A brief overview of object oriented methodology was followed by our model in order to describe the system components and their relations. An example of user feature was described in details in our model.

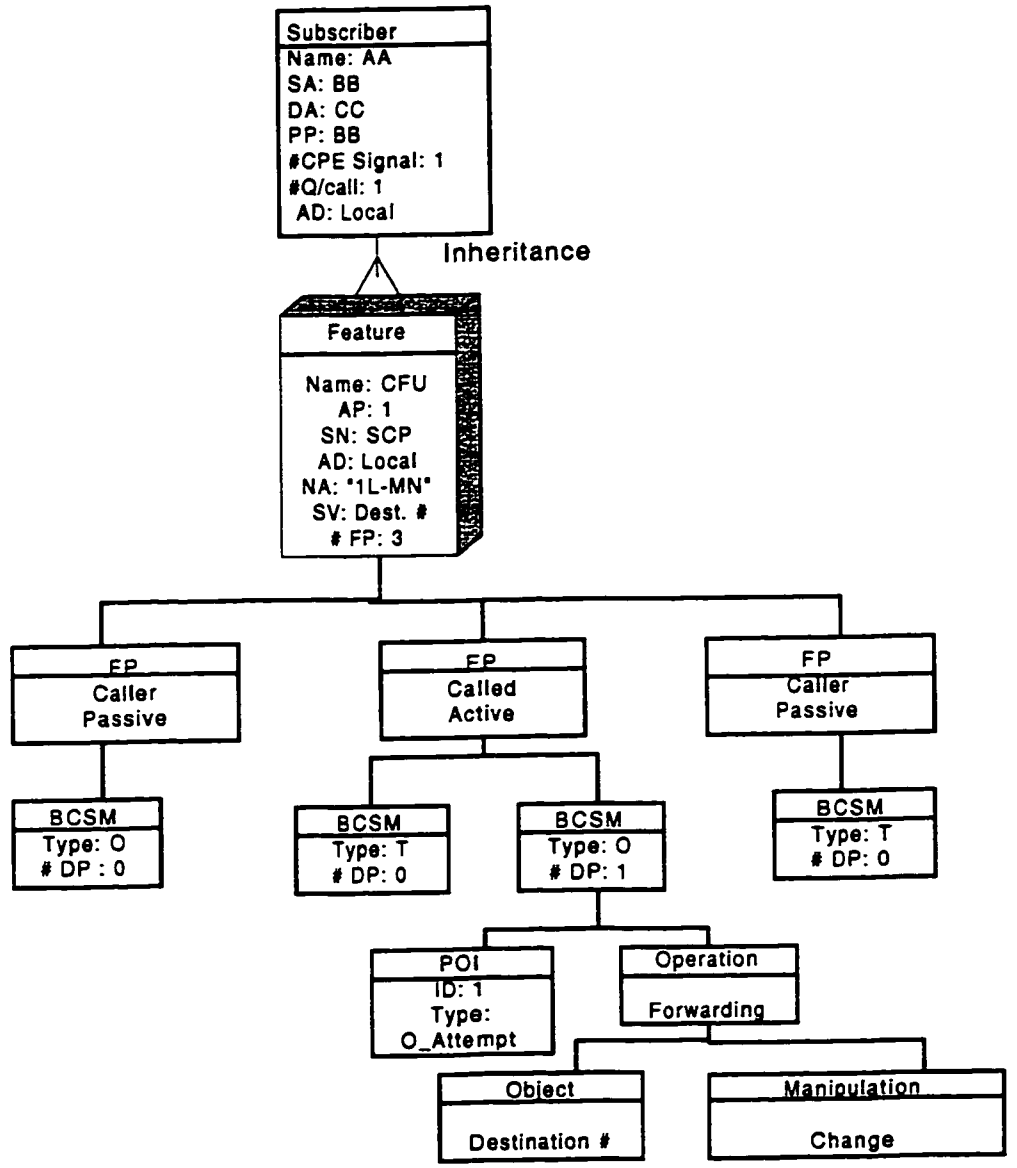


Figure 5.8: The CFU specification

Chapter 6

DETECTION TECHNIQUE

6.1 INTRODUCTION

The main part of our approach is the detection technique, which consists of three parts: filtering algorithm, feature instantiation and detection algorithm. Figure. 6.1 shows the whole detection process. The filtering component is responsible for determining the interaction-prone call scenarios. It uses the features specifications described using the description model that was explained in Chapter 5. It creates the list of formal participant combinations and reduces this list by applying pre-defined conditions.

The items of this list are used for features instantiation in order to provide the list of Actual Participants (AP). The detection component uses the APs list and the features

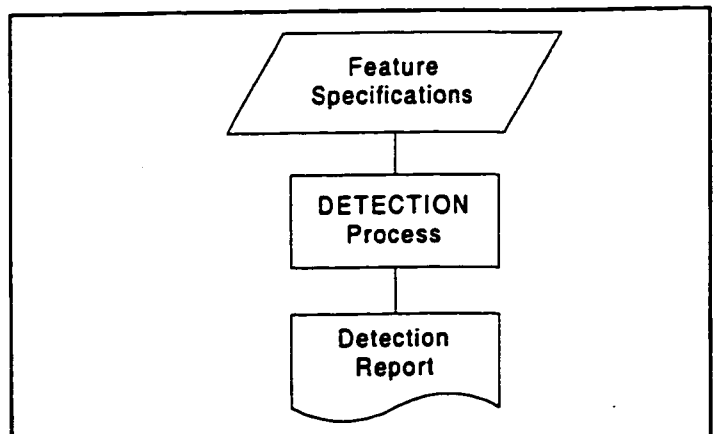


Figure 6.1: The Feature Interaction Detection System

specifications in the object-oriented template to detect the interactions between the given features.

6.2 FILTERING TECHNIQUE

The idea of filtering the interaction-prone scenarios before applying any detection method was proposed in the EURESCOM project [5] and been worked on by Keck from University of Stuttgart [29].

According to this idea, it is just wasting of time and resources to process all the possible scenarios between given features in order to detect the interactions. Another drawback is that processing all the possible combinations between the features under consideration may lead to a state explosion. To have an efficient detection method, the combination of filtering technique with any other detection method is required. The following notations are used in the rest of this chapter:

- Combination: It is a collection of FPs in any possible order.
- Call Scenario: A collection of FPs than has the semantic of a call /service.
- Interaction-prone Scenario: A call scenario, which is interaction prone.

Assume there are two features, one feature has n1 Formal Participants (FP) such as CW that has three FPs, first caller part, called part, and second caller part named 1, 2, and 3 respectively. The other feature has n2 FP such as TWC that has three FPs called 4, 5, and 6.

There are totally $n = n1 + n2$ FPs, e.g. CW and TWC have six FPs together.

The numbers of possible combinations of FPs is n^n.

For example for the combination of the CW and TWC n=n1+n2=6, the number of possible combinations is:

$$(\text{Number of FPs})^{(\text{Number of FPs})} : \; 6^6 = 46656 : \text{FPs combinations}$$

Processing all these combination is costly and time consuming.

Since the above number is huge for showing the detail of the example, we have chosen the number to be 3, to clarify the detail:

For n= 3 there are: $3^3 = 27$ FPs Combination as follows:

111 222 333 112 221 331 121 212 313 211 122 133 113 223 332

131 232 323 311 322 233 122 211 311 123 213 312

However, only topologically different combinations have the meaningful semantics of the service application, which have been highlighted in the above example. It means (121) combination has the same topology as (131) combination and the only difference is the chosen identifier for the second participant. Therefore, between 27 combinations only five topologically different combinations are important for further process.

The number of call scenarios can be obtained by applying a formula. Applying the formula to different n, the results are shown in Table 6.1. For our example – the combination of CW and TWC with totally 6 FPs - the number of topologically different combinations (call scenario) derived from applying the above formula is: 203

Therefore, the filtering algorithm applies the topological conditions to derive these 203 call scenarios properly and store them as a list.

The list is created using the following recursive algorithm:

1. The only valid combination of one element is 1

2. In order to create a combination with N+1 elements, for each combination with N elements:

   A) Append each of the existing elements (as far as they are different) to this combination with N elements

   B) Append a different one, which is not already contained in the combination with N elements

N = 1: 1 - (Step 1)

N = 2: 11 - existing element appended (2.A)

12 - different element appended (2.B)

N=3: From 11: 111 - Existing (2.A)

112 - new (2.B)

From 12: 121 - existing (2.A)

122 - existing (2.A)

123 - new (2.B)

The flow of filtering algorithm is shown in Figure 6.2.

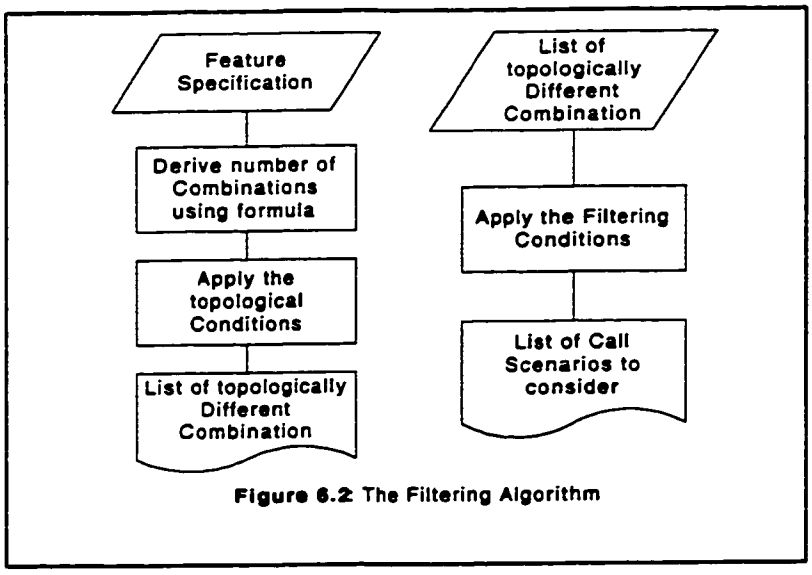


Figure 6.2 The Filtering Algorithm

N	F(n)
1	1
2	2
3	5
4	15
5	52
6	203
7	877

N	F(n)
8	4,140
9	21,147
10	115,975
11	678,570
12	4,213,597
13	27,644,437
14	190,899,322

Table 6. 1: Result of applying the formula on different number

Applying this recursive algorithm for the number 3 results in the following list: **{ 111, 112, 121, 211, 123 }** as shown previously.

However, not all items in this list are combinations leading to interactions between the two features. For example, if the formal participants of two features are completely different, the environment of the two features will be completely isolated from each other, so trying to find the interaction between them is useless. Let us consider again two features, CW and TWC. As we mentioned previously each feature has 3 FPs. One item of the created list is (123456):

	CW			TWC		
	FP1	FP2	FP3	FP4	FP5	FP6
One item:	1	2	3	4	5	6

As it is shown, there is no common actor in two services, and there will not be any interaction. Furthermore, even if there is common FPs for some features, the scenarios may not necessarily be interaction-prone. Based on the Keck method [29], to extract the

interaction-prone scenarios out of the list of automatically generated scenarios, the following conditions must be applied:

1) The two service scenarios must have at least one common actual participant. It is necessary, that the set of each two features must not be disjoint, i.e., at least one actor of one service scenario must be at the same time an actor of the second one.

2) At least one of the features must be able to trigger the execution of the second one. There must exist at least one actual participant that (a) is a participant of both features (this is condition 1) and (b) contains the BCSM that triggers one of the features and (c) also contains this BCSM as a number of the other feature.

3) One pair of associated BCSMs containing an initial detection point of one feature must be common to both feature scenarios. It is sufficient to study interaction scenarios where the association of the O- and T-BCSM containing the initial trigger detection point of one feature, also is present in the second feature scenario.

Besides these conditions there may be additional constraints; for example avoidance of activation of two instances of the same feature for one participant, which is a logical assumption.

Applying the above conditions, the number of scenarios is reduced. The steps of the filtering method are:

1) Use the total number of formal participants of some features and apply the formula to get the number of topologically different scenarios.

2) Create the list of topologically different scenarios.

3) Apply the conditions on the list items in order to reduce the list.

4) Apply special conditions.

The result of this part is a list of interaction-prone scenarios. Applying all the steps of filtering algorithm in our example will result in:

Total number of Formal Participant = 6

Number of scenarios = 46656

Number of topologically different scenarios = 203

The number of scenarios after condition 1 is: 178

The number of scenarios after condition 2 is: 151

The number of scenarios after condition 3 is: 14

List of interaction-prone call scenarios:

{121211, 121211, 121212, 121213, 121221, 121222, 121223, 121231, 121232, 121233, 121234, 122111, 122112, 122113}

The filtering component uses as input the following information items of the features:

- Name of the feature
- Number of Formal Participant
- BCSMs (Type, Initial Detection Point)

6.3 FEATURE INSTANTIATION

The second phase in our approach, is the instantiation of the features. This part connects the filtering process to the detection process. The described features are instantiated using

the elements of the list, which is created by applying the filtering algorithm. In this case, the instantiation is the assignment of the actual participants to the formal participant identifiers in order to obtain the list of APs. Figure 6.3 shows two features, which have been described using the object-oriented template.

We assume the related parts of each formal participant have been defined already. As the purpose of this section is to explain the instantiation technique, the details of the features are not considered. Applying the items of the filtered list means assigning the ids to the formal participants:

	FP1	FP2	FP3	FP4	FP5	FP6
E.g.: one item in the list is: 121233	1	2	1	2	3	3
and another is: 121234	1	2	1	2	3	4

This means that in the first case there are only three APs and in the second case there are four APs. In the second case:

AP1: FP1 and FP3 (AP1 carries out the roles of FP1 and FP3)

AP2: FP2 and FP4 (AP2 carries out the roles of FP2 and FP4)

AP3: FP5 and AP4: FP6

Intuitively, we notice that the overlapping in the duties on one participant may lead to feature interactions. The result of this step for one feature instantiation is: {AP1, AP2, AP3, AP4}, where each AP is an object with the associated attributes and components.

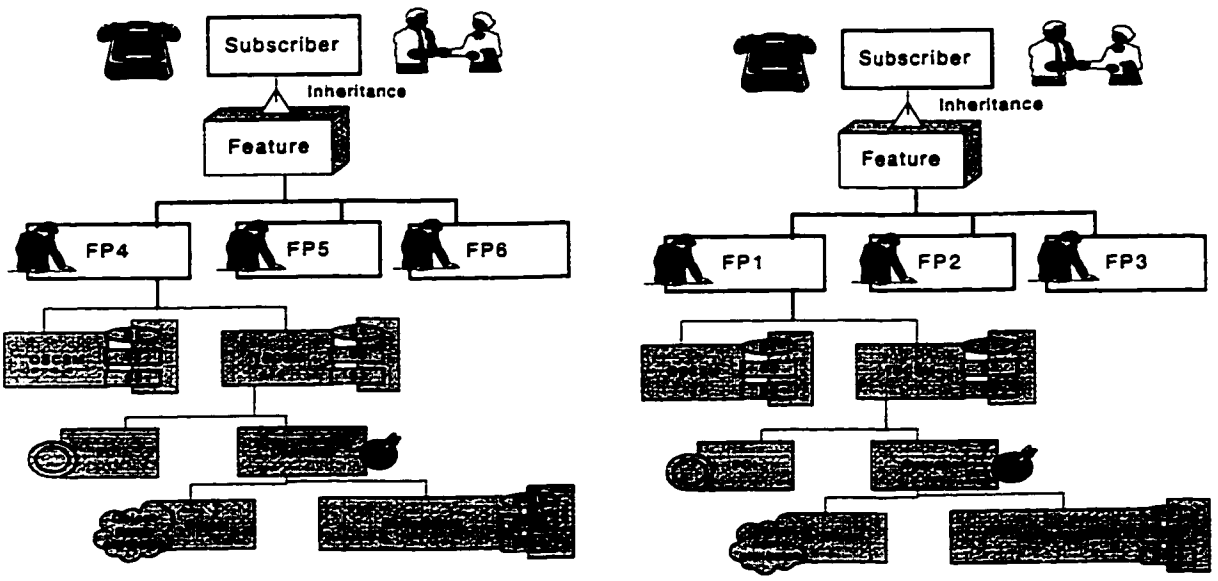


Figure 6.3: Feature Specification

6.4 DETECTION ALGORITHM

The detection algorithm is applied on the result of features instantiation, which are the members of AP list to process each AP carefully and detect feature interactions. The detection algorithm consists of a set of possible conditions. Whenever a condition is satisfied, a specific feature interaction is detected. The conditions are verified through the comparison of common parameters of the features described using the object-oriented template, in different scenarios.

6.4.1 CONDITIONS

In this subsection, the conditions of our detection algorithm are explained. In each case, the same examples used in Chapter 4 for describing the causes of interactions, are referred. The features in these examples are specified using the object-oriented template

and the related procedures for detecting their interaction are described. The relation of causes and conditions is shown in Figure 5.1.

6.4.1.1 Are the naming assumptions compatible?

The naming assumption as described in Chapter 4, can cause an interaction. We devised a procedure to detect this cause. One attribute of a feature as explained in Chapter 5, is the "NAMING". If this attribute is "NONE", it means, the current feature is not sensitive to the "Naming Assumption", otherwise, we have to compare the Naming fields of two features and see whether they are the same or different. If one or both features are not sensitive to the "Naming Assumption", further process for this condition is not necessary. However, when the features under process are sensitive to naming assumption, they must not violate the assumptions of each other, and different naming assumption shows the interaction between the features.

EXAMPLE

F1.name: "Originating Call Screening"

F2.name: "Call Forwarding Unconditional"

F1.Naming: "1L-1N" (It assumes each line has one number as the identifier)

F2.Naming: "1L-MN" (It means more than one number can be used to reach a line. CFU implicitly defines another number for reaching a destination. If user A forwards his calls to user B, another number for reaching B is A, and caller C can dial either B, or A to reach B)

PROCEDURE

```
IF (Naming of F1 != "NONE") and (Naming of F2 != "NONE")
    IF (Naming of F1 (is not equal to) Naming of F2)
        Report the Interaction
    ELSE
        There is no Interaction related to the Naming Assumption
```

6.4.1.2 Are the features under process in the same network component?

It is sufficient to compare the supporting-network components in which the features reside. The developer will set the supporting network fields of the two features. The comparison of these fields of the features under process recognises whether they can be activated at the same time and can have access to the necessary information, provided by the supporting network component. If they are in different supporting network components, there will be a feature interaction.

EXAMPLE

Operator service and Originating Call Screening.

F1.name: "Operator service"

F2.name: "Originating Call Screening"

F1.Supporting-Network: "Remote Switching" (It has no access to the user profile)

F2.Supporting-Network: "SCP" (Originating call screening needs to check the outgoing numbers against the screening list in the SCP to see whether they are allowed to be processed.)

PROCEDURE

```
IF (Supporting Network of F1  !=  Supporting Network of F2)
        Report the interaction
ELSE
        There is no interaction related to the supporting network
```

6.4.1.3 Are the features under control of different administrative domains?

It is sufficient to compare the administrative domains of the two features as described in the Chapter 4, and see whether they can be activated at the same time without any interference. If two features are under the control of different administrative domains, they will interact.

EXAMPLE

Long Distance Calls and Message Rate Charge services.

F1.name: "Long Distance Calls"

F2.name: "Message Rate Charge Service"

F1.Admin-Domain: "External" (There is no access to the status of the call connection on the external segment)

F2.Admin-Domain: "Local"

PROCEDURE

```
IF (Admin Domain of F1  !=  Admin Domain of F2)
        Report the Possibility of Interaction
ELSE
        There is no Interaction related to the administrative domain
```

6.4.1.4. Does a feature have a non-atomic operation?

The existence of one non-atomic operation in one feature prevents not only the processing of any other feature, but also the processing of the basic call. Therefore, it is worth to check this case before going to further tests. The specified operation for each feature as described previously has one of two possible modes (atomic , non-atomic). So, it is enough to check the operations described for the active participants to see whether they have atomic or non-atomic mode.

EXAMPLE

AIN-based Services and POTS.

F1.name: "AIN-based service" (The provisioning of these services is not an atomic operation)

F2.name: "POTS"

F1.FP1.OBCSM.Op.Manipulation.Mode: "Non-atomic operation"

Feature numbered one has one non-atomic operation in the originating basic-call-state model of its first formal participant. Since there is one non-atomic operation, the other feature can not be successful.

PROCEDURE

```
FOR (All formal participants of the features under process) {
        IF (There is non-atomic operation)
            Set the case as the interaction-Prone}
IF (There is Interaction-Prone case)
        Report the Interaction
ELSE
        There is no Interaction related to the Non-Atomic operations
```

The next steps of the algorithm are related to the feature behaviour. To perform these checks, the possible combination of the formal participants (from the list of combinations) must be used to create the actual instances of the features. The instantiation will overlap the duties for some actual participants (common formal participant) and therefore make the interaction-prone situations. On each set of instances and for all the actual participants, the following verifications are to be done:

6.4.1.5 Are the features activated under the same triggering conditions?

If one Actual Participant is responsible for activating the two features, the important issue is the detection point in which they must be activated. The triggering conditions can make an ambiguous situation. If the triggering conditions are the same, it is necessary to look at the activation priorities of the features. In the case of having the same triggering conditions, there are some further tests to be done.

EXAMPLE

Answering Call and Call waiting

F1.name: "Answering Call"

F2.name: "Call Waiting"

F1.FP1.OBC.Tr.type: O-Active

F2.FP1.OBC.Tr.type: O-Active

If the combination of the FPs leads to a situation in which there is a common active FP in two features – an AP that activates two features -, there is the same trigger type (O-active). It is a case that must be processed further.

PROCEDURE

```
FOR (Actual Participants with the activation mode of two features) {
        Check their BCSMs and
        IF (Triggering of F1 (is equal to) Triggering of F2)
            Follow by Check numbered 6.4.1.6
        ELSE
      Follow by Check numbered 6.4.1.10 }
```

6.4.1.6 Do the features have equal activation priority?

If two features have equal priority for activation, and if they are to be activated in the same detection points they may interact. So, by checking the priority fields and the triggering fields of two features, these conflicts will be detected.

EXAMPLE

Answering Call and Call waiting

F1.name: "Answering Call"

F2.name: "Call Waiting"

F1.FP1.OBC.Tr.type: O-Active

F2.FP1.OBC.Tr.type: O-Active

F1.Priority = 1;

F2.Priority = 1;

Two features have one formal participant with the originating basic-call-model in which the trigger is o-active and they have equal activating priorities. These two features will interact.

PROCEDURE

```
FOR (Actual Participants with the activating mode of two features) {
        Check their BCSMs and
        IF (Triggering of F1 (is equal to) Triggering of F2)
                IF (Priority of F1 (is equal to) Priority of F2)
                Follow with Check 6.4.1.7
                ELSE
                        Follow with Check 6.4.1.10   }
```

Now, we must check all the possible conflicts in the operations of the features. Checking the compatibility of the operations of two features with the same triggering conditions and with equal activation priorities is sufficient, because it creates the problem of resource limitation and/or timing and race condition. If in one Actual Participant, two features with the same detection point are activated and they have equal priority, the interaction is related to the operations of the features. In this case, if the operations are the same, the limit on the resources is one of the main problems to be considered. As previously mentioned, the limited number of CPE signals, the number of query per call and the limited physical resources are the causes of interaction. The tests numbered 6.4.1.7 and 6.4.1.8 are related to checking the limits for CPE signal and Query per call.

6.4.1.7 Are there enough CPE-signalling available for both features?

To check the safety of combination of two features for the CPE signals, it is sufficient to count the number of the signals used by the features. According to the Table 4.1, the operations are send/receive signals by the features. The total number of required signals for the feature, must be checked against the limits described for the subscriber.

EXAMPLE

Three-way Call and Call waiting

F1.name: "Three-way Call"

F2.name: "Call Waiting"

F1.FP1.OBC.Tr.type: O-Active

F2.FP1.OBC.Tr.type: O-Active

F1.FP1.OBC.Operation: Send (Signal (flash-hook))

F2.FP1.OBC.Operation: Send (Signal (flash-hook))

F1.Priority = 1;

F2.Priority = 1;

In this case both features want to use the same signal, and they will interact. The procedure in this case is similar to the procedure for condition 6.4.1.8.

6.4.1.8 Are there enough queries per call available for both features?

Same as checking the signal limitation, the total number of query per call used by the features is to be determined and checked against the query limitation described for the subscriber.

EXAMPLE

Originating Call Screening and Area Number Calling

F1.name: "Originating Call Screening"

F2.name: "Area Number Calling"

F1.FP1.OBC.Operation: Send (Query)

F2.FP1.OBC.Operation: Send (Query)

PROCEDURE

```
FOR (Actual Participants with the activating mode of two features) {
        Check their BCSMs and
        IF (Triggering of F1 (is equal to) Triggering of F2)
                IF (Priority of F1 (is equal to) Priority of F2)
                        IF (Operation of F1 (is same as) Operation of F2)
                          IF (The operation is (send Signal))
                              Count the # of signals
                              Compare against the limit for subscriber.
                          IF (The operation is (send Query))
                              Count the # of query
                              Compare against the limit for subscriber.
                          ELSE
                              Follow with Check 6.4.1.10  }
```

6.4.1.9 Are the operations compatible?

If two features are activated for one Actual Participant with equal/none priorities, the compatibility of two operations must be checked in different manner. The compatibility of two features is based on the semantics of their operations. Different contradictory operations have been experienced, in which the activation of one prevents the activation of the other one. Table 6.2 shows some contradictory operations.

To make sure there is no interaction based on incompatible operations, the operations of the features under process must be checked against all the above examples. Subsections A, B, and C present three cases chosen from the above Table.

A. Call Control Operations

By checking the operation of two features to realise whether they contain the control operations, the conflict can be detected. If both features are supposed to control the call, they will interact.

EXAMPLE

Three-way calling and 911

F1.name: "Three-way Calling" (wants to put second part on hold to call the third part)

F2.name: "911" (wants the absolute control of the call)

F1.FP1.OBCSM.Op: "Session (Control)"

F2.FP1.OBCSM.Op: "Session (Control)"

OPERATION1	OPERATION2
Control	Control
Read	Write
Ring	Protect-From-Ring
Reserve	Protect-From-Reserve
Forward	Forward
Release	Protect-From-Release
Use	Protect-From-Use
Write	Protect-From-Write
Read	Protect-From-Read

Table 6.2: Examples of contradictory operations

PROCEDURE

```
FOR (Formal participants with the activating mode of two features) {
        Check their BCSMs and
        IF (Operation of F1 (is same as) Operation of F2)
                IF (The operation is (Session (Control))
                        Report the Interaction
                ELSE
                        Follow with the Check 6.4.1.10 }
```

B. Write and Read Data

One of the causes of interaction as explained in Chapter 4, is race condition. This problem arises when one data item must be read and write by two parallel processes. In

this case different problems such as "read after write" or "write after read" may arise. So if the objects under manipulation of the operations are common and one feature wants to read the data while the other one wants to write it, there is always the possibility of write after read or read after write, which leads to the failure of one feature.

If the data under manipulation is the status of the line, and if one feature changes the status of the line in a wrong time, there will be an interaction to report. Another similar case is when the data under manipulation is the status of the user and one feature uses this information item to perform its task while another operation is changing it.

EXAMPLE

F1.name: " Call Waiting" (Write the status of the line idle while it is busy)

F2.name: "Automatic Call Back" (Read the status of the line)

F1.FP1.OBCSM.Op: "Write (Data)"

F2.FP1.OBCSM.Op: "Read (Data)"

Data = "Status of the line"

PROCEDURE

IF (Object of F1 (is the same as) Object of F2)

IF (Object of Operation 1 (is equal to) Object of Operation 2)

IF (F1 read the Object (and) F2 write the Object)

Report the interaction }

C. Read and Protect-From-Read

If the operations of two features are related to data and both operations manipulate common data and if the mode of data in one operation is public and in another operation is private, an interaction will arise. When one operation captures the data, the other operation will not be able to progress and the feature will not behave properly.

EXAMPLE

Calling Number Delivery and Unlisted Number.

F1.name: "Calling Number Delivery" (wants to use dialling number in public)

F2.name: "Unlisted Number" (keep the number private)

F1.FP1.OBCSM.Op: "Use (Data (Public mode))"

F2.FP1.OBCSM.Op: "Use (Data (Private mode))"

PROCEDURE

```
IF (Object under process of F1 (is the same as) Object under process of F2)
        IF (Object is (data))
            IF (Mode of data in F1 is (private) (and) Mode F2 write the data)
                Report the interaction
}
```

6.4.1.10 Does the result of one operation affect the criteria of the other one?

In different cases, such as the "similar triggering conditions and different priority", "different activation priority" and "activation of one feature in each Formal Participant", checking the necessary operations of two features in different ways is required. The common characteristic of all these cases is that they do their operations one after another or in parallel. A possible problem that can arise here is the result of one operation may change the required assumptions of the other feature. Therefore, it is required to check the influence of one operation on the triggering criteria of the other one. In the feature descriptions, in the description of the detection point of the basic call model of the active formal participant, the situation of the other participants and their lines as the important criteria or dynamic assumptions of the feature about its working environment are specified. After executing of each operation, the criteria must be checked.

PROCEDURE

```
FOR  (All the Operation of all the formal participants of all features) {
        Apply the result of operation
        Check the criteria of other operation
        IF (The required conditions on other FP (is not) satisfied)
            Report the interaction
    }
```

6.4.2 THE ENTIRE DETECTION ALGORITHM

The complete detection flow is shown in Figure 6.4 and the algorithm is as follows:

```
FOR (all the Actual Participants of all Features under process)
    IF (More than one Feature are activated in this FP)
        IF (Similar Detection Point)
            IF (Same Activating Priority)
                IF (Similar Operation)
                    Check Resource Availability (CPE-Signal, Query/Call)
                    Check Data Availability
                ELSE (Different operation)
                    Check the Operation Compatibility
            ELSE (Different Activating Priority)
                Check the Result of one operation on Criteria of the other
        ELSE (Different Detection Point)
            Check the Result of one operation on Criteria of the other
    ELSE (Maximum one feature is activated in this FP)
        Check the Result of one operation on Criteria of the other
```

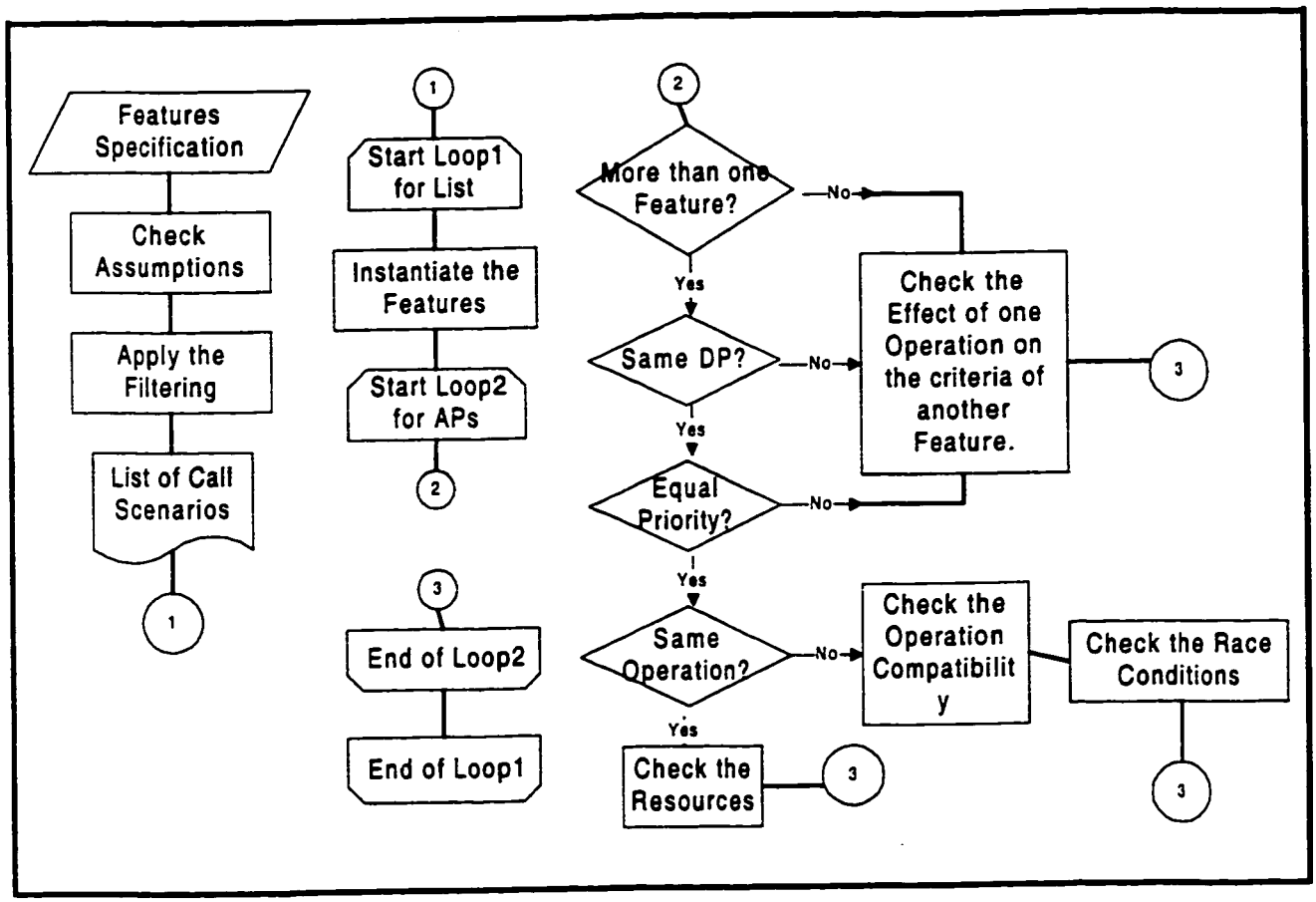


Figure 6.4: Detection Flow

6.5 CHAPTER SUMMARY

In this chapter, the detection technique, including the filtering algorithm, feature instantiation, and detection algorithm were described. All the components of the detection technique use the features specifications described in object-oriented template. The filtering component derives the interaction-prone scenarios. Feature instantiations processes the results of the filtering algorithm and derive the list of Aps, which serves as the input for detection algorithm. The detection component uses the APs list to detect the interactions between the given features. Detection algorithm consists of sufficient procedures for detecting causes of FI and relatively the FI.

Chapter 7

FID TOOL AND APPLICATION

7.1 INTRODUCTION

To validate our approach we implemented a Feature Interaction Detection (FID) tool. FID tool is a complete implementation of the approach presented in the previous chapters. The previous explanations on features modeling and the detection process provided us with the requirements for the tool. To develop the tool, the object-oriented methodology has been used. In analysis step, the requirements of the tool have been analysed to find the relevant components of the tool and as well as the way to design the system. In the design step, the necessary objects and their relations have been defined. Finally, in the implementation step, the system has been implemented using an object-oriented language JAVA [27].

In order to provide all the facilities to validate our approach, the tool consists of eight components: Editor, Choose-Features, Filtering, Detection, Report, View, Tool Evolution and Help.

In the rest of this chapter, we explain the architecture of the tool and its applications. The user interface of the FID tool can be found in the Appendix.

7.2 ARCHITECTURE OF THE TOOL

Analysing the requirement of the system, the architecture of the system was designed.

7.2.1 CLASS HIERARCHY

To implement the FID tool, the suitable user interface, data structures and object classes were implemented. The relation of the object classes at the implementation level is shown in Figure 7.1

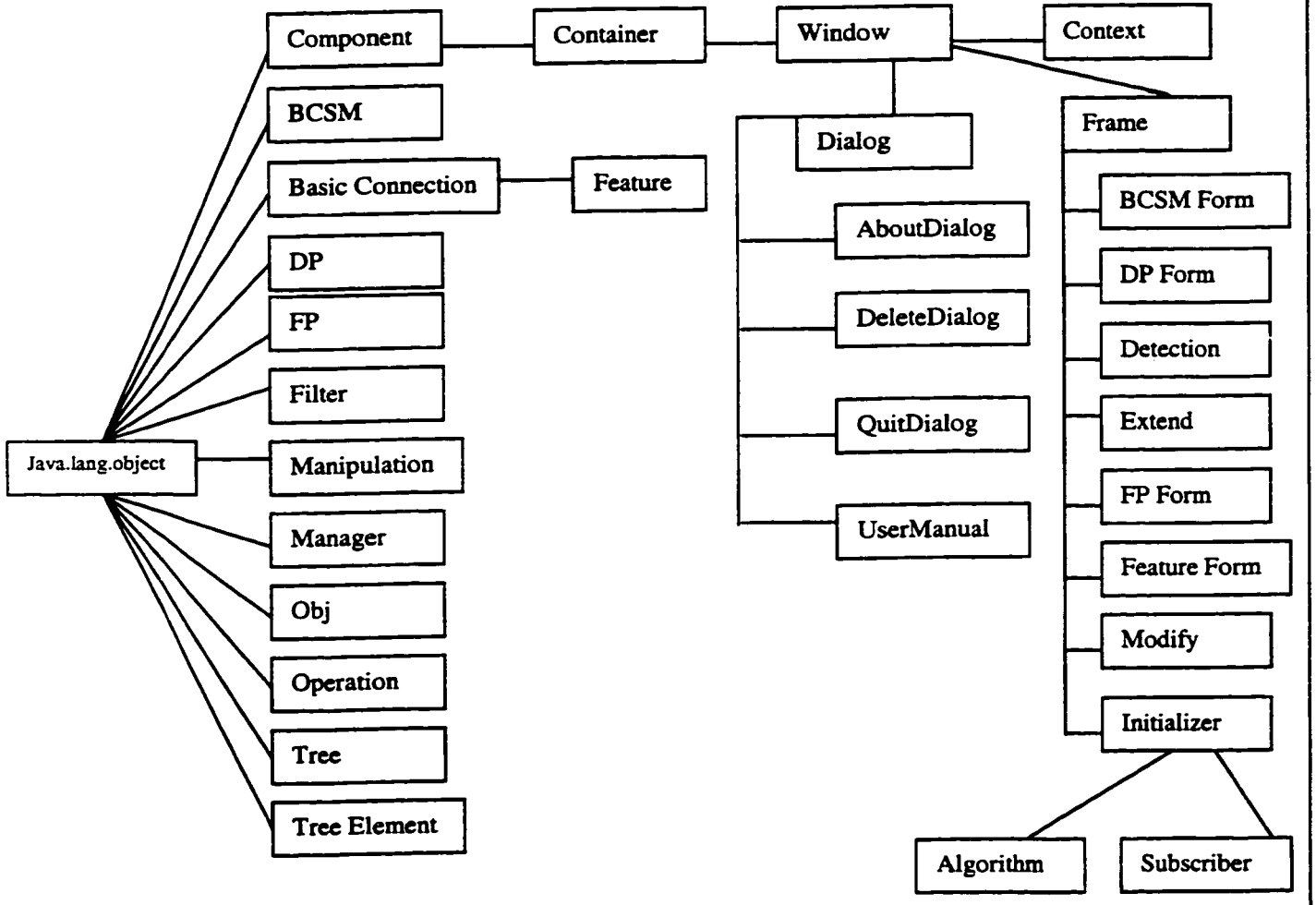


Figure 7.1: FID Class Hierarchy

The architecture of the system is composed of its inputs, processes and outputs. However, for each system there are some important requirements from the user point of view, which affect the architecture of the system.

7.2.2 SYSTEM INPUT

We proposed an object-oriented template in Chapter 5 to model the feature specifications. Since the input of the FID system is the features specifications, the user of our approach must be able to edit the input. Once entered the specification, the user must be able to modify and remove them whenever it is needed. The tool must automatically keep track of the relations between the features and their components and provide user-independent care for updating them in the case of modifications.

Also, the tool must compile the input information and give the possible error messages and let the user correct the information before saving them. In other words, the FID system input provides the following facilities:

- EDIT FEATURE SPECIFICATION
- MODIFY FEATURE SPECIFICATION
- REMOVE FEATURE SPECIFICATION
- KEEP AND UPDATE RELATION BETWEEN THE FEATURE COMPONENTS.
- COMPILE THE INPUT INFORMATION AND GIVE THE ERROR MESSAGES

These functionalities are the services of Feature-Form and its related classes and Modify class.

7.2.3 FILTERING PROCESS

In Chapter 6, we described the filtering algorithm and its necessity before applying any kind of detection algorithm. The FID tool provides the facility to apply the filtering algorithm on the feature specifications. The result of the filtering algorithm is saved in a data structure used by the detection algorithm. To do so, the FID tool provides:

- FILTERING ALGORITHM COMPONENT.

This functionality is the service of Filter class.

7.2.4 DETECTION PROCESS

The main goal of the approach and the tool is to detect interactions between the specified features. The FID tool provides a component, which is the implementation of the detection algorithm. This component uses the result of the filtering algorithm to instantiate the features and provides the list of actual participants. Then, the detection algorithm uses the list of actual participants to apply the conditions and detect the interactions between the features. So, the tool provides:

- THE DETECTION COMPONENT

This functionality is the service of Algorithm and Detection, Initializer and Manager classes.

7.2.5 USER REQUIREMENTS

In addition to the above described components of the system, some of the important required characteristics such as, facility of user interaction, reusability, efficient performance must be considered.

7.2.5.1 EFFICIENCY

To provide better efficiency and performance, as soon as a feature and its associated components are entered to the system; it is saved for further process. Using this facility, the user does not have to enter the specification of a feature several times. The user is able to choose from a list of pre-entered features to detect the possible interactions between them.

Furthermore, to give the fast result to the user, the tool provides the option of choosing between the detecting "only the first interaction" and "all the possible interactions" that can happen between the features under process. So, to cover these cases, the tool provides:

- SAVE THE FEATURE SPECIFICATION FOR FURTHER USE
- DETECT THE FIRST INTERACTION BETWEEN THE FEATURES

These functionalities are the services of Tree, Tree-Element classes and Subscriber and Feature files.

7.2.5.2 USER GUIDE

The FID tool provides an environment where the user is able to interact with the system easily. The help component is provided to guide the user to become quickly and easily familiar with the system. Regarding the special purpose of the FID tool and the complexity of its input items – feature specifications -; the user is able to view the contents of the already entered features specifications. Furthermore, there is a facility to have a hard copy of the results of the processing of a combination of some features. The tool provides:

- HELP THE USER IN USING THE TOOL

- BROWSING THE CONTENT OF FEATURE SPECIFICATION
- CREATE A SEPARATE FILE FROM THE RESULTS OF THE PROCESSING SOME FEATURES

These functionalities are the services of UserManual, AboutDialog classes.

7.2.5.3 REUSABILITY

In Chapter 4 we discussed all the experienced causes of the feature interactions. Based on the possible ways to detect those causes we defined sufficient information for feature description. The important question is "How to take into account the new causes of feature interactions". In the other words, how to update the method as well as the FID tool without changing the code. To update the tool, a component that gets and integrates the new reference information is required. The reference information is mostly about the "ASSUMPTIONS", "RESOURCES", and "OPERATIONS" as the causes of interactions. So, the tool provides:

- KEEP THE REFERENCE INFORMATION
- GET THE NEW REFERENCE INFORMATION FROM THE USER
- USE THE REFERENCE INFORMATION DYNAMICALLY IN OTHER COMPONENTS SUCH AS EDITOR AND DETECTION ALGORITHM.

7.2.6 IMPLEMENTATION DECISIONS

Some decisions before implementing any software, must be made. First of all the language for writing the code, then the data structure for keeping and retrieving the information must be chosen. The following subsections explain our decisions.

7.2.6.1 LANGUAGE

Although our approach, and the requirements of the system narrow our options among which the suitable languages for implementing the tool can be chosen, but still there are some choices.

In Chapter 5, to describe the feature description template, the object-oriented paradigm was chosen, and its advantages briefly explained. Considering the requirement analysis and for better reusability, system maintenance, and easy update the best-suited languages are C++ and JAVA. We chose JAVA because of its special features in comparison to other programming languages [27]:

- JAVA provides a security system to keep its programs well behaved.
- JAVA provides built-in multi-threading and the programs can be designed to do several things at once.
- JAVA has the built-in internetworking capabilities and allows means of accessing multiple computers and distributed applications.
- JAVA has its own garbage collection features and the programmer does not have to take care of memory pointers and memory leaks.
- JAVA incorporates advanced exception handling facilities.
- JAVA is an architecture independence language, and the programs written in Java are runnable on any computer architecture.

Because of those advantages we chose Java as the programming language to implement our tool.

7.2.6.2 DATA STRUCTURES

Another important issue is to choose the appropriate data structures for saving and implementing the information. A proper data structure has a distinguishable effect on the running time. Furthermore, it makes easier the updating and accessing the data. The relation of the information items is another important issue that is strictly related to the data structure. The chosen data structure must provide the means of keeping and updating the logical relations between the data components. The most important data in our system are the subscriber specification, feature specification and reference information.

PERMANENT DATABASES

As explained previously, it is required to permanently store the entered specifications on the hard disk. Since the entered specifications have a special structure, the best data structure to save, retrieve, update and delete the information items is a database management system. However, a database management system will add some overhead to the system and may slow it down. To have the advantages of a database management system while avoiding its disadvantages, we simulated all the tasks and the logic of a Relational Database Management System (RDBMS) in a text file. In an RDBMS the important point is the key identifier for each databases and relation key in order to make the relation among databases.

A) Subscriber file

The subscriber file is a text file named "Subscriber_Data.txt", which has the structure of a database. Each Subscriber has only one record in this database. The key identifier, which acts like a relation key is the subscriber name. The structure of the subscriber record is:

SUBSCRIBER RECORD:

B) Feature file

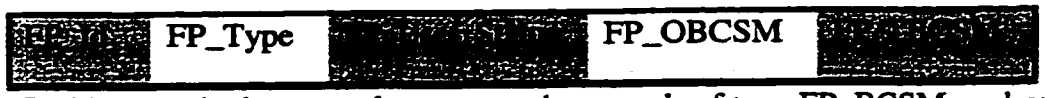
The feature file is a text file named "Feature_Data.txt", which has the structure of a database. Each feature has only one record in this database. The feature name as the key identifier and the subscriber name as the relation key to with the subscriber file are present in each record.

It indicates each feature has only and only one subscriber while each subscriber may have more than one active feature. The structure of the feature record is:

FEATURE RECORD:

The last item of the feature record is a list containing the following structure:

FP LIST ITEM:



In this item the last two elements are the records of type FP_BCSM as shown below:

FP_BCSM :

In the FP_BCSM record, the last item is the DP list, which contains the following structure:

DP LIST ITEMS:

DP ID	Trigger Type		Operation

Two last elements of DP list are TC list and operation records, have the following architectures:

TC LIST ITEM:

FP ID	FP Condition

OPERATION:

OP Name	Object Name	Object Mod	Manipulation	Manipulation Mod

C) Reference file

In describing the operation of the feature and when the detection process is called, the reference file is used in the DP editing environment.

This file contains all the possible operations that can be used to define the feature operation. It also contains the contradictory operations, which are used in detection algorithm to check the operation compatibility condition in Chapter 6.

This file is basically a text file but has a structure of a database. Each record in this database has two fields containing the name of operations. The structure of the record in this file is as follows:

Operation Name	Operation Name

DYNAMIC DATA STRUCTURES

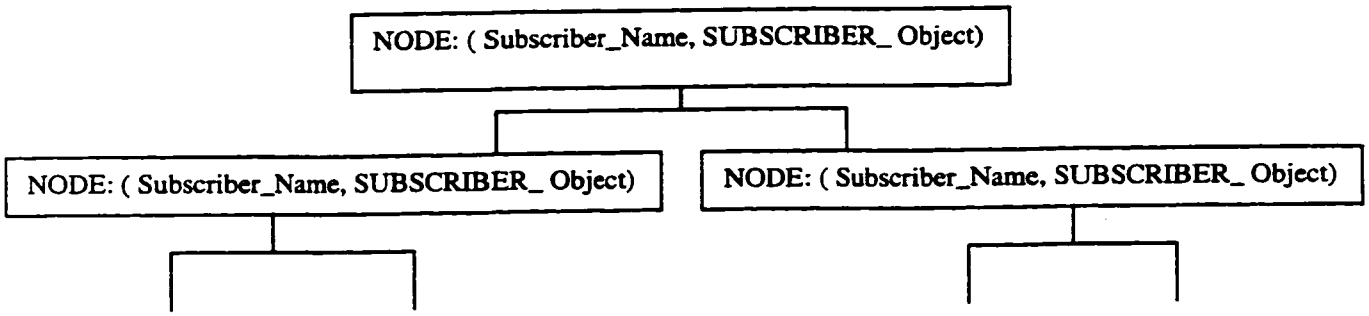
Subscriber file, feature file, and reference file reside on hard disk. Retrieving information directly and dynamically from these files is a time consuming and inefficient task. To

prevent the tool from being slow, when the tool is initialised, all the contents of the related files are extracted and put in a faster and temporary data structure.

The best data structure, which provides fastest retrieve and is automatically sorted, is a binary tree. So, we chose binary tree structure for intermediate storage.

A) Subscriber file

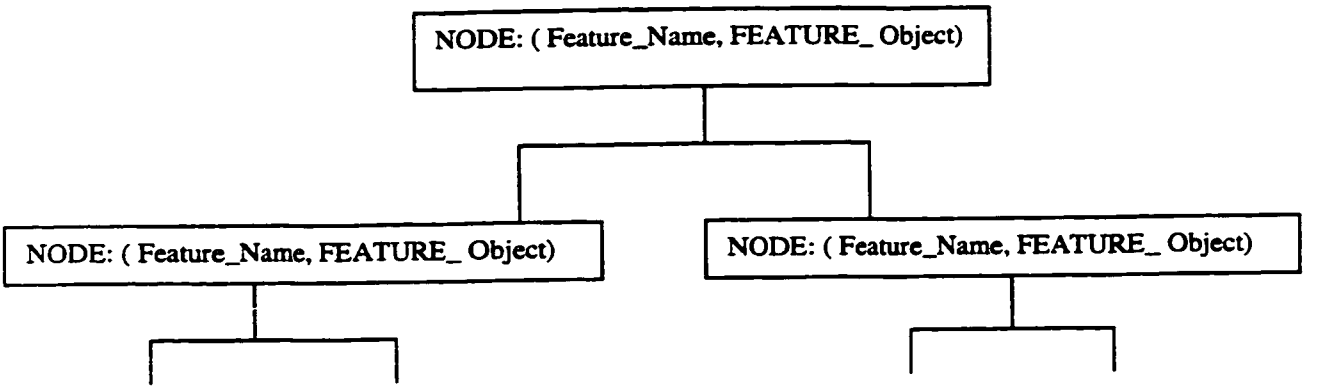
When the tool is initialised and before appearing the user interface main page, the content of the "Subscriber_Data.txt" is read and saved in a binary tree. The structure of this tree is:



Each node has a KEY and an ELEMENT. The key in this tree is the subscriber name and the element is the subscriber object.

B) Feature file

Same as the Subscriber file, when the tool is initialised and before appearing the user interface main page, the content of the "Feature_Data.txt" is read and saved in a binary tree. The structure of this tree is:



Each node has a KEY and an ELEMENT. The key in this tree is the subscriber name and the element is the subscriber object.

C) Reference file

The content of the reference file is extracted at the beginning of the program and put into a two dimensional array. Because there is no search on this file and the only necessary access is sequentially.

7.2.7 TOOL EVOLUTION

As previously described, the causes of interactions, which are classified into assumptions, resources, and feature operations are general and can be extended to cover new technologies and IN capability sets. For example new assumptions may be required for the correct behaviour of the features, the features may need new resources and new operations may be defined. The tool provides the possibility of updating reference information using provided data by the user, based on the new technologies. Therefore, the tool can be updated according to the new features and new causes of FIs. To add new operation to the reference information, the user of the FID tool can simply use the

"EXTEND" component of the tool and enter the new contradictory or non-contradictory operations as shown in Figure 7.2.

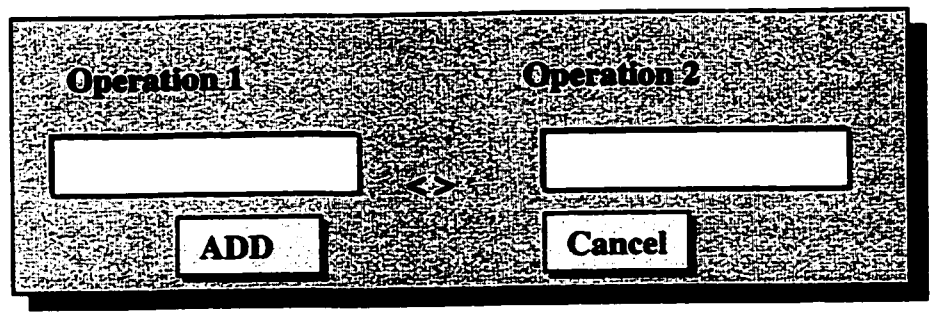


Figure 7.2: Add Operation Environment

Having entered the new contradictory operations, the reference information is updated. This information is used for editing data in Detection Point form for describing the operations of the feature and also for detection the interactions between the features based on the "operation compatibility" condition as described in Chapter 6.

7.3 APPLICATIONS

In this section, we will validate our approach and FID tool. To validate our approach, we must check it against the known feature interactions. It is important to know that the experienced feature interactions are not the only possible ones. There are maybe a lot more inexperienced feature interactions. So, we intend to check the ability of our approach to detect new feature interactions.

7.3.1 BELLCORE BENCHMARK

The Bellcore benchmark [4] developed by Bellcore presents two different ways of categorising feature interactions:

- By nature of the interaction which is divided into:

- Type of features involved
- Number of users involved
- Number of network components involved

- By causes of the interaction

In this benchmark a collection of features and their interactions have been presented. We used the features in this benchmark and their combinations to test our tool.

7.3.2 EUROPEAN BENCHMARK

In the EURESCOM project P509 [5], a benchmark for evaluating the approaches proposed and elaborated in the project has been developed. This benchmark is based on a new definition of feature interaction from which a classification of interactions is inferred in a rather straightforward manner. Same as Bellcore benchmark, a set of features has been introduced and a collection of experienced feature interactions has been presented. We used the features in this benchmark and their combinations to test our tool.

7.3.3 FI CONTEST

A feature interaction detection contest was held during the fifth international workshop on feature interactions. The contest had two phases. The first phase required the contestants to analyse interactions among ten features. The second phase required analysis of the interactions among two new features and the original ten [18]. In addition to the European and Bellcore benchmarks, we also used the features introduced in the contest to validate our tool.

7.3.4 VALIDATION OF OUR FID TOOL

Since the FID tool is an implementation of our approach, we used it as a means to validate the approach. In that manner, the tool also was checked and debugged and additional facilities including better messages and error prompts for the user were added in this phase.

In this subsection, all the combinations checked by the FID tool are presented. In each case, the reason of feature interaction is explained. Notice that in the rest of this section the following acronyms have been used.

(EB) stands for "European Benchmark" and (BB) stands for "Bellcore Benchmark"

There are features and features combinations common in both benchmarks. The common examples are identified by both acronyms EB ad BB.

- (EB) Advice of Charge Start (AOC-S) / Premium Rate plus (PRM+)

  One feature reads data and another feature writes that data which is "charging rate".

- (EB) Connected Line Presentation (COLP) / Connected Line Restriction (COLR)

  One feature uses data in public mode and another feature makes the same data private mode.

- (EB) Originating Call Screening (OCS) / Abbreviated Dialing (ABD)

  Naming assumptions are different. OCS assumes one line has one Number as the identifier while ABD violates this assumption.

- (EB) Call Waiting (CW) / Three-way Calling (TWC)

  Two features have common operation on a common object, which is prevented by the signal limitation. (Sending flash hook signal)

- (EB) Credit Card Calling (CCC) / Voice Mail (VM)

  Both features have common operation on a common object, which is prevented by the signal limitation. (Receiving # signal)

- (EB) Call Waiting (CW) / Call Forwarding on Busy (CFB)

  Triggering conflict with the common operation (control the call).

- (EB) Call Forwarding Unconditional (CFU) / CFU

  Triggering conflict with the common operation, (control the call).

- (EB) Call Waiting (CW) / Call Waiting (CW)

  Trigger conflict and common operation that is joining the new leg.

- (EB) Call Forwarding Unconditional (CFU) / Terminating Call Screening (TCS)

  Naming assumptions are different. CFU use name alias while the other believes in one number one line.

- (EB) Calling Number Delivery (CND) / Unlisted Number (UN)

  One feature uses data in public mode and another feature makes the same data private.

- (EB) Wake up Call (WUC) / Call Forwarding on No Reply (CFNR)

  Common Operation of sending signal (Ring signal)

- (EB) Do Not Disturb (DND) / Emergency Response Service (ERS)

  Common Operation of sending signal (Ring signal)

- (EB) Call Waiting (CW) / Universal Personal Telecommunications (UPT)

  Common operation (call controlling).

- (EB) Call Waiting (CW) / Answer Call (AC)

  Common operation (call controlling).

- (EB) Call Completion on Busy Subscriber (CCBS) / Hotel Room Billing (HRB)

  Two features are under the control of different administrative domain

- (EB) Call Completion on Busy Subscriber (CCBS) / Call Back (CB)

  Common operation (call controlling).

- (EB) IN SCP-access / ISDN time-outs

  Common operation sending a signal

- (EB) Terminating Key Code Protection(TKCP) / CFU

  Both features have common operation, which is controlling the call

- (EB) Delayed Message Delivery-No Reply (DMD-NR) / TCS

  Common operation and common object under process, which is data (Source address)

- (BB) 911 / Three-way Call (TWC)

  Both features want to control the call.

- (BB) Terminating Call Screening (TCS) / Automatic Recall (ARC)

  Naming Assumptions are Different.

- (BB) Originating Call Screening (OCS) / Area Number Calling (ANC)

  Naming assumptions are different.

- (BB) Operator Services (OS) / Originating Call Screening (OCS)

  Supporting Network mismatch.

- (BB) Multi-location business Service Extension Dialing (MBS-ED) / CENTREX

  Different Supporting Network components cause missing the information.

- (BB) Call Forwarding Unconditional(CFU) / Originating Call Screening (OCS)

  Naming assumptions are different.

- (BB) Call Waiting (CW) / Personal Communication Services (PCS)

  Naming assumptions are different.

- (BB) OCS / MDNL with Distinctive Ringing (MDNL-DR)

  Naming assumptions are different.

- (BB) Call Waiting (CW) / Automatic Call Back (ACB)

  Operations are not compatible. One is using the status of the line, while other changes it.

- (BB) Automatic Call Back (ACB) / Automatic Recall (ARC)

  Both features want to control the call.

- (BB) Long Distance Calls (LDC) / Message Rate Charge Service (MRC)

  Two features are under the control of different administrative domain

- (BB) Calling from Hotel (HRC) / Message Rate Charge Service (MRC)

  Two features are under the control of different administrative domain

- (BB) Billing in AIN release 0

  Limited number of Query per call prevents activation of two features to work well together.

- (BB) AIN based services (AIN) / POTS

  One of the features has non-atomic operations and prevents any other feature to be activated.

7.3.5 NEW FEATURE INTERACTIONS

To measure the ability of our approach in detecting new feature interactions, we randomly chose different combinations of already described features in the system to see whether there are interactions between them or not. The followings are some of the new feature interactions detected by our tool.

- TCS / ABD: Two features will interact because of the different naming assumptions.
- WUC / CFU: Two features will interact because of different administrative domains.
- HRB / CB: Two features will interact because of different administrative domains.

- CFU / ARC: Both features want to control the call.
- TCS / MDNL-DR: Different naming assumptions leads to feature interaction.
- ACB / CFU: Different naming assumptions leads to feature interaction.
- *69 / UN: Different modes of access to directory number- public and private.
- OCS/ 1-800: Different naming assumptions leads to feature interaction
- *69 / LDC: Different Administrative domains cause the interaction.
- *69 / HRB: Different Administrative domains cause the interaction.

7.4 CHAPTER SUMMARY

In this chapter, the tool development cycle was explained in details. The architecture of the tool was described. The application of our approach to experienced feature interactions in order to validate our approach was presented. Furthermore, to show the ability of our approach for detecting new feature interactions, more combinations of known features were processed and a list of new detected FIs was presented.

Chapter 8

CONCLUSION

8.1 MAIN CONTRIBUTIONS

Along with more and more emerging IN-service platforms for telecommunication services in fixed and mobile networks, feature interactions between concurrent IN services need to be handled. This can be achieved through an efficient approach for feature interaction detection.

The basis of any solution for the feature interaction problem in IN is a way to detect where and why the interactions happen between a combination of features. Based on the study of previous works and their advantages and disadvantages we chose a pragmatic approach. In our approach the main parts are:

- **Determination of the causes of feature interactions:** By studying several examples introduced in benchmarks and also some of the previous FI detection techniques, the causes of FIs were determined, generalised, and classified. Our effort in this step was toward realising the possible for FIs.

- **Definition of the sufficient information for feature description:** Based on the derived causes of FIs, the sufficient information for feature description was defined. We only use the information that captures the essence of behaviour of the feature and

helps us to detect the causes of interactions. We neglect the details of feature design and implementation. One of the main concern is this step was the simplicity.

- **Modelling the feature description using an object-oriented template:** In modelling the feature description, we chose the object-oriented paradigm in order to describe the relations between the components precisely and provide high reusability and the facility for future updates.

- **Adapting the filtering method to derive the interaction prone call combinations:** Successful management of FIs in the IN, requires a combination of approaches. One of the drawbacks of formal methods is the risk of state explosion when the parallel processes are running together. We avoid this by adapting the filtering method and processing only the interaction-prone scenarios.

- **Instantiating the features using the filtered combinations:** This step connects the filtering component to the detection component. It gets the output of the filtering algorithm and provides the input for the detection algorithm.

- **Establishing a detection algorithm based on sufficient conditions for feature interactions:** The detection algorithm checks for the presence of all the possible causes of interactions. It is based on verifying the sufficient conditions. Whenever a condition is satisfied, the presence of an interaction is detected. The algorithm covers all the realised causes of interactions and can easily take into account new causes of feature interactions.

- **Implementing an automatic tool to validate our techniques:** The feature interaction tool is an automatic means for applying the proposed approach. It provides

all the required components for using our method including an environment for editing and compiling the input features specifications, filtering component to apply the filtering algorithm, and a detection component to apply the detection algorithm. The tool has been implemented using JAVA in approximately 6000 lines of code.

It also provide all the sufficient facilities for guiding the user in using the system as well as viewing the contents of the features specifications and creating the sufficient reports. Using the tool-extend component the user can add the necessary reference information to the tool in order to update its different parts specially the editor and the detection algorithm without any change in the code.

Our approach has been presented to the Intelligent Network community. In fact, one paper [50] describing our approach was published and presented at the "IEEE International Conference on Computer Communications and Networks" in October 1999. A second paper, more complete and detailed, has been submitted for the IEEE Communication Magazine [51].

8.2 MAIN ADVANTAGES OF OUR APPROACH

The proposed method is a pragmatic method; it is based on the experience and observed cases. Despite the fact that, there is no prediction about the coming features, their interactions and the new technologies, our approach has the following advantages:

- It has the simplicity of the pragmatic methods.
- It has the accuracy of the formal method, because it considers all the possible scenarios, which can happen between two features.

- Feature description is simple and does not need deep detailed design information about the features.
- Modelling template embraces all feature components. Since the model is based on the object-oriented methodology, it can evolve easily.
- The approach covers all feature interactions introduced by Bellcore Benchmark.
- The approach covers all feature interactions introduced by European Benchmark.
- It detects new feature interactions, which have not been experienced yet.
- Using the filtering idea, it avoids state explosion, because it reduces the number of scenarios by applying different conditions.
- The detection algorithm is simple.
- The tool is user friendly and provides useful prompts and messages for the user.
- As the most important advantage, the tool can be extended easily and takes into account the new features and the new causes of interaction without any change on the program.

8.3 POTENTIAL EXTENSIONS AND FUTURE WORK

Extension from IN networks to general networks

The developed approach is based on the concept of IN and its components. To develop the approach further, a study of the non-INs is required. This study will probably lead us to an extension of the approach and the tool.

Feature interaction resolution

As a complete solution for the feature interaction problem, a detection approach must be followed by another approach such as resolution. Since the current approach and the FID tool provide the required information about the place and reason of interactions between the features, a resolution approach can simply use this information to completely solve the problem.

Combination of more than two features

Despite the fact that, through this thesis we have mostly chosen the combination of two features in order to describe different situations, our approach does not have any restriction related to the collection of more than two features. During the implementation of FID tool, our effort was aimed toward adding the required functionality to different components in order to handle the combinations of more than two features. It is worth to carefully investigate the ability of FID tool in handling the combination of more than two features.

Future capability sets

Our work is mainly based on the ITU-T standardised IN capability set 1 (CS-1). As described in Chapter 2, the evolution of IN comes with new capability sets.

CS-2 scope provides a significant challenge to specify adequate details for multi-vendor compatibility. In any higher capability set, more IN functionality will be added and new technologies will come to the play. Hence, the concepts of IN and mapping of the

adjacent planes may evolve. So, it is useful to verify the ability of the present approach in the face of higher capability sets and extend it, if it is required. Mapping our model for feature description to the new features can do this verification. Also new causes of feature interactions can be examined to see whether they can be included in the three defined categories – assumptions, resources, and operation - or the introduction of another category is required.

REFERENCES

[1] William A. Shay, Understanding Data Communications and Networks, P W S Publishers, 1997.

[2] Uyless Black, The Intelligent Network, Customising Telecommunication Network and Services, Printice Hall , 1998.

[3] ITU-T, Recommendation Q.1200, "Q-series Intelligent Network Recommendation Structure", 1993.

[4] E.J. Cameron, N.D. Griffeth, Y.J. Lin, M.E. Nilson, W.K. Schnure, and H. Velthuijsen, « A Feature Interaction Benchmark for Intelligent Network and Beyond », Feature Interaction in Telecommunication Systems, L.G. Bouma and H. Velthuijsen (eds.), pp. 1-23, Amesterdam, IOS Press, May 1994.

[5] K. Kimbler and H. Velthuijsen, « Feature Interaction Benchmark », Proc. of the Third Feature Interaction Workshop (FIW 95), Japan, Oct. 1995.

[6] ITU-T, Recommendation Q.1211, Introduction to Intelligent Network Capability Set 1.

[7] ITU-T, Recommendation Q.1208, General Aspects of the Intelligent Network Application Protocol, 1993

[8] ITU-T, Recommendation Q1204, Intelligent Network Distributed Functional Plane Architecture, 1993

[9] ITU-T, Recommendation Q1203, Global Functional Plane For Intelligent Network CS-1, 1993

[10] ITU-T, Recommendation Q1202, Intelligent Network – Service Plane Architecture CS-1, 1993

[11] Jan Cameron, F. Joe Lin, « Feature Interaction in the New World » , Proc. of the Fifth Feature Interaction Workshop (FIW 98), Lund, Sweden, Oct. 1998.

[12] L. Bouma and H. Velthuijsen, Feature Interactions in Telecommunication Systems, eds. Amsterdam, IOS Press, May 1994.

[13] M. J. Butler, « Feature Interaction Analysis Using Z », Abo Academic University, Finland, October 1993.

[14] M. Calder, « What Use are Formal Design and Analysis Methods to Telecommunication Services? », Proc. of the Fifth Feature Interaction Workshop (FIW 98), Lund, Sweden, Oct. 1998.

[15] Jane Cameron, Kong Cheng, F. Joe Lin, Hong Liu and Bob Pinherio, A Formal AIN Service Creation, Feature Interactions Analysis and Management Environment: An Industrial Application, IOS Press, 1997

[16] Jan Cameron, Hugo Velthuijsen, « Feature Interactions in Telecommunications Systems», IEEE Communication Magazine, August 1993.

[17] Stephan Chen, Masanobu Fujioka, and Gerard O'Reilly, Intelligent Network for the global market place. IEEE Communication Magazine, March 1993.

[18] Nancy Griffeth, Bell Labs, Feature Interaction Contest of 5the International Workshop on Feature Interactions, Soka University

[19] Jose M. Duran and John Visser, International standards for intelligent Networks, IEEE Communication Magazine, February 1992.

[20] M.Faci and L.Logrippo, "Specifying Features and Analyzing their Interactions in a LOTOS Environment", Second International Workshop on Feature Interactions in Telecommunication Software Systems, IOS Press, 1994, pp. 136 - 151.

[21] The First International Workshop on Feature Interactions in Telecommunication Software Systems, Florida 1992.

[22] The Second International Workshop on Feature Interactions in Telecommunication Systems, L.G. Bouma and H. Vethuijsen (Eds), IOS Press, 1994.

[23] The Third International Workshop on Feature Interactions in Telecommunication Software Systems, K. E. Cheng and T. Ohta (Eds), IOS Press 1995.

[24] The Fourth International Workshop on Feature Interactions in Telecommunication Networks, P. Dini, Raouf Boutaba and Luigi Logrippo (Eds), IOS Press 1997.

[25] The Fifth International Workshop on Feature Interactions in Telecommunication and Software Systems, K. Kimbler and L.G. Bouma (Eds), IOS Press 1998.

[26] James J. Garrahan, Peter A. Russo, Kenichi Kitami, and Roberto Kung, Intelligent Network Overview, IEEE Communications Magazine, March 1993.

[27] Paul M. Tyma, Gabriel Torok, and Troy Downing, "Java Primer Plus, supercharging web applications with the Java programming language ", Waite Group Press, 1996.

[28] Eric Kuisch, Ronald Janmaat, Harm Mulder, and IKO Keesmaat, « A Practical Approach to Service Interactions », IEEE Communication Magazine, August 1993.

[29] Dirk O. Keck, « A Tool for the Identification of Interaction-Prone Call Scenarios », Proc. of the Fifth Feature Interaction Workshop (FIW 98), 1998.

[30] Dirk O. Keck and Paul J. kuehn, « The Feature and Service Interaction problem in Telecommunication systems: A survey » IEEE Transactions on Software Engineering, Vol. 24, No. 10, October 1998.

[31] Dirk O. Keck, « Identification of Call Scenarios with Potential Feature Interaction », Proc. of the Int'l Workshop on Advanced Intelligent Networks, (AIN 96), T. Margari (ed.), pp. 42-55, March 1996.

[32] Bryce Kelly et al., "Feature Interaction Detection Using SDL Models", IEEE GlobeCom 1994, pp. 1857 - 1861.

[33] Yow-Jian Lin, Nancy D. Griffeth, Managing Feature Interactions in Telecomunications Systems, IEEE Communications Magazine, August 1993.

[34] F. Joe LIN and Yow-Jian LIN, « A building Block Approach to Detecting and Resolving Feature Interactions », Proc. of the Third Feature Interaction Workshop (FIW 94), Amesterdam, Netherlands, May 1994.

[35] Fuchun Joseph Lin, Hong Liu, and Abrajit Ghosh, « A Methodology for Feature Interaction Detection in the AIN 0.1 Framework », IEEE Transactions on Software Engineering, Vol. 4, No. 10, October 1998.

[36] Yuan Peng "Modeling of Intelligent Network Using SDL and an approach for Feature Interaction Detection", Master of Computer Science, Dept of Computer Science, Concordia University, April 1998

[37] Y. Peng, F. Khendek, P. Grogono and G. Butler, « Feature Interaction Detection Technique Based on Feature Assumption », Proc. of the Fifth Feature Interaction Workshop (FIW 98), Lund, Sweden, 1998.

[38] J. Riordan. An Introduction to Combinatorial Analysis, John Wiley and Sons, New york, 1958.

[39] B.Stepien and L.Logrippo, "Status-Oriented Telephone Service Specification", Theory and Experiences for Real-Time System Development. AMAST Series in computing, Vol.2, World Scientific, 1994, pp.265-286.

[40] B.Stepien and L.Logrippo, "Feature Interaction Detection using Backward-Reasoning with LOTOS", Protocol Specification, Testing and Verification, XIV Proceedings of the 14th International Symposium, Vancouver, 1995, pp. 71 - 86.

[41] William Stallings, Data and Computer Communications, Fourth Editions, Addison Wesley, 1996.

[42] Software Engineering Fifth Edition, Ian Sommerville, Lancaster University, Addison Wesley, 1997

[43] Simon Tsang and Evan H. Magill, Learning To Detect and Avoid Run-Time Feature Interactions in Intelligent Networks, IEEE Transactions on Software Engineering, Vol 24, No. 10, Oct. 1998.

[44] Yasuki Wakahara, Masanobu Fujioka, A Method for Detecting Service Interactions, IEEE Communication Magazine, August 1993.

[45] Tae Yoneda and Tadashio ohta, « A Formal Approach for Definition and Detection of Feature Interaction», Proc. of the Fifth Feature Telecommunication Systems» Proc. IFIP Int'l Working Conf. Intelligent Networks, pp. 68-83, Copenhagen, Denmark, Aug. 1995.

[46] P. Zave, « Feature Interaction and Formal Description in Telecommunication», Computer, Vol. 26, No. 8, pp. 20-29, Aug 1993.

[47] Pamela Zave, A Formal Specification of some important 5ESS Features, AT&T Bell Laboratories, 1993.

[48] Pamela Zave, Calls Considered Harmful and Other Observation: A tutorial on Telephony, AT&T Laboratories, 1997

[49] Pamela Zave, Formal Description of Telecommunication Services in Promela and Z, AT&T Laboratories- Research Shannon laboratory.

[50] A. Sefidcon and F. Khendek, "A Pragmatic Approach for Feature Interaction Detection in Intelligent Networks", IEEE Conference on Computer Communication and Networks, Oct. 1999.

[51] A. Sefidcon and F. Khendek, "Detecting Feature Interactions in Intelligent Networks", submitted for IEEE Communication Magazine

[52] K. Kimbler, « Addressing the Interaction Problem at the Enterprise level », Proc. of the Fourth Feature Interaction Workshop (FIW 97), IOS Press, 1997.

APPENDIX: FID TOOL INTERFACE

In this section, the whole picture of the system is drawn. The main components of the system and the relations between them are presented. The tool provides various components to properly accomplish the determined jobs.

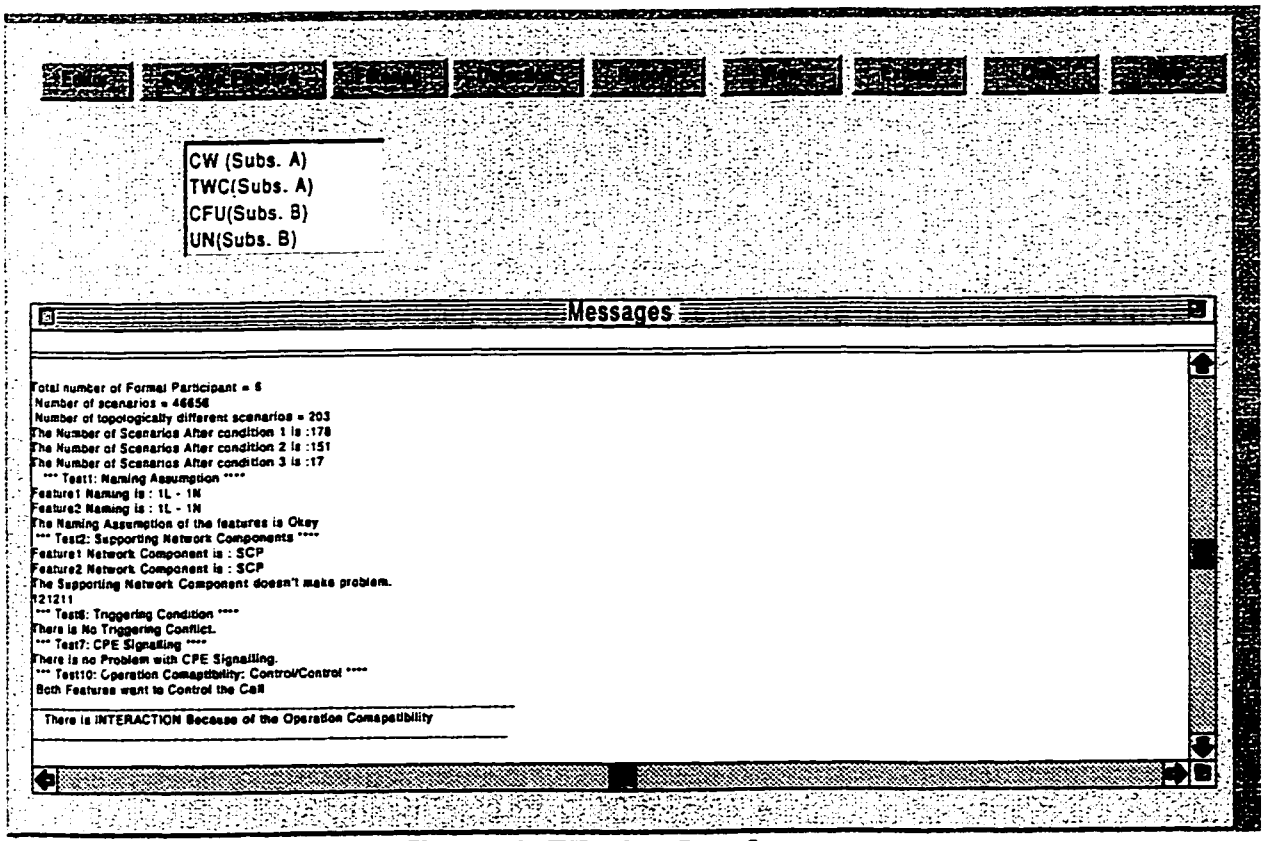


Figure 1: FID User Interface

Each component provides a facility for the user of the tool to do some parts of the job. The user must enter the features specifications by Editor, choose from already entered features into the system, filter the interaction-prone call scenarios between the chosen features by filtering component and detect the interactions between the chosen features by running the detection algorithm on the features. Figure 1 shows the FID user interface.

1) EDITOR

The Editor provides the complete environment to add, modify and remove Subscriber and Feature specification. It saves all this information permanently in the appropriate databases and keeps track of the relations among the feature, its associated components and its subscriber.

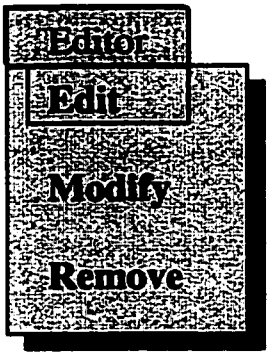


Figure 2: Editor Menu

Clicking the EDITOR button, the editor menu shown in Figure 2 is appeared. From the Editor menu, the user can choose the Edit in order to enter the specifications. Choosing the Edit, the window shown in Figure 3, is appeared. In this window, the "NEW SUBSCRIBER" option to introduce a new subscriber and its associated features can be chosen. Having chosen the "NEW SUBSCRIBER" the subscriber editing form, shown in Figure 4, is appeared. In this form, the user must complete some of the attributes. Some other attributes can be chosen from a predefined list.

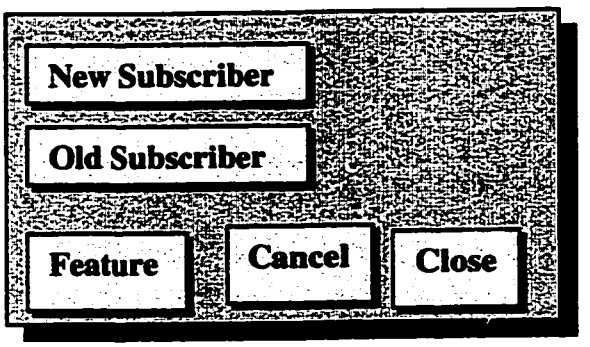


Figure 3: Edit Options

Note that in this form and the other ones the sign "↕" on the attributes shows the capability of choosing from a list.

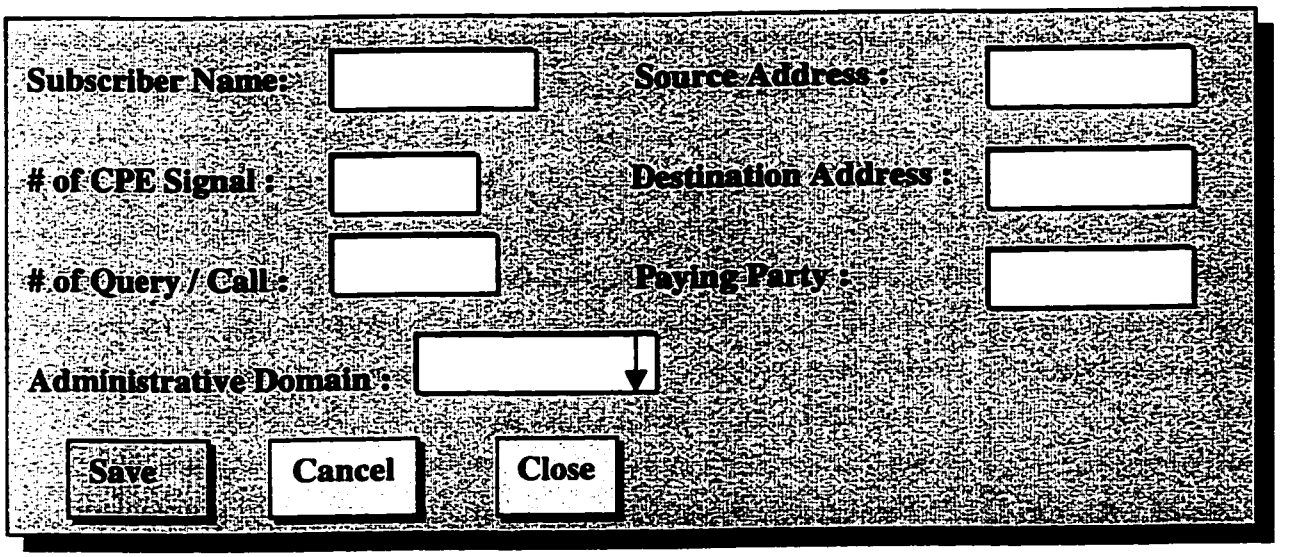


Figure 4: Subscriber form

By choosing the "OLD SUBSCRIBER" button in the window shown in Figure 3, a list of already entered subscribers is added to the window as shown in Figure 5. The user can choose one of the subscriber symbolic names from the list, and then press the

"FEATURE" button to edit the feature specification. In this case the window shown in Figure 6 is appeared to let the user describe the feature characteristics.

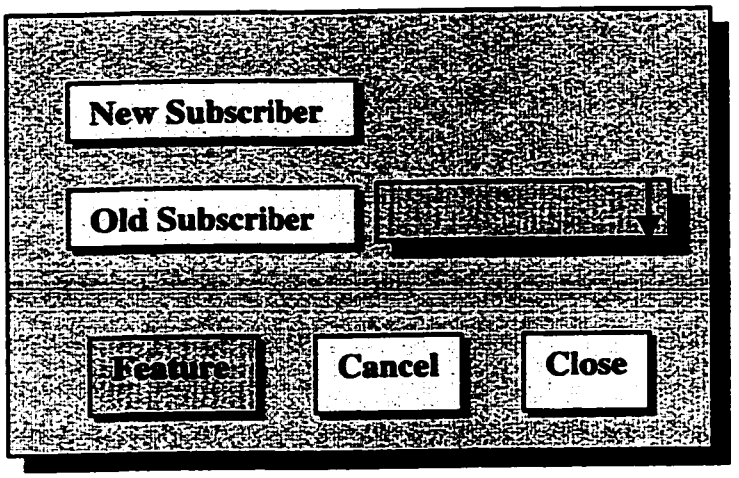


Figure 5: To enter feature specification for an old subscriber

Obviously, the "CLOSE" button, in each form close that window without saving its contents. The "CANCEL" button clears the text boxes and deselects the chosen items from the lists.

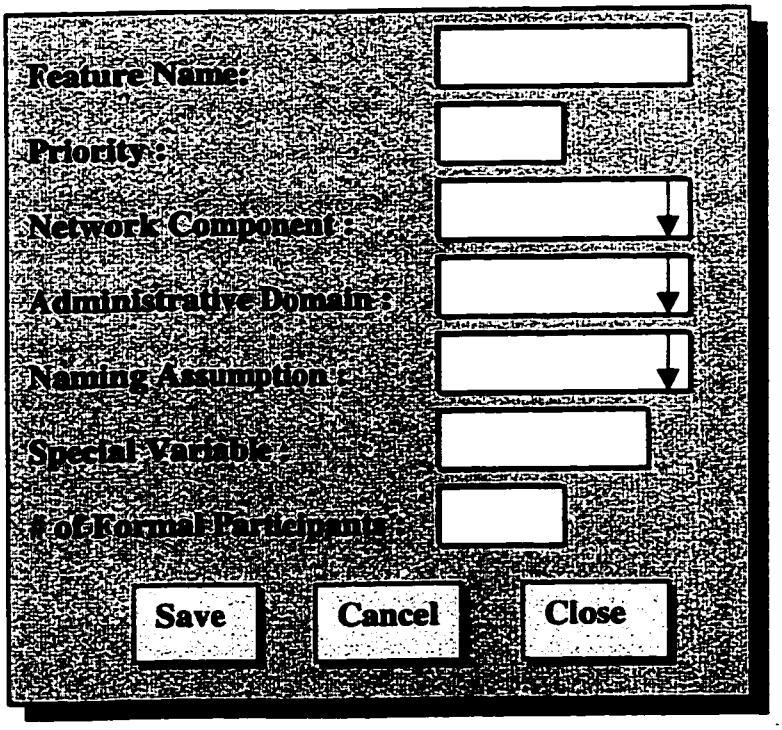


Figure 6: Feature form

Finally, the "SAVE" button activates the compiler process in order to validate the entered information by the user. If there is any problem with the information, the compiler sends an error message to the user that is shown at the bottom of the main page in Figure 1. Getting these messages, the user can modify the information and try to save it again. The compiler checks the information each times the "SAVE" button is pressed.

Based on the special relation defined in our model, for Feature - Formal participants, after specifying the feature, its formal participants must be described. Once the Feature form is completed and the "SAVE" button is pressed, in addition to the compile routine, the edit windows for the formal participant are appeared. The number of editing windows for formal participants is the same number of formal participants defined in the Feature form by the user. For example for the TWC service, with three formal participants, after pressing the save button in the feature form, three windows for editing the characteristics of the formal participants are appeared. FP form is shown in Figure 7.

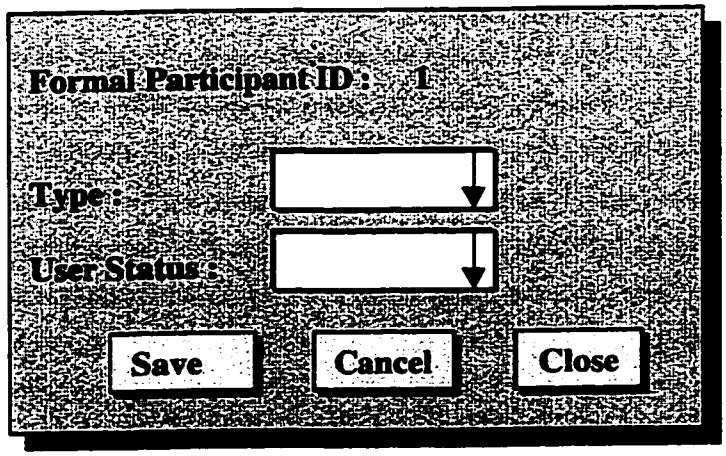


Figure 7: Formal Participant Form

In the Formal Participant form, the FP id is automatically given and the other attributes can be chosen from the relevant lists. The "SAVE" button in these windows not only activates the compiler routine but also provides the BCSMs editing environment, which is

shown in Figure 8. Since there are two types of BCSM, two windows are appeared, one for originating side and one for the terminating side. According to its role, one formal participant may need OBCSM, TBCSM, or both of them.

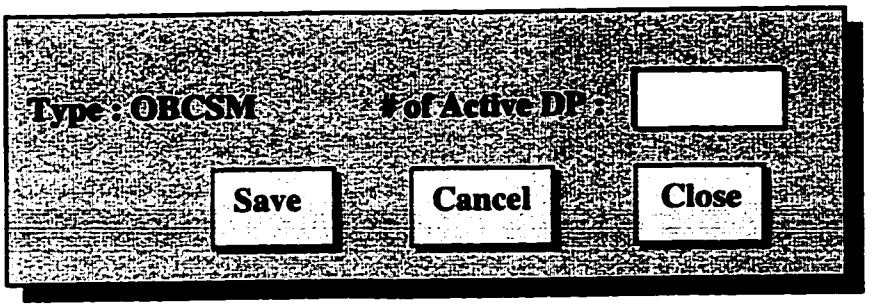


Figure 8: Basic Call State Model Form (Originating Side)

If a formal participant is related to the originating side, it can close the TBCSM window and vice versa. In the BCSM form the important attribute is the number of active detection point. If the formal participant is not activating the feature and just participates in the scenario, it does not have any active detection point in its BCSM (either originating or terminating). In this case, the user has to enter "0" in the text box to specify there is no active detection point. With no detection point, the save button will save the contents of that BCSM for the relevant formal participant.

In the case of having one or more active detection points, pushing the "SAVE" button, the necessary DP editing environment are provided. The Detection Point form is shown in Figure 9.

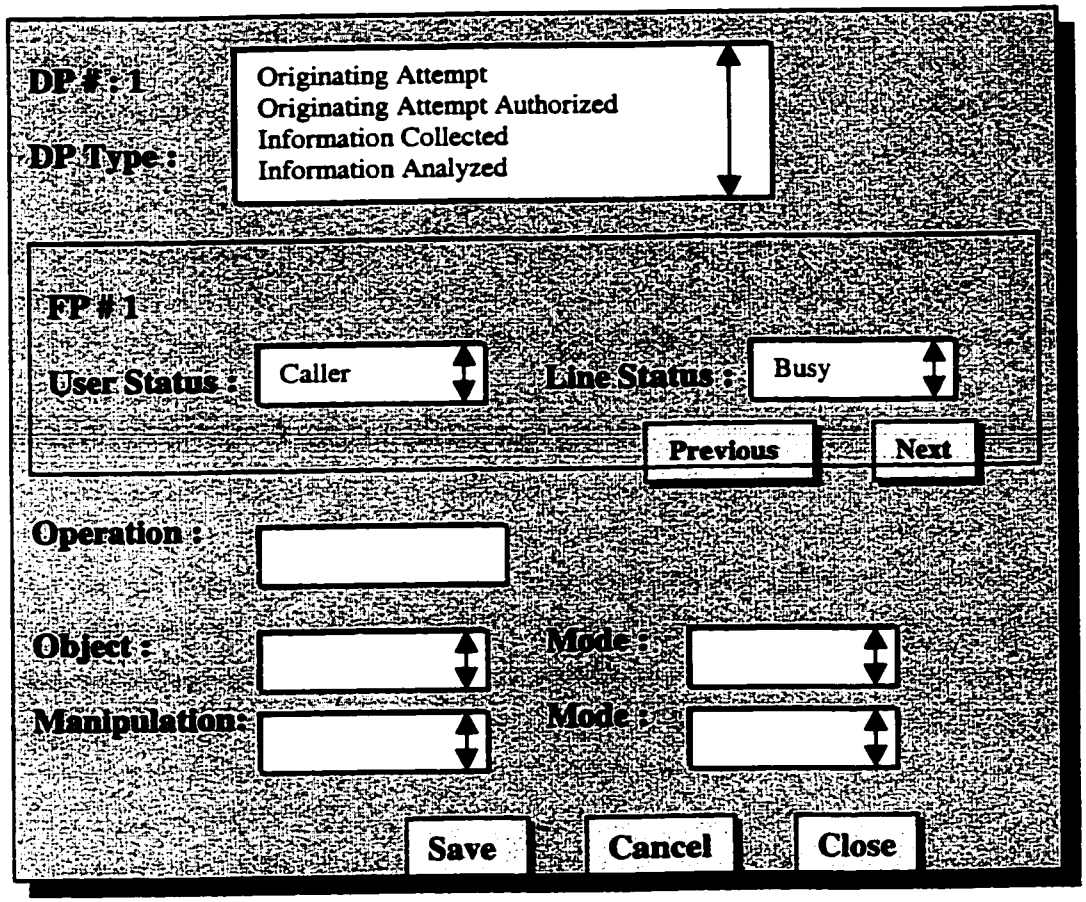


Figure 9: Detection Point Form

In DP form, the tool automatically assigns the DP number, and the other attributes can be chosen from the predefined lists. The "SAVE" button will compile the entered information and save the contents of the form and close the window.

Pushing "SAVE" button for the second time on the BCSM, FP, and Feature forms will save their contents and close the windows.

To prevent user from confusing the relations of the forms, at the top of each form, its precedents forms ids are shown. For example at the top of DP form, the following title shows its relation with its precedents: subscriber name, feature name, and FP id.

Choosing Modify from the window shown in Figure 1, the list of saved features or subscribers correspond to the user selection is appeared. Modify window is shown in Figure 10. The user must choose the feature, or subscriber button, to activate the relevant list, then the user must choose the desired name of feature or subscriber, and finally press the "MODIFY" button.

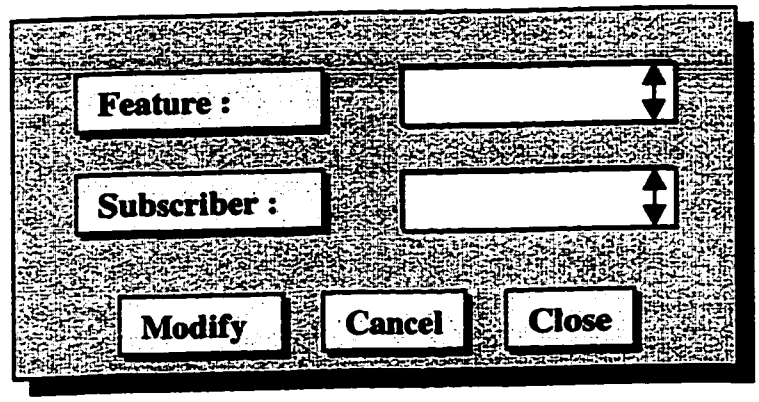


Figure 10: Modify window

Pressing the "MODIFY" button, all the editing windows containing pre-defined attributes, which are explained in the edit section, are appeared. Having the editing windows the user can do the required modification and save them again. The "SAVE" button in this case will also activate the compiler and update all the related files and databases.

Pressing the "REMOVE" button from window shown in Figure 1, the same window shown in Figure 10 is appeared, except that, instead of the "MODIFY" button there is the "REMOVE" button. The user has to choose the feature, or subscriber button, to activate the relevant list, then choose the desired name of feature or subscriber, and finally choose the "REMOVE" button. In this case, a confirmation window is appeared in order to let the user make the final decision about removing the chosen information. The

confirmation window is shown in Figure 10 in which selecting the "YES" button will remove the chosen specification from the relevant file and databases.

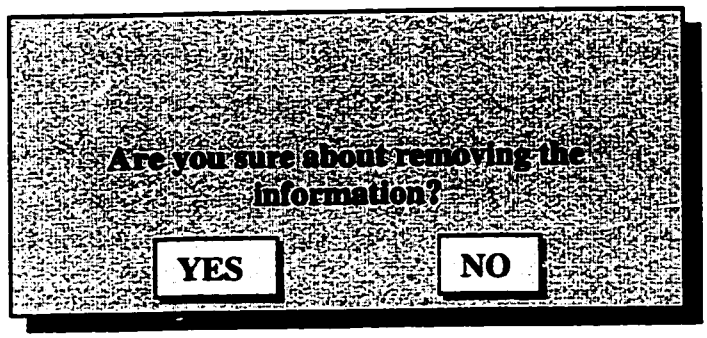


Figure 11: Confirmation window

2) CHOOSE FEATURES

As soon as the features specifications are entered, they are saved on disk for future usage. By pressing the "FEATURES" button from the main menu, a list containing the names of the pre-described features along with the names of their subscribers appears on the main page. The user can choose the desired features from this list, which is a multi-selection list.

In this list, the names of the features are associated with the name of their subscribers. Figure 1 shows the main page of the tool after pressing the "CHOOSE FEATURES" button from the main menu, and from the window is the list of pre-described features.

3) FILTERING

After selecting features from a list, the user can run the filtering algorithm in order to create the list of interaction prone call scenarios. The filtering component extracts the information of the chosen features from the appropriate database using the mathematical analysis to create a list of topologically different combinations. Each combination represents a collection of formal participants who play the roles for the features.

Application of the filtering method is done on several steps. Each step corresponds to one condition and it must be satisfied if the combination is to be interaction-prone. After applying each condition the number of items in the list is reduced.

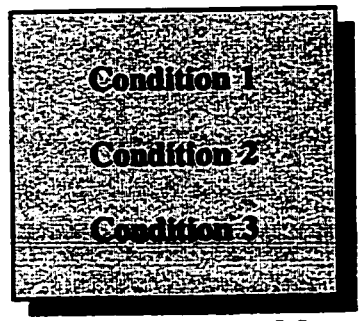


Figure 12: Filtering Menu

Choosing the "FILTERING" button from the main menu in Figure 1, the menu shown in Figure 12 is appeared. From this menu the filtering conditions can be chosen and run one after another. The result will be provided to the user in "MESSAGE" box in Figure 1.

4) DETECTION

Detecting the interaction is the main goal of the tool. After applying the filtering algorithm on the features specifications, creating and reducing the number of interaction-prone call scenarios, the detection algorithm can be applied to the items of the reduced list (each item is a combination of the formal participants). By feature instantiation the formal participant will be replaced by the actual participants and the detection algorithm will be run. The result will be a report of interactions between the selected features and the causes of these interactions, which in turn can be used for further process such as resolution. Pressing the "DETECTION" button from the main menu shown in Figure 1 the detection menu shown in Figure 13 is appeared.

"FIRST INTERACTION" from the detection menu runs the detection process just to find the first occurrence of the feature interaction.

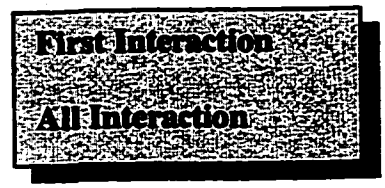


Figure 13: Detection Menu

However, the "ALL INTERACTION" option follows all the conditions of the detection process to find all the possible interactions that can happen between the selected features. The result of the process is shown in the "MESSAGE" window of the FID main page in Figure 1.

5) REPORT

In most cases, it is necessary to submit the result of the processing some features for further study such as resolution. It is worth to have the result of the process in a printable file. The tool provides this facility by creating a report. The result of the filtering and the detection processes will be saved into this printable file. To do so, the user has to select the desired features from the features list, then press the "CREATE REPORT" in order to have the result of the process in a printable file identified by the names of the selected features.

6) VIEW

The user may need to look at the content of the specified features. The "VIEW" button from the main menu shown in Figure 1 provides this facility for the user. To see the contents of a feature, the user must choose the desired feature and then push the "OPEN"

button from the view menu. The result is the feature window and all its associated components.

7) HELP

The Help component of the tool provides complete guidelines for the user. It helps the user in walking through different sections of the tool and to understand the correct order of running the tool components. In addition to the user manual it provides the interactive prompt to keep track of the order for applying the processes to help the users in understanding their mistakes. Pressing the "HELP" button from the main menu of Figure 1, the help menu shown in Figure 14 is shown. The "ABOUT THE TOOL" give information about the tool, while "USER GUIDE" provides useful comments for the user.

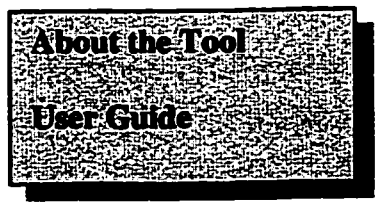


Figure 14: Help Menu