ETSI
TECHNICAL
REPORT

ETR 137

July 1995

Source: ETSI TC-NA

Reference: DTR/NA-061101

ICS: 33.080

Key words: IN, service

Intelligent Network (IN); Service and service feature interaction service creation, service management and service execution aspects

ETSI

European Telecommunications Standards Institute

ETSI Secretariat

Postal address: F-06921 Sophia Antipolis CEDEX - FRANCE
Office address: 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
X.400: c=fr, a=atlas, p=etsi, s=secretariat - **Internet:** secretariat@etsi.fr

Tel.: +33 92 94 42 00 - Fax: +33 93 65 47 16

Contents

Blank page

Foreword

This ETSI Technical Report (ETR) has been produced by the Network Aspects (NA) Technical Committee of the European Telecommunications Standards Institute (ETSI).

ETRs are informative documents resulting from ETSI studies which are not appropriate for European Telecommunication Standard (ETS) or Interim European Telecommunication Standard (I-ETS) status. An ETR may be used to publish material which is either of an informative nature, relating to the use or the application of ETSs or I-ETSs, or which is immature and not yet suitable for formal adoption as an ETS or an I-ETS.

This ETR is based on CCITT Recommendations Q.1201 [1] to Q.1290 [13] as given in CCITT COM XI-R 164, 1992.

Blank page

1 Scope

This ETSI Technical Report (ETR) deals with the definition of service and service feature interaction. It provides a definition of the interaction problem and a setting of the scope of interaction studies.

Intelligent Network (IN)-structured networks are used to provide rapid deployment of new services and service features to telecommunications users. These services and service features are mostly specified by different designers and implemented on different networks. This increases problems occurring during the life cycle of each service because the resources (network resources, services data) are influenced by other services and service features and thus such a service behaves differently.

For the services to behave according to their specification and to fulfil the constraints imposed by sharing of resources among different services, methods for the detection and resolution of interaction situation are to be found.

It is to be noted that a service is supported within a **service addressing domain**. Consequently, interaction can only be addressed (and solved) within such a given addressing domain. Interaction between two network-specific services, e.g. interaction between a call rerouteing in a Private Branch Exchange (PBX) and a service in the public network, cannot be solved as far as the services are not to be considered as multi-network, i.e. applied to a multi-network addressing domain.

The purpose of this ETR is to propose methods independent of both services and physical grouping of the functional entities involved in the call. It covers interaction between services and service features provided by IN-structured networks. The concepts and solutions proposed are **not** limited to a specific capability set.

Interactions between IN-based and switch-based services and service features can be found in ETR 186-1 [14].

A subset of topics which are candidates for standardization should be determined from this ETR. For instance, subclause 6.3.3.5, related to automation of spotting addresses, a competitive domain and no standardization may be foreseen: these ideas are mentioned in this ETR as a part of the general interaction solving process and a help to understand it as a whole. On the contrary, general topics such as vocabulary or description of the interaction life cycle, as well as some specific topics such as trigger priority table or communication between interaction managers, have to be studied as candidates for standardization.

2 References

This ETR incorporates by dated and undated reference, provisions from other publications. These references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this ETR only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies.

[1] CCITT Recommendation Q.1201 (1992): "Principles of intelligent network architecture".

[2] CCITT Recommendation Q.1202 (1992): "Intelligent network - Service plane architecture".

[3] CCITT Recommendation Q.1203 (1992): "IN global functional plane architecture".

[4] ITU-T Recommendation Q.1204 (1993): "Intelligent network distributed functional plane architecture".

[5] ITU-T Recommendation Q.1205 (1993): "Intelligent network physical plane architecture".

[6] ITU-T Recommendation Q.1208 (1993): "General aspects of the intelligent network application protocol".

[7] ITU-T Recommendation Q.1211 (1993): "Introduction to intelligent network capability set 1".

[8] ITU-T Recommendation Q.1213 (1993): "Global functional plane for intelligent network CS-1".

[9] ITU-T Recommendation Q.1214 (1993): "Distributed functional plane for intelligent network CS-1".

[10] ITU-T Recommendation Q.1215 (1993): "Physical plane for intelligent network CS-1".

[11] ITU-T Recommendation Q.1218 (1993): "Interface Recommendation for intelligent network CS-1".

[12] ITU-T Recommendation Q.1219 (1994): "Intelligent network user's guide for capability set 1".

[13] ITU-T Recommendation Q.1290 (1993): "Glossary of terms used in the definition of intelligent networks".

[14] ETR 186-1: "Intelligent Network (IN); Interaction between IN Application Protocol (INAP) and Integrated Services Digital Network (ISDN) signalling protocols; Part 1: Switching signalling requirements for IN Capability Set 1 (CS1) service support in a Narrowband ISDN (N-ISDN) environment".

3 Definitions

For the purposes of this ETR, the following definitions apply:

service: Offer by a public or private service provider to its customers in order to satisfy a telecommunication requirement.

service feature: Specific aspect of a telecommunication service that can also be used in conjunction with other telecommunication services/service features as part of a commercial offering. It is either a core part of a telecommunication service or an optional part offered as an enhancement to a telecommunication service.

switch-based service: Service of which logic is located on a switch outside IN, and that can be either a standardized Integrated Services Digital Network (ISDN) supplementary service or an operator-specific service for analogue lines[1)].

service life cycle: Description of both stages and steps involved during the complete life of any service, in a service independent manner.

network operator: Entity responsible for the development, provisioning and maintenance of real-time services and for operating the corresponding networks.

service provider: Entity who provides services to its subscribers on a contractual basis and who is responsible for the service offered.

service subscriber: Entity that contracts for services offered by service providers.

service user: Entity external to the network that uses its services.

service addressing domain: A number of networks (e.g. public, private, fixed, mobile) controlled by different service operators, within which a service is visible.

interworking: Processing of several services which have mutual influence.

interaction: Interference of an entity with the intended and expected behaviour either of another entity, or of another instance of itself.

interaction life cycle: Set of moments through which an interaction gets, from its spotting to its resolution.

interaction germination: Data modification or initialization which prepares an interaction manifestation that may occur either later on in the same call or in a further call.

interaction manifestation: Moment when an interworking between services causes a situation viewed as unsatisfactory from any of the actors, which occurs during service invocation. It happens:

- either when a service disturbs or inhibits the expected execution of another service considered separately (or of another instance of the same service);
- or when the joint accurate executions of services creates new unforeseen behaviours which would not appear in case of executions considered separately.

interaction spotting: Analysis of the new service model, in conjunction with already existing service models, in order to find as many interaction cases as possible.

[1)] There are currently projects on the way, for example Switch to Computer Application Interface (SCAI) or Computer Supported Telecommunication Applications (CSTA), that propose to supply services via linking computers to the switch. The wording "switch-based services" is thus to be considered carefully. The impact of these projects on interaction studies is for further study.

interaction avoidance: Set of techniques aiming at the elimination of interaction cases, e.g. development of more advanced terminals, introduction of new signalling capabilities in the network, redefinition or restriction of service functionalities, or determination of acceptable service interworking.

interaction tracking: Moment when an interaction germination can be observed during service management (data modification or service activation) before any interaction manifestation occurs. It allows arming of mechanisms to enable interaction treatment.

interaction watching: Observation of service execution, in order to ensure that no interaction manifestation occurs and, in the opposite case, to trigger mechanisms designed to take care of this manifestation.

interaction care: Alleviation of an interaction manifestation after its detection by interaction watching.

interaction treatment: Processing of complementary mechanisms designed to solve an unsatisfactory interworking situation, which may have germinated either in the same call or in a previous call.

technical service interaction: Interworking situation which results either in the dysfunction of one of the services involved or in the immediate subsequent dysfunction of another service. In such a case, one of the services does **not** fulfil its technical basic requirements.

policy service interaction: Interworking situation whose effect is unacceptable to the service customer (i.e. service user or service subscriber) from an economical, sociological or ergonomical point of view, or is deemed as unacceptable to business by the service provider or the network operator. Such a situation occurs **even if** each of the services does fulfil its technical basic requirements.

execution arbitrator: Entity which has, during service execution and according to the choices previously made, to watch and deal with interaction manifestation.

management arbitrator: Data manager able to track germination and to treat it according to the choices previously made.

local interaction: Interaction situation when a unique Basic Call Process (BCP) is involved in the call and has triggered several Service Logic instances (SLs).

distributed interaction: Interaction situation when several BCPs are involved in the call, each one connected to a unique SL.

4 Abbreviations

For the purposes of this ETR, the following abbreviations apply:

ARC	Automatic ReCall
BCP	Basic Call Process
BCSM	Basic Call State Model
CCF	Call Control Function
CFU	Call Forwarding Unconditional
CW	Call Waiting
DFP	Distributed Functional Plane
DP	Detection Point
FIM	Feature Interaction Manager
GFP	Global Functional Plane
GIM	Global Information Model
IN	Intelligent Network
IPFIM	Inter-Platform FIM
ISDN	Integrated Services Digital Network
ISUP	ISDN User Part
LOTOS	Language Of Temporal Ordering Specification
PBX	Private Branch Exchange
PSTN	Public Switched Telephone Network
SCE	Service Creation Environment
SCF	Service Control Function
SEE	Service Execution Environment
SDL	Specification and Description Language
SIB	Service Independent building Block
SIM	Service Information Model
SL	Service Logic instance
SLIM	Service Logic Interaction Manager
SMF	Service Management Function
SSF	Service Switching Function
TCS	Terminating Call Screening
TKCS	Terminating Key Code Screening

5 The service and feature interaction problem

NOTE: In the following text, in order to facilitate the reading, the term "service" is currently used instead of "service feature".

5.1 General introduction

This ETR gives a general overview of the service and feature interaction problem.

Clause 6 proposes a view from the service plane, which tries to answer to three questions that occur when considering service interaction:

- a solution, which one? (i.e. the solution is not unique, and a choice is to be made);

- a solution, when? (i.e. in which moment of the interaction life cycle the problem is to be solved);

- a solution, how? (i.e. by which means can the problem be discovered and solutions be found).

An approach combining three mechanisms is proposed:

- spotting during service creation, which is outside the scope of this ETR, but whose principles are briefly described;

- implementing a management arbitrator, which is proposed without further development;

- implementing an execution arbitrator.

This last mechanism implies a set of solutions impacting on the Distributed Functional Plane (DFP), solutions built around execution managers and improved signalling, which are further described in clause 8.

5.2 Situations of interworking between services

5.2.1 Two services involved

Considering the introduction of a new service in the network, when this new service interworks with an already existing one, three situations can be distinguished:

- when the new service interworks with an existing service without one influencing the other, its introduction has **no impact**, from an interaction point of view, on the considered service: the two services are functionally independent;

- when the aim of the new service is to modify an existing service, there is a **desired interworking** (e.g. Freephone modifies the charging of the basic telephony service): the two services are functionally complementary;

- when an undesired consequence of the mutual influence of the new service with already existing services is observed, such an **undesired interworking** is called **interaction**. In this case, services may be either functionally complementary (some precautions have to be taken to get a correct interworking), or functionally incompatible (the second request shall be refused or shall override the earlier one).

5.2.2 More than two services involved

Multiple interactions, i.e. interactions involving more than two services, may be divided into three cases:

- bilateral interaction (i.e. considering S1-S2, S2-S3 and S3-S1 independent interactions);
- multilateral interaction (i.e. considering global S1-S2-S3 interaction);
- roundabout interaction (i.e. considering the case where S1-S2 interaction causes an interaction S1-S3 or S2-S3).

This ETR does not consider specific requirements for addressing such multiple interactions.

5.3 A thesaurus of examples

Basic examples referred to in this ETR may be found in annex A. These are the following:

- Call Forwarding Unconditional (CFU) and Terminating Key Code Screening (TKCS):
    - first case: a user forwards his calls to a line screened by a key code;
    - second case: a user has his line screened by a key code and makes his calls forwarded;

- CFU and Automatic ReCall (ARC):
    - first case: a user B has requested a CFU to C, a user A places a call to B which is forwarded to C, who uses ARC;
    - second case: a user B has requested a CFU to C, B calls A, and A tries to use ARC;

- CFU and Terminating Call Screening (TCS): interworking between these two services presents 36 different cases to scan.

6 View from the service plane

6.1 A solution: which one?

6.1.1 A non-deterministic problem

If an end user sets both a Call Forwarding on Busy (CFB) service and a Call Waiting (CW) service on the same line, only one service can operate when a call is received in a busy situation. Such an easy example of undesired interaction seems easy to manage, and a number of solutions spring to the mind: one could forbid CFB and CW to be simultaneously active on the same line, or one could specify that CFB is to be invoked only if CW can not be invoked (for instance, if the user's line is fully busy, i.e. when the user is already involved in two calls), or one could even imagine to alter CFB, which could be activated only if the user does not accept the CW and hence would become Call Forwarding on No Reply.

This example shows that interaction may proceed from a choice. Another common example, interaction between Freephone and CFU (CCITT Recommendation Q.1201 [1]), shows that service interaction is not to be considered as a technical problem, and that solving an interworking problem is not deterministic at all; this interaction may be handled in three different ways, independently of any technical consideration:

- freephone calls attempts shall be forwarded like other terminating calls;

- a freephone destination shall **not** be selected for freephone calls, if it has activated CFU;

- a freephone destination shall be selected for freephone calls, **even** if it has activated CFU.

6.1.2 Goals of a feature

Considering example of CFU and ARC in subclause A.3.2, it may be noticed that the choice of a solution depends on the goal the service user has when activating CFU. Such a user may want to forward his calls either because he wants to be got through at the forwarded-to network address, or because he wants not to be disturbed by any phone call.

The coexistence of several possible goals for a given feature makes more complex the treatment of a policy interaction, for what fits a user may be unpleasant for another one.

6.1.3 Service provider's choices

The treatment of an interaction case has to be considered in two different steps:

- as a first step, during the specification phase, the service provider's technical staff analyses the interworking between the two services and deducts the interaction cases which may be a problem for any of the actors;

- as a second step, both technical and commercial staffs make choices amongst the different possible solutions, taking into account:

    - technical possibilities such as network capabilities;
    - ergonomical constraints (which solution will be the easiest for the users?);
    - sociological data (which solution will be the most admissible for the users?);
    - economic factors (does the perfect ergonomical solution bring an excessive extra cost?);
    - user's requirements.

But the service provider's choice may also be to let the user himself decide of the way the two services will complete together, i.e. to leave to the final user the choice of the interaction problem which have been brought up.

6.1.4 Customer's choices

According to the pair of interacting services, the customer's choice may intervene in different states of the service life cycle.

In the case of interaction between CFU and ARC (figure A.4), the user may choose between three solutions when he subscribes to the service. In the case of interaction between CW and CFB, the user may choose at activation time if he prefers to give the priority to the CW with CFB only in case of full busy, or to give the priority to the CFB.

Another example is given by the case of a user having set up his subscription profile such that he has a wake-up call at 05.30 a.m. each working day. One day, he activates a CFU to a friend. What happens when the morning wake-up call arrives? One can consider that probably the CFU is activated because the user accidentally forgets to deactivate it after having spend a couple of hours at a friend's place, thus the CFU has to be overridden. But one can also think that maybe the user is still staying at a friend's place (e.g. watching the cat while the friend is away) and actually wants his wake-up call forwarded. Typically, that kind of interaction choice may be included in the subscriber's profile.

6.2 A solution: when?

6.2.1 Interaction life cycle

An interaction is characterized by an unexpected and somewhat defective service demeanour, whereas service design shall always result in a healthy system behaviour.

When investigating the reasons of such a defective demeanour, it appears that, for several cases mentioned in annex A, an interaction between two services is characterized by an execution problem generated by a **previous data modification**. In every case, the activation of a service (in annex A, CFU) is actually at the origin of the interaction:

- in figure A.1, from A's point of view, because of the activation of CFU, there is a dysfunction when activating TKCS;

- in figure A.4, from C's point of view, because of the activation of CFU, there is a dysfunction when invoking ARC;

- in figure A.5, from A's point of view, there is an annoyance (not really a dysfunction) because of the activation of CFU.

This remark leads to introduce the concept of **interaction life cycle**.

To help to a better understanding, in subclause 6.2.2, an interaction manifestation is compared with another kind of defective demeanour: human illness. Both schemes include the same phases and the same relationships between these phases, making the life cycles quite similar.

6.2.2 Comparison between interaction life cycle and disease life cycle

Figure 1 describes the following moments:

- without intervention, the disease follows its "wild life cycle": contamination (e.g. by a virus) prepares an illness;
- medical research discovers the process "contamination/illness";
- a first set of solutions is inspired by results of medical research. Prophylaxis avoids contamination by appropriate means. This applies for instance to:
  - cases where vaccination impedes contamination from provoking any manifestation; and
  - cases where the contamination process is known, allowing implementation of all possible techniques to impede transmission of the virus;
- a second set of solutions is prompted by pharmaceutical research, which enables to finalize a care in order to alleviate illness manifestation. Diagnosis sees illness, and triggers such a care. This curative treatment applies to the cases where the contamination process is not overcome, so making prophylaxis inefficient;
- a third set of solutions can be finalized. Here as well, the contamination process is not overcome (e.g. prophylaxis has been inefficient), but pharmaceutical research implements methods to detect contamination and an appropriate treatment is triggered to prevent illness.

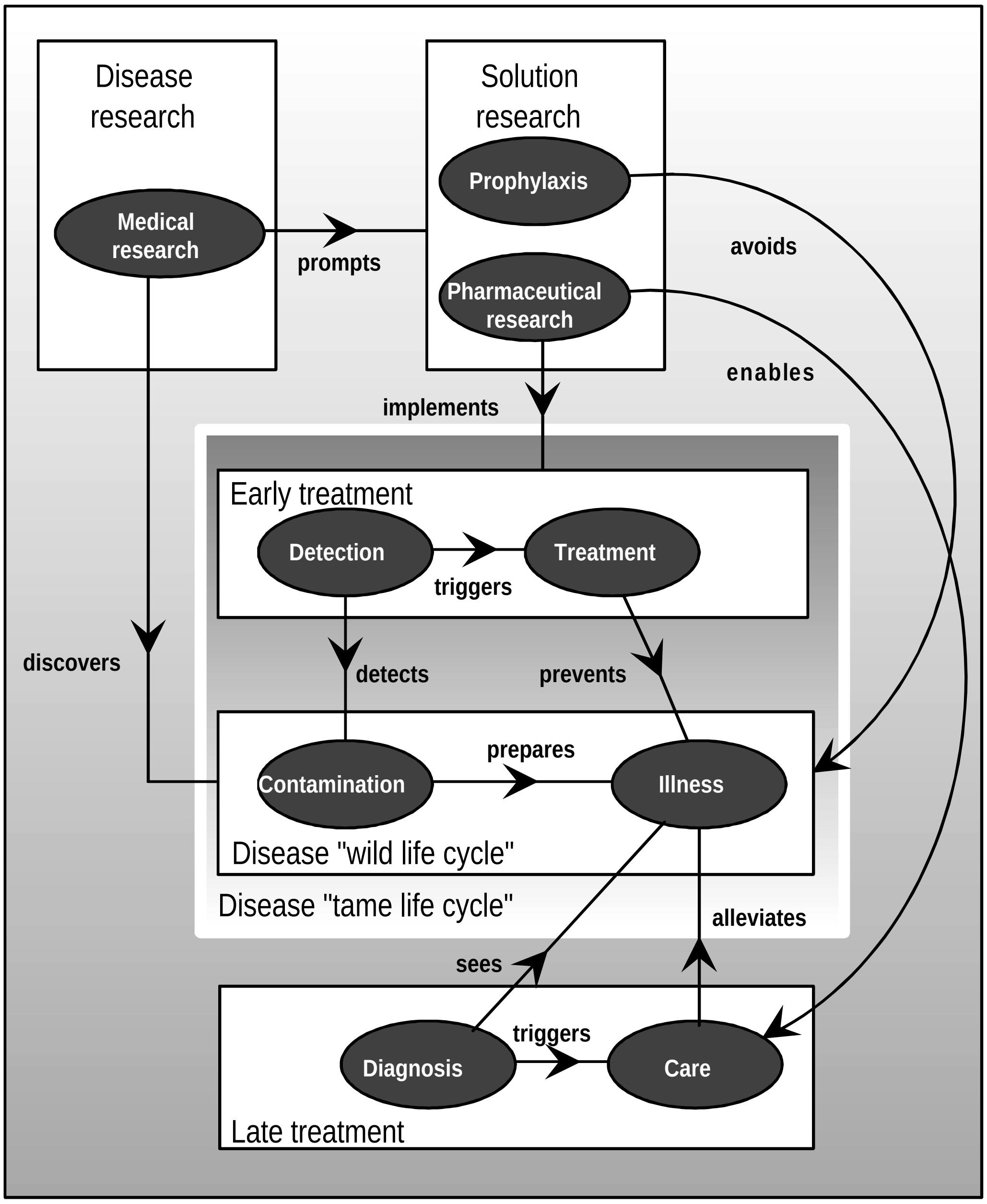


Figure 1: A disease life cycle

This model can be applied to the service interaction process.

6.2.3 The phases of the interaction life cycle

6.2.3.1 Interaction "wild life cycle"

As a first step, consider interaction between two conflicting services left to itself, without any kind of intervention of the service provider, in the frame of its so-called "wild life cycle" (figure 2).

6.2.3.1.1 Interaction germination

Interaction germination is defined as a data[2)] modification or initialization which prepares an **interaction manifestation** that may occur either later on in the same call or in a further call, implying risks of dysfunction.

Interaction germination can take place either at service management (data modification or service activation) or at service invocation.

6.2.3.1.2 Interaction manifestation

Interaction manifestation, defined as the moment when an interworking between services causes a situation viewed as unsatisfactory from any of the actors, occurs during service invocation. It happens:

- either when a service disturbs or inhibits the expected execution of another service considered separately (or of another instance of the same service);
- or when the joint accurate executions of services creates new unforeseen behaviours which would not appear in case of executions considered separately.

6.2.3.2 Interaction research

Interaction research takes place during service creation, and is therefore performed in the Service Creation Environment (SCE). It is made of three parts:

- **spotting** aims at finding as many interaction cases as possible, by analysing the interworking between the service to be introduced and the existing ones;
- **handling study** aims at choosing, for each case, a solution among the possible ones (according to network capabilities, ergonomical constraints, sociological data, economical factors and user's requirements);
- **avoidance** aims at decreasing the number of cases to be handled.

2) The notion of data is to be considered in a broad sense, to integrate the concept of resource.

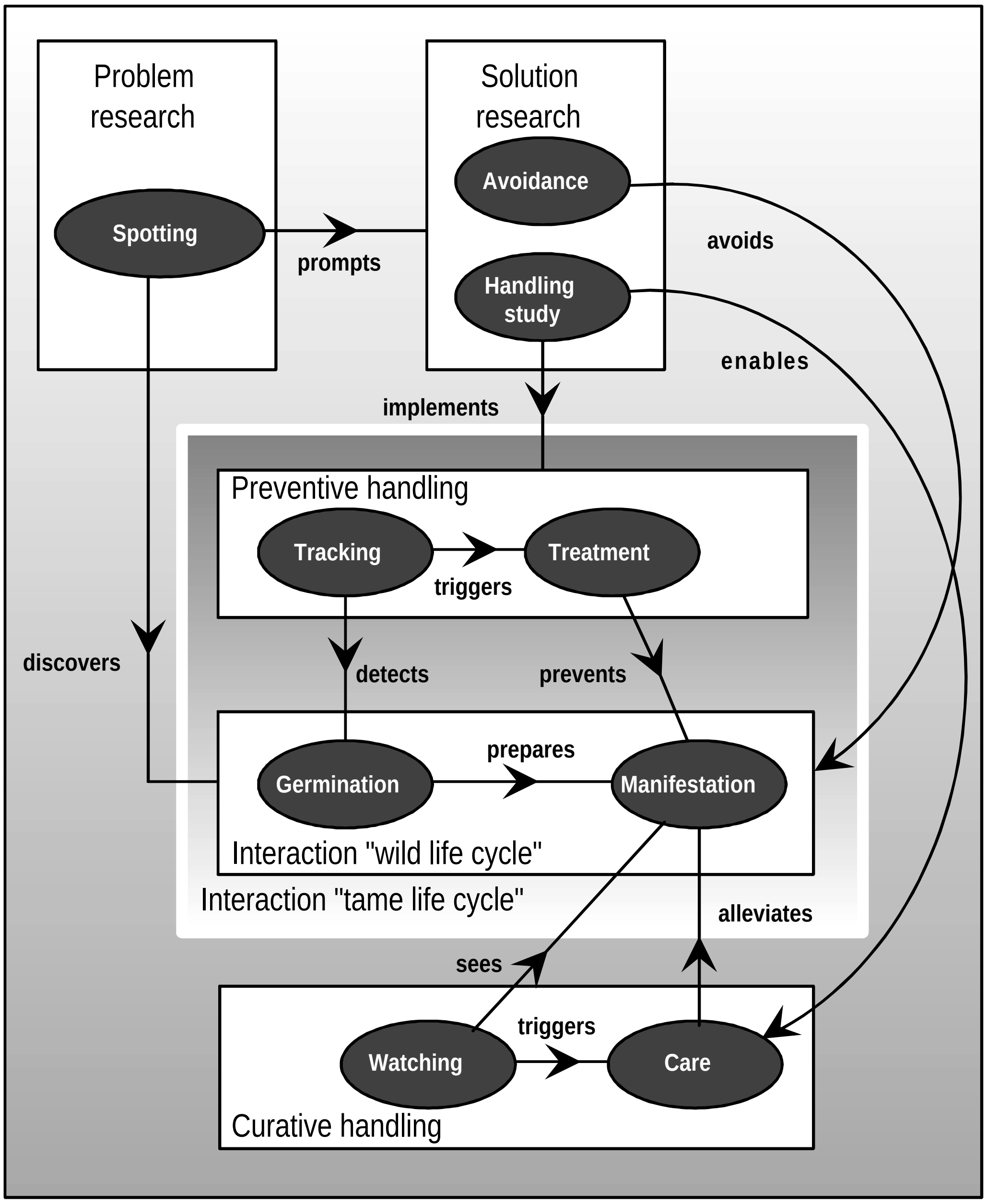


Figure 2: The interaction life cycle

6.2.3.2.1 Interaction spotting

Interaction spotting takes place during the various phases of service creation. It consists in the analysis of the new service model, in conjunction with already existing service models, in order to find as many interaction cases as possible.

One of the basic aims of **interaction spotting** is the discovery of potential **interaction manifestations**, as well as **interaction germinations** which prepare them, in order to facilitate subsequent **interaction handling**.

Note that we use the term **interaction spotting**, to characterize the moment when an interaction is foreseen at service **creation**, in order to make a distinction with "detection" at service **execution**, for which we shall propose other wordings. In the technical literature, the term detection is often used in both cases.

6.2.3.2.2 Interaction handling study

Interaction handling study is performed by both service provider's marketing and technical staffs. It evaluates the gravity of each spotted case and, for those that can not be solved by avoidance techniques, deduces handling mechanisms necessary to their detection and treatment:

- **late (**or **curative) handling** watches interaction manifestation and tries to alleviate its effects;

- better, **early (**or **preventive) handling** intends to forbid germination by tracking harmful data configurations.

6.2.3.2.3 Interaction avoidance

Interaction avoidance takes place during service creation. It aims at eliminating, once and for all, some interaction cases, thus can be compared to prophylaxis methods. We consider as interaction avoidance the development of more advanced terminals, the introduction of new signalling capabilities in the network, as well as the redefinition or restriction of service functionalities, or the determination of acceptable service interworking.

6.2.3.3 Curative handling

Curative handling, acting at execution level inside the Service Execution Environment (SEE), watches and remedies to an **interaction manifestation** already visible.

6.2.3.3.1 Interaction watching

Interaction watching is the observation of service execution, in order to ensure that no **interaction manifestation** occurs and, in the opposite case, to trigger mechanisms designed to take care of this manifestation.

6.2.3.3.2 Interaction care

Interaction care includes the processing of any mechanism designed to solve an **interaction manifestation** after its detection by **interaction watching**. Since the manifestation is already visible by one of the actors, it can not be avoided, and its effects can only be alleviated. For instance, the calling user may be prompted to take a decision related to the completion of his call.

6.2.3.4 Preventive handling

Preventive handling intends to avoid any **interaction manifestation** by implementing mechanisms tracking **interaction germination** as soon as it occurs. Interaction process is then totally controlled and interaction performs its so-called "tame life cycle". Such a handling also takes place inside the SEE.

6.2.3.4.1 Interaction tracking

Interaction tracking is defined as the moment when an **interaction germination** can be observed during service management (data modification or service activation) before any **interaction manifestation** occurs. It allows arming of mechanisms which will enable **interaction** treatment.

An observation mechanism may have been introduced in the service logic, and an observable event may have been fabricated, in order to enable **interaction tracking**.

6.2.3.4.2 Interaction treatment

Interaction treatment is the processing of complementary mechanisms designed to solve an unsatisfactory interworking situation, which may have germinated either in the same call or in a previous call.

Such a processing is a consequence of **interaction tracking.** However, it may take place either **before** (if it is preventive), **while** or **after** (if it is curative) **interaction germination.** Refer to the examples:

- in cases 19 to 24 in clause A.4 (case **before**), if B wants to activate a CFU to C while C has entered B in his blacklist, these services will include incompatible data, and a clever management is to forbid the data modification that creates this incompatibility;

- in figure A.2 (case **while**): a possible solution is to take into account a predetermined invocation order;

- in figure A.4 (case **after**): a possible solution to the conflict is to not take into account the second service.

6.2.3.5 Some semantic precision

The word "detection" is not use to name the phases of the interaction life cycle, for it is a generic term usually turned to fit any case, from the moment when an interaction is **foreseen** at service **creation** to the moment when its **manifestation** is noted at service **execution**.

If "detection" may be kept as a generic name (though it implies some ambiguities), three distinct activities in interaction resolution are christened after three terms keeping the semantic peculiarities of common English language:

- **spotting**, which implies an idea of location of targets and an idea of investigation, seems perfectly applicable to the active task to be performed during service creation;

- **tracking**, which implies an idea of following a (moving) target, fits to the active task of detecting germination at service execution;

- **watching**, which implies an idea of keeping under observation and an idea of continuous attention, suits the passive task of noticing interaction manifestation after it occurred.

6.3 A solution: how?

6.3.1 Towards an integrated solution

Refer to the interaction life cycle and to its comparison with a disease life cycle:

- as a first step, the process "contamination ➜ illness" is discovered by medical research. By analogy, the "germination ➜ manifestation" process, so-called "wild life cycle", is spotted. The **spotting** result includes the description of each interaction manifestation, but also the characteristics of the associated germination;

- this discovery is followed by a phase in which decisions of disease handling have to be made, according to the results of pharmaceutical research. This corresponds to the **study of interaction handling** in which choices are made, taking into account technical, sociological, ergonomical and economical constraints. The solution of an interaction case never proceeds from determinism, and the service provider may choose among open possibilities, which may be implemented via three techniques:

  - prophylaxis propositions intend to suppress the contamination by appropriate means, such as vaccination or impeding transmission of the virus. This corresponds to the so-called **avoidance techniques**, implemented in the SCE: service specifications refinements, underlying network adaptations and signalling systems enhancements are proposed to impede germination. Thus interaction "wild life cycle" is suppressed;

- when prophylaxis is insufficient, one may pin all hopes on a curative treatment to alleviate illness manifestations. This corresponds to the implementation of an execution arbitrator which has, during service execution and according to the choices previously made, to watch and deal with interaction manifestation. It is implemented in the SEE;

- when pharmaceutical research offers a preventive treatment, methods to detect contamination are implemented and the treatment is triggered to prevent illness. This corresponds to the implementation of a **management arbitrator**, data manager able to track germination and to treat it according to the choices previously made. It is implemented in the SEE.

None of these methods is self-sufficient, and only a mix may get through the whole set of interaction problems, in order to tackle the service interaction problem as early as possible in both environments (SCE and SEE) by merging different compatible approaches.

6.3.2 Spotting in the SCE

Since spotting is a prerequisite for avoidance or handling, the investigation shall primarily consist in spotting all potential interaction manifestations and germinations. Spotting is a complex activity, due to:

- the increasing number of new services;

- the intrinsic complexity of many services and of the different ways in which services can interact;

- the fact that interaction can occur between more than two services;

- ergonomical and sociological aspects involved by the notion of unwanted or unforeseen behaviours.

Due to this complexity, spotting is further studied in subclause 6.4.

6.3.3 The resolution techniques

6.3.3.1 Avoiding interaction

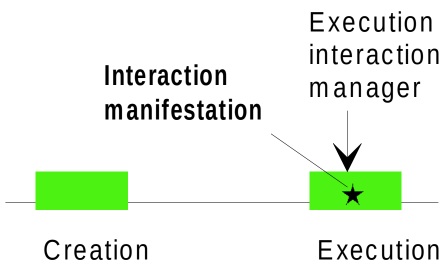
After spotting activity has been performed, one has to investigate if the interaction problem can be avoided, applying the techniques listed above (service specifications refinements, underlying network adaptations and signalling systems enhancements). One shall take into account the various actors' interests, looking for the best compromise between them.

Note that when an interaction is avoided during the creation process, the decision taken and the causes of such a decision shall be carefully recorded. A future evolution of either the network or the services might involve reconsidering restrictions brought to service functionalities.

6.3.3.2 Tracking interaction germination

Interaction resolution may be confided to a manager (machine, functional entity) which intervenes at service execution (figure 3a). In such a case, interaction is solved when its manifestation occurs, with the traditional scheme: some causes generate effects, thus it is necessary to repair damages.

But if interaction is managed as soon as germination is detected, i.e. a specific management arbitrator is invoked at each time data are created or modified, the need of intervention at the call processing level is minimized (figure 3b). This process looks more mature: having detected the causes before possible problems, there is no effect (it could be compared to a treatment applied after a systematic virus screening, and efficient before any symptom is visible). It restricts as far as possible the need of intervention at call processing level.



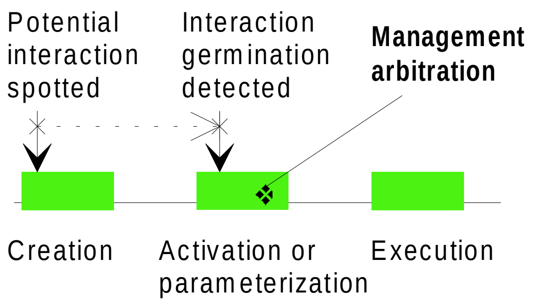

Potential interaction spotted
Interaction germination detected
Management arbitration
Creation
Activation or parameterization
Execution

Figure 3a: Interaction process without management at germination

Figure 3b: Interaction process with management at germination

Such a prevention can be performed by detecting germination, thus allowing the tracking entity to trigger intervention of a management arbitrator, designed to solve interaction cases during service registration or activation, by preservation of data consistency either inside a user's profile or between several user's profiles. A leading role may be devoted to the user, who could have to take a decision related to the current service context.

For the user as for the service provider, consequences are positive:

- the user is no more informed of a difficulty when he places or receives a call, but when he modifies his profile, i.e. at a moment when he is generally more free-minded;

- the system reaction time can be longer in a management phase than in a call processing phase, for real time constraints are not the same, and this simplifies the service provider's task.

Applying this principle to the pairs of services taken in example above:

- for case in figure A.1, the CFU user may be asked, when activating the service, to give the forwarded-to line's key code: the germination is definitely detected as soon as the activation, i.e. during user's management;

- for case in figure A.4 germination may be anticipated by asking the user to choose the interaction behaviour either at provision or at activation.

Some feed-back is to be given to the SCE, in order to provide information about the problems met at execution.

6.3.3.3 Watching interaction manifestation

But there is a limit to interaction management at germination: in figure A.5, no detection is possible in a management phase (nothing could indicate that, in a future call, a link between users A and B will exist).

Furthermore, performing though the tools available in the SCE could be, they can not allow a service designer to be exhaustive in the stage one completion and in the spotting of interaction. Experience learns that all the possible goals of a service may not be foreseen by a service designer, all the more so the tricky policy interaction cases.

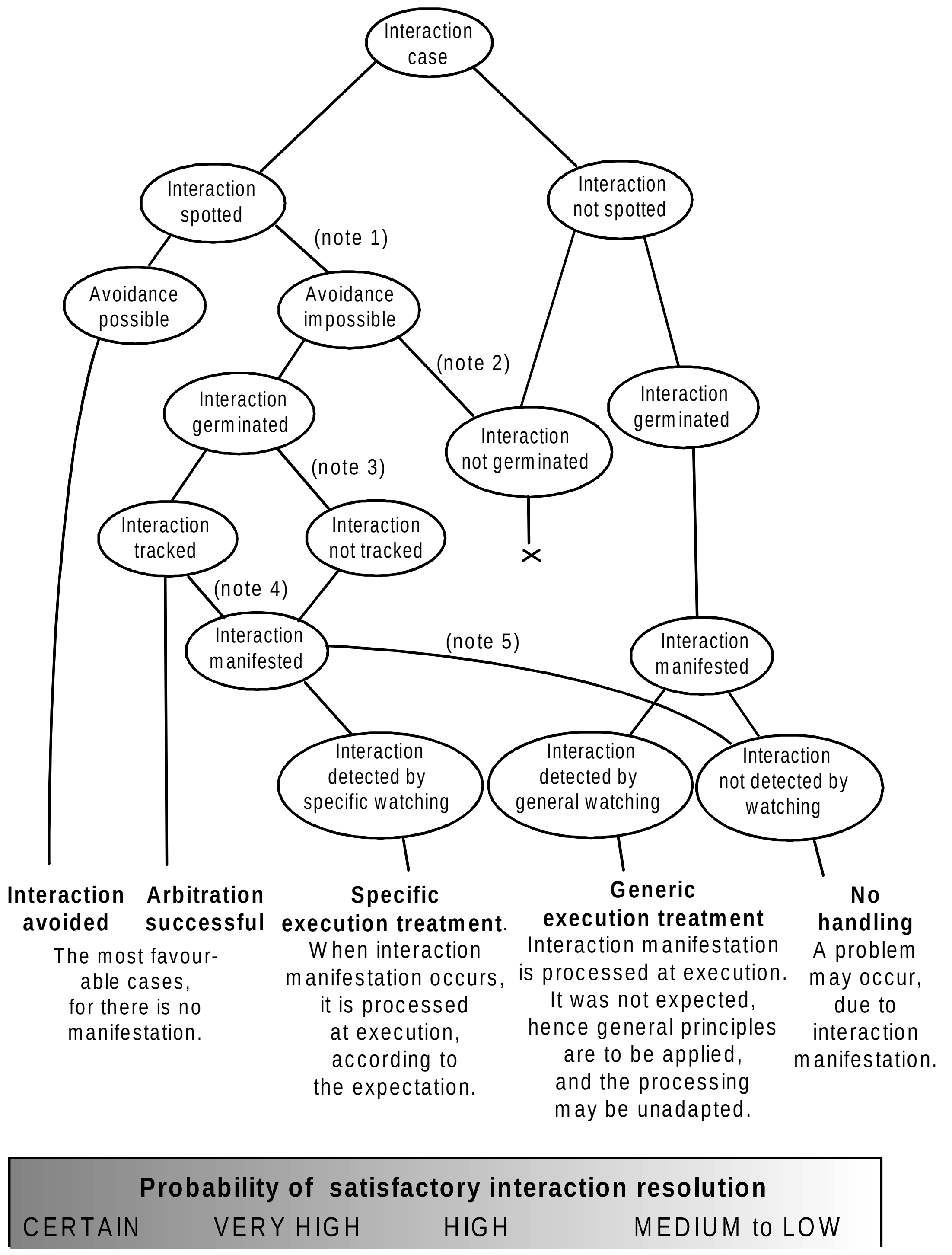
A mechanism in call processing is thus necessary, and spotting results may lead to adapt the execution arbitrator. Triggered by a watching entity, execution arbitration (similar to a curative medicine absorption) aims at resolving interaction at invocation, i.e. when its manifestation occurs. It takes care of cases for which germination can not be detected (even if it has been spotted) and for which a management arbitrator would be either less efficient or too costly with respect to the real problem. It might be extended to unspotted interaction cases.

Here as well, some feed-back is to be given to the SCE, in order to provide information on the problems encountered during service registration or activation.

Such an interaction management at service execution level concerns the DFP and is developed in subclause 8.1. Two main approaches, the Feature Interaction Manager (FIM) and the negotiation concept, are described.

6.3.3.4 Recapitulation of the possible scenarios

Figure 4 summarizes the possible ways to solve an interaction case, indicating the probability of satisfaction associated to each of the scenarios.


Interaction case
Interaction spotted
Interaction not spotted
(note 1)
Avoidance possible
Avoidance impossible
(note 2)
Interaction germinated
Interaction not germinated
Interaction germinated
(note 3)
Interaction tracked
Interaction not tracked
(note 4)
Interaction manifested
(note 5)
Interaction manifested
Interaction detected by specific watching
Interaction detected by general watching
Interaction not detected by watching
Interaction avoided
The most favour-able cases, for there is no manifestation.
Arbitration successful
Specific execution treatment. When interaction manifestation occurs, it is processed at execution, according to the expectation.
Generic execution treatment Interaction manifestation is processed at execution. It was not expected, hence general principles are to be applied, and the processing may be unadapted.
No handling A problem may occur, due to interaction manifestation.
Probability of satisfactory interaction resolution
CERTAIN
VERY HIGH
HIGH
MEDIUM to LOW

NOTE 1:	Avoidance techniques do not apply.
NOTE 2:	For these call instances, data configuration was such that no interaction occurred.
NOTE 3:	Germination can not be detected.
NOTE 4:	The service designer's choice is to intervene at service execution. It may occur either when management arbitration is too complex or too costly, or when processing at execution is foreseen to be sufficient.
NOTE 5:	Interaction has been spotted, but no handling has been implemented. It means that the service designer's choice was not to treat interaction, e.g. because the cost of the treatment is not worth the candle compared to the minor consequences of the manifestation.

Figure 4: Possible scenarios for interaction resolution

6.3.3.5 Automating and standardizing interaction management

As seen above, it is not difficult to imagine several solutions to resolve an undesired interaction. Furthermore, it appears clearly that interaction is to be taken into account as soon as its detection is possible. Two main problems remain:

- between two services enriched with several features, there is a very large number of possible interactions, which leads to avoid to consider all possible cases with the intention to solve interactions on a case by case basis;

- in open network scenarios, service designers having shallow knowledge of the network and no responsibility to preserve its efficiency could be empowered to define how the network is to handle their calls.

The solution of these two problems requires:

- a high level of automation, both in the computerized spotting of the possible desired interactions themselves and in the setting of automatic mechanisms that allows to manage the interaction occurring during processing and charging of the calls;

- a good level of standardization and a **coordination authority**, to allow the interworking of complex services over different networks, or in the same network over different service providers[3].

The coordination authority could manage an administrative environment to aid concurrent, independent service development by screening developers from implementation level conflicts with other services or service features. For example, this administrative environment should support assignment of global attributes, library functions available to developers, monitoring facility for all global system data, abstract data definitions, etc. It should also receive feed-back from the interaction arbitrators and from other network sources, in order to be able to formulate recommendations regarding the base of rules and the service descriptions (see figure 7).

The achievement of this authority may be difficult:

- certification of a service, i.e. guarantee that all possible interactions are identified and handled, is likely to be an issue;

- it may be technically difficult to come to a common understanding on the principles to be applied to the policy interaction cases;

- in addition to the technical difficulties, the service providers, especially in case of competition, may consider that information on planned services is sensitive.

A co-ordination is required for each service addressing domain. International standardization bodies can take care of the technical aspects of this co-ordination for the international services.

---

[3] It is to be remarked that interaction is not only related to service call processing, but also covers service charging. For instance, when a mobile makes a call, the radio part is charged on the mobile account, even if the called number is a Freephone number; for the caller, the call is not totally free any more. If the service provider wants this radio part to be charged on the Freephone account, a charging interaction management is necessary.

6.4 Spotting in the SCE

6.4.1 Two kinds of approaches

As explained earlier, the interaction investigation shall primarily consist in spotting all potential interaction manifestations and germinations.

Two kinds of methods can be applied: **heuristic** and **formal**. One cannot expect complete spotting of all interaction cases from the development of a single and unique method. Formal methods are necessary to help the service designer in performing the long, detailed and exhaustive checking of all the possible interactions. But all service requirements cannot be easily expressed in a formal language, and in some cases, interworking between services will lead to strange behaviours which can be analysed only by specialists. So better investigation of the resulting behaviour will be obtained by integrating heuristic and formal methods into a consistent set.

For both kinds of methods, service providers may decide to build tools for a common analysis of service interaction, that could be partially standardized.

6.4.2 Formal approach

Spotting takes place during the specification phase, and could be located in the SCE. Analysis by the service designer should permit to identify desired and undesired interactions between IN-based services. The process could be extended to identification of interactions between IN and non-IN services.

As each service can include a set of features making it richer but more complicated, specifications become long and detailed, making it difficult to identify which among the features of the new service may interact with some of the features of already existing services. Considering that interactions between more than two services are to be analysed, as well as interactions between more than two parties involved in a call, it becomes obvious that the exponential growth of the number of services makes it impossible to spot interaction through non-automatic ways. The complexity which results of the deployment of many independently conceived services implies the need of precise specifications and improved methods to integrate these services and to spot interactions.

In order to allow efficient spotting of interaction cases, the description of a service can be modelled according to an adapted formal description method. The following paragraphs aim at giving examples of such a use of formal descriptions, without intention neither to be exhaustive nor to recommend the mentioned techniques.

For instance, the behaviour of the underlying network and services in this network may be modelled by concurrent communicating Specification and Description Language (SDL) processes allowing the construction of a complete system representing the network and services behaviours. Then, to verify the correctness of the system behaviour, the desired requirements of the services and general properties of the network may be expressed in a formal way using branching time temporal logic, before checking whether the complete system verifies these formal properties. Different other tools may be used, such as the Language Of Temporal Ordering Specification (LOTOS) specification or Object-Oriented methods. The latter leans on the fact that introducing a new service may imply modifications of basic network concepts, and consequently of definition of existing services, so provoking interactions. The use of Object-Oriented techniques during the analysis activity before the formalization of specifications may help to investigate evolving concepts and relations between service definitions, and to spot potential interactions by analysis of the common use of data or resources (e.g. conflict between service invocations and the basic call).

After the spotting of interaction through its manifestation, i.e. when the violation of a requirement is detected, the verification tool gives the scenarios which lead to this manifestation: the germination can then be investigated.

6.4.3 Heuristic approach

Heuristic methods are based on knowledge rather than formalisms. Some ideas are expressed in this subclause, which are to be considered as a guidance for service providers.

6.4.3.1 Phases of service life cycle to be examined

The **service life cycle** is the description of both stages and steps involved during the complete life of any service, in a service-independent manner. It is considered the basis defining the possible behaviour of a service at all times, the stages identified covering all aspects of the service life (including its "death").

Figure 5 highlights the set of phases and transitions of the service life cycle in which interaction may occur and is therefore to be spotted by service designers during service creation.

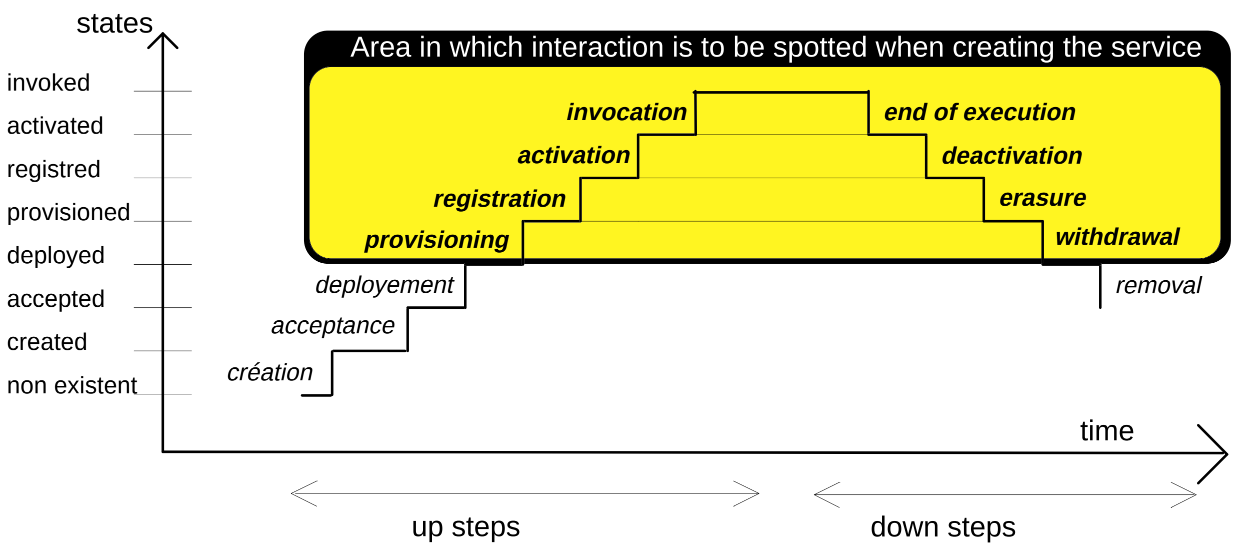


Figure 5: Interaction and service life cycle reference model

6.4.3.2 Combinative state/transition examination

The example of interaction between CFU and TKCS (figures A.1 and A.2) shows that two services may present several different kinds of interaction, depending either on the event occurring in the processing of the first service and on the considered phase of the second service.

These cases may be revealed by considering the service life cycle as a finite state machine with a state/transition model, looking for all the possible events due to a service S2 that may have an impact on the service S1 execution. Hence all the combinations between the **event** of a service S1 (i.e. a **transition** as introduced above) and the **state** of a service S2 have to be considered.

The finite state machine permits to define, in an intuitive way, notions of "superior" and "inferior" states, from which may be derived a classification of the transitions. A transition is said:

- **upward** when final state is superior to initial state;
- **downward** when final state is inferior to initial state;
- **stationary** when final state is equal to initial state.

An exhaustive interaction study will consist in the search of the impacts of five upward or downward S1 transitions (initialization, activation, invocation, end of execution, deactivation) and of three stationary S1 transitions (modification of initialization data, modification of activation data and modification of invocation data) on four S2 states (subscribed, initialized, activated and invoked), as shown by figure 6.

6.4.3.3 Services invocation order

The interaction of a service S1 with a service S2 may not be the same as interaction of the service S2 with the service S1.

Refer to figures A.4 and A.5: in the first case, when CFU has been invoked before ARC (the call forwarded is the direct call), the interaction has no impact; but when ARC has been invoked before CFU (the call forwarded is the reverse call), an interaction mechanism is necessary.

Thus, for each pair of services, two ways of interaction have to be considered: interaction of S1 with S2(i.e. impact of S2 transitions on S1 states) and interaction of S2 with S1 (i.e. impact of S1 transitions on S2 states). The combinative state/transition examination is to be performed twice for each pair of services.

6.4.3.4 Interaction of a service with itself

A service interaction may occur when a service interacts with itself (for instance, CFU). Hence the combinative state/transition examination is to be performed for the service against itself.

6.4.3.5 Interaction of services pertaining to several users

The way an interaction case pertaining to a single user is solved depends on the user's interpretation whether this interaction is positive or negative. When two users are involved in a call, the problem becomes more complex, for user A's view may be different from user B's view. For instance, in the example of ARC and CFU in figure A.4, A wishes to see his call back completed while B could wish not to be disturbed to all, even if he previously decided to place a call to A.

The complexity of the problem is growing when considering a service where multiples users are involved, and the spotting of interaction becomes complex. A good spotting method is the permutation of the call actors. An example is given in clause A.4.

6.4.3.6 Rules applying to data

6.4.3.6.1 Service Information Model (SIM) and Global Information Model (GIM)

When a new service is introduced, a SIM is defined, in consistency with the GIM. Some of the data defined for the new service are already existing in the GIM, some others have to be created and included in the GIM. Each data of the GIM contains the list of all the services using them. When a new service uses data already defined in the GIM, there are potential interactions with each of the services from the list. When a service is modified, for instance completed with a new feature, the SIM is modified, and the same principles apply.

Moreover, some rules concerning the data structure may be defined and used for the expertise. For instance, when a Public Switched Telephone Network (PSTN) phone number used as Freephone translated number, if the PSTN subscriber withdraws his subscription or if his telephone number is changed, the modification must be taken into account in the Freephone database: in such a case, integrity constraints have an impact on interaction between two services.

6.4.3.6.2 Rules to access a shared database

The same data defined in the GIM may be accessed by both network part and management part, which entails an interaction between management applications and real time applications to access a shared database. The service designer has thus to define the rules to access a shared database and to manipulate data, and the links between data.

The access to data in a shared database is executed by means of transactions, units of consistency characterized by the ACID properties (Atomicity, Coherency, Isolation, Durability). The standard transaction type mechanisms solve a large part of interaction problems relevant to access concurrence, within the various functions including data bases: Service Management Function (SMF), Service Control Function (SCF), Service Switching Function (SSF).

But besides these general rules, rules related to the existence of different actors (network operator, service provider, service subscriber, end user) are to be set. For each data, access rules (write, read) are defined for each actor. To simplify this definition, data may be grouped according to some characteristics. From these data definition, the interaction between actors can be observed, and rules about priority to access the same data can be determined, according to the access type (read, write) and, on a same access, according to the actor.

6.4.3.6.3 Data associated to the service life cycle

To each state of the service life cycle, correspond data. The amount of information managed by the system increases at each upward transition, as defined in subclause 6.4.2.3, since new data are supplied by the customer or by the service provider:

- provision data, recorded during service instance provisioning, remain till service instance withdrawal;
- registration data, initialized at registration, disappear at erasure;
- activation data, initialized at activation, disappear at deactivation;
- invocation data are ephemeral and invocation context dependant.

Table 1 indicates, for each kind of transition, whether data are created, modified or suppressed.

Table 1: Evolution of data for each type of transition

	Data creation	**Data modification**	**Data suppression**
Upward transition	usually	sometimes	NO
Stationary transition	NO	YES	NO
Downward transition	NO	sometimes	usually

It has to be noted that data related to a state are persistent during an upward transition. For instance, during the invocation of the service, four types of data (provision, registration, activation and invocation) are used. Nevertheless, whatever the transition may be, it is possible, according to the service, that data related to an underlying state are modified (for instance, when recording activation data as implicit registration data for the next activation).

Figure 6, which shows as an example a service instance with two invocations during the first activation and a second activation without invocation, recapitulates:

- the states that apply to a service instance;
- the upward and downward transitions (black arrows);
- the stationary transitions (white arrows);
- the different types of modifiable data.

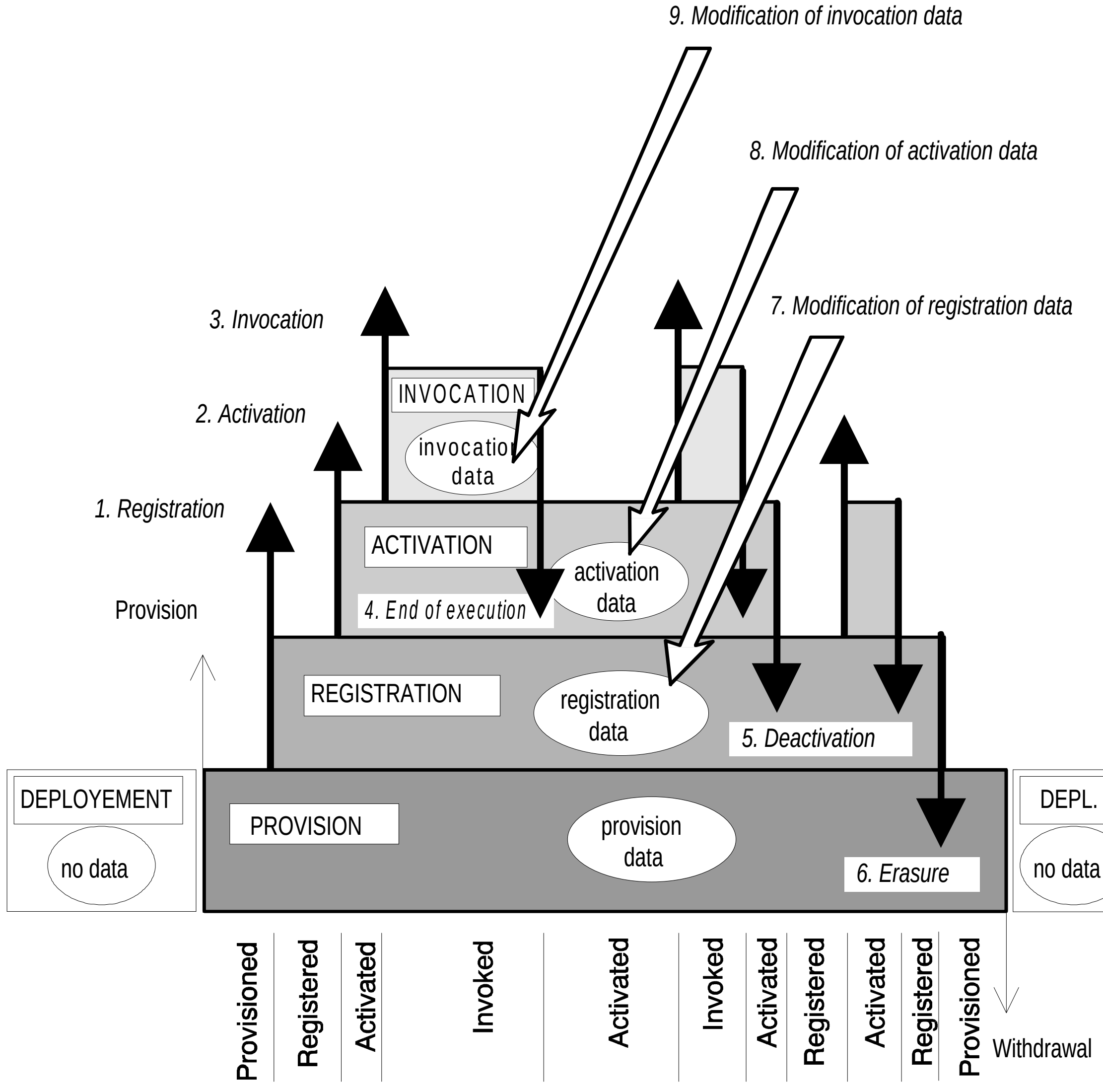


Figure 6: Impact study of the S1 transitions on the S2 states

6.4.3.7 Rules applying to service behaviour

6.4.3.7.1 General principles

The automated analysis of the stage one description of a new service to be introduced may use a rule based method. Such a rule based approach is well adapted to SCE, due to its flexibility (rules can be easily added or modified) and its ability to address complex situations by using heterogeneous criteria.

From some examples mentioned in annex A, one can notice that knowledge of both services interacting was a precondition for spotting the interaction cases. In other words, the desired handling cannot be identified before the specific call scenarios are identified. This implies that the definition of a first set of rules require a previous analysis of a large number of interaction cases.

6.4.3.7.2 Example of possible rules

Two kinds of rules may be defined, logically demonstrated rules and simplifying service provider's choices. The first set could be standardized, while the second one is peculiar to one service provider.

Examples of such rules applied to service invocation are given hereafter (they are not demonstrated, but could be assumed as true):

Rule #1: No routeing restriction applies to outgoing calls which are not charged on calling line.

Rule #2: Calls placed by an ARC mechanism are not to be considered as incoming calls from the first caller's point of view, but as pseudo-outgoing call.

This rule, to be considered as a service provider's choice, means that calls placed by an ARC mechanism are not submitted to incoming restrictions applying to first caller's line, due to the fact that the first caller did intend to place a call to the person using ARC: according to this rule, an ARC call will override first caller's TCS or TKCS.

Rule #3: Before rerouteing a call, the service logic shall scan whether a routeing restriction applies to the new called number.

This rule is directly inferred from **service typology**: when N services exist and a new service is added, the number of interaction descriptions (N) may be decreased by using the service typology, which allows to group several services on a single case study. Only P descriptions (P < N) are thus necessary. It proves that the rule base grows when a new service is designed, and/or when the service typology is refined. For instance, when adding the Sub-addressing service, two rules at least are to be added:

Rule #4: Each service using **calling** line identity may take into account the expanded sub-address information.

Rule #5: Each service using **called** line identity may take into account the expanded sub-address information.

6.4.3.7.3 Outcome of the rules

The outcome of the expertise could be a verdict which must be considered as a help for the service designer, who will have to take into account the service provider's operation choices. For instance, a Freephone call whose Freephone number is expanded by a sub-address may be routed even if the sub-address cannot be added to the translated number: in such a case, the rule #5 is not respected.

The expertise verdict should include:

- the lists of services with which the interaction **has**, or **may have**, or **does not have** an impact;
- values for filling up the FIM tables as mentioned in subclause 8.2.5;
- other hints about the interactions and their impact.

6.4.4 Result of the automated spotting phase

During this phase of analysis, the aim is to spot the greatest number of possible interactions between the new service and the existing ones. The verdict permits (figure 7):

- enrichment of principles and rules used in information model analysis and service typology analysis;
- introduction in the network of new generic mechanisms applying to the execution environment;
- determination of the improvements of the signalling necessary to handle these mechanisms;
- modifications to the new service stage one description.

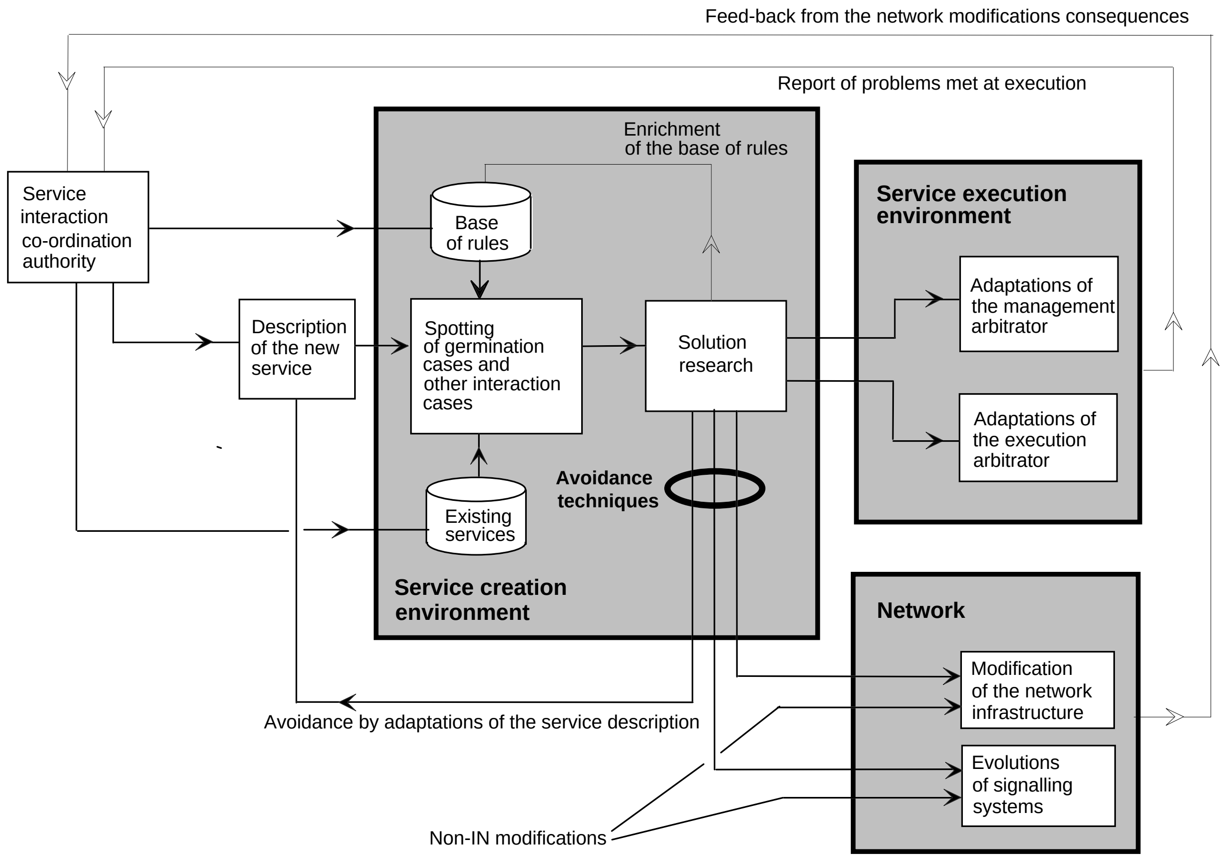

Feed-back from the network modifications consequences
Report of problems met at execution
Enrichment of the base of rules
Service interaction co-ordination authority
Base of rules
Service execution environment
Adaptations of the management arbitrator
Description of the new service
Spotting of germination cases and other interaction cases
Solution research
Adaptations of the execution arbitrator
Avoidance techniques
Existing services
Service creation environment
Network
Modification of the network infrastructure
Avoidance by adaptations of the service description
Evolutions of signalling systems
Non-IN modifications

Figure 7: General scheme of interaction management in a multi-provider environment

7 View from the Global Functional Plane (GFP)

7.1 Interaction between Service Independent building Blocks (SIBs)

This subclause describes how part of the interaction issues can be solved at GFP level.

Interaction management in the GPF consists in the investigation whether two or more SIBs have impact on each other. It is performed by comparing the descriptions of different SIBs, using the following rules:

- each SIB works on a set of basic properties, such as resources, data, etc.; there is only a very low probability for a SIB based on the manipulation of resources to interact with a SIB based on the manipulation of data;

- the mutual use of the same data or resources by two SIBs shall be exactly identified, for it may cause problems, for instance when the order in which the SIBs could be active is not defined;

- when a new SIB is being defined, the already existing SIBs have to be taken into account to avoid two SIBs to perform the same or almost the same tasks.

7.2 Interaction handling methods in the SIB definition phase

Interaction may occur when two SIBs use the same data or resources, which may occur either when both SIBs are active at the same time or when they are active sequentially.

For instance, one may consider the two SIBs "charging" and "user interaction". The first SIB increments a charging counter every minute, for a given call. The second SIB has to present the charging counter value to the user at the end of the call. If no specific precaution is taken, "user interaction" SIB may read the counter value before the final update by "charging" SIB. This specific precaution is a statement that could be written: **"user interaction" shall not be active on a charging data before or at the same time that "charging" is active**.

Interaction may be solved by preventing such a kind of situations, which leads to the following rules:

- two SIBs shall be prevented to manipulate the same data or resources (the mutual use of the same data may cause problems when the sequence in which the SIBs should be active is not defined);

- when two SIBs are felt in an interaction situation, this situation shall be prohibited in decreasing one of the SIBs functionality, that leads to increase its atomicity and to define smaller SIBs with more specific tasks.

8 View from the DFP

The view from the DFP is related to execution arbitration mechanisms mentioned in subclause 6.3, aiming to care interaction manifestation.

NOTE: In the following, the term **execution arbitrator** is replaced with **interaction manager**, to stick to a terminology already used in the first set of ITU-T Recommendations. Such a terminology is ambiguous, since we mentioned earlier than interaction could be considered either at management time (provision, registration, activation) or at execution time. Hence, in the future, it may be necessary to adopt a new terminology and to replace the terms build from FIM (i.e. Service Logic Interaction Manager (SLIM), Inter-Platform FIM (IPFIM)) by new terms built from IEA (**Interaction Execution Arbitrator**). Furthermore, new considerations related to the use of IMA (**Interaction Management Arbitrator**) have to be developed.

8.1 Concept of interaction manager

8.1.1 Separation of service action from service interaction

When a new service is added that interacts with existing services, modification of the existing service logics is to be avoided. In other words, service action is to be separated from service interaction.

For this purpose, the concept of interaction manager is introduced. This entity is a link between services and BCP that terminates user and network signals and provides a set of basic call processing activities.

Depending on the responses it receives from services and on its rules and data, the interaction manager distributes messages from the BCP only to the relevant services and synthesizes various answers received from the services into a unique answer to be sent back (figure 8). In that case, services cannot directly manipulate resources and they cannot respond directly to the BCP.

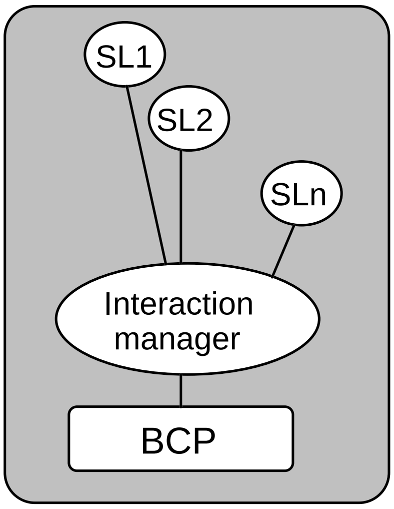


Figure 8: Concept of interaction manager

8.1.2 SCF and SSF oriented approaches

Two different approaches of interaction care can be identified to implement such an interaction manager: a SCF oriented approach and a SSF oriented approach.

- The **SCF oriented approach** tends to concentrate the additional mechanisms for service interaction within the SCF, in order that the SSF could see multiple SLs, acting on the same call, as only one. The interaction management entity that controls the simultaneous execution of multiple Service Logic Programmes in the same SCF is called SLIM.

- The **SSF oriented approach** tends to concentrate the additional mechanisms for service interaction in the SSF, in order to preserve call integrity (given that the SSF/Call Control Function (CCF) has a view of the caller's activity, e.g. interrupting or hanging up, and a view of non-IN switch-based services) and possibly to avoid a growing complexity of the SCF. The interaction management entity in the SSF that provides mechanisms to support multiple concurrent instances of IN services on a single call is called FIM[4].

In this ETR, a "shared approach", combining both basic approaches, is presented.

8.1.3 Service interaction typology

This subclause introduces a global concept of service interaction typology.

8.1.3.1 Two chronological situations

Two SLs invoked on the same call can be either sequentially invoked (sequential interaction) or simultaneously invoked (simultaneous interaction).

Sequential interaction applies to the case where the first service logic is no more involved in the call when the second is invoked. This does not mean that an interaction mechanism is not necessary. subclause 8.1.3.2 shows that interaction, **whether sequential or simultaneous**, may entail major impacts to be considered in the service interaction management.

Several examples of simultaneous and sequential interactions are provided hereafter.

8.1.3.2 Overall call model

Two situations of functional interaction typology can be considered: **local** and **distributed.**

[4] The term should be "Service Interaction Manager (SIM)", but the terminology spread in ITU-T is adapted from the American one, which frequently uses feature instead of service. In this ETR, it has been decided to use service as often as possible, but it was felt necessary to keep the names FIM and IPFIM, which have become accepted through use.

8.1.3.3 Local interaction

An interaction situation is **local** when a unique BCP is involved in the call and has triggered several SLs (figure 9).

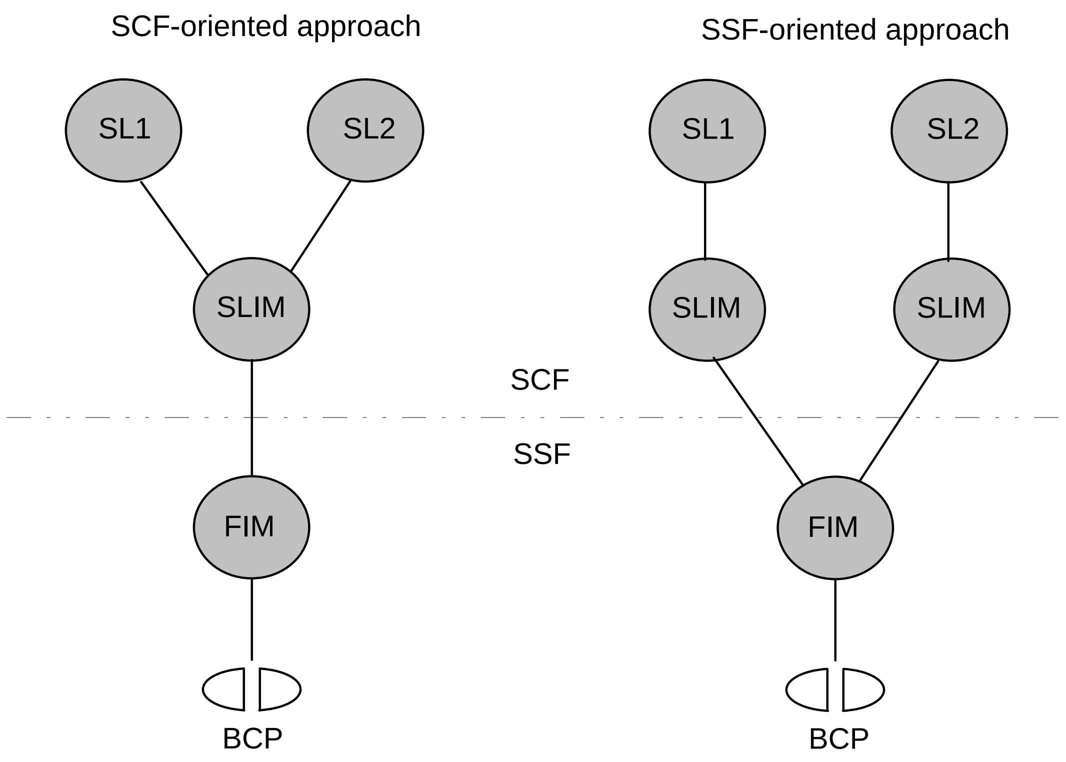


Figure 9: Two approaches of local interaction

EXAMPLE 1: **sequential local interaction**. A terminating call is routed to a subscriber's line on which are active both TKCS and CFU (figure A.2). The services are triggered at the same Detection Point (DP), the call instance is processed by a unique BCP and connected to two SLs.

EXAMPLE 2: **simultaneous local interaction**. In this situation, the services are triggered at the same DP, the call instance is processed by a unique BCP and connected to two SLs simultaneously invoked. This case is mentioned only as a reminder. It may not occur, for even if two services are triggered at the same DP, they are triggered according to different criteria and not simultaneously. It is up to the Basic Call State Model (BCSM) (in a fixed order scheme) or to management information (provided at DP activation with a given criteria) to describe the invocation order.

8.1.3.4 Distributed interaction

An interaction situation is **distributed** when several BCPs are involved in the call, each one connected to a unique SL. Only the case where two BCPs are involved is addressed hereafter (figure 10).

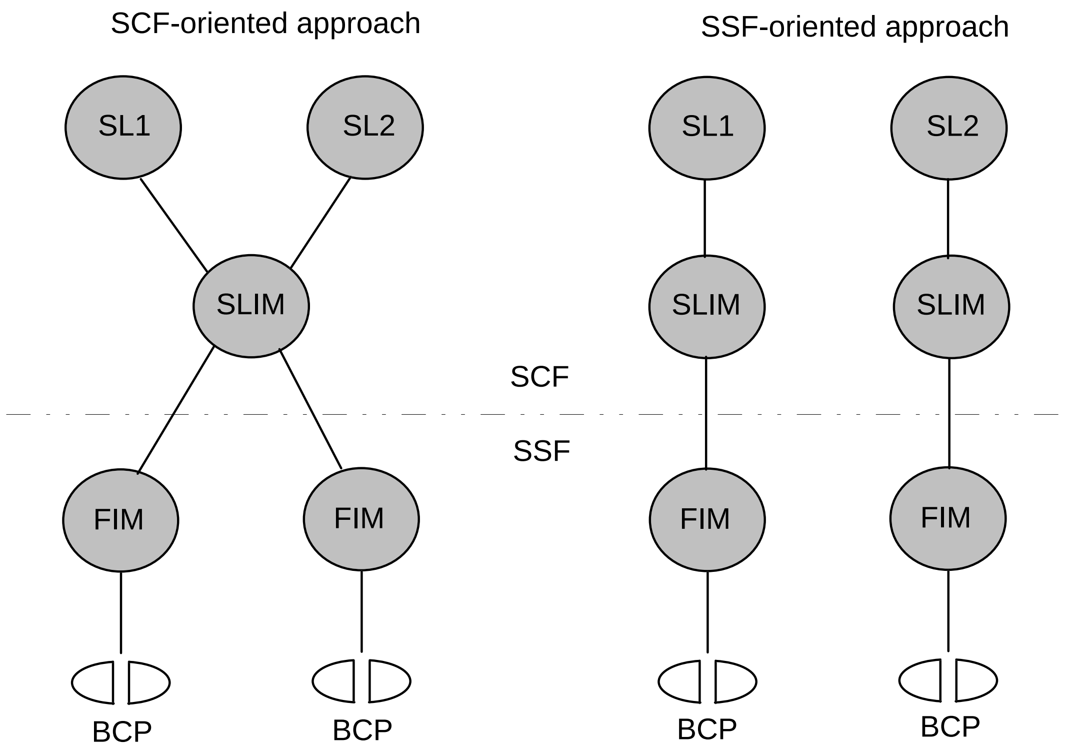


Figure 10: Two approaches of distributed interaction

One may note that all the situations depicted in figures 9 and 10 may be derived from the generic model given in figure 10/right. By linking the SLIMs, one obtains figures 9/left and 10/left. By linking the FIMs, one obtains figures 9/left and 9/right. Thus a generic mechanism can be produced, which requires capability of exchanging information between SLs (SLIM level), BCPs (FIM level) and basic call segments, as well as coordination between these three elements.

EXAMPLE 3: **sequential distributed interaction**. A call may encounter a Call Forwarding service (Unconditional, on No Reply, Selective) and be re-routed on a line where Distinctive Ringing is active. The call instance may be seen as two sub-call instances connected together through the network, with two SLs sequentially invoked. The two services have, in terms of interaction, "**no impact**".

EXAMPLE 4: **simultaneous distributed interaction**. Assume a user asking an Automatic Alternate Billing (AAB) against a called party with a TCS invoked from another SSF. The AAB service logic remains invoked until the called party's answer. When the incoming call is submitted to TCS, the two SLs are simultaneously invoked, while the two sub-call instances are connected together through the network. According to service provider's operation choices, either the TCS does not apply to the AAB service, or the AAB service calling line identity may be included in the TCS list.

8.1.3.5 Distributed interaction involving two BCPs within the same SSF

Generally, two SSFs may be involved in the call, but this is not mandatory. It is possible to physically connect two BCPs within the same SSF.

To avoid to connect two BCPs within the same SSF for a single call, such a connection may be logical and not physical (figure 11).

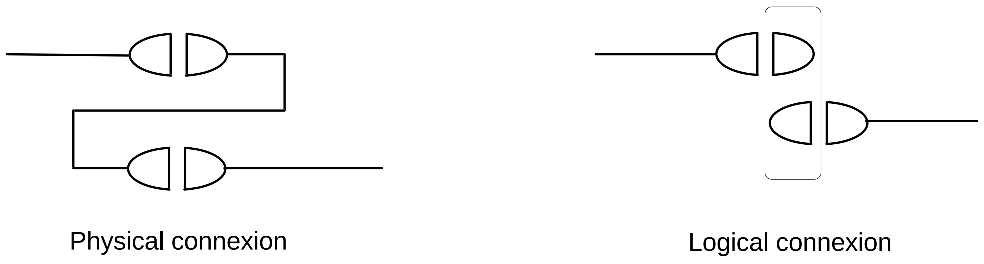


Figure 11: How to connect two BCPs within the same SSF

EXAMPLE 5: **sequential distributed interaction involving two BCPs within the same SSF**. When a subscriber uses a Call Distribution (CD) for managing the incoming calls between different locations, he may use the Call Forwarding No Reply (CFNR) for completing the call if a location does not answer. If CFNR is triggered in the same SSF as CD, the interaction is distributed, and the two SLs are sequentially involved on the same call instance. Nevertheless, in this case the two services have, in terms of interaction, "**no impact**".

EXAMPLE 6: **simultaneous distributed interaction involving two BCPs within the same SSF**. When a caller uses Charge Card Calling (CCC) to get through to a Premium Rate (PRM) number, the SSF has to involve two SLs on the same call instance, with two controlling relationships. Both services have to be notified of the answer or release of the terminating part of the call.

8.1.3.6 Complex interaction

A call can be made of several sub-calls connected throughout the network, and some of these sub-calls can be involved in local interaction. Such a case can be called complex interaction.

EXAMPLE 7: **complex interaction.** For this example, consider a tricky case merging examples 1, 3 and 6: a caller uses Account Card Calling (ACC) for getting through to a Split Charging (SPL) number, the terminating call is routed to a line on which are active both TKCS and CFU, before being forwarded to a line where Distinctive Ringing (DRI) is active. A sequential distributed interaction, involving two BCPs within the same SSF, will be kept during the whole call attempt, while a sequential local interaction and another sequential distributed interaction are encountered, three SLs being successively involved together in the call (ACC and SPL permanently, and TKCS, CFU and DRI in turn).

8.2 Interaction management mechanisms in the DFP

8.2.1 Co-ordinating interaction manager

Figure 10/right shows no link between the involved managers, though such a link seems nonetheless necessary to coordinate the process. The entity which ensures the consistency of operation and dispatches all pertinent information among the other managers, as shown in figure 12, is named IPFIM[5]. It has the knowledge of the services implemented on all the platforms and, in case of distributed interaction, the ability of taking decisions on precedence and pre-emption.

NOTE: The location of IPFIM within DFP is for further study. It could be a separate FE, a functionality distributed upon several SCFs and SSFs, and/or a global view of all the SLIMs and FIMs.

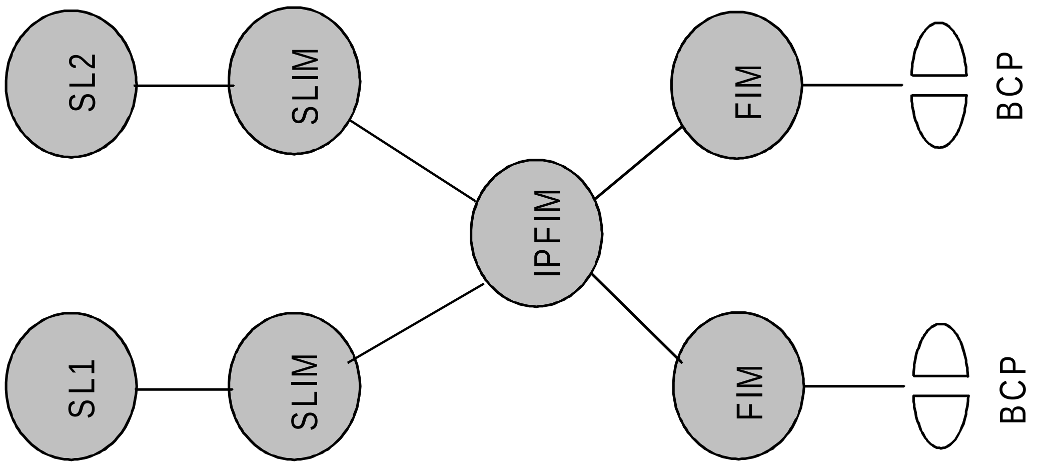


Figure 12: IPFIM

8.2.2 Concept of negotiation of interaction

Among the possible tracks that may be followed to build an IPFIM, the concept of negotiation can be quoted. It enables potentially conflicting users of a system to co-operate by negotiating agreements to execute operations.

In a negotiating agent model, each party (i.e. object, such as user, terminal equipment, service, switch, etc.) is represented by a process called "agent" that knows its preferences and tries to establish a communication accordingly. Agents make proposals for a communication in a form understood by all parties, these proposals are examined by the other agents that can modify them to make them consistent with their goals. No communication can be established until all agents have agreed i.e. have made a common proposal.

The negotiating agent model addresses the resolution of specification-level interactions that arise because parties have conflicting interests or because different parties use conflicting services.

[5] For consistency sake, the term should be "IPSIM", but the terminology spread in ITU-T is adapted from the American one, which frequently uses feature instead of service. In this document, it has been decided to use service as often as possible, but it was felt necessary to keep the names FIM and IPFIM, which have become accepted through use.

8.2.3 Communication between interaction managers

8.2.3.1 Need for additional data to be conveyed between managers

Consistent operation of a service introduces the concept of mapping on the BCSM. IN events are defined to occur at clearly defined points in the BCSM in order to allow clean interaction.

As a first step, a partial mapping considers the starting event and the ending conditions of the service, and priority and exclusion mechanisms constitute a basis for the interaction control.

As a second step, fully mapping considers all of the service's relationship to the BCSM, including significant DPs where state changes occur, such as user signalling, or DPs over the party's participation is exercized such as splitting a party away from the connection point. The IPFIM may provide the SLIM with a global view of the service's progress, thus the mapping allows to get a consistent understanding of when services operate with respect of the BCSM, and interaction may be managed at each DP.

In order to provide information needed by the IPFIM to make decisions, additional data about services available to the users have to be exchanged between the different managers.

The following mechanisms apply to all types of interaction mentioned in subclause 8.1.3.

8.2.3.2 Identification of services (SID)

A basic premise for working with two sets of services on separate platforms is the need for consistent definition of service and consistent operation of those services. Consistent definition means that a service identifier (SID) is assigned to a service so that its operation is understood well enough to consistently apply it across all service provider's offers. It is understood that sometimes vendors provide several "flavours" of a single service. When the operation of these flavours begins to diverge from a consistent view of either the initial invocation of the function or its operation in light of the BCSM, a new service identifier is needed. Furthermore, when a service is commonly defined, all service providers need to insure that there is an agreement upon starting DP and, in case of fully mapped service, common reference DPs.

8.2.3.3 Identification of service categories (BID)

Alternatively, (critical) service behaviour can be identified and categorized, and services grouped under each relevant category. As an example, a category could be "Exclusive Emergency", to indicate emergency services that must not be affected by (i.e. that are exclusive of) any other service, hence escape to screening and restriction services. Such a categorization, so-called **service typology**, has already been mentioned for other purposes in subclause 6.4.3.7.2.

Under this approach, behaviour identifiers (BID) could be formulated to characterize each category. Interaction managers could then exchange BIDs when required to manage interactions. This approach requires the full disclosure of available services from one manager to another. It has the advantage of requiring the exchange and processing of a smaller, hopefully in a significant way, number of parameters. To evaluate the method, the decrease of the number of parameters is to be balanced with the increase of complexity.

8.2.3.4 Identification of active services (ASID)

Considering the interaction problem of ARC with CFU (figures A.4 and A.5), it may be solved by stating that the CFU service logic has to be informed that the call is an ARC, in order to force the CFU and present the call to the real caller. This proves that a service S1, in order to interact with a service S2, may need to be informed of the **remote invocation** of S2.

The Active Services Identification (ASID) indicates which services have already been invoked on the call and how many times (for example, Call Forwarding may be invoked several times on the same call). It also indicates, for each service, whether invocation still persists or not.

The ASID shall appear in:

- the initial operation from SSF to SCF;
- some ISDN User Part (ISUP) messages such as **PassAlongMessage** (PAM) used to inform remote SSFs (in case of distributed interaction, this bi-directional message can inform S1 of the invocation of S2, or conversely);
- the report of these ISUP messages from the SSF to the SCF.

8.2.3.5 Identification of pending services (PSID)

The Pending Services Identification (PSID) indicates which services are activated, i.e. available to the user, and may be invoked on the call. The list may be complete or limited to the service mapped on a specific DP.

For instance, considering solution 4 in subclause A.3.2, ARC invocation is to be forbidden when the call has been initiated by a line where CFU is activated. In such a case, the call from B to A could bear an indication that CFU is activated on B's line.

8.2.3.6 Identification of inhibited services (ISID)

The Inhibit Services Identification (ISID) indicates which services are to be inhibited. For instance, considering solution 2 in subclause A.3.2, CFU may be overridden when the call is generated by ARC. In such a case, the call could bear an indication that an encountered diversion service is to be overridden (or inhibited)[6].

If it is considered that the information between managers is to be conveyed by signalling, the ISID should also indicated whether inhibition applies only to the call or shall be persistent. For instance, in annex A, interaction between CFU and TCS has been analysed and a proposal has been made: **if (A) wants to activate CFU to (B) whereas (B) has put (A) in (B)'s blacklist, the command is rejected, and CFU is not possible**. Considering now the case where (B) puts (A) in his blacklist after CFU has been activated, the calls placed to (A) are forwarded to (B) and systematically rejected. The interaction manager at (B) side could order a persistent inhibition of (A)'s CFU using ISID.

The ISID shall appear in:

- some ISUP messages such as **PassAlongMessage** (PAM);
- the report of these ISUP messages from the SSF to the SCF.

8.2.3.7 Flexible call context

Some information about the call must be kept for the whole call duration. This information is stored in a call context which must be flexible, since its content changes according to the invoked services.

The flexible call context mechanism is common both to the SSF oriented approach and the SCF oriented approach. It applies to all types of interaction mentioned in subclause 8.1.3.

This flexible call context is sent:

- to the SCF, at each invocation;
- to a remote SSF, in order to avoid triggering some services incompatible with the already invoked ones; the remote SSF has to send this context to the remote SCF. Thus, it has to be carried through the network signalling, and for this purpose, ISUP has to be improved, either by modifying existing messages or by creating new messages.

---

6) Note that this example uses both BID and ISID.

The main purposes of the flexible call context are:

- Characterizing the call: a mark "IN type of call" is determined either by the subscriber's line characteristics or after translation in the first exchange. When recognizing this mark, the transit exchange or the local exchange immediately sends the context to the SSF.

- Conveying useful information related to both calling and called parties, which may include caller's geographic location, calling line identity, account to be charged, extra dialled numbers, flags such as "authentication performed", and so on. For example, when an authentication is requested for one service, it is kept in the call context to avoid requesting the same information if another service requires it. For another example, considering the interaction of Malicious Call Identification (MCID) with Freephone or Account Card Calling (ACC), in both cases, the usual identities memorized by MCID are not sufficient: in case of a Freephone call, the dialled number is needed as well as the translated number, and in case of ACC, the card number is to be recorded as well as the calling line identity.

8.2.3.8 Open issues

The following points have to be considered:

- the amount of information to be conveyed could be limited by the signalling capabilities. In such a case, a refinement of the type of information to be conveyed is to be studied;

- the described mechanisms apply to interaction of services active on the same call. If interaction occurs between services active on different calls[7)], complementary mechanisms are to be implemented;

- information about other user's profile may also be needed, as well as information about the configuration of multiple party calls.

8.2.4 Interaction management at the SCF level

The interaction management in the SCF is performed by the SLIM. The SLIM manages interactions among multiple Service Logic Programme instances simultaneously active on a single call in the same SCF. It is common both to the SSF oriented approach and the SCF oriented approach.

8.2.5 Interaction management at the SSF level

8.2.5.1 General

The interaction management in the SSF is performed by the FIM, in which several mechanisms are included. They are common both to the SSF oriented approach and the SCF oriented approach.

Two approaches permit to solve the possible incompatibility between services: either it is part of the DP processing, and the decision is made before a SL is invoked, or it is independent of DP processing, and decisions about compatibility and precedence are made after invocation. The second approach, more flexible, requires nevertheless more complex mechanisms. The first approach is developed below.

---

7) When the history of a call is used by service logic for a later call, interaction may occur between services active on different calls. Assume for example a user having subscribed to both Terminating Call Screening (TCS) and Automatic ReCall (ARC). A first call is placed by A to B, who is temporarily unable to answer. Then a second call is placed by a user C to B, but is rejected by TCS because C is on B's blacklist. ARC is conceived for returning the latest terminating call without knowing the calling number. If the services TCS and ARC have been designed independently, ARC has memorized C's number as "latest terminating call" information, and if B invokes ABC, the call will be returned to somebody he deliberately tried to avoid. The last call depends on call history, while the two first ones can control the recording of this history.

8.2.5.2 Trigger priority table

The trigger priority table consists in arranging all services at one DP (for instance Selective Call Rejection, TKCS and Distinctive Ringing) in a priority order. This is a specific case of local interaction.

The trigger priority table may be represented as shown in table 2.

Table 2: Trigger priority table

P 1	S1>S2
P 2	S3
P 3	S6>S7>S8
P 4	S9
...	

Such a table can be compiled at spotting time. It should be part of the administration of the services.

8.2.5.3 Exclusion table

For each pair of services, it is to be considered whether the interaction is allowed or not. For that purpose, the SSF handles a table of service exclusion. For each pair (S1,S2), an exclusion mark indicates whether the interaction is allowed or not. The exclusion table defines an **exclusion relation** (S1 excludes S2) and conversely a **compatibility relation** (S1 compatible with S2). It applies to all types of interaction mentioned in subclause 8.1.3.

For filling up the exclusion table, the following requirements have to be taken into account:

- the compatibility relation is nor **reflexive**, nor **anti-reflexive**. A service may exclude itself: for some services (for instance, Account Card Calling), a new instance of S1 may not be invoked after S1 has been invoked. But the contrary is also possible (for instance, CFU);

- the compatibility relation is not **symmetrical**. If S1 may be invoked after S2, it does not imply that S2 may be invoked after S1. As an example, considering in figure A.4 the interaction between ARC and CFU, if a user B has requested a CFU to C, if he calls a user A and if A tries to use ARC against B, the call would be forwarded to C;

- the compatibility relation is not **transitive**. If S1 may be invoked after S2, and if S2 may be invoked after S3, this does not imply that S1 may be invoked after S3: considering S1 as Automatic Alternate Billing (AAB), S2 as CFU and S3 as TKCS, the invocation of AAB after CFU is possible, the invocation of CFU after TKCS is possible (see Annex A), but the invocation of TKCS when AAB has been invoked is not trivial and may be refused by the service provider: the service logic is not able to recognise the recorded announcement asking to compose a key code, and this information should be carried in the signalling. To deal with this non-transitivity, more complex tables should be handled, with several entries. A possible limitation to this assumption consists in forbidding interaction between more than p services at the same time, i.e. invocation of more than p services on the same call.

8.2.5.4 Trigger algorithm

When a triggering condition is detected on a DP, the trigger algorithm shall:

- match trigger criteria;
- consider the first priority service from the trigger priority table;
- scan the **exclusion table** for any exclusion between the service and the other services already invoked on the same call;
- if none is detected, invoke the SL;
- if exclusion exists, consider the next higher priority service and try again.

This algorithm applies to all types of interaction mentioned in subclause 8.1.3.

8.3 Interaction with non-IN services

Interworking of IN-based services with switch-based services may be partially described in stage one descriptions. Interaction may be handled either by IN or by the network. In both cases, the complementary signalling mechanisms mentioned above may be used and completed.

In case of handling by IN, services are embedded in the switching systems with a proprietary implementation. The switch may be viewed at a higher level as having an SCF structure with their own service interaction manager not standardized with the IN recommendations.

From a modelling point of view, i.e. in the Service Plane, GFP, and DFP, the difference between IN and switch-based services is to be clarified. It is expected that switch-based services could be modelled with the INCM. It is only in the way the services are designed that they could be called "IN" (for services which are implemented on the basis of a service logic interacting with a basic call modelled according to the BCSM) or "non-IN" (for services implemented either directly within the basic call or on the basis of a service logic interacting with a basic call not modelled according to the BCSM). This raises the following issue: "what is intrinsically different for a non-IN service regarding service interaction?" (the answer should not be "these non-IN services are already existing and include a number of interaction mechanisms", as far as these mechanisms may become part of the SLIM/FIM/IPFIM). Interactions between IN-based and switch-based services and service features are covered in ETR 186-1 [14].

Annex A: A thesaurus of examples

A.1 General

This annex provides some basic examples which are referenced in the main body of this ETR.

A.2 CFU and TKCS

A.2.1 First case

The TKCS service enables a subscriber to have incoming calls screened by a user defined key, i.e. key code. Callers are required to enter this key. The subscriber may activate and deactivate the service.

Consider the interaction between CFU and TKCS, when a user wants to make his calls forwarded to a line screened by a key code.

This situation is a interworking case: the caller A could be connected to an announcement asking him to compose a key code he his not aware of, because he does not know the line he gets through to (figure A.1).

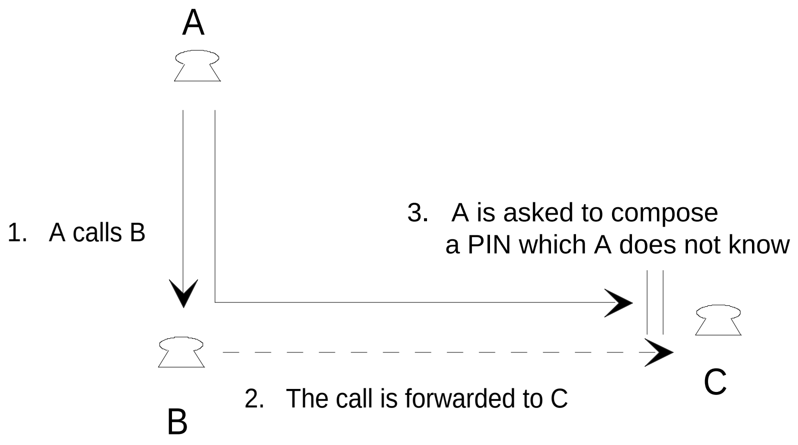

A
1. A calls B
3. A is asked to compose a PIN which A does not know
C
2. The call is forwarded to C
B

Figure A.1: Invocations of call forwarding and TKCS on two different lines

A.2.2 Second case

Considering the same pair of services, if a user has his line screened by a key code and makes his calls forwarded, it is to be decided whether the call forwarding is to be considered prior to the key code screening or not (figure A.2).

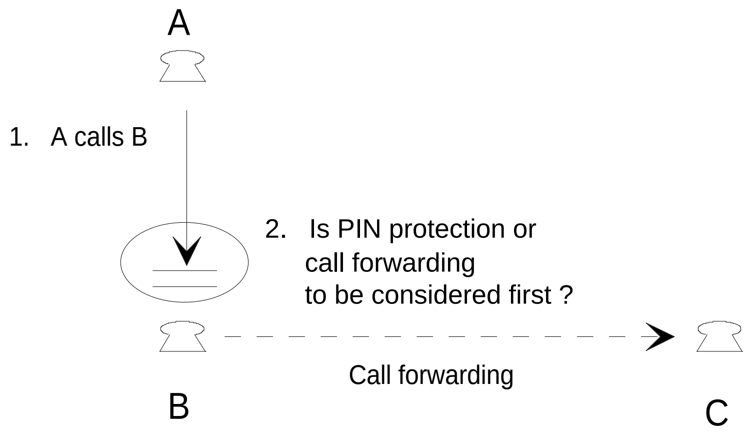

A
1. A calls B
2. Is PIN protection or call forwarding to be considered first ?
Call forwarding
B
C

Figure A.2: Invocations of call forwarding and TKCS on the same line

A.3 CFU and ARC

A.3.1 First case

ARC allows a user using a control procedure to call back the calling party of the last incoming call.

Consider the interaction between ARC and CFU. If a user B has requested a CFU to C, and if a user A places a call to B, the call is forwarded to C, who may use ARC against A. Interworking is acceptable: if he uses ARC, C gets through to the user who placed to previous call. One may consider that there is no interaction problem (figure A.3).

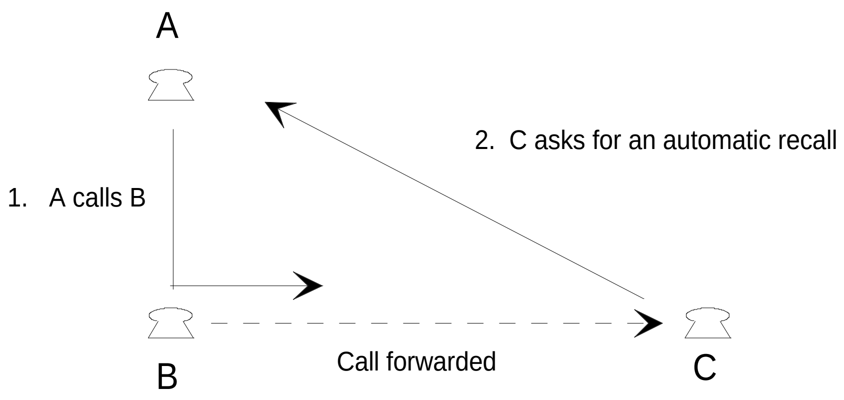


Figure A.3: Invocations of call forwarding then ARC

A.3.2 Second case

But if B calls A, and if A tries to use ARC against B, the call would be forwarded to C that will probably will not be aware of the call and may not understand the situation (figure A.4). Interaction solution is not obvious, and several possibilities may be envisaged:

a) the call back may "normally" be routed to C's line; this plain solution may disturb C who doesn't know where the call may come from (but it is the best if B and C lines are owned by the same person);

b) the call back may be routed to B's line, where is located A's caller; this choice is the best from A's point of view, for it tries to make the return call as efficient as possible; one may also consider that B wants to get in touch with A, since B has just called A, which justifies CFU's overriding;

c) one may mix previous solutions, ringing B first (where is probably located the caller), then C if B doesn't answer; in such a case, the CFU is converted in CFNR;

d) one may forbid the calls generated by ARC if the calling party has activated a call forwarding.

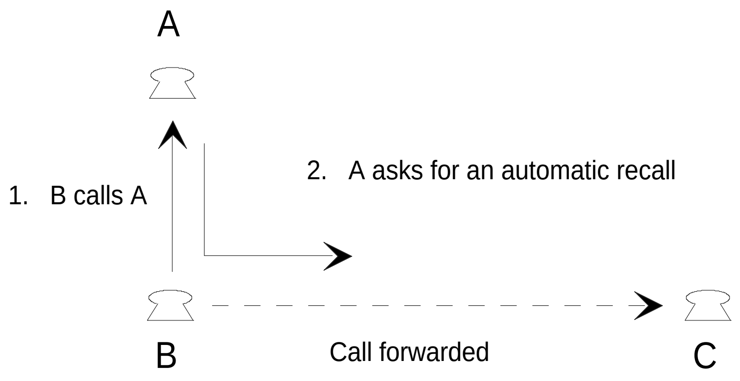


Figure A.4: Invocations of ARC then call forwarding

A.4 CFU and TCS

The case of interworking between CFU and TCS is complex, because in each service, several actors may be found:

- the CFU actors are "caller", "called" and "forwarded-to";
- the TCS are "screener" and "screened".

The permutation reveals that interworking between these two services presents 36 different cases to scan, among which 6 (#19 to 24) may be merged. Table A.1 shows the results of actors permutation, considering the scenario "B forwarding his calls to C".

Table A.1: Interworking between TCS and CFU

#	"screened"	"screener"	caller	calling	result
1	A	B	A	B	No impact, call rejected
2				C	No impact
3			B	A	No impact
4				C	No impact
5			C	A	No impact
6				B	If CFU authorizes call back from C to B, the call is completed. Otherwise, busy tone is sent to C.
7	A	C	A	B	Call is forwarded, but rejected when terminating at C
8				C	No impact, call rejected
9			B	A	No impact
10				C	No impact
11			C	A	No impact
12				B	If CFU authorizes call back from C to B, the call is completed. Otherwise, busy tone is sent to C.
13	B	A	A	B	No impact, call forwarded
14				C	No impact
15			B	A	No impact, call rejected
16				C	No impact
17			C	A	No impact
18				B	If CFU authorizes call back from C to B, the call is completed. Otherwise, busy tone is sent to C.
19 to 24	B	C			If B wants to activate a CFU to C whereas C has put him in his blacklist, the services will include incompatible data.
25	C	A	A	B	(see figure A.5) The call is forwarded, but A may be connected to C against his willing whereas C is in A's blacklist, therefore whereas A has all the best reasons to avoid C.
26				C	No impact
27			B	A	No impact
28				C	No impact
29			C	A	No impact, call rejected
30				B	If CFU authorizes call back from C to B, the call is completed. Otherwise, busy tone is sent to C.
31	C	B	A	B	No impact, call rejected
32				C	No impact
33			B	A	No impact
34				C	No impact
35			C	A	No impact
36				B	No impact, call rejected

Case #25 shows a very tricky case which would probably escape a heuristic investigation (figure A.5). User B having forwarded his calls to C, A is a user who has the best reasons to avoid C. If A calls B, the CFU will make the call forwarded to C, and A and C will be linked. If it is considered to be avoided, one may state that a call is not to be forwarded to a line which is in caller's blacklist.

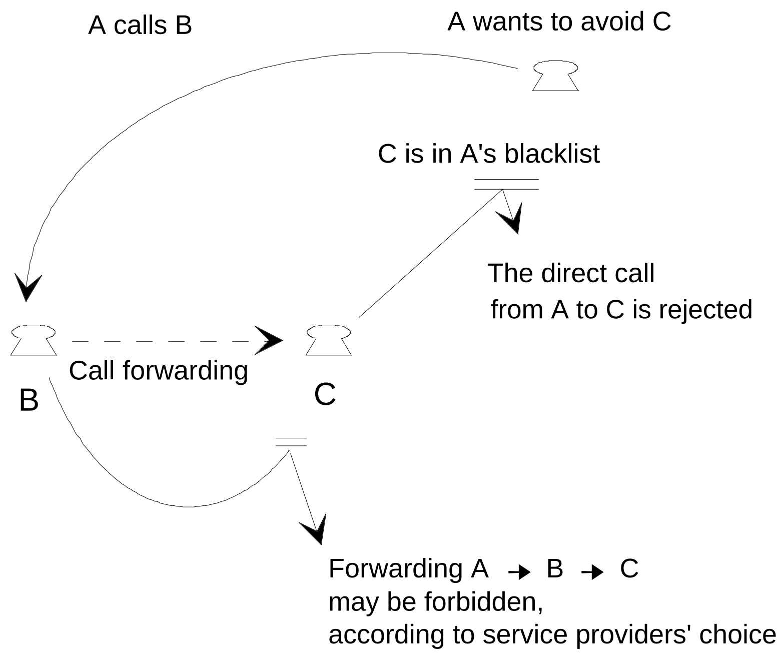


Figure A.5: Complex interaction between CFU and TCS

History

Document history	
July 1995	First Edition
February 1996	Converted into Adobe Acrobat Portable Document Format (PDF)

ISBN 2-7437-0163-3
Dépôt légal : Juillet 1995