Service Specification Using Z

Mícheál Mac an Airchinnigh, K&M Technologies Ltd.

Michael Butler, Broadcom.

11th October 1993

1 Executive Summary

Z is a formal description language based on set theory and predicate logic [12]. In SCORE, we have investigated, through a number of case studies, how Z may be used to support service creation. Z may be used in the specification stage of service creation as it provides a succinct and unambiguous way of specifying services. Experience has shown that formal notations such as Z may also be used to aid the requirements gathering stage of the service creation process; by building partial formal models based on requirements, these requirements can be clarified and further elaborated before a more complete specification is produced.

1.1 Conceptual Model

Guidelines on the use of Z for telecoms service specification are described in this report. We have taken the view that a service is provided by a system. A standard approach to system specification was taken, whereby we focus on some clearly defined notions—entities—and build appropriate relations between them. We refer to the entities and the relationships between them as a *conceptual model*. The conceptual model is represented in Z using mathematical structures. Commonly used mathematical structures used in service specification are sets and functions. Functions are used to describe mappings from elements of one domain to another and are used in many different contexts. For instance, a function could be used to specify communications connections by mapping A-numbers to B-numbers, or could be used to describe a freephone service by mapping freephone numbers to real numbers, or could be used to describe the relationships between service subscribers and service providers. The conceptual model is developed by identifying entities and relationships between them from the service requirements, and gradually elaborating and refining the conceptual model by posing questions of the requirements.

1.2 Operations

The conceptual model serves as the state space of a system, and transitions on the state space are described by *operations*, which also form part of a specification. An operation is described by a predicate relating possible before-states to possible after-states, as well as some input parameters and output parameters. The effect of invoking an operation is to update the state based on the before-state and the input parameters, resulting in an after-state and output parameters according to the specification. When the conceptual model consists of sets and functions, then such state changes usually involve adding or removing elements from sets, and adding, removing, or updating mappings in functions.

A system provides a service to it's environment (i.e., users and others systems) by interacting with that environment. The environment interacts with a system by having a view of the system state, and by changing the system state through operations.

So the definitions of the operations are based on the required interactions as identified from the service requirements and their intended effect. Often we find that the conceptual model may be inadequate to fully specify the intended effect of an operation, and needs to be refined.

1.3 Invariants

Invariants are used to describe "healthiness" conditions that the conceptual model should satisfy. These invariants may represent constraints imposed by the requirements, or may be necessitated by the mathematical structures we use to describe the conceptual model. It is important that the operations of a specification preserve all invariants, and mathematical reasoning may be used to check this. The B-tool has been evaluated in order to see how it can support this reasoning. As well as performing syntax-checking and type-checking, the B-tool acts as a proof assistant. It generates a list of proof obligations from a specification, tries to prove some of those obligations based on a library of proof rules, and flags any obligations that it fails to generate. The library of proof-rules is easily extended by the tool user.

1.4 Structuring Specifications

An important feature of Z is the *schema*. A schema is a box containing a piece of mathematical specification. Schemas can be combined in a variety of ways, thus providing a powerful means of structuring large specifications. An object-oriented version of Z has also been developed, providing yet more mechanisms for structuring Z specifications.

1.5 Rationale

The mathematical structures provided by Z allow us to write service specifications that are abstract, independent of implementation detail, and closer to

the user's point-of-view of the service. For instance, when describing Intelligent Network services in Z, we don't introduce any of the detail to do with interactions between the distributed network entities such as Switching Control Point, Service Control Point, SIB's, etc. We only introduce enough detail to the specification as is necessary to describe the service from the user point-of-view. The specification is thus easier to validate against the user's requirements. The mathematical nature of Z means that the specifications produced are rigorous and unambiguous. It also allows us to reason about our specification thus ensuring greater quality. For instance, we can ensure the consistency of the specification by ensuring that invariants are preserved by operations as mentioned above. In SCORE, we have also investigated how Z can be used the detection of potential service interaction problems. This work is reported in [**?**].

1.6 Case Studies

Z has been applied in two case studies in SCORE: the IN-Services case study and the EuroBridge case study. For the IN-Services case study, specifications in Z of several IN-services, including the Follow-Me-Diversion service, have been produced. The approach taken in this case study was to first specify the basic connection service in Z. The supplementary services (e.g. Follow-Me-Diversion) were then specified as extensions of this. The interactions between the supplementary services and the basic service, and between different supplementary services, were then analysed for potential problems and unwanted side-effects. It was found that the Z specifications were amenable to such analysis. For the EuroBridge case study, specifications of some multi-media services have been produced. The approach taken in this case study was to solicit general requirements on multi-media services from EuroBridge and other sources. These requirements were then used to develop a generic model in Z of multi-media services, and the generic model was instantiated to a number of multi-media applications including a point-of-information service and a medical support system. Use of Z provided an effective way of adding structure and rigour to the process of eliciting and analysing requirements, and feedback from preliminary formal models was used in refining the requirements to produce a more complete specification. The specifications produced in both these case studies would serve as sound bases for implementing the example services.

2 Introduction

Z is a formal description language based on set theory and predicate logic [12]. In this report, we provide an overview of how Z may be used to describe telecommunications services. Z may be used in the specification stage of service creation as it provides a succinct and unambiguous way of specifying services. Experience has shown that formal notations such as Z may also be used to aid the requirements gathering stage of the service creation process; by building partial formal models based on requirements, these requirements can be clarified and further elaborated before a more complete specification is produced.

We take the view that a service is provided by a system. We choose to exhibit a standard approach to system specification, whereby we focus on some clearly defined notions—entities—and build appropriate relations between them. We refer to the entities and the relationships between them as a *conceptual model*. The conceptual model is represented in Z using mathematical structures. The conceptual model serves as the state space of a system, and transitions on the state space are described by *operations*, which also form part of a specification. It is usual, moreover, to specify an *initialisation* condition for the system

A system provides a service to its environment (i.e., users and other systems) by interacting with that environment. The environment interacts with a system by having a view of the system state, and by changing the system state through operations.

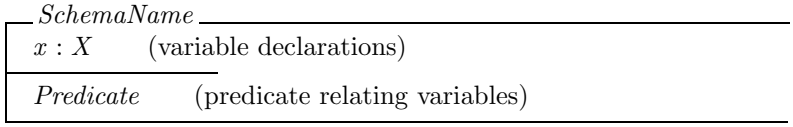
Invariants are used to describe "healthiness" conditions that the conceptual model should satisfy. These invariants may represent constraints imposed by the requirements, or may be necessitated by the mathematical structures we use to describe the conceptual model. It is important that the operations of a specification preserve all invariants, and mathematical reasoning may be used to check this.

We will not attempt to introduce all the constructs of the Z language, rather we present a series of example Z specifications, explaining the important constructs as they are introduced. In the next section, we introduce the schema notation, which is an important mechanism for structuring Z specifications. We then illustrate how the mathematical structures of Z may be used to specify important service concepts such as the relationships between service actors, connection control, directory service, green-number service, and multi-media user interface.

A glossary of the symbols used can be found at the end of the chapter.

3 Specification and Schemas

A Z schema consists of a name, variable declarations, and a predicate:



In this section, we illustrate the schema notation of Z by specifying a simple counter system. Firstly, assume that max is some constant natural number:

$$max : \mathbb{N}$$

The state space for the counter is specified by the schema *Counter*:

Counter

$$ctr : \mathbb{N}$$

$$ctr \leq max$$

The state space consists of a single component named ctr of type natural number representing the current value of the counter. The predicate part of *Counter* is an invariant constraining the value of the counter never to be greater than the fixed value max.

An operation to increment the counter is defined by the following schema:

Increment

$$\Delta Counter$$

$$ctr < max$$
$$ctr' = ctr + 1$$

The declaration $\Delta Counter$ means that this operation will change the state space *Counter*. The new value of the counter is referred to as ctr', while the old value is ctr. The predicate part of this schema simply states that provided ctr is less than max initially, the new value of ctr is the old value plus 1. (There is an implicit conjunction between successive lines in the predicate part.)

Similarly, an operation to decrement the counter is defined by:

Decrement

$$\Delta Counter$$

$$ctr > 0$$
$$ctr' = ctr - 1$$

An important property of an operation is it's *precondition*. This is a condition on the initial value of the state variables describing when the operation will result in a valid new state (i.e., one that satisfies the state invariants). For example, the precondition of the *Decrement* operation is as follows:

pre *Decrement*

$$ctr : \mathbb{N}$$

$$ctr > 0$$

So the decrement operation may be validly invoked only if $ctr > 0$.

An operation to set the value of the counter is specified by the schema *SetCounter*:

SetCounter

$\Delta Counter$
$c? : \mathbb{N}$

$c? \leq max$
$ctr' = c?$

Here $c?$ is an input parameter to the operation. (Conventionally input parameter names are suffixed with '?'.)

An operation to display the counter value is defined by:

Display

$\Xi Counter$
$c! : \mathbb{N}$

$c! = ctr$

Here $\Xi Counter$ means that the *Counter* state-space is not changed by the operation. The parameter $c!$ is an output parameter. (Conventionally output parameter names are suffixed with '!'.)

4 Roles and Inter-relationships

Let us introduce formal names for the roles of service end-users ($u \in USR$) and service subscribers ($s \in SUB$). Multiple elements from a domain will be designated by a mixture of sub- and superscripts. Thus,

$$u_1, u_2, \ldots, u_n$$

denotes a collection of service end-users. We will later associate such a collection with a specific subscriber, s_j, say. In which case, the collection might be designated by

$$u_j^1, u_j^2, \ldots, u_j^n$$

Similarly, entities such as service provider ($p \in PRV$) and service ($x \in SRV$) may be furnished with appropriate formal names.

4.1 Subscribers and end-users

Suppose we wish to associate subscribers with users. We may declare a function f from subscribers to users:

$$f : SUB \nrightarrow USR.$$

Here, f maps subscribers to users and, since f is a function, a subscriber may be mapped to at most one user. An entry in f will have the form

$$s \mapsto u.$$

Rather than a single end-user, we take the view that there is a collection of service end-users associated with a service subscriber. The totality of such associations is modelled by a 'system of subsribers', $SUBSYS$. Formally we write:

$$\begin{array}{|l} \hline SUBSYS \\ ss : SUB \pfun \mathbb{P}\, USR \\ \hline \end{array}$$

Now subscribers are mapped to sets of users.
We may consider this model to denote a sort of 'table' where each entry has the form

$$s \mapsto \{u_1, u_2, \ldots, u_n\}$$

The model permits entries of the form

$$s \mapsto \{\}$$

which indicates that the particular subscriber, s, has (currently) no associated end-users. This is a common feature (called a *singularity*) in a specification which may be interpreted as denoting a particular point in time of end-user registration in the system.

Often, it is of particular importance to model a system in the state of equilibrium where all such singularities have been eliminated. We describe such a model as a 'derivation' and write

$$\begin{array}{|l} \hline SUBSYS_1 \\ ss_1 : SUB \pfun \mathbb{P}_1\, USR \\ \hline \end{array}$$

Here, subscribers are mapped to non-empty sets of users.

Such stable models are invertible. In other words, given ss_1, we may make use of the inverse $(ss_1)^{-1}$, where

$$(ss_1)^{-1} \in USR \pfun \mathbb{P}\, SUB$$

The model of subscribers and end-users, $SUBSYS$, also accommodates the possibility that the same end-user, u, may be associated with two different subscribers, s_j and s_k, say. In other words, a table such as

$$ss = \left[\begin{array}{lcl} \ldots & & \ldots \\ s_j & \mapsto & \{u, u_j^1, \ldots\} \\ \ldots & & \ldots \\ s_k & \mapsto & \{u, u_k^1, \ldots\} \\ \ldots & & \ldots \end{array}\right]$$

is perfectly acceptable. This captures, for example, the possibility of a consultant u working for two different companies s_j, s_k.

Given a model, such as that of the relationship between subscribers and end-users, we are in a position to exercise it. One may think of such exercising as the definition of operations on the model. Alternatively, it is often preferable to describe the process by posing suitable questions and ascertaining whether there is sufficient information to obtain the answers.

QUESTION 4.1 *Which subscribers are recorded in the system?*

Formally, the response is

$$\operatorname{dom} ss$$

It is formulated by an 'inbuilt' mathematical operator/function—the domain function.

QUESTION 4.2 *Is the subscriber s recorded in the system?*

Again the response is immediate:

$$s \in \operatorname{dom} ss$$

There are many such operators/functions which are immediately available. Rather than exercise this model exhaustively, we will now introduce other relationships and pose other questions.

4.2 Providers and services

Since we are dealing with service specification, it seems appropriate to introduce a model in which the concept, at least, appears. We choose to model the relation between service providers and the services that they offer. Formally, we write:

PRVSYS

$$ps : PRV \pfun \mathbb{P}\, SRV$$

It is obvious that this model is structurally identical (isomorphic) to that of subscribers and end-users. Thus, we are led to examine the interpretation to be given to singularities such as

$$p \mapsto \{\}$$

and entries of the form

$$ps = \begin{bmatrix} \ldots & & \ldots \\ p_j & \mapsto & \{s, s_j^1, \ldots\} \\ \ldots & & \ldots \\ p_k & \mapsto & \{s, s_k^1, \ldots\} \\ \ldots & & \ldots \end{bmatrix}$$

In the latter case we are forced to examine the meaning of two different providers offering the same service. Is this feasible, realistic?

Suppose that we wish to denote the addition of a new service, s, offered by the service provider p. We designate this by the expression

$$ps \oplus \{\ p \mapsto (ps(p) \cup \{s\})\ \}$$

Here, the override operator ($\oplus$) replaces the previous mapping for p with a new value, leaving other entries unchanged. For validity, it is necessary that the provider p already be recorded in the system ps, a pre-condition which is immediately expressed by the form

$$p \in \operatorname{dom} ps$$

Pragmatically, to construct formal specifications from such expressions, one needs to wrap them up in syntactic sugar and provide a sprinkling of meaningful names to bind them to the reality of the application in hand.

For the operation in question, the addition of a new service, we might provide a form that is reminiscent of a procedure in a programming language:

AddNewService

$\Delta PRVSYS$
$p? : PRV$
$x? : SRV$

$ps' = ps \oplus \{\ p? \mapsto (ps(p?) \cup \{x?\})\ \}$

Then, the pre-condition, that the service provider already be recorded in the system, is specified by

pre *AddNewService*

$PRVSYS$
$p? : PRV$
$x? : SRV$

$p \in \operatorname{dom} ps \land \ldots$

The ellipsis indicates that there are other conditions which we may wish to have satisfied. For example, we may wish to ensure that the new service being offered, $x?$, by provider p is NOT already one of those being offered by that provider:

$$\neg\ x? \in ps(p?)$$

It is worth noting that the validity of the expression for the addition of the new service does not depend on such a condition!

5 Connection Control

5.1 Conceptual Model

We wish to model the control of connections between service end-users. A connection will be modelled as a set of users, so the type *Conn* is defined as follows:

$$Conn = \mathbb{P}\, USR.$$

The set of established connections is then modelled by

$$established \quad \in \quad \mathbb{P}\, Conn.$$

If $\{a, b\} \in established$, then a connection is said to be established between users a and b. The complete model of the connection control system is defined as follows:

Connections

$$request : USR \pfun USR$$
$$established : \mathbb{P}\, Conn$$
$$active : \mathbb{P}\, USR$$

$$request \cap ID_{USR} = \{\}$$
$$(\mathrm{dom}\, request) \cap (\bigcup established) = \{\}$$
$$\mathrm{disjoint}\ established$$
$$(\forall\, cn \in established \bullet \#cn = 2)$$
$$active = (\mathrm{dom}\, request)\ \cup\ (\bigcup established)$$

The variable *request* records connections that have been requested but not yet established; if $request(a) = b$, then user a has requested a connection with user b. The variable *established* records all established connections, while *active* records all those users that are either requesting a connection, or involved in an established connection. The five invariants respectively constrain the system as follows:

- A user cannot request a connection with itself.
- A user cannot be both requesting a connection and involved in an established connection; this constraint could be relaxed to allow passage to multi-way connections.
- No user can be involved in more than one connection.
- Each established connection consists of only two distinct users; this constraint could be relaxed to allow multi-way connections.
- *active* is simply defined as the set of users requesting a connection or involved in an established connection.

This model of connection control was developed by first deciding on a way of modelling connections. Given this, we were in a position to elaborate the requirements by asking questions such as "can a user be involved in more than one connection?", or "is there a limit to the number of users that may be involved in a connection?". The model chosen allowed us to see clearly what questions should be asked. The constraints chosen are not imposed by the model, but rather by a set of requirements.

5.2 User Operations

A user $a?$ may request a connection with a user $b?$ using the following operation:

Request_Connection

$$\Delta Connections$$
$$a?, b? : USR$$

$$a? \notin active \land a? \neq b?$$
$$request' = request \cup \{a? \mapsto b?\}$$
$$established' = established$$

User $a?$ can only request a connection if it is not already active, and cannot request a connection with itself. The effect of the operation is to record the request, and leave existing connections unchanged. Note that the invariant

$$active = (\mathrm{dom}\, request) \ \cup \ (\bigcup established)$$

means that $a?$ is implicitly added to *active*.

A user with whom a connection has been established may accept that connection using the following operation:

Accept_Connection

$$\Delta Connections$$
$$b? : USR$$

$$(\exists\, a : USR \bullet$$
$$\quad request(a) = b?$$
$$\quad b? \notin active$$
$$\quad request' = \{a\} \lhd request$$
$$\quad established' = established \cup \{\ \{a, b\}\ \}\ )$$

The user $b?$ may accept the connection only if it is not already active. The effect of the operation is to delete the request and add the connection to those already established. ($\{a\} \lhd request$ represents the function *request* with the mapping for a removed.)

Finally, we describe an operation for clearing a connection. A user may invoke the clear operation if it has requested a connection, or if it is involved in an established connection. We specify these two cases separately and then

combine them using a special Z schema operator. The first case is where the user is clearing a request that has not yet been established:

Clear_Requested_Connection

$$\Delta Connections$$
$$a? : USR$$

$$a? \in \mathrm{dom}\, request$$
$$request' = \{a?\} \lhd request$$
$$established' = established$$

The second case is where the user is clearing an established connection:

Clear_Established_Connection

$$\Delta Connections$$
$$a? : USR$$

$$(\exists\, cn : Conn \bullet$$
$$\quad cn \in established \land a? \in cn$$
$$\quad request' = request$$
$$\quad established' = established \setminus \{cn\}\ )$$

The two cases are combined to form a single operation using the schema disjunction operator as follows:

$$Clear\_Connection \quad = \quad Clear\_Requested\_Connection \lor Clear\_Established\_Connection$$

The effect of the schema disjunction operator is to merge the two declarations, and produce a predicate which is the logical disjunction of the predicates of the component schemas.

The precondition of the *Clear_Connection* operation is the disjunction of the respective pre-conditions of *Clear_Requested_Connection* and *Clear_Established_Connection*:

pre *Clear_Connection*

$$Connections$$
$$a? : USR$$

$$a? \in \mathrm{dom}\, request$$
$$\lor$$
$$a? \in (\bigcup established)$$

The separate cases for the clear operation were determined by our conceptual model. Because of our model, a user could either be in the domain of *request*, the range of *request*, in some established connection, or in none of these. We decided that a user wouldn't need to clear a connection if it is not active ($a \notin (\mathrm{dom}\, request) \cup (\bigcup established)$), which means we only provide for it being in the domain of *request* or in some established connection.

Observation: Clearly the conceptual model is adequate for describing the required operations. But there may be other operations for which it is inadequate. For instance, we do not model which user of an established connection requested that connection. If we wished to add some billing operations to the specification, then such information would be necessary in order to determine which users to charge.

6 Directory service

In this part of the document, we address the specification of information structures which are inherently a part of any telecommunication system.

Let us consider a simple model of a directory service. We may consider this to be a mapping from persons to telephone numbers:

TELEPHONE_DIR

$$\delta_p : PERSON \pfun NUM$$

But, in law, a company is also a person, and hence, we need make no distinction between human person and non-human person. Again, we know from experience that one person may have many numbers and, alternatively, many persons may *de facto* share the same number. Hence, our model must be modified to give

TELEPHONE_DIR

$$\delta_p : PERSON \pfun \mathbb{P}\, NUM$$

6.1 Persons, names and identifiers

The treatment of location leads to consideration of coordinate functions such as latitude and longitude pairs. Rather than choose a conventional target domain of R^3 for the coordination functions, using cartesian coordinates or spherical coordinates, we might prefer or even use in addition a different frame of reference such as (state, region, area, time). Whatever the choice, we are aware of a relationship between locations and their 'names'. This relation is captured by the notion of a 'directory'.

Persons exist at a location (and point in time). They certainly do not 'live' in a computing system. For purposes of billing, etc., we must keep a record of persons by recording their personal data. Of the latter, the person's name is of key significance:

$$PERSON \pfun Pnm$$

In addition, since names are not usually unique, it is customary to give each person a unique identifier

$$PERSON \pfun Pid$$

An association of name and identifier will also be modelled as a directory. Thus, we may immediately state the following requirements, backed up with supporting models.

REQUIREMENT 6.1 *The system must keep a registry of persons names and their unique identifiers.*

JUSTIFICATION 6.1 *A current telecommunications subscriber is aware of the existence of telephone directories, facsimile directories, etc. We may consider this new registry to be an appropriate abstraction which correctly generalises such directories.*

In keeping with the person-oriented view of the requirements, we will model this as

$$\delta_p \in P\_DIR = Pnm \nrightarrow Pid$$

This is termed the person-oriented view in the sense that persons first consider themselves as being named and only afterwards as being identified (abstractly).

From the system point of view we may invert the model to obtain

$$\delta_p^{-1} \in (Pnm \nrightarrow Pid)^{-1} = Pid \nrightarrow Pnm$$

The model δ_p assumes that every person is uniquely named. Since, in reality, such is not the case, then we may do one of two things:

1. Use the standard technological solution which maps unique identifiers onto names:

$$Pid \nrightarrow Pnm$$

2. Alternatively, we may partition the original δ_p^{-1} by elaborating a name into a structure, thereby achieving uniqueness.

Since, such detail is of not of importance at this stage we will assume simple uniqueness of person names.

Using the directory (or register, catalogue, etc.,) as paradigm, it is simple to populate the 'specification world' with requirements and models. For example, we will want to keep a register of services:

REQUIREMENT 6.2 (SERVICE–REGISTER) *The system must keep a registry of services and their unique identifiers.*

JUSTIFICATION 6.2 *In the case of a freephone service, current network operators maintain a register of services and service providers.*

Without repeating all of the analysis and commentary used for the 'telephone directory', it is probably sufficient just to mention the model

$$\delta_s \in S\_DIR = Snm \pfun Sid$$

Similarly, for ISDN, whether simple or broadband, it seems reasonable that the system keep track of terminal equipment:

REQUIREMENT 6.3 (TERMINAL–REGISTER) *The system must keep a registry of terminal equipment and their unique identifiers.*

JUSTIFICATION 6.3 *In mobile telephone systems there is a register of mobiles kept, each of which may be considered to be identified by a unique electronic serial number (ESN).*

Naturally, there is a corresponding model:

$$\delta_t \in T\_DIR = Tnm \pfun Tid$$

It is surely clear that, given a description of some system whether a telecommunications one or not, we may build models around entities, entity names and (unique) entity identifiers. Such models fall into the general class of directory models. Other names in use are register, catalogue, etc., depending on the application.

6.2 Directory use

Let us suppose that a user wishes to obtain the telephone number[1] of a particular person. It may be the case that, in an implementation, the response is (i) a voice message, (ii) a window display on a workstation, (iii) ...

For convenience we reproduce the 'telephone directory' model:

TELEPHONE_DIR

$$\delta_p : PERSON \pfun \mathbb{P}\, NUM$$

In its most simple form, the directory inquiry is a simple lookup operation:

Lookup

$$\Xi TELEPHONE\_DIR$$
$$p? : PERSON$$
$$n! : \mathbb{P}\, NUM$$

$$n! = \delta_p(p?)$$

[1] We use the idea of telephone number solely for convenience. From the above discussion, it is clear that all sorts of identifiers may stand in its place.

with the usual pre-condition. The result of the inquiry is a set of telephone numbers. There are clearly cases when this **is** the desired result, for instance, in ISDN. However, in other situations/implementations (such as hunting lines) the user expects only one 'free line'. We may model this, using a non-deterministic choice:

Lookup

$$\Xi TELEPHONE\_DIR$$
$$p? : PERSON$$
$$n! : NUM$$

$$n! \in \delta_p(p?)$$

The effect of the non-deterministic choice is to return a single number from a non-empty set. There is no interpretation as to what such a choice might mean in a real application. For example, if we really want to model the 'free line' then we must introduce another model of the system state which keeps track of busy lines, etc.

Again space prohibits further extensive development here. To complete this part on directory services we now consider the typical freephone service and demonstrate the notion of lateral elaboration of a model.

7 Green number service

Green number service is the term used in the Intelligent Network literature to describe the generalisation of the '800' number service of the US [1] pp.105 *et seq*. The basic service is succinctly described as follows:

> "Calls containing a special access code (for example in the USA 800, in West Germany 130) trigger the interrogation (via CCS7) of a database to determine the real (target) number. The real number is used for call setup".

We introduce the abstract model of a *green number* directory:

GN_DIR

$$\delta_g : GNUM \nrightarrow \mathbb{P}\, NUM$$

Annotations:

- There may be one or more real numbers associated with a given green number g, e.g., $g \mapsto \{n_1, n_2, \ldots, n_k\}$
- There may be two different green numbers which have the same real number associated with them, e.g., $g_j \mapsto \{n\}, g_k \mapsto \{n\}$

¿From the structure of the model it is clear that making a freephone call is exactly equivalent to the lookup operation of a directory service. However, it is of interest to examine more closely the interpretation that is to be given to the set of numbers returned. Indeed, we will want to choose just one number from the set, recalling our earlier brief discussion on the non-deterministic choice.

One possible interpretation relies on the notion of time. The number to be selected is to be determined by the time when the freephone call is made:

GN_DIR

$$\delta_g : GNUM \pfun (TIME \pfun NUM)$$

For the lookup operation, we will use a time-selector function, π_t, which has the intended meaning that the real number to be selected is that for which the time at which the freephone call is made, t, is greater than or equal to the corresponding time index in the directory:

Lookup

$$\Xi GN\_DIR$$
$$n? : GNUM$$
$$t? : TIME$$
$$n! : NUM$$

$$n! = \pi_{t?}(\delta_g(n?))$$

Yet to be done is the complete specification of the time-selector operation $\pi_{t?}$.

A different interpretation, especially important in the European context, is based on locality. Assuming a European-wide freephone service, it is natural to select the real number based on the locality of the originating freephone call. The corresponding model and associated operations is similar to that of the time-based system above.

Clearly, we may mix the two approaches, incorporating both time and location.

8 Multimedia user interface

Services must, of course, be delivered to the end-user. Ultimately, it is upon the end-user interface that the success of broadband communications will depend. For this reason, it seems appropriate that one address the specification of a 'typical' modern user interface or, as more recently termed, a mediaphone [10].

For simplicity we may assume that a multimedia service interface may be abstracted as a conventional 'workstation' which has been described as an "*à la carte* [Broadband ISDN] terminal" [4] p. 185. Naturally, such a general model will require considerable tailoring (and re-interpretation) if one were to adapt it for a simple 'audio' phone.

8.1 The screen

Let us begin by proposing that the service end-user is confronted by a workstation screen ($s \in SCREEN$), absolutely essential for any sort of video. The screen is divided up into different areas of interest which we shall consider to be windows ($w \in WINDOW$), each of which is uniquely identified ($wi \in Wid$).

Windows which are open and on display to the user will be termed 'displayed windows' ($d \in DSPWINS$). Such windows have a particular visibility order ($v \in VISORDS$) or priority which is completely determined by an end-user's 'current' focus of attention. In conventional jargon one speaks of the 'current' window as the active window.

For convenience, and particularly to conserve screen space and avoid clutter, the end-user will have closed some windows ($c \in CLSWINS$), their very existence perhaps being determined by icons or names. Naturally, such closed windows do not have a visibility order.

We model the screen as an ordered collection of displayed windows, their visibility order, and the closed windows:

$$\begin{array}{|l} \textit{SCREEN} \\ \hline d : DSPWINS \\ v : VISORDS \\ c : CLSWINS \\ \hline \end{array}$$

For the collection of displayed windows we will use the simple idea of an association between unique window identifiers and windows:

$$d \in DSPWINS = Wid \pfun WINDOW$$

where a window will be considered to be nothing more than something which we shall call a window specification ($s \in WINSPEC$), yet to be determined:

$$w \in WINDOW = WINSPEC$$

The visibility order of the displayed windows may be modelled as a sequence of the corresponding window identifiers

$$v \in VISORDS = \mathrm{seq}\ Wid$$

A typical sequence such as

$$v = \langle\, wi_1, wi_2, \ldots, wi_n \,\rangle$$

is to be interpreted from left to right, indicating that in this particular case wi_1 denotes the current window.

Finally, the closed windows are modelled simply by a set of window identifiers

$$c \in CLSWINS = \mathbb{P}\ Wid$$

The various components of the screen are not independent. For example, the window identifiers which 'index' the 'table' of displayed windows ($d \in DSPWINS$) **must** be exactly those which are listed in the visibility order structure ($v \in VISORDS$). Such necessary constraints on the screen model are formally specified by an invariant:

inv *SCREEN*

$$SCREEN$$

$$\begin{array}{l} \mathrm{dom}\, d = \mathrm{ran}\, v \\ \mathrm{dom}\, d \cap c = \{\} \wedge \ldots \end{array}$$

The invariants are interpreted as follows:

- the window identifiers which 'index' the 'table' of displayed windows ($d \in DSPWINS$) **must** be exactly those which are listed in the visibility order structure ($v \in VISORDS$), and
- the window identifiers which 'index' the 'table' of displayed windows ($d \in DSPWINS$) **must** be totally disjoint from those which indicate the set of closed windows ($c \in CLSWINS$), and
- there may be other constraints which are either necessary, or which we may wish to impose.

Given such mathematical models, there are inevitable operations which may be performed, giving rise to expressions which correspond to questions which we might wish to pose. However, since we have already illustrated this aspect of the specification earlier in the document, we will address instead operations which a service end-user might wish to perform at the interface.

8.2 User operations

Let us suppose that a particular end-user is being called. We will first suppose that the system indicates the call by alerting the user in some manner, which we will initially consider to be a 'closed window' ($wi?$). The end-user acknowledges the call by clicking on the appropriate icon, say, thereby causing it to be 'opened'. Such an action will affect the currently displayed windows, their visibility order and the collection of closed windows:

AckCall

$$\Delta SCREEN$$
$$wi? : Wid$$

$$\begin{array}{l} d' = d \cup \{wi? \mapsto \ldots\} \\ v' = \langle\, wi? \,\rangle^\frown v \\ c' = c \setminus \{wi?\} \end{array}$$

The necessary pre-condition is, of course,

pre *AckCall*

$$SCREEN$$
$$wi? : Wid$$

$$wi? \in c$$

Note the presence of the ellipsis in *AckCall*. This is an indication that we do not have sufficient information to state precisely what happens when the user acknowledges the call. In other words, the current model is deficient with respect to this particular operation.

9 Telecom Experiences with Z

The use of formal specification notations such as Z is becoming increasingly popular in the telecoms industry. In this section, we outline some experiences with the use of Z in the telecoms industry as reported in the literature.

The ARISE project (part of the RACE I program) investigated how Z could be used to promote reuse [2]. It was shown that Z could be used to provide abstract, succinct specifications of reusable components thus making them easier to reuse. A number of case studies involving the specification in Z of telecoms services were undertaken. As part of an overall design methodology, Z was integrated with the HOOD methodology [6]. It was shown how Z can be used to describe the functional behaviour of HOOD objects. The advantage of using a formal notation such as Z are that a common vocabulary between specifier and designer is provided, designs can be formally verified, and the abstraction provided can be used as a basis for reusability. Some prototype tool support for the combined Z/HOOD approach was also developed.

AT&T Bell Laboratories have undertaken case studies involving the specification in Z of an existing PABX (Private Automatic Branch Exchange) [13]. The aim of these case studies was to understand how to specify real switching systems, and to understand how best to combine partial specifications so that different aspects can be described independently and then combined. Operations of a PABX are very complex because of the many varied conditions that may apply. It was found that use of the schema calculus of Z allowed the complexity to be managed. Formulating invariants and proving they were maintained by each operation was seen as a good way of ensuring consistency of behaviour. Through the use of invariants in Z, some constraints were discovered that had not been documented in the system manuals and probably weren't intended in the original design.

Z has been used to fully specify the Open Distributed Processing (ODP) Trader Object [8] and the specification produced has been proposed as a standard for the object. The developers of this Z specification of the Trader Object have found that Z is suited to both the Information and Computational viewpoints of ODP standards.

British Telecom have used Z in Telecommunications Network Management [11]. ISO has produced standard descriptions of Managed Objects written in English. British Telecom are concerned with conformance of systems they develop to these standards, and felt the best way to approach this was to develop formal specifications of the standardised managed objects. It was found that Z was very well suited to specifying managed objects. A small hierarchy of managed objects has been specified in Object-Z (an object-oriented version of Z), and these have been implemented in C++. British Telecom are also developing a method for conformance testing based on Z [3], and are developing tools for automating the production of C++ implementations from Object-Z specifications [9].

Bell-Northern Research, Canada have investigated the use of VDM (a formal notation similar to Z) for the specification of OSI Managed Objects [7]. It was found that VDM provided a good way of describing relationships between managed objects, descriptions that are lacking in current standards.

The Australian Overseas Telecommunications Corporation (OTC), in collaboration with the University of Queensland, have also investigated the use of Object-Z for the specification of telecommunications services [5]. A large case study was undertaken involving the specification in Z of a system which would support the implementation of intelligent communications services. This case study in fact resulted in suggested improvements to Object-Z.

10 Glossary of Symbols

Logic

$\land, \lor$	logical-and, logical-or
$\neg$	negation
$\Rightarrow$	logical implication
$(\exists\, x \bullet P)$	exists an x such that P
$(\forall\, x \bullet P)$	forall x, P holds

Sets

$\{\cdots\}$	set delimiters
$\in, \notin$	set membership, non-membership
$\cup, \cap$	set union, set intersection
$\mathbb{P}\, S$	powerset of S
$\mathbb{P}_1\, S$	$\mathbb{P}\, S$ excluding the emptyset
$\mathbb{N}$	set of natural numbers
disjoint S	no two sets of S have elements in common
$S \subseteq T$	S is a subset of T
$S \times T$	cartesian product of S and T

Relations and Functions

$S \leftrightarrow T$	set of relations from S to T
$S \nrightarrow T$	set of partial functions from S to T
$x \mapsto y$	mapping of element x to element y
$f(x)$	application of function f to element x
f^{-1}	inverse of function f
$\mathrm{dom}\, f, \mathrm{ran}\, f$	domain of f, range of f
ID_S	identity function on set S
$f \oplus g$	function f overridden by function g
$S \lhd f$	function f with all elements in S removed from its domain

Sequences

$\mathrm{seq}\, S$	set of sequences of type S
$\langle \cdots \rangle$	sequence delimiters
$s \frown t$	s concatenated with t

References

[1] Wolf-Dietrich Ambrosch, Anthony Maher, and Barry Sasscer, editors. *The Intelligent Network, A Joint Study by Bell Atlantic, IBM and Siemens.* Springer-Verlag, Berlin, 1989.

[2] ARISE. A Reusability Approach to Software Engineering, Chapter 5 – Formal Methods and Reuse. ARISE Report VIII, 1992.

[3] E. Cusack and C. Wezeman. Deriving Tests for Objects Specified in Z. In *7th Z User Meeting*, December 1992.

[4] G. Dicenet. *Design and Prospects for the ISDN.* Artech House, London, 1987. David Oliver (trans.), from the French *Le RNIS, Techniques et Atouts*, Masson, Paris, 1987.

[5] R. Duke, G. Rose, and G. Smith. Transferring Formal Techniques to Industry: A Case Study. In *FORTE'90*, November 1990.

[6] HOOD. HOOD Reference Manual. Technical Report Issue 3.0, European Space Agency, 1989.

[7] L.S. Marshall and L. Simon. Using VDM to Specify Managed Object Relationships. In *FORTE'92*, October 1992.

[8] ODP. A Specification of the Basic ODP Trader in Z. Expert Contribution to SC21 WG7, November 1992.

[9] G-H.B. Rafsanjani and S.J. Colwill. From Object-Z to C++: A Structural Mapping. In *7th Z User Meeting*, December 1992.

[10] Srinivas Ramanathan and P. Venkat Rangan. Adaptive Feedback Techniques for Synchronized Multimedia Retrieval over Integrated Networks. *IEEE/ACM Transactions on Networking*, 1(2):246–60, April 1993.

[11] S. Rudkin. Modelling Information Objects in Z. In *International Workshop on ODP*, October 1991.

[12] J.M. Spivey. *The Z Notation*. Prentice Hall, 1989.

[13] P. Zave. Towards Formal Specification of Real Switching Systems. In *XIV International Switching Symposium*, October 1992.