| | |
|---|---|
| Author: | Timo Pärnänen |
| Document: | tcap |
| Date: | 06/11/1998 |
| Version: | 0.2 |

# TCAP

The tcap module implements the ITU-T Recommendations Q.771-774 [1][2][3][4] (Signalling System No. 7 - Transaction Capabilities Application Part). The TCAP is a set of communication capabilities that provides an interface between applications and a network layer service.

## 1 Introduction

The TCAP is a connectionless remote procedure call (RPC) and part of it is quite similar to the remote operation service element (ROSE), published in ITU-T X.219 and X.229. The TCAP operates on top of SCCP (Signaling Connection Control Part) and supports database access by the SS#7 (Signaling System No. 7) switches, and the applications that reside in the switch use TCAP to access information.

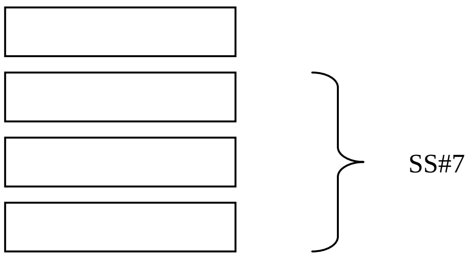
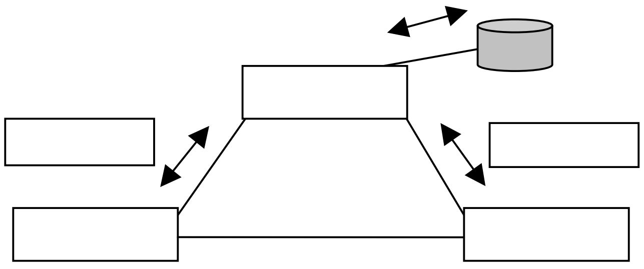


**Figure 1.** The location of TCAP module in a SS7 protocol stack.

The TCAP defines the format and fields that allow a service switching point (SSP) or a service control point (SCP) to formulate a TCAP access request to another node and a database (figure 2).



## 2 Architecture

The figure 3 describes the architecture of TCAP specified in ITU-T Recommendations. The TCAP is divided into two sub-layers, component and transaction sub-layers. Both layers include two or more blocks.

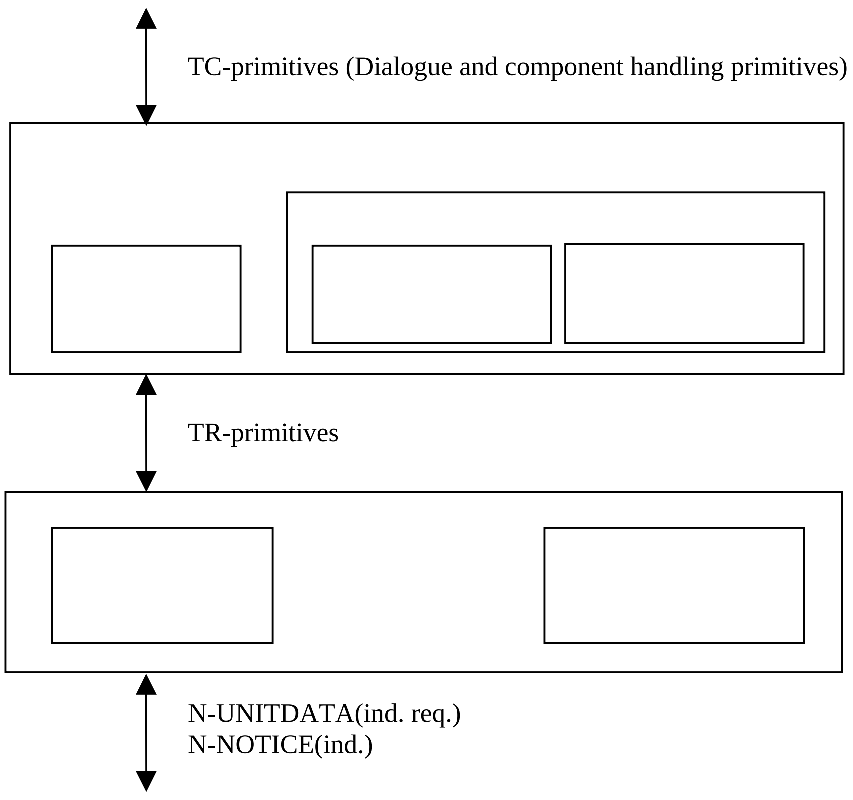


**Figure 3.** Overview block diagram of TCAP (ITU-T / Q.774).

In our implementation of TCAP, this structure is simplified by combining different blocks together. TSM and DHA have identical states, these and CCO are all combined to one class called Dialogue(Impl), Protocol(Impl) class plays role of TCO and ISMprotocol class implements actions defined in ISM block. Relations of these classes are shown in the figure 5.

### 2.1 Modules and interfaces

TOVE TCAP implementation is distributed with CORBA (ORBacus [6]). A TCAP adapter stays on top of TOVE SCCP and therefore it is implemented with C++. The purpose of the adapter is to transform messengers from a conduit graph to CORBA requests and vice versa. Other part of this TCAP implementation is implemented with Java (JDK 1.1.6 [7]).

TC-User (Transaction Capabilities) interface is implemented according to OMG Document, Interworking Between CORBA and TC systems [5]. The

TcPduProviderFactory interface is used to create new TC sessions for CORBA objects and to register and de-register CORBA objects to receive dialogs from TC/SS#7 entities. TcPduProvider interface is used to communicate with the TC/SS#7 stack by the TC-User. It provides both TC dialog handling and TC component handling primitives.

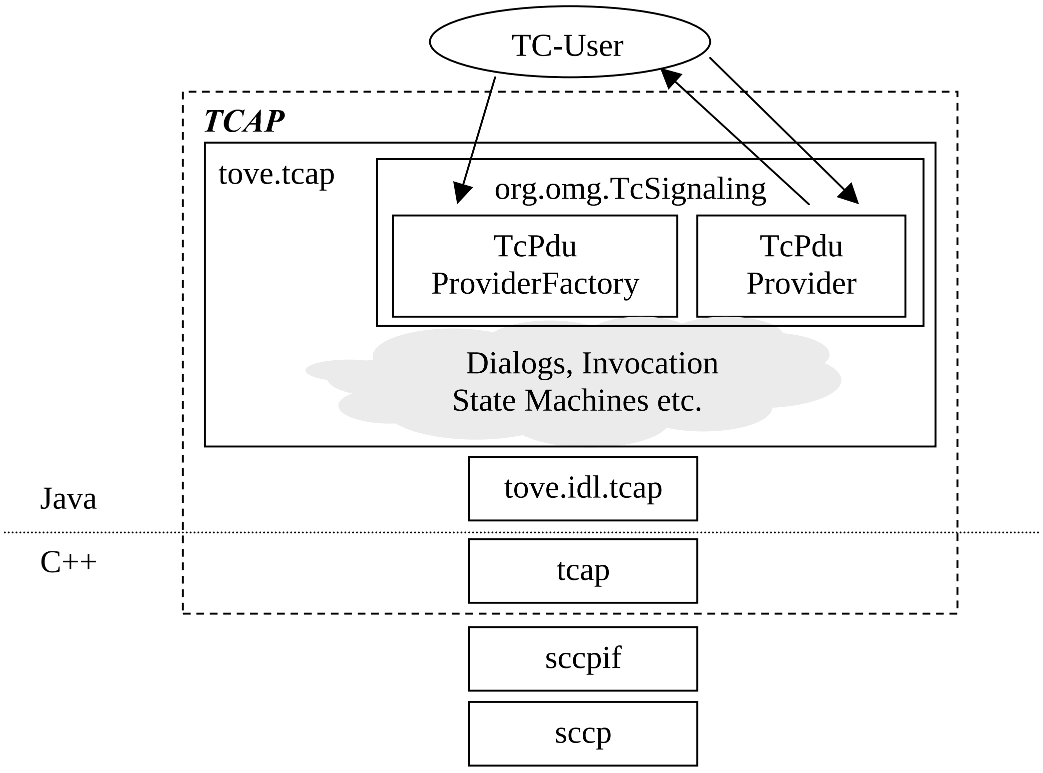


**Figure 4**. Modules and interfaces.

| | | |
|---|---|---|
| Upper protocol: | ingw (TC-User) | |
| Upper module: | tove.ingw | |
| Upper interface: | org.omg.TcSignaling | |
| Module: | tove.tcap (Java package) | Internal interface: tove.idl.tcap |
| | tcap (C++ module) | |
| Lower interface: | sccpif | |
| Lower module: | sccp | |

**Table 1.** The upper and lower interfaces and modules used by tcap module.

## 2.2 Object relations at runtime

Following figure 5 shows runtime object relations in TCAP module. Additionally there are message and component objects and state machine objects for ISM and Dialogue.

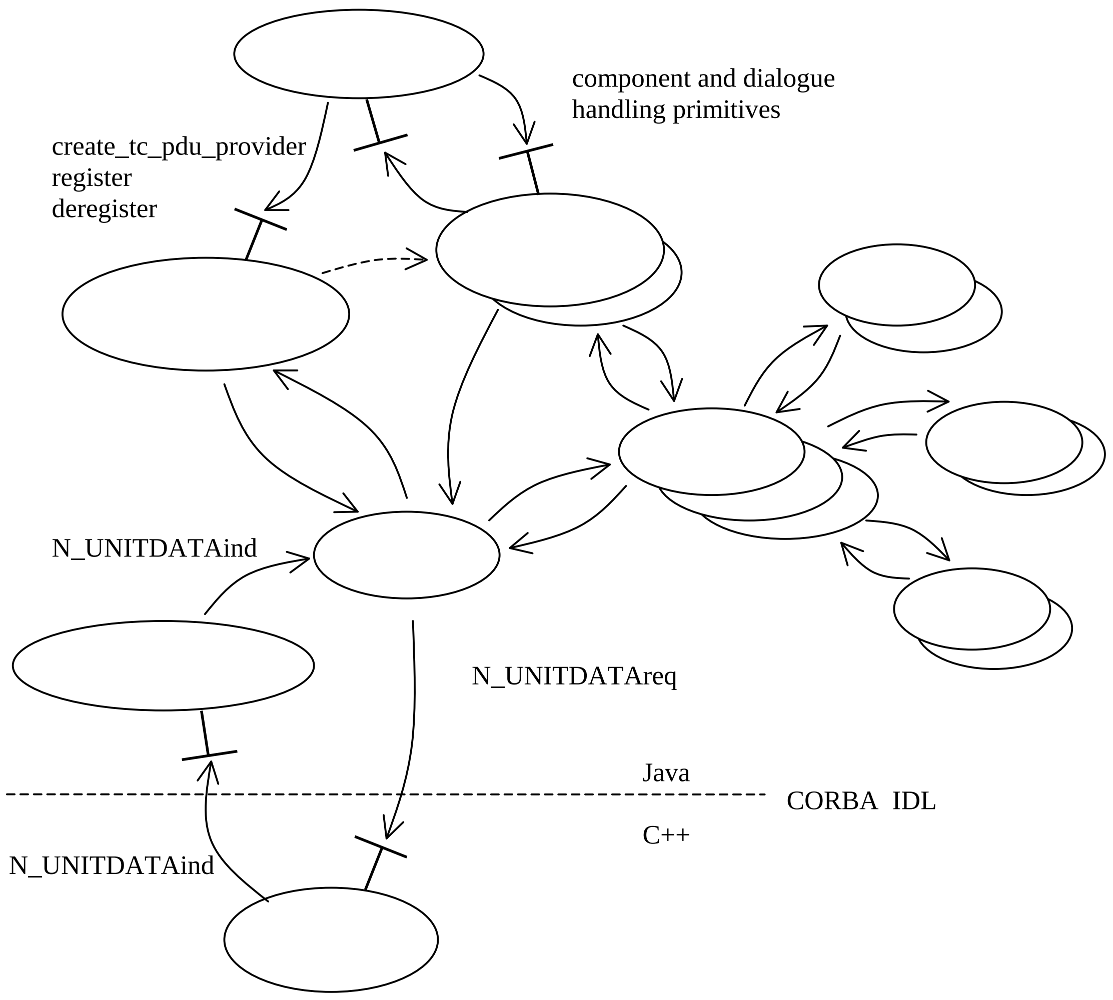


**Figure 5.** Object relations at runtime.

CORBA requests from TCAP adapter are received in CORBAInputHandler object. This object constructs message objects (N_UNITDATAind) from incoming CORBA requests. Created message is started at new Thread and it is accepted to a protocol when run method is called by the Java Virtual Machine. Threads are used to prevent CORBA requests to block.

The protocol creates and maintains dialogues, and maps remote and local dialogue ID pairs. The TCAP protocol decodes messages from CORBAInputHandler, and directs them to the right dialogue. When incoming message is a BEGIN message, the protocol checks, is there a TC-User registered with an address founded in the message. Outgoing messages TCAP encodes and then sends as CORBA requests to the TCAP adapter.

The dialogue uses SendInterface implemented by a TcPduProviderImpl to send dialog and component handling primitives to the TC-User. The user can set several components and then send them with a dialogue handling primitive. The dialogue stores and delivers components, and maintains invocation state machines (ISM).

## 2.3 Class hierarchy

The figure 6 shows component and message class hierarchies of the TCAP implementation. The grayish area of the class hierarchy is generated from ASN.1 (Abstract Syntax Notation One) description by Snacc for Java tool. A dashed line describes interface inheritance (italic font). Snacc generated message classes (Begin, End, Continue) have not a common base class (except ASN1Type interface). That makes it more difficult to use. Due to that problem, another message hierarchy is created, which have Snacc classes as attributes.

Components have a same kind of hierarchy as messages. Using these hierarchies, components can be stored easily inside messages. Messages get also common messenger features and in that way they can be transmitted between protocol and dialogue.

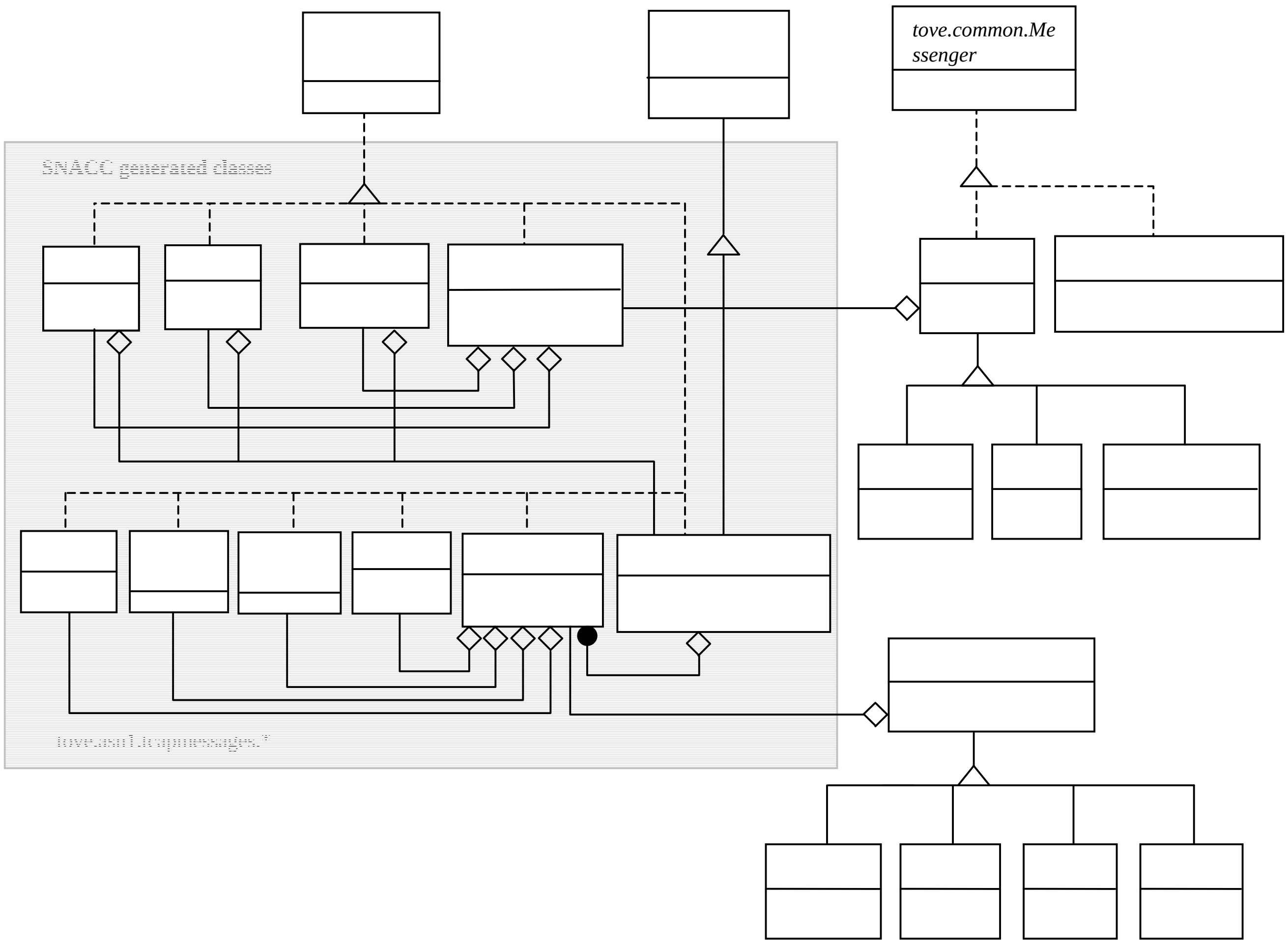


**Figure 6.** Class hierarchy of messages and components.

The figure 7 shows class hierarchies of the TCAP protocol, dialogue and ISM. All of these protocols extends common protocol implementation base class, and thus inherit a synchronous accept method and state machine features.

A part of actions for the protocol and dialogue are implemented as methods in messages and components.

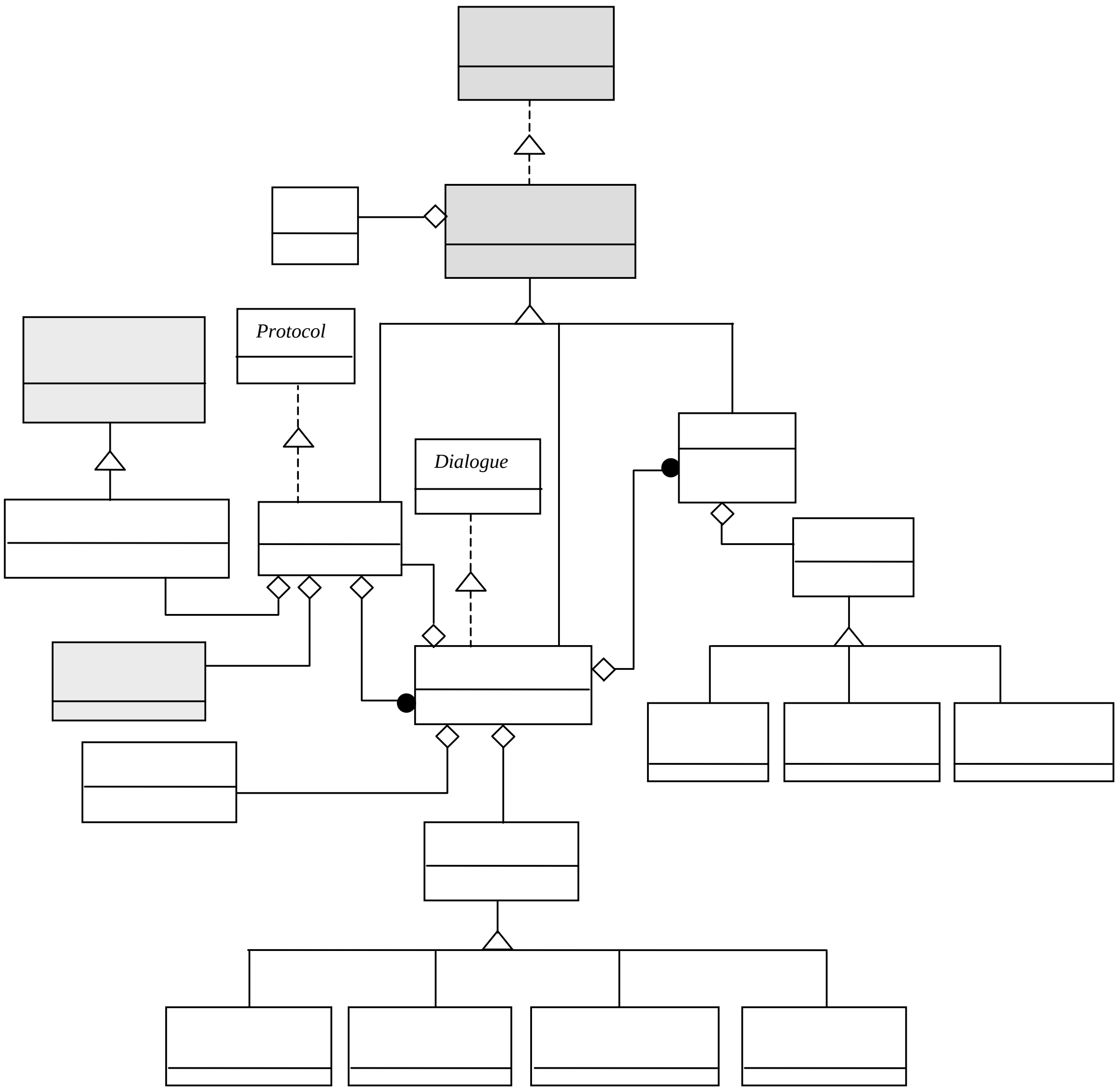


**Figure 7.** Class hierarchies for Protocol, Dialogue and ISM.

## 3 Implementation details

The following list describes some implementation details used in the TCAP:

- Allocation of IDs
  - Dialogue IDs are generated by the TCAP, but invoke IDs are allocated by the TC-User.
  - Transaction ID and dialogue ID are combined together, so there is one and only ID (dialogue ID) to identify dialogues. No separate transaction layer exists in this implementation.

## 4 Features implemented

This version of the TCAP does not implement all features described in relevant Recommendations by ITU-T and OMG Document. Table 2 lists unimplemented parameters (that are the not transferred and negotiated) in TC-primitives between the TCAP and user. Table 3 compares some implemented and unimplemented features in the TCAP protocol.

| **Dialogue handling** |
|---|
| Qos, DialogPortion, TerminationType |
| **Component handling** |
| OperationClass, LinkedID |

**Table 2.** Unimplemented parameters in TC-primitives.

| **IMPLEMENTED** | **NOT IMPLEMENTED** |
|---|---|
| Operation Class 1<br>(Both success and failure are reported) | Operation Classes 2-4 |
| Structured Dialogue<br>(BEGIN, END, CONTINUE) | Unstructured Dialogue<br>(UNI) |
| Normal end | Pre-arranged end, abort of dialogue |
| TC-primitives (except Notice and Abort) | TR-primitives<br>TC-NOTICE, TC-ABORT<br>Return Result Not Last component |

**Table 3.** Implemented and not implemented protocol features.

Additionally an Application Context Name is not supported in register/deregister CORBA methods in the TcPduProviderFactory interface. Unimplemented CORBA methods in the TcPduProvider interface are: uni_req, set_dialog_qos, get_dialog_qos u_abort_req, result_nl_req and destroy.

Due to lack of features in used Snacc ASN.1 tool for Java, some part of ASN.1 description (Q.773) are changed e.g. OPERATION and ERROR macros are replaced with more simpler types.

## 5 Known bugs and flaws

- Unimplemented features mentioned in previous chapter.
- The order of components changed in TCAP message coding. The reason can be found in Component Portion class (generated by Snacc). It uses a Hastable, which is unordered container.
- Decoding errors can not be specified and localized due to Snacc generated code. Only one common ASN1Excepion is thrown when error occurs.
- There might be memory leaking because destroy method is unimplemented. The TcPduProvider should know which dialogs it owns.
- Type of reject indication is not based on a problem type, but u_reject_ind is always used instead of r_reject_ind.

## 6 Future development

First feature that should be implement is destroy method in the TcPduProvider implementation. This means that TcPduProvider should know which dialogs it owns. The provider has to implement some kind of map of dialogs, which is synchronized with the dialogue map in the TCAP protocol.

Some problems may be fixed if ASN.1 tool will be changed or new version of Snacc for Java is released. Also some unimplemented features like missed parameters (table 2) could be implemented.

## 7 References

[1] ITU-T Recommendation Q.771, Signalling System No.7 – Functional Description of Transaction capabilities, 03/93

[2] ITU-T Recommendation Q.772, Signalling System No.7 – Transaction capabilities Information Element Definitions, 03/93

[3] ITU-T Recommendation Q.773, Signalling System No.7 – Transaction capabilities Formats and Encoding, 03/93

[4] ITU-T Recommendation Q.774, Signalling System No.7 – Transaction capabilities Procedures, 03/93

[5] OMG Document: telecom/98-10-03, Interworking Between CORBA and TC Systems, Revised (Final) RFP Submission, October 19, 1998

[6] ORBacus For C++ and Java, version 3.0, Object-Oriented Concepts, Inc., 1998.

[7] Java Development Kit ™ 1.1.6, Sun MicroSystems, Inc., 1998.