International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.625

(10/2010)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Broadband, triple-play and advanced multimedia services – Advanced multimedia services and applications

Architecture for network-based speech-to-speech translation services

Recommendation ITU-T H.625

ITU-T H-SERIES RECOMMENDATIONS

AUDIOVISUAL AND MULTIMEDIA SYSTEMS

CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	H.260–H.279
Related systems aspects	H.280–H.299
Systems and terminal equipment for audiovisual services	H.300–H.349
Directory services architecture for audiovisual and multimedia services	H.350–H.359
Quality of service architecture for audiovisual and multimedia services	H.360–H.369
Supplementary services for multimedia	H.450–H.499
MOBILITY AND COLLABORATION PROCEDURES	
Overview of Mobility and Collaboration, definitions, protocols and procedures	H.500–H.509
Mobility for H-Series multimedia systems and services	H.510–H.519
Mobile multimedia collaboration applications and services	H.520–H.529
Security for mobile multimedia systems and services	H.530–H.539
Security for mobile multimedia collaboration applications and services	H.540–H.549
Mobility interworking procedures	H.550–H.559
Mobile multimedia collaboration inter-working procedures	H.560–H.569
BROADBAND, TRIPLE-PLAY AND ADVANCED MULTIMEDIA SERVICES	
Broadband multimedia services over VDSL	H.610–H.619
Advanced multimedia services and applications	**H.620–H.629**
IPTV MULTIMEDIA SERVICES AND APPLICATIONS FOR IPTV	
General aspects	H.700–H.719
IPTV terminal devices	H.720–H.729
IPTV middleware	H.730–H.739
IPTV application event handling	H.740–H.749
IPTV metadata	H.750–H.759
IPTV multimedia application frameworks	H.760–H.769
IPTV service discovery up to consumption	H.770–H.779

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T H.625

Architecture for network-based speech-to-speech translation services

Summary

Recommendation ITU-T H.625 defines the system architecture for network-based speech-to-speech translation (S2ST) on the basis of Recommendation ITU-T F.745 and serves as a technical introduction to the subsequent definitions of detailed system components and protocols. The scope of this Recommendation is to describe the functional architecture and mechanisms of network-based S2ST, interface protocols between S2ST modules, and a workflow of the network-based S2ST system.

History

Edition	Recommendation	Approval	Study Group
1.0	ITU-T H.625	2010-10-14	16

Keywords

Automatic speech recognition (ASR), machine translation (MT), modality conversion markup language (MCML), speech-to-speech translation (S2ST), text-to-speech synthesis (TTS).

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

Table of Contents

Introduction

Speech-to-speech translation (S2ST) technologies, which translate speech in a source language to speech in a target language, are an effective means of facilitating communication beyond language boundaries for those who do not speak a common language. Developing an S2ST system requires the construction of automatic speech recognition (ASR), machine translation (MT) and text-to-speech (TTS) synthesis components. Many ASR, MT and TTS systems for different language pairs have been developed independently all over the world. If those systems could be connected together through a network, then S2ST systems spanning a greater number of language pairs can be achieved. To realize this network-based S2ST technology by connecting various distributed components, this Recommendation describes how to combine those individual S2ST systems and defines architectural requirements for the network-based S2ST system.

Recommendation ITU-T H.625

Architecture for network-based speech-to-speech translation services

1 Scope

This Recommendation defines the following items for designing a platform of the network-based speech-to-speech (S2ST) system:

– a functional architecture and mechanisms of network-based S2ST;

– interface protocols between S2ST modules; and

– a workflow of the network-based S2ST system.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T F.745] Recommendation ITU-T F.745 (2010), *Functional requirements for network-based speech-to-speech translation services.*

[IETF RFC 2279] IETF RFC 2279 (1998), *UTF-8, a transformation format of ISO 10646*.

[IETF RFC 2396] IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI): Generic Syntax*.

[IETF RFC 2616] IETF RFC 2616 (1999), *Hypertext Transfer Protocol – HTTP/1.1*.

[IETF RFC 2818] IETF RFC 2818 (2000), *HTTP Over TLS*.

[IETF RFC 3550] IETF RFC 3550 (2003), *RTP: A Transport Protocol for Real-Time Applications*.

[W3C XML1.0] W3C XML1.0 (2008), *Extensible Markup Language (XML) 1.0*.

[W3C XML Schema] W3C XML Schema Part 2 (2004), *XML Schema Part 2: Datatypes Second Edition*.

[ISO 639-1] ISO 639-1 (2002), *Codes for the representation of names of languages – Part 1: Alpha-2 code*.

[ISO 639-2] ISO 639-2 (1998), *Codes for the representation of names of languages – Part 2: Alpha-3 code.*

[ISO 639-3] ISO 639-3 (2007), *Codes for the representation of names of languages – Part 3: Alpha-3 code for comprehensive coverage of languages.*

[ISO 639-5] ISO 639-5 (2008), *Codes for the representation of names of languages – Part 5: Alpha-3 code for language families and groups.*

[ISO 639-6] ISO 639-6 (2009), *Codes for the representation of names of languages – Part 6: Alpha-4 code for comprehensive coverage of language variants*.

NOTE – In this Recommendation, "[ISO 639-1,2,3,5,6]" refers collectively to the five parts of ISO 639 listed above.

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 adaptive differential pulse code modulation (ADPCM) [b-ITU-T G.701]: Compression algorithms that achieve bit rate reduction through the use of adaptive prediction and adaptive quantization.

3.1.2 machine translation (MT) [ITU-T F.745]: Text in a source language is converted by computers into text in a target language what has the same meaning as the original text in the source language.

3.1.3 modality conversion (MC) [ITU-T F.745]: The conversion of data to different formats and languages using automatic speech recognition (ASR), machine translation (MT) and text-to-speech (TTS) systems.

3.1.4 modality conversion markup language (MCML) [ITU-T F.745]: An XML schema that serves as a data description for data exchanged among modality conversion modules.

3.1.5 modality conversion protocol (MCP) [ITU-T F.745]: The communication protocol which transfers data between modality conversion (MC) clients and servers using HTTP(S)/RTP as defined in [IETF RFC 2616], [IETF RFC 2818] and [IETF RFC 3550]. This protocol transfers the MCML comprising of Multimodal Information (MI) data which is input into MC clients by users and MC results which are obtained by MC servers.

3.1.6 multimodal information (MI) [ITU-T F.745]: The information input into MC clients by users via multimodal sensors.

3.1.7 multipurpose Internet mail extensions (MIME) [b-ITU-T J.200]: An application layer protocol. It features a content architecture to facilitate multimedia data such as text other than US-ASCII code, sound, image, etc. to be handled in Internet mails.

3.1.8 N-best [ITU-T F.745]: The most likely "N" hypotheses obtained from modality conversion engines.

3.1.9 pulse code modulation (PCM) [b-ITU-T G.701]: A process in which a signal is sampled, and the difference between each sample of this signal and its estimated value is quantized and converted by encoding to a digital signal.

NOTE – The estimated values of the signal are calculated by a predictor from the quantized difference signal.

3.1.10 speech-to-speech translation (S2ST) [ITU-T F.745]: Speech in a source language is translated into speech in a target language.

3.1.11 text-to-speech synthesis (TTS) [b-ITU-T P.10]: A process generates a speech signal from text codes. It is usually composed of the two parts:

– a language-dependent text processing part (the high level processing part), which generates from the character string (by reading rules, vocabulary and semantic analysis) a set of phonetic, prosodic, etc., parameters which are used by:

  • an acoustical signal generating part, the synthesiser itself, which generates the audible speech.

3.2 Term defined in this Recommendation

This Recommendation uses the following term:

3.2.1 automatic speech recognition (ASR) [b-ITU-T P.10]: A system that can recognize continuous speech, often having phoneme-sized references, using lexical, syntactic, semantic, and pragmatic knowledge, and reacts appropriately (therefore having interpreted the message and found the corresponding action to be taken).

NOTE – This definition is the same as given for *continuous speech understanding system* in [b-ITU-T P.10].

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

ADPCM	Adaptive Differential Pulse Code Modulation
ASR	Automatic Speech Recognition
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identifier
IPA	International Phonetic Alphabet
MC	Modality Conversion
MCML	Modality Conversion Markup Language
MCP	Modality Conversion Protocol
MI	Multimodal Information
MIME	Multimodal Internet Mail Extensions
MT	Machine Translation
PCM	Pulse Code Modulation
RTP	Real-time Transport Protocol
S2ST	Speech-to-Speech Translation
TTS	Text-To-Speech synthesis
UCS	Universal Character Set
URI	Uniform Resource Identifier
UTC	Universal Time, Coordinated
UTF-8	UCS Transformation Format-8
XML	Extensible Markup Language

5 Conventions

In this Recommendation:

– The expression "**is required to**" indicates a requirement which must be strictly followed and from which no deviation is permitted if conformance to this Recommendation is to be claimed.

– The expression "**is recommended to**" indicates a requirement which is recommended but which is not absolutely required. Thus this requirement need not be present to claim conformance.

– The expression "**can optionally**" indicates an optional requirement which is permissible, without implying any sense of being recommended.

6 System functional architecture

6.1 System overview

This clause defines the functional architecture of network-based S2ST systems. Figure 6-1 shows the network-based S2ST system with data flow. The processing steps for a network-based S2ST are as follows:

Step 1: A user's speech is input into an S2ST client

Step 2: A speech signal of the user's speech is transferred to an ASR module

Step 3: A transcription of the original speech obtained by the ASR module goes back to the original S2ST client and is transferred simultaneously to an MT module

Step 4: A translation into a target language for the original speech obtained by the MT module goes to a target S2ST client and is transferred simultaneously to a TTS module

Step 5: A synthesized speech signal in the target language is transferred to the target S2ST client

Step 6: The synthesized speech is output from the target S2ST client

Data can be transferred through the same network, or different ones. To help with understanding the data flow of network-based S2ST, the case in which all data is transferred using the same network is illustrated in Figure 6-1.

Figure 6-2 shows the functional model extracted from Figure 6-1.

The system functional architecture for network-based S2ST requires the following components: S2ST clients, and ASR, MT and TTS servers. The S2ST clients function as MC clients, which request modality conversion of multimodal information input by users and which receive the modality conversion results from MC servers. The ASR, MT and TTS servers function as MC servers that convert the modality of the multimodal information input by users.

To achieve communication between MC clients and servers, a communication protocol is required, namely the Modality Conversion Protocol (MCP) with the Modality Conversion Markup Language (MCML). Focusing on communication via MCP, MCP clients send MCML to the MCP servers, and MCP servers receive MCML from the MCP clients.

Figure 6-3 shows the logical architecture of MCP client and server. This architecture is specified in a single MCP client and server combination. Each of the engines for ASR, MT and TTS is identified as modality conversion (MC) function in the MCP server.

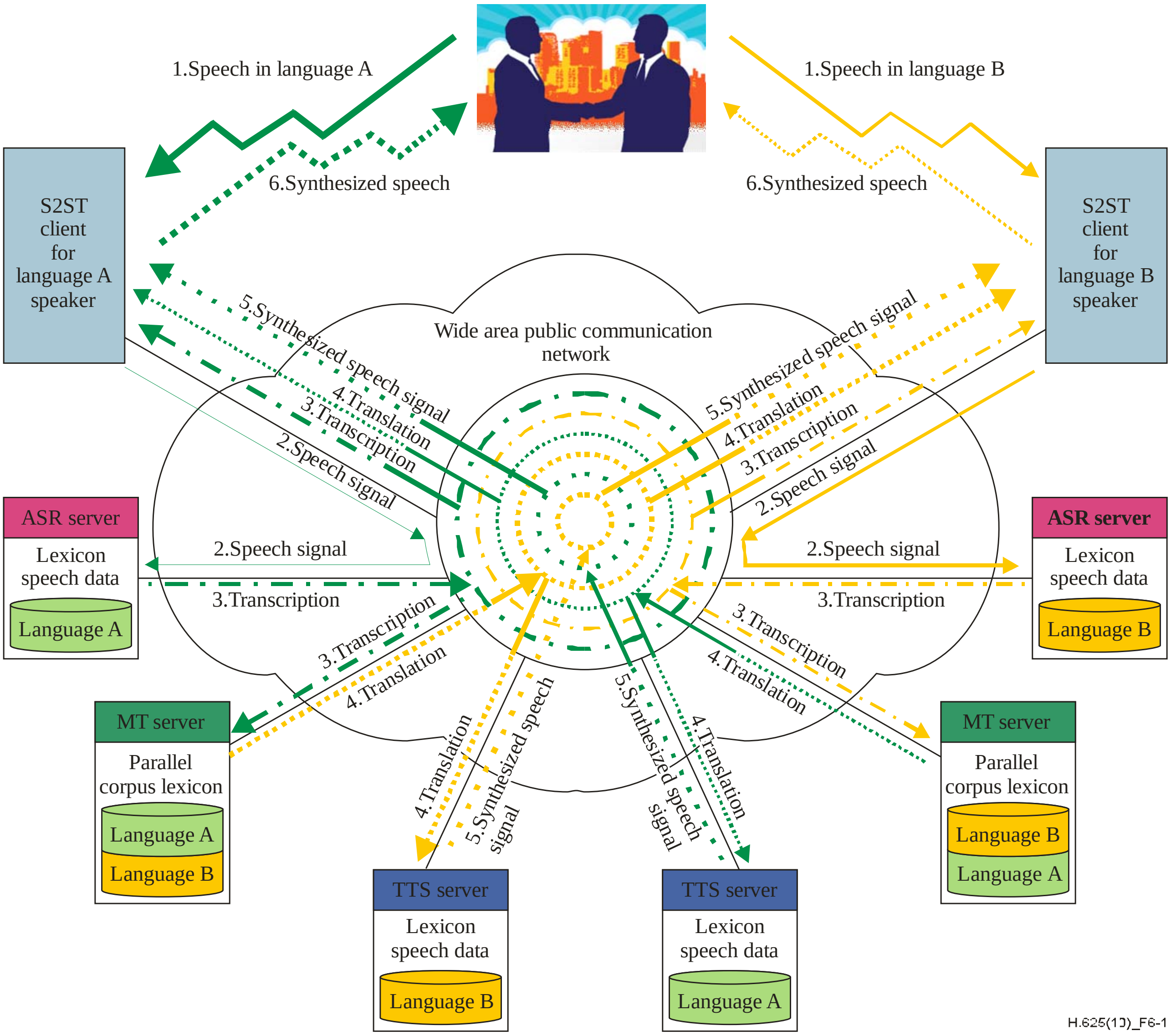

Figure 6-1 – A network-based S2ST system with data flow

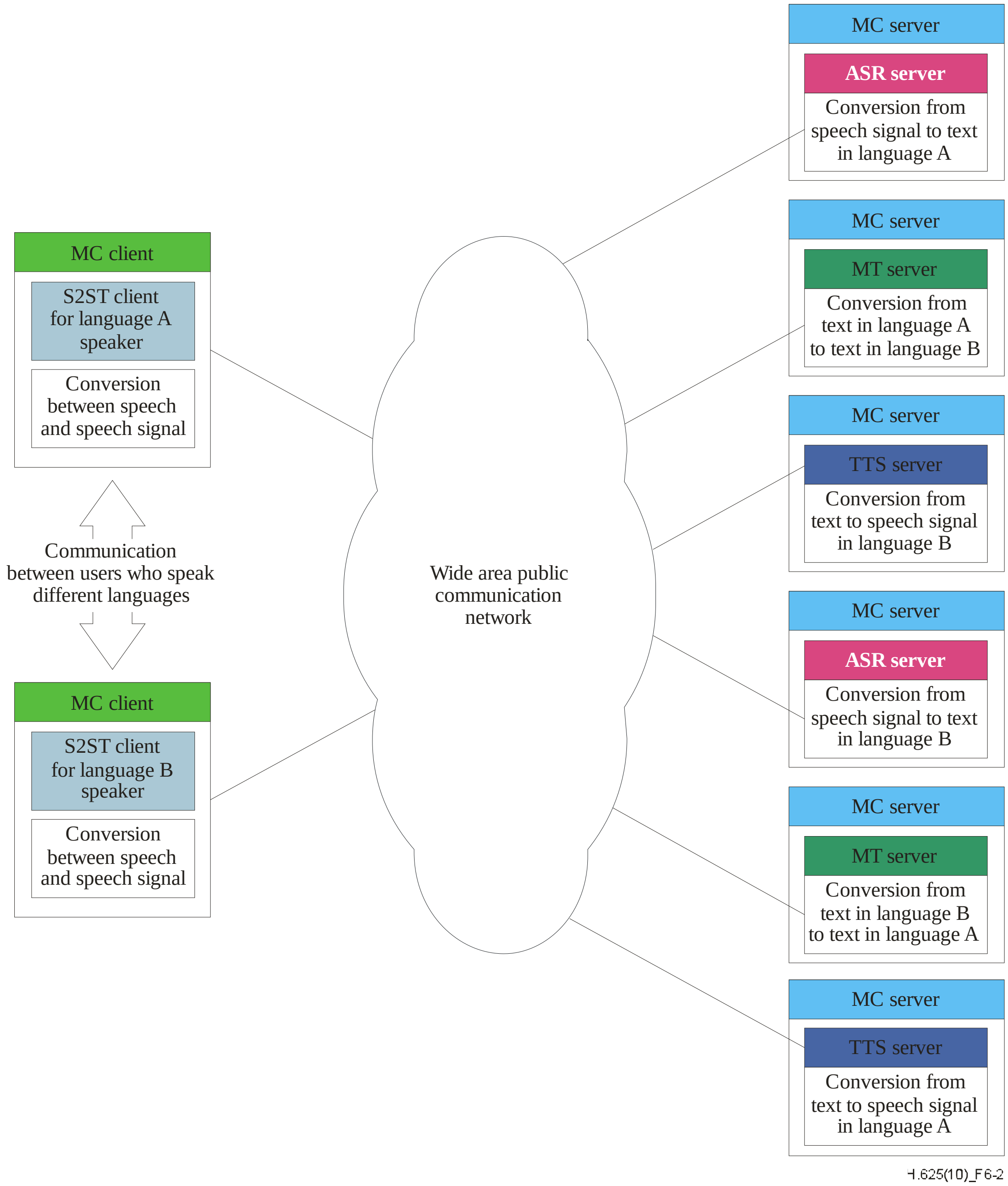
MC server
ASR server
Conversion from speech signal to text in language A
MC server
MT server
Conversion from text in language A to text in language B
MC server
TTS server
Conversion from text to speech signal in language B
MC server
ASR server
Conversion from speech signal to text in language B
MC server
MT server
Conversion from text in language B to text in language A
MC server
TTS server
Conversion from text to speech signal in language A
MC client
S2ST client for language A speaker
Conversion between speech and speech signal
Communication between users who speak different languages
MC client
S2ST client for language B speaker
Conversion between speech and speech signal
Wide area public communication network
H.625(10)_F6-2

Figure 6-2 – Functional model of the network-based S2ST system

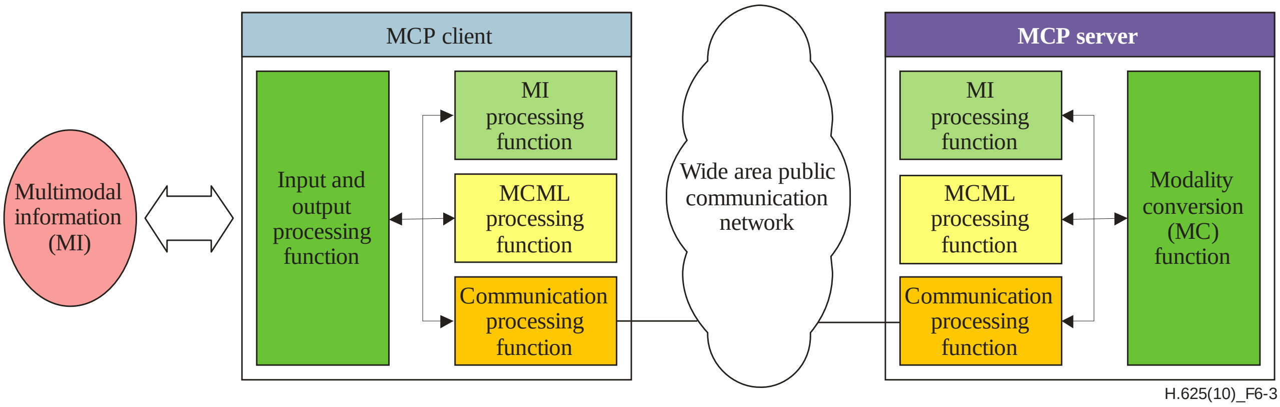
Multimodal information (MI)
MCP client
Input and output processing function
MI processing function
MCML processing function
Communication processing function
Wide area public communication network
MCP server
MI processing function
MCML processing function
Communication processing function
Modality conversion (MC) function
H.625(10)_F6-3

Figure 6-3 – Logical architecture of MCP client and server

To accomplish network-based S2ST, there are two approaches to communication via MCP, namely: MC client-server communication (Figure 6-4), and MC relay communication (Figure 6-5). To realize relay communication between MC servers, each server needs to be switched to MCP client mode from MCP server mode, enabling the information transfer. To confirm the quality of the output from each MCP server, the MCP server output can be sent back to each client using relay communication.

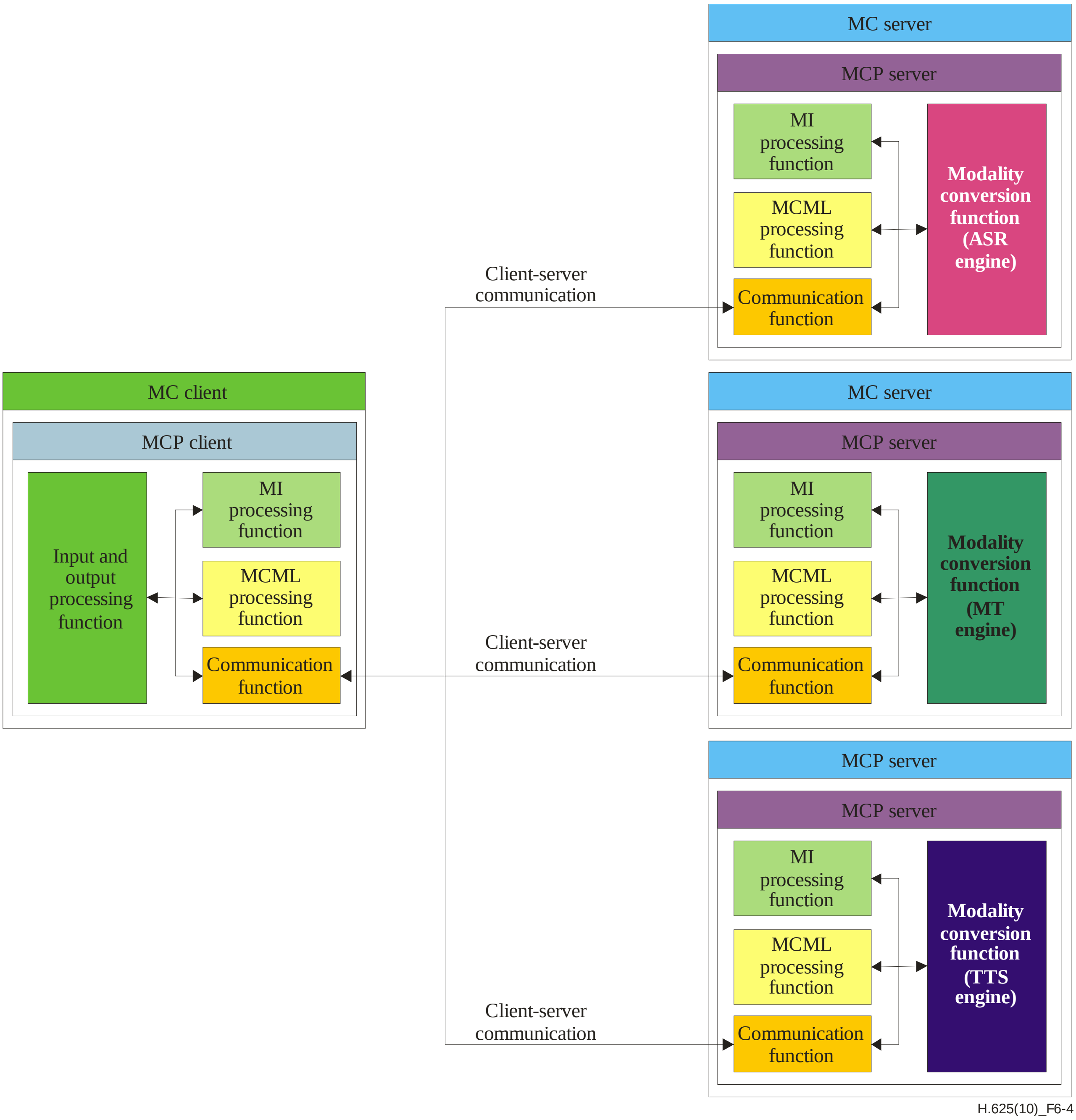

Figure 6-4 – MC client-server communication for network-based S2ST

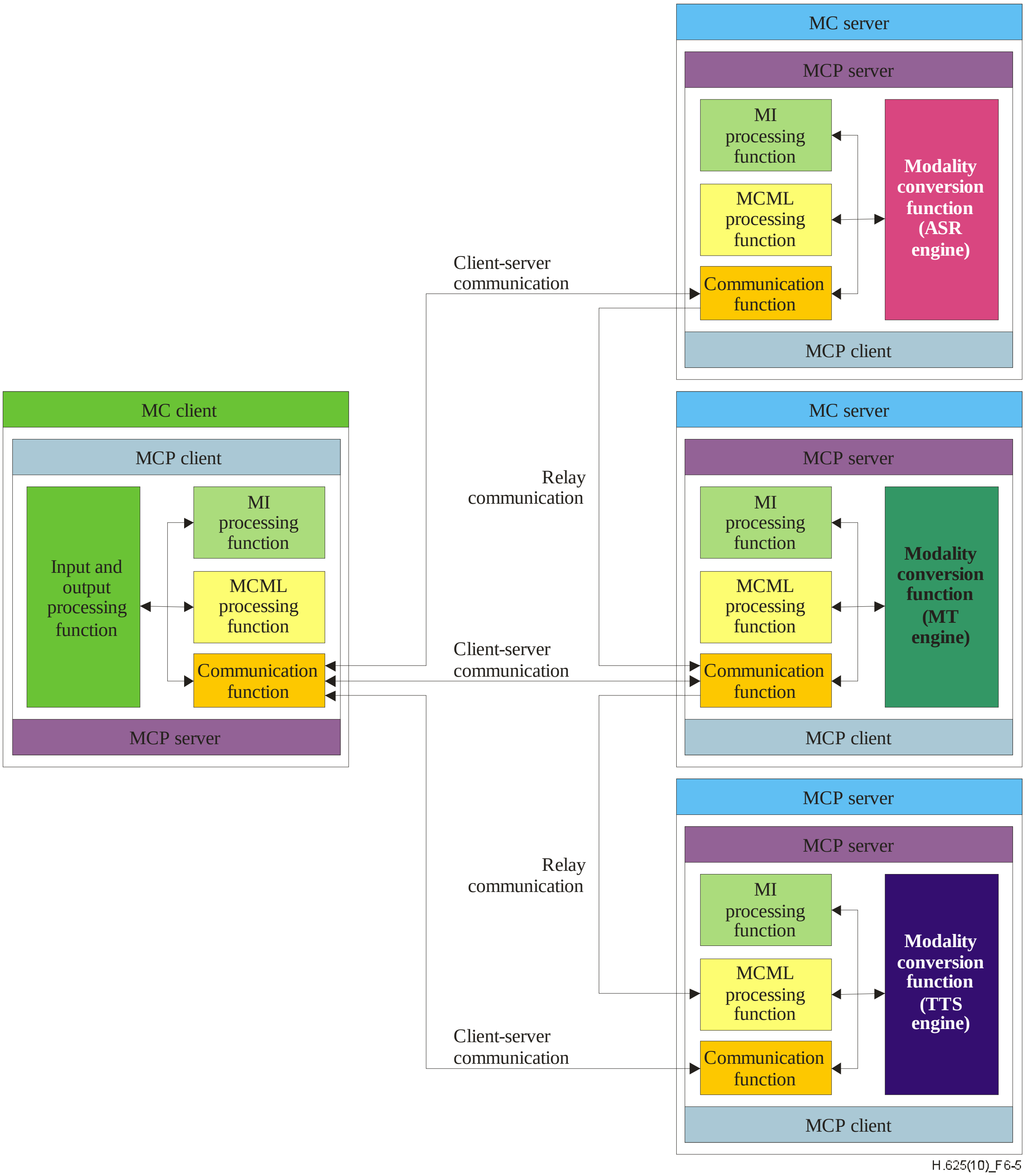

Figure 6-5 – MC relay communication for network-based S2ST

6.2 Functional components

The functional architecture of the modality conversion (MC) is required to consist of the following components: MC client and MC server. The clients and servers contain a set of MI processing, MCML processing, and communication functions. These clients and servers have an input and output processing function and a modality conversion function, respectively.

6.2.1 MCP client

The MCP client function is required to request modality conversion of the MI input by users. The MCP client has two different transfer functions: one is for user and device information as well as MI and MC results; and the other is for MC history through relay services. The MCP client is required to have the component functions described in the following subclauses.

6.2.1.1 Input and output processing function

The input and output processing function is required to input MI from users and output MI to users.

6.2.1.2 Multimodal information processing function

The MI processing function is required to encode the MI digitized by the input and output processing function into feature expressions. This function can be implemented either in the MCP client or the MCP server.

6.2.1.3 Communication processing function

The communication processing function is required to handle MCP to send MCML formatted data to the MCP server.

6.2.2 MCP server

The MCP server responds to MCP client requests for modality conversion (MC) of multimodal information (MI). The MCP server is required to be composed of the functions described in the following subclauses.

6.2.2.1 Communication processing function

The communication processing function is required to handle the multimodal communication protocol (MCP) to receive the MCP client requests, and to send modality conversion (MC) results of MI back to the MCP client from the MC server.

6.2.2.2 Multimodal information processing function

This is the same function described in clause 6.2.1.2. This function is needed in the MCP server, in case it is not implemented in the MCP client.

6.2.2.3 Modality conversion function

The MC function is required to convert the modality of MI in accordance with the MCP client request. Each processor for MC is denoted as an MC engine.

6.3 Protocols

Modality conversion (MC) is realized by communication between the MCP client and the MCP server, by exchanging modality conversion markup language (MCML) using the modality conversion protocol (MCP). The architecture described in clause 6.2 (Figure 6-1) is required to be supported by the communication protocols shown in Figure 6-6.

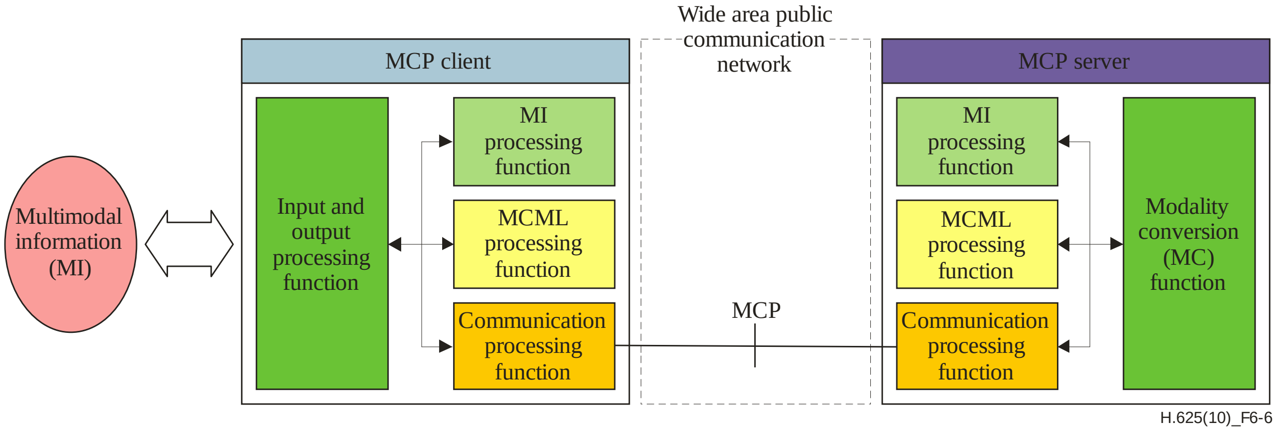
Wide area public communication network
MCP client
MCP server
Multimodal information (MI)
Input and output processing function
MI processing function
MCML processing function
Communication processing function
MCP
MI processing function
MCML processing function
Communication processing function
Modality conversion (MC) function
H.625(10)_F6-6

Figure 6-6 – Interface between MCP client and MCP server

6.3.1 MCP

Modality conversion protocol (MCP) is a communication protocol based on HTTP [IETF RFC 2616] or HTTPS [IETF RFC 2818]. MCML containing MI and MC results is embedded into MIME format with MI binary data attached. RTP [IETF RFC 3550] is used as an option. Figure 6-7 shows the MCP structure.

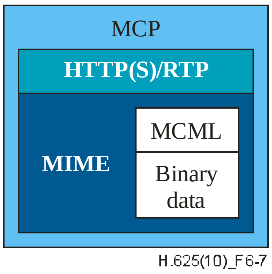

Figure 6-7 – MCP structure

6.3.2 MCML

The modality conversion markup language (MCML) is an XML schema [W3C XML Schema]. The message exchanged between MCP clients and servers is formatted in XML [W3C XML1.0] with the MCML schema.

MCML includes:

- user profile, user's demand for output and device information of the MCP client;
- service type and output information of the MCP server; and
- history of the MC process through multiple MCP servers.

UTF-8 [IETF RFC 2279] encoded text is required for MCML. The MCML schema is defined using the diagram and table representations described below, and its text representation is contained in Annex A.

(1) MCML Root

The root structure of MCML is shown in Figure 6-8. The root element of MCML is annotated as "MCML". An attribute of this element is required to indicate the MCML version information. The MCML element holds three child elements, i.e., "User", "Server" and "History". Table 6-1 shows the definition of the attribute and child elements.

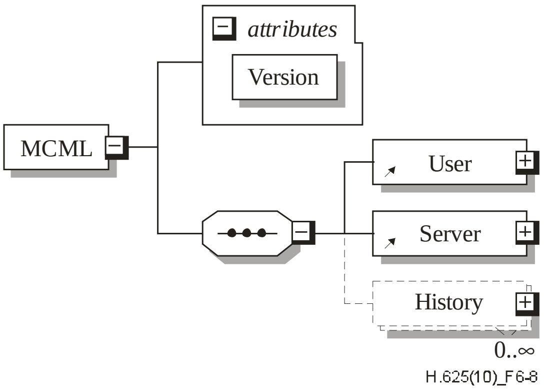

Figure 6-8 – MCML root structure

Table 6-1 – Definition of attributes and child elements of an "MCML" element

MCML				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	Version	Version information of MCML	1.0	1
Children	User	See (2)		1
	Server	See (3)		1 or more
	History	See (4)		0 or more

(2) User

The "User" element contains user and device information for the MC client. The structure of this element is shown in Figure 6-9. The communication between a single transmitter and single/ multiple receivers is required to be through MCML.

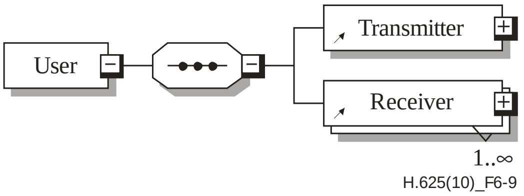

Figure 6-9 – Structure of a "User" element

Table 6-2 – Definition of the child elements of a "User" element

User			
Annotation	**Name**	**Definition**	**Quantity**
Children	Transmitter	See (2-1)	1
	Receiver		1 or more

(2-1) Transmitter and receiver

Transmitter:

This "Transmitter" element contains device and user information for an MC client, which sends MI to MC servers. The child elements are "Device" and "UserProfile", shown below in Figure 6-10 and in Table 6-3. Multiple users' profiles are required when sharing a single device among multiple users.

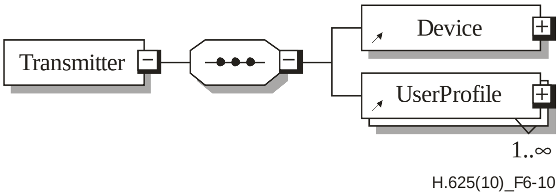

Figure 6-10 – Structure of a "Transmitter" element

Table 6-3 – Definition of child elements of a "Transmitter" element

Transmitter			
Annotation	**Name**	**Definition**	**Quantity**
Children	Device	See (2-1-1)	1
	UserProfile	See (2-1-2)	1 or more

Receiver:

This "Receiver" element contains the device and user profile information of MC clients which receives MC results. To receive the MC results from MC servers, the receiver device functions as an MCP server.

The "Receiver" element has the same structure as the "Transmitter" described above in this clause. The MCML is required to handle not only a single "Receiver" but also multiple "Receivers" in order to send MI and MC results to multiple MCP servers.

(2-1-1) Device

The "Device" element has the MC client's device information. The structure of the "Device" is shown in Figure 6-11. The child element is named "Location" and the attributes of this element are defined in Table 6-4.

To communicate between MC clients and servers, their addresses in the network are required. The child element of "Location" element is the "URI" element which has the information of the URI [IETF RFC 2396], and the URI identifies the location of the MC client in the network.

Since the environment for users may change depending on the user's location, the "GlobalPosition" element is required which contains the physical location of the MC client device in the real world. The "GlobalPosition" has a set of attributes consisting of "Longitude" and "Latitude".

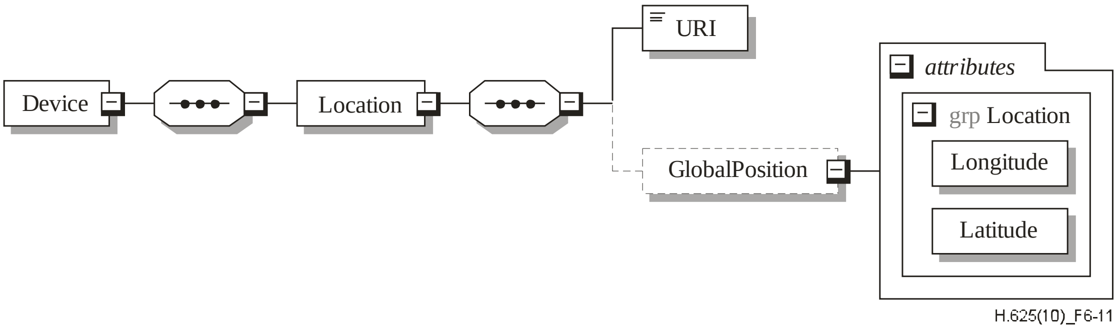

Figure 6-11 – Structure of "Device" element

Table 6-4 – Definition of child elements of a "Location" element and attributes of a "GlobalPosition" element

Location				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Children	URI	Uniform resource identifier	string	1
	GlobalPosition	See below		0 or 1
GlobalPosition				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	Longitude	The longitude of the physical location of the client device in the real world	float	1
	Latitude	The latitude of the physical location of the client device in the real world	float	1

(2-1-2) UserProfile

The "UserProfile" element represents personal information about the user as shown in Figure 6-12. The attributes of the "UserProfile" element consists of "ID", "Age", and "Gender". The "ID" attribute enables MC servers to identify the user and enhance the performance of MC engines. Each of the attributes has personal information about the user. The definition of the attributes of the "UserProfile" element is described in Table 6-5. To share a single device by multiple users, a function to handle multiple user profiles is required.

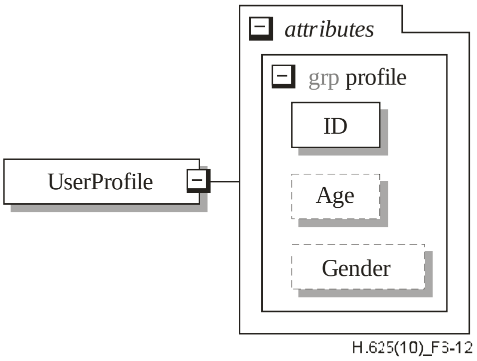

Figure 6-12 – Structure of a "UserProfile" element

Table 6-5 – Definition of attributes of a "UserProfile" element

UserProfile				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	ID	Identification name or number of the user	string	1
	Age	User's age	integer	0 or 1
	Gender	User's gender	"female"/ "male"	0 or 1

(3) Server

The "Server" element contains the type of service provided by MC servers, Request information including MI data input into the MC client and Response information including MC results output from the MC server. The structure of the "Server" element is shown in Figure 6-13.

The attributes and child elements of the "Server" are defined in Table 6-6. "Request" or "Response" is alternatively selected. When the MC client requests MC servers to convert modality, the MCML that is sent to the MC server contains a "Request" element. When the MCP servers send MC results back to the MC client, the MCML contains a "Response" element.

To achieve relay communication, MC results are transferred directly to the next MC server. To deal with multiple MC processes sequentially, multiple "Request" elements are to be listed with the associated process order attribute in the MCML sent by the transmitter. In case MCML data is directly transferred from an MC server to another MC server through the relay communication, the MCML, including the "Request" element, is sent to another MC server.

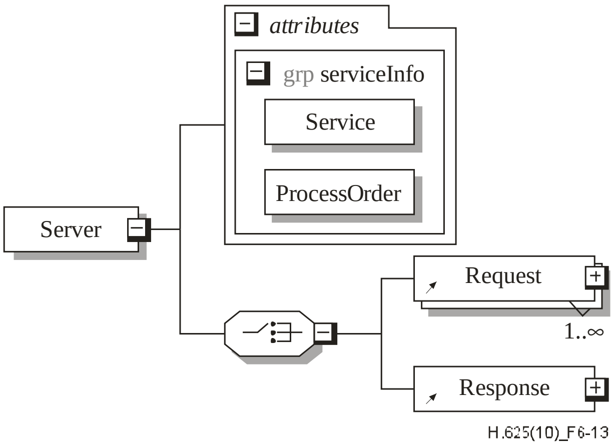

Figure 6-13 – Structure of a "Server" element

Table 6-6 – Definition of attributes and child elements of a "Server" element

Server				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	Service	Service type provided by MCP server: "ASR", "MT", "TTS", "SignRec" are reserved for MCML	string	1
	ProcessOrder	The process order for services accomplished by the combination of multiple MCP servers	integer	1
Children	Request	See (3-1)		1 or more
	Response	See (3-1)		1

(3-1) Request and response

The "Request" element indicates request information with MI data that is sent from the MC client to the MC server. The structure of the "Request" element is shown below in Figure 6-14. The child elements are listed in Table 6-7. The "Response" element includes MC results sent from the MC server. The "Response" element contains: "Routing", "Output" and "Error" as shown in Figure 6-14. These child elements of the "Response" and attributes of the "Error" element are described in Table 6-7.

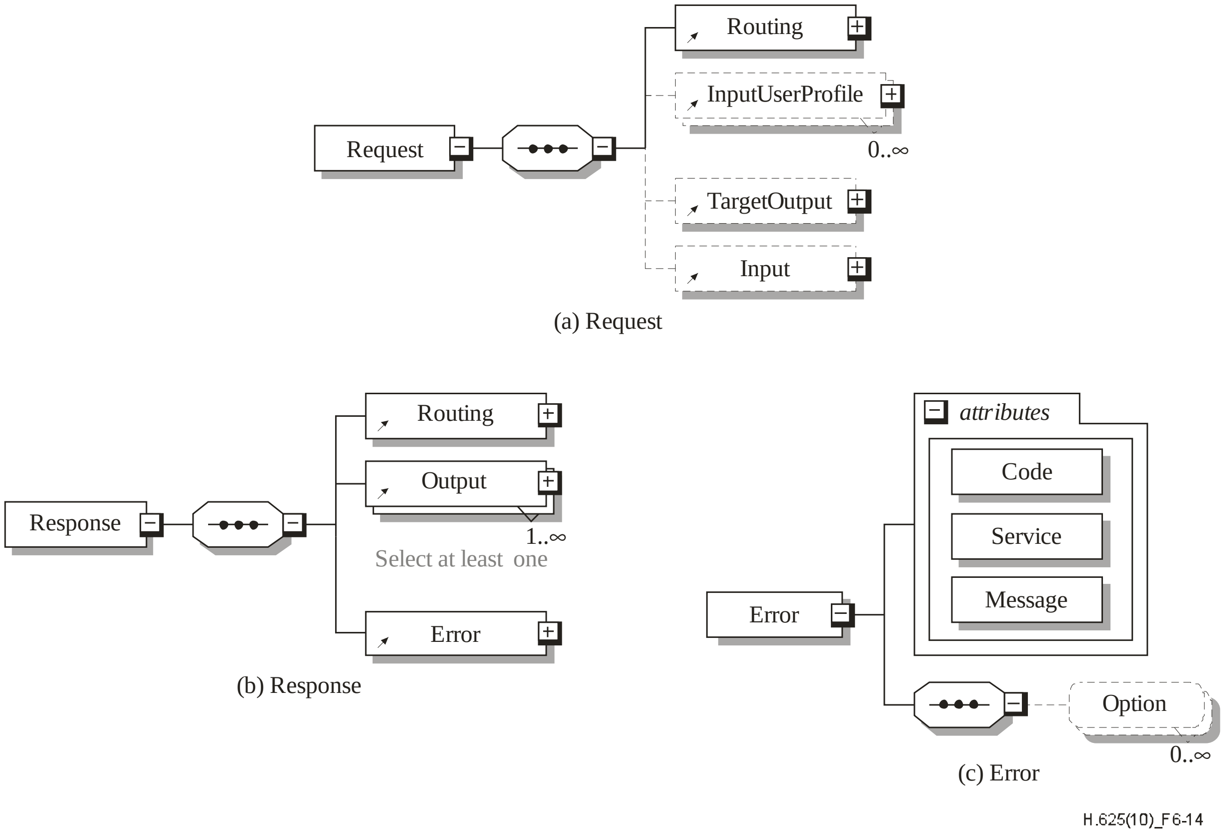

Figure 6-14 – Structure of request, response and error elements

Table 6-7 – Definition of child element of "Request", "Response" and "Error" elements

<table>
<tr><th colspan="5">Request</th></tr>
<tr><th>Annotation</th><th>Name</th><th colspan="2">Definition</th><th>Quantity</th></tr>
<tr><td rowspan="4">Children</td><td>Routing</td><td colspan="2">See (3-2)</td><td>1</td></tr>
<tr><td>InputUserProfile</td><td colspan="2">See (3-3)</td><td>0 or more</td></tr>
<tr><td>TargetOutput</td><td colspan="2">See (3-4)</td><td>0 or 1</td></tr>
<tr><td>Input</td><td colspan="2">See (3-5)</td><td>0 or 1</td></tr>
<tr><th colspan="5">Response</th></tr>
<tr><th>Annotation</th><th>Name</th><th colspan="2">Definition</th><th>Quantity</th></tr>
<tr><td rowspan="3">Children</td><td>Routing</td><td colspan="2">See (3-2)</td><td>1 or more</td></tr>
<tr><td>Output</td><td colspan="2">See (3-5)
The structure of the output is the same as the Request's input</td><td>1 or more</td></tr>
<tr><td>Error</td><td colspan="2">See below</td><td>1</td></tr>
<tr><th colspan="5">Error</th></tr>
<tr><th>Annotation</th><th>Name</th><th>Definition</th><th>Value</th><th>Quantity</th></tr>
<tr><td rowspan="3">Attributes</td><td>Code</td><td>Error type</td><td>string</td><td>1</td></tr>
<tr><td>Service</td><td>Service type</td><td>string</td><td>1</td></tr>
<tr><td>Message</td><td>Message to be prompted for error type</td><td>string</td><td>1</td></tr>
<tr><td>Children</td><td colspan="4">Option</td></tr>
</table>

(3-2) Routing

The "Routing" element is a child element of "Request" and "Response". The "Routing" element contains the routing information for relay services through multiple MC servers. The child elements indicate "From" and "To" as shown in Figure 6-15. Each of the child elements has an attribute "URI", as explained in Table 6-8. Multiple "To" elements are required to send data to multiple MC clients and servers.

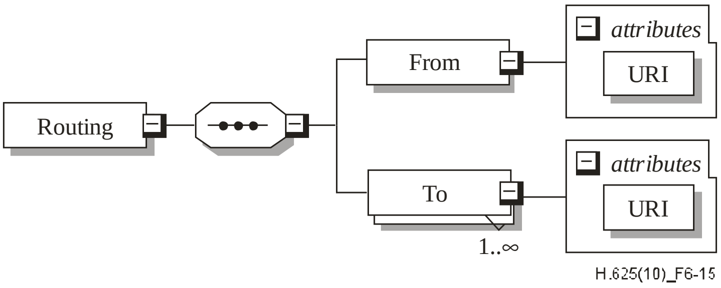

Figure 6-15 – Structure of a "Routing" element

Table 6-8 – Definition of attributes and child elements of "Routing" and "From"/"To" elements

Routing			
Annotation	**Name**	**Definition**	**Quantity**
Children	From	The location in the network of the server which sends a request	1
	To	The location in the network of the server which receives a request	1 or more

From/To				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	URI	Uniform resource identifier	string	1

(3-3) InputUserProfile

The element of "InputUserProfile" contains modality information for the user's input into the MC client. The structure of the "InputUserProfile" is as shown in Figure 6-16. The attributes are the same as those in Table 6-5 for "UserProfile". The child element is named "InputModality" and its structure is shown in Figure 6-17. The definition of children of "InputModality" is given in Table 6-9.

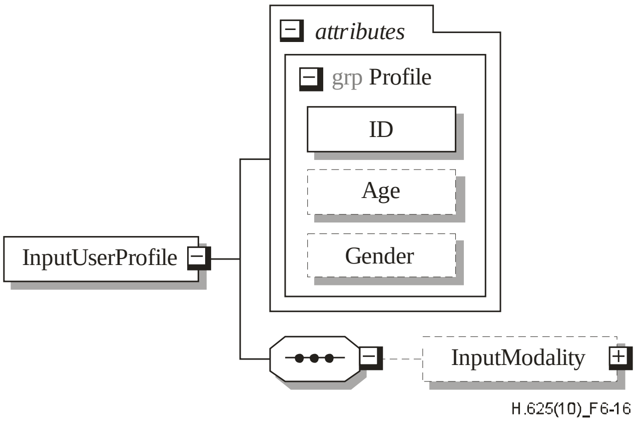

Figure 6-16 – Structure of a "InputUserProfile" element

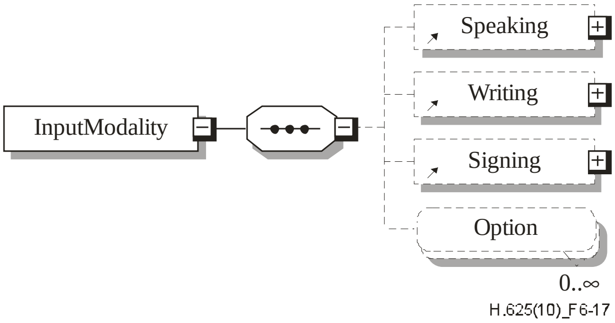

Figure 6-17 – Structure of a "InputModality" element

Table 6-9 – Definition of child elements of "InputModality" element

InputModality			
Annotation	**Name**	**Definition**	**Quantity**
Children	Speaking/ Writing	The modality for communication	0 or 1
	Signing	The modality for communication. The details of this modality is for further study	0 or 1
	Option	The additional modalities for communication can be designed under agreement between an MCP client and MCP servers	0 or more

The "InputModality" contains information for each modality. The child elements of "InputModality" are "Speaking", "Writing", "Signing" and "Option". The child element of each modality element is the "Language" element. The structure of the "Language" element associated with the "Speaking" element is shown in Figure 6-18. The definition of each element of the "Language" attribute is described in Table 6-10. The fluency of each language for respective modalities is contained by these child elements of the "InputModality" element and the information of the fluency contributes to the performance of MC engines.

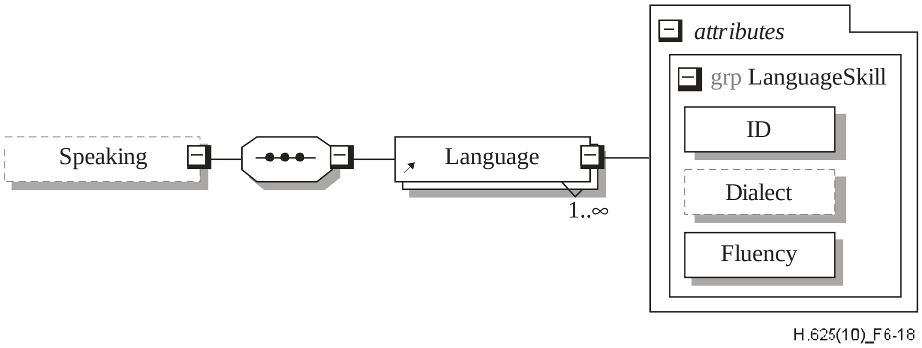

Figure 6-18 – Structure of a "Language" element

Table 6-10 – Definition of attributes of a "Language" element

Language				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	ID	The identifier for language	ISO 639 [ISO 639-1,2, 3, 5, 6]	1
	Dialect	The identifier for dialect which is not categorized by ISO 639 [ISO 639-1,2,3,5,6]	string	0 or 1
	Fluency	The fluency level for the modality is rated from "1" to "5", where "5" shows the most fluent level. Level 5 indicates fluency of native speakers.	integer	1

NOTE – The languages are identified by ISO 639 codes, which are the set of international standards that lists short codes for language names [ISO 639-1,2,3,5,6].

If the user's language is linguistically different from the standard language, the information contained in the "Dialect" element aids MC engines. To use the "Dialect" element, MC servers are required to know the meaning of the identifier in this element.

(3-4) TargetOutput

The "TargetOutput" element contains information about what type of output from the engine is demanded by the MC client as shown in Figure 6-19. The child elements are named "HypothesisFormat", "LanguageType" and "Option" as defined in Table 6-11. Additional demands to the MC servers can optionally be added.

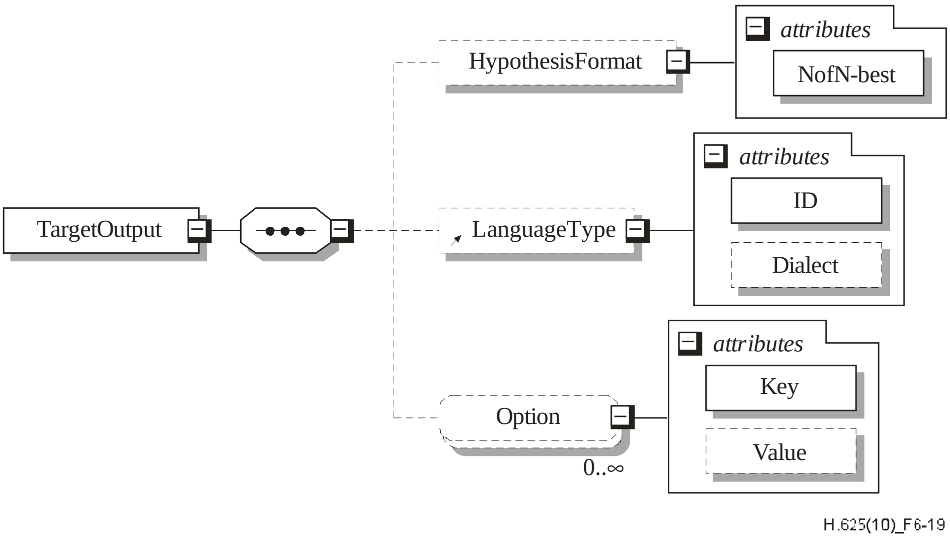

Figure 6-19 – Structure of a TargetOutput element

Table 6-11 – Definition of attribute/child elements of "TargetOutput", "HypothesisFormat" and "LanguageType" elements

TargetOutput				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Children	HypothesisFormat	See below		0 or 1
	LanguageType			0 or 1
	Option	Another demanded format can be added		0 or more
HypothesisFormat				
Attributes	NofN-best	N of the N-best hypotheses	string	1
LanguageType				
Attributes	ID	The identifier for language	ISO 639 [ISO 639-1, 2, 3, 5, 6]	1
	Dialect	The identifier for dialect which is not categorized by ISO 639 [ISO 639-1, 2, 3, 5, 6]	string	0 or 1

(3-5) Input

The children of the "Input" element are the "Data" and "AttachedBinary" elements, as illustrated in Figure 6-20. These are defined in Table 6-12. Multiple binary data are attached to MCML. To link each binary data to MI for each modality, the "AttachedBinary" element has information which links the data format described in the child element of the "Data" element with the attached binary data.

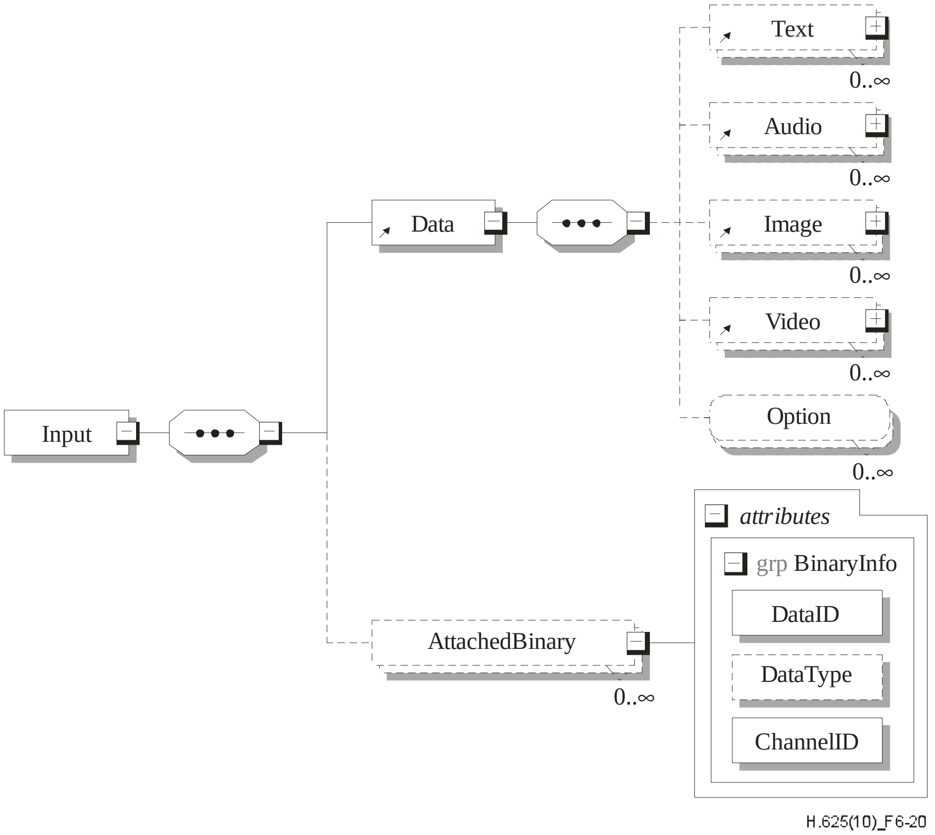

Figure 6-20 – Structure of an "Input" element

Table 6-12 – Definition of attributes and child elements of "Input", "Data" and "AttachedBinary" elements

Input				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Children	Data	See (3-5-1)		1
	AttachedBinary	Another demanded format can be added		0 or more
Data				
Children	Text/Audio/ Image/Video	See from (3-5-1) to (3-5-3)		0 or more
	Option	Additional modalities can be freely added		0 or more

Table 6-12 – Definition of attributes and child elements of "Input", "Data" and "AttachedBinary" elements

Input				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
AttachedBinary				
Attributes	DataID	The identifier for language	string	1
	DataType	One of the child elements of the "Data". Types "text", "audio", "image", "video" are reserved for MCML.	string	1
	ChannelID	Identifier for input channel	integer	1

(3-5-1) Text

The "Text" element is a child element of the "Data" element. The structure of the "Text" element is shown in Figure 6-21. To cope with multi-channel inputs, the "Text" element has an attribute named "ChannelID" whose value identifies the channel of the text input, as shown in Figure 6-21. The attributes and child elements of "Text" are defined in Table 6-13.

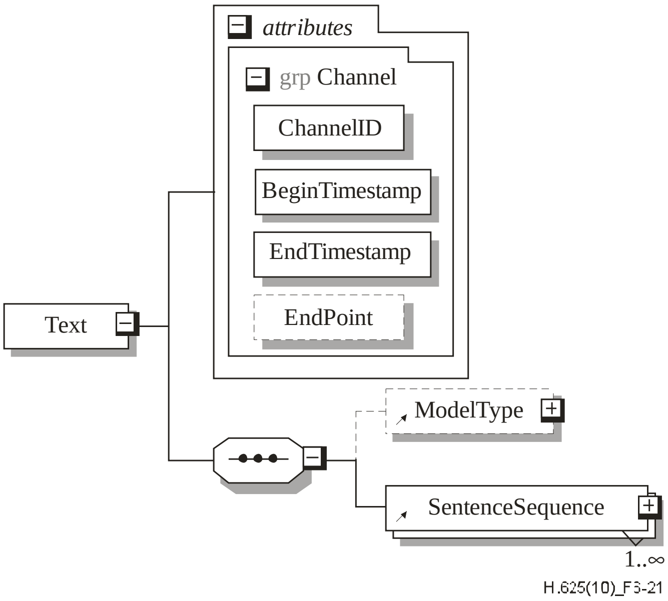

Figure 6-21 – Structure of a "Text" element

Table 6-13 – Definition of attributes and child elements of a "Text" element

Text				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	ChannelID	The identifier for channel	integer	1
	BeginTimestamp	The timestamp for the audio input (UTC)	dateTime	1
	EndTimestamp	The timestamp for the stop of audio input (UTC)	dateTime	1
	EndPoint	Flag for input of signal, such as "endpoint" indicating the end of input	dateTime	0 or 1
Children	ModelType	See (3-5-1-1)		0 or more
	SentenceSequence	See (3-5-1-2)		1 or more

(3-5-1-1) ModelType

One of the child elements of "Text" is named "ModelType", as shown in Figure 6-22 with its definition in Table 6-14. This element contains target domains in the "Domain" element and user information in the "Personality" element which contribute to enhancing the performance of MC engines.

When data is input directly to the user device without MC, the "ModelType" is not used for input. This value does not exist in such a case.

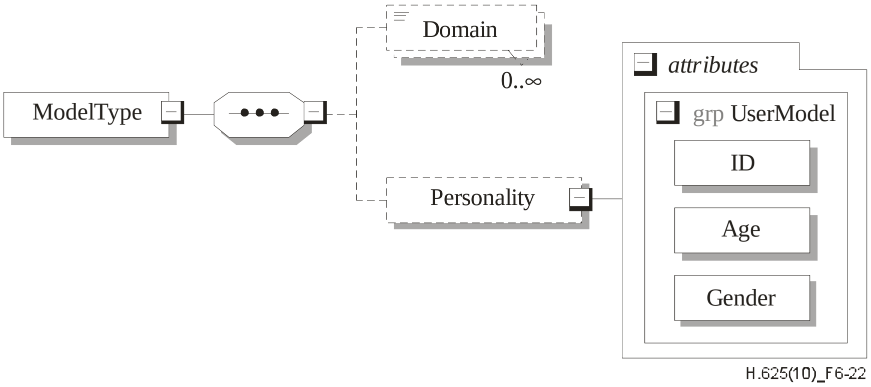

Figure 6-22 – Structure of a "ModelType" element

Table 6-14 – Definition of attributes and child elements of "ModelType" and "Personality" elements

ModelType				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Children	Domain	The domain for MI input such as "Travel" domain	string	0 or more
	Personality	See below		0 or more
Personality				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	ID	Identification name or number of the user	string	1
	Age	User's age	integer	1
	Gender	User's gender	"female"/ "male"	1

(3-5-1-2) SentenceSequence

The "Text" element is required to handle not only plain text but also ASR results. To cope with multiple ASR hypotheses generated from multiple utterances, the child element named "SentenceSequence" has attributes which represent information for ASR results and a child element named "Sentence", as shown in Figure 6-23. The attributes and children are listed in Table 6-15.

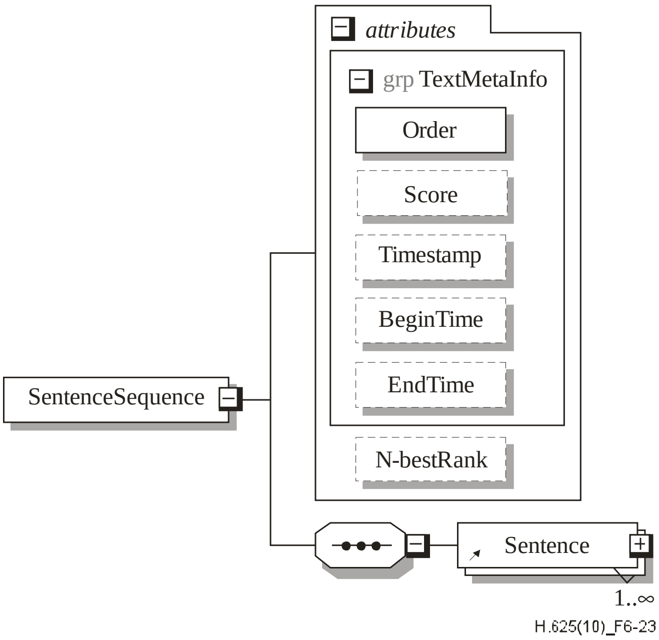

Figure 6-23 – Structure of a "SentenceSequence" element

Table 6-15 – Definition of attributes and child elements of a "SentenceSequence" element

SentenceSequence				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	Order	The order of MI input	integer	1
	Score	Score given by the MC engine such as a confidence score of ASR output	float	0 or 1
	TimeStamp	The start time of text input or that of ASR output given by user device (UTC)	dateTime	1
	BeginTime	The start time of text input or ASR output that is measured from the start of the sentence sequence. For Sentence Sequence, this is set to 0 [msec]	integer	0 or 1
	EndTime	The stop time of text input or ASR output relative to the BeginTime [msec]	integer	0 or 1
	N-bestRank	The rank in the ASR N-best hypotheses	integer	0 or 1
Children	Sentence	See (3-5-1-3)		1 or more

Although ASR output does not always start from a sentence beginning and ending at a sentence ending, the beginning and the ending of speech input are dealt with as the beginning and the ending of a sentence. MCP servers that use MC results obtained by ASR servers are required to detect the real boundary of sentences.

(3-5-1-3) Sentence

The child element of the "SentenceSequence" is named "Sentence", which contains information of each sentence. The structure of the "Sentence" element is shown in Figure 6-24. The "Sentence" element has three child elements named "Function", "Surface" and "Chunk". The attributes of the "Sentence" element and its child elements are listed in Table 6-16. The "Surface" element contains a sequence of chunk surfaces in a sentence, as defined in Table 6-16.

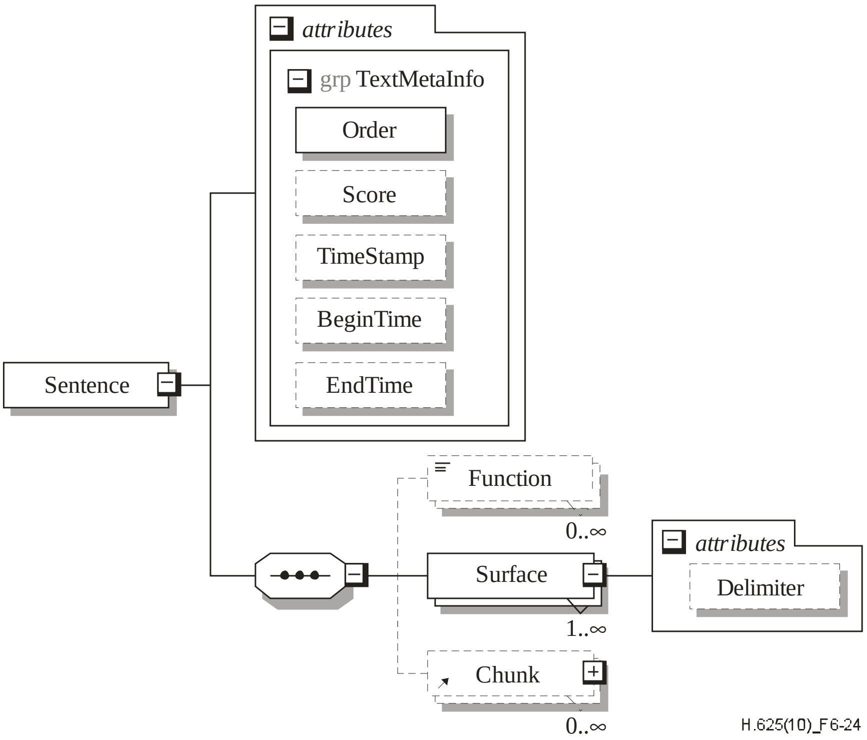

Figure 6-24 – Structure of a "Sentence" element

Table 6-16 – Definition of attributes and child elements of a "Sentence" element

Sentence				
Annotation	**Name**	**Definition**		**Quantity**
Attributes	Order/Score/ TimeStamp/ BeginTime/ EndTime	The same attributes as those of the "SentenceSequence". The "N-bestRank" is not considered for the "Sentence" element. See (3-5-1-2)		
Children	Function	Class for sentence such as dialog act (No attributes and children)		0 or more
	Surface	See below		1 or more
	Chunk	See (3-5-1-4)		0 or more
Surface				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	Delimiter	Symbols indicating chunk boundary	string	0 or 1

(3-5-1-4) Chunk

The structure of the "Chunk" element is shown in Figure 6-25. The definition of attributes and child elements is given below in Table 6-17.

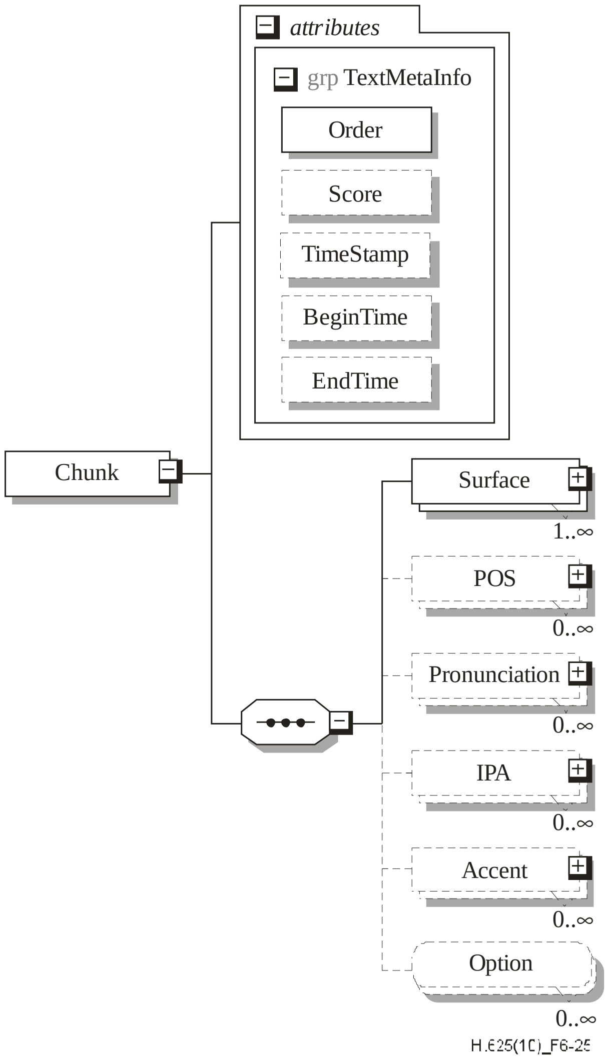

Figure 6-25 – Structure of "Chunk" element

Table 6-17 – Definition of attributes and child elements of "Chunk" element

Chunk				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	Order/Score/ TimeStamp/ BeginTime/ EndTime	The same attributes as those of the "SentenceSequence". The "N-bestRank" is not considered for the "Sentence" element. See (3-5-1-3)		
Children	Surface	Surface expression of Chunk	string	1 or more
	POS	Part-of-Speech or its complementary information for Chunk	string	0 or more
	Pronunciation	Pronunciation of Chunk	string	0 or more
	IPA	International Phonetic Alphabet for Chunk	string	0 or more
	Accent	Accent for Chunk	string	0 or more
	Additional information which depends on language can be added when agreements between clients and servers exist			

(3-5-1-5) Surface, POS, Pronunciation, IPA and Accent

The child elements of the "Chunk" element are named "Surface", "POS", "Pronunciation", "IPA" and "Accent", and all of them have the same attributes. As an example, the attributes of "Surface" are shown in Figure 6-26. The attributes of the child elements are listed in Table 6-18.

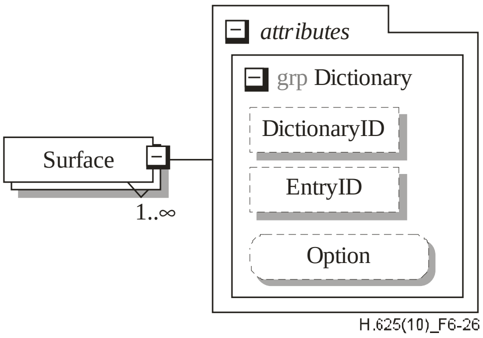

Figure 6-26 – Structure of a "Surface" element

Table 6-18 – Definition of attributes of "Surface", "POS", "Pronunciation" and "Accent" elements

Surface/POS/Pronunciation/Accent				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	DictionaryID	Identifier for chunk dictionary	string	0 or 1
	EntryID	Identifier for entry in a dictionary	string	0 or 1
	Optional language information can be added when agreements between clients and server exist			

Generally, the "Surface" of "Chunk" contains words, while some TTS engines need to use phrases as chunks. This element can be freely designed corresponding to the MCP servers' demands. The tuples of phonemes, sub-words, or words are supposed to be contained by the "Surface" of "Chunk" element.

(3-5-2) Audio

The "Audio" element is a child element of the "Data" element. The "Audio" element has three child elements, "ModelType", "Signal" and "Feature", as shown in Figure 6-27. The "Feature" element is optional.

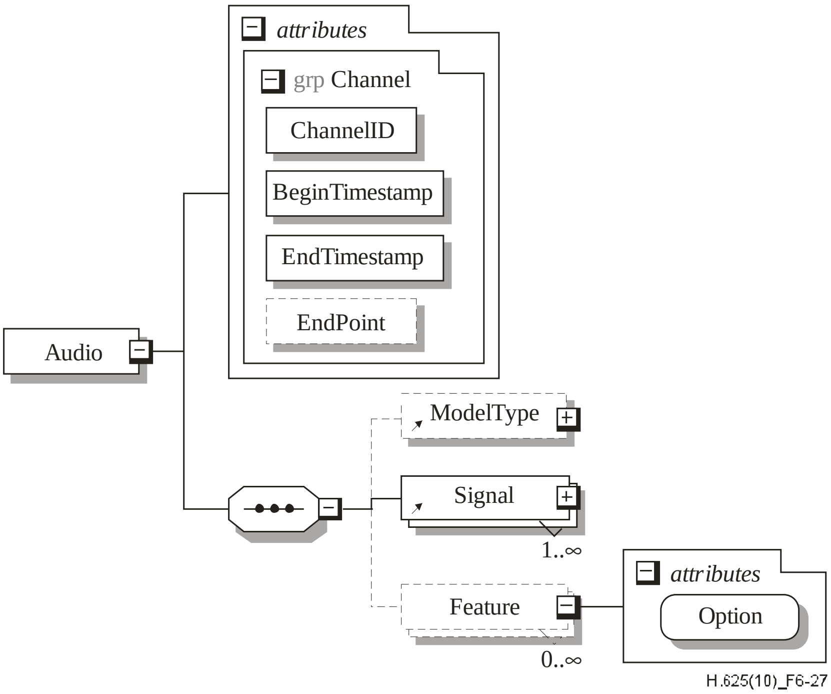

Figure 6-27 – Structure of an "Audio" element

To cope with multichannel inputs, the "Audio" element has an attribute named "ChannelID", whose value identifies the channel of audio input. The definition of the "Audio" element is described in Table 6-19. The definition of "ModelType" is described in (3-5-1-1). Timestamps are given by user devices.

Table 6-19 – Definition of attributes and child elements of an "Audio" element

Audio			
Annotation	**Name**	**Definition**	**Quantity**
Attributes	ChannelID/ BeginTimestamp/ EndTimestamp/ EndPoint	These attributes are the same as those of the "Text" element, as shown in Table 6-13.	
Children	ModelType	See (3-5-1-1)	0 or 1
	Signal	See (3-5-2-1)	1
	Feature	Optional model feature	0 or 1

(3-5-2-1) Signal

The attributes of the "Signal" element are shown in Figure 6-28 and listed in Table 6-20.

Element "Feature" is not defined in this Recommendation. Any features can be used for this communication when agreements on feature expressions exist between clients and servers.

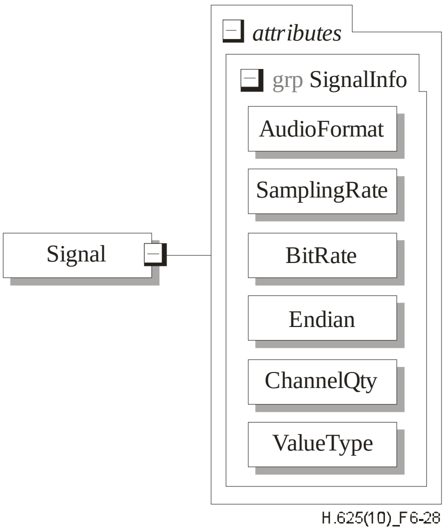

Figure 6-28 – Structure of a "Signal" element

Table 6-20 – Definition of attributes of a "Signal" element

Signal				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	AudioFormat	The format of digitalized speech by Input Method, "raw PCM" and "ADPCM" are reserved for MCML. Use of PCM formatted audio data is required for MCML, while use of ADPCM is optional.	string	1
	SamplingRate	Sampling rate [Hz]	integer	1
	BitRate	Bit rate	integer	1
	Endian	Endian type	"little"/ "big"	1
	ChannelQty	Number of channels	integer	1
	ValueType	Float or integer	"integer"/ "float"	1

(3-5-3) Image and video

The "Image" and "Video" elements have the same attributes. The structure of the "Image" element is shown in Figure 6-29. The definitions of the attributes are listed in Table 6-21. To cope with multichannel inputs, the "Image" and "Video" elements have an attribute named "ChannelID", whose value identifies the input channel.

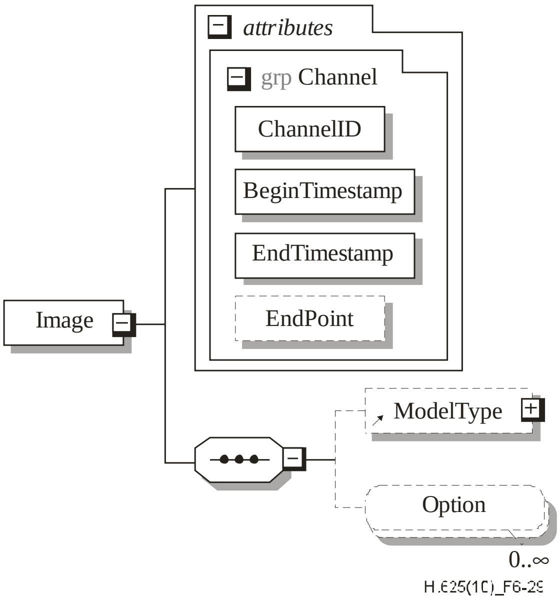

Figure 6-29 – Structure of an "Image" element

Table 6-21 – Definition of attributes and child elements of "Image" and "Video" elements

Image/Video				
Annotation	**Name**	**Definition**	**Value**	**Quantity**
Attributes	ChannelID/ BeginTimestamp/ EndTimestamp/ EndPoint	These attributes are the same ones that the "Text" element has in Table 6-13		
Children	ModelType	See Figure 6-22 and Table 6-14		
	Option	Optional signals and features can be added		

(4) History

The MCP client requests modality conversion (MC) by using the "Request" element. In response to this MCP client request, the MCML server responds using the "Response" element to provide MC results. In order to record all processes through multiple combinations of MCP servers, all the information embedded into both the "Request" and "Response" elements are recorded in element "History". The structure of the "History" element is shown in Figure 6-30. The structure of the "Request" and "Response" elements is described in (3-1).

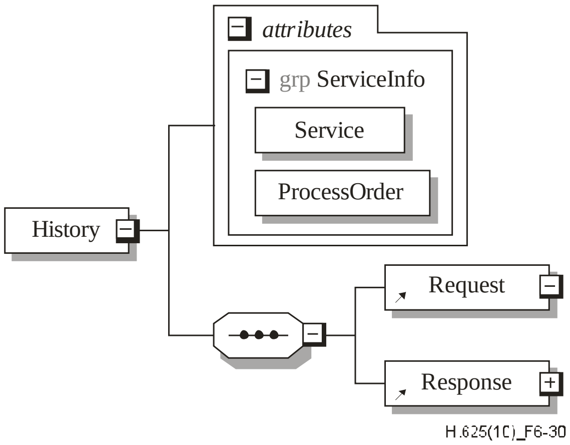

Figure 6-30 – Structure of a "History" element

6.4 General workflow for modality conversion

It is recommended that the MC architecture in Figure 6-3 be controlled according to the data flow and workflow in Figures 6-31 and 6-32.

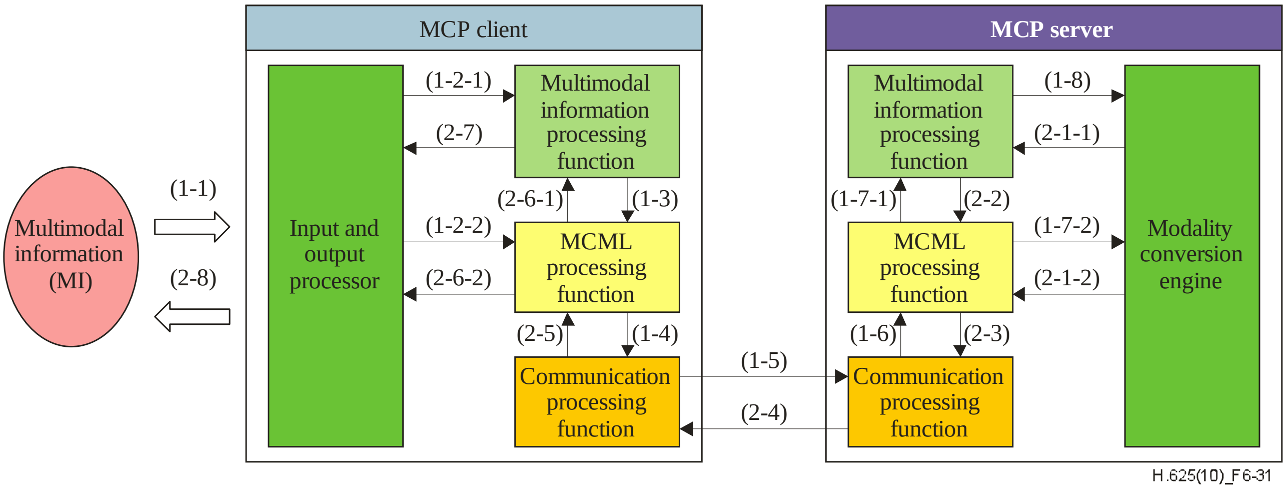
MCP client
MCP server
Multimodal information (MI)
(1-1)
(2-8)
Input and output processor
(1-2-1)
(2-7)
Multimodal information processing function
(2-6-1)
(1-3)
(1-2-2)
(2-6-2)
MCML processing function
(2-5)
(1-4)
Communication processing function
(1-5)
(2-4)
Multimodal information processing function
(1-8)
(2-1-1)
(1-7-1)
(2-2)
MCML processing function
(1-7-2)
(2-1-2)
(1-6)
(2-3)
Communication processing function
Modality conversion engine
H.625(10)_F6-31

Figure 6-31 – Data flow for modality conversion

The general data flow and workflow are described with their associated steps in the following list. In this description, a module that has a particular function is denoted as "function" processor.

(1-1) Multimodal information (MI) is input into the MCP client.

(1-2-1) MI is digitized by the input and output processor and sent to the MI processor or,

(1-2-2) MI is digitized by the input and output processor and directly sent to the MCML processor.

(1-3) Digitized MI is encoded by the MI processor and the encoded MI is sent to the MCML processor.

(1-4) Digitized or encoded MI is embedded in MCML format by the MCML processor and sent to the communication processor.

(1-5) MCML including digitized or encoded MI is sent to the MCP server using MCP via the communication processor.

(1-6) Digitized or encoded MI in MCML format is received by the communication processor and sent to the MCML processor.

(1-7-1) Digitized MI extracted from MCML by the MCML processor is sent to the MI processor.

(1-7-2) Encoded MI extracted from MCML by the MCML processor is directly sent to the MC engine.

(1-8) Digitized MI is encoded by the MI processor and the encoded MI is sent to the MC engine.

(2-1-1) Digitized or encoded MI is converted in modality by the MC engine and the MC result is sent to the MI processor.

(2-1-2) The MC result is directly sent to the MCML processor.

(2-2) The MC result is encoded by the MI processor and the encoded MC result is sent to the MCML processor.

(2-3) The encoded MC result or the MC result is embedded in MCML format by the MCML processor and sent to the communication processor.

(2-4) MCML is sent to the MCP client using MCP via the communication processor.

(2-5) MCML including the MC result and/or the encoded MC result is received by the communication processor and sent to the MCML processor.

(2-6-1) The MC result is extracted by the MCML processor and sent to the MI processor.

(2-6-2) The MC result is sent to the input and output processor.

(2-7) The MC result decoded by the MI processor is sent to the input and output processor.

(2-8) The MC result is output by the input and output processor of the MCP client.

MC is accomplished either by a single MCP server, or by a combination of multiple MCP servers. The order of processing by MCP servers is required to be assigned by the MCP client before processing. Even if an undefined MC server may be connected to S2ST servers, communication can be achieved as far as the servers support MCP.

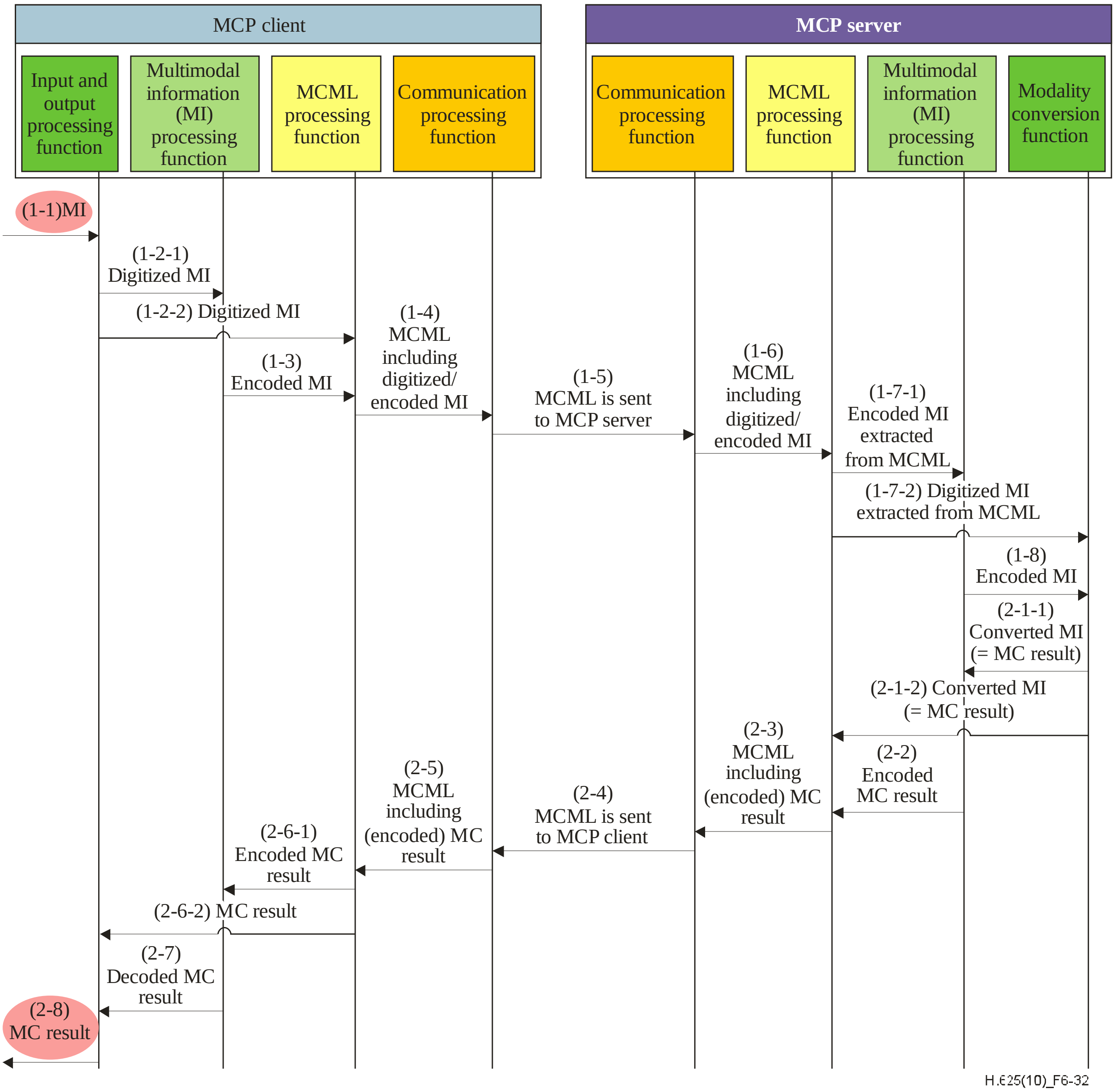
MCP client
Input and output processing function
Multimodal information (MI) processing function
MCML processing function
Communication processing function
MCP server
Communication processing function
MCML processing function
Multimodal information (MI) processing function
Modality conversion function
(1-1)MI
(1-2-1) Digitized MI
(1-2-2) Digitized MI
(1-3) Encoded MI
(1-4) MCML including digitized/ encoded MI
(1-5) MCML is sent to MCP server
(1-6) MCML including digitized/ encoded MI
(1-7-1) Encoded MI extracted from MCML
(1-7-2) Digitized MI extracted from MCML
(1-8) Encoded MI
(2-1-1) Converted MI (= MC result)
(2-1-2) Converted MI (= MC result)
(2-2) Encoded MC result
(2-3) MCML including (encoded) MC result
(2-4) MCML is sent to MCP client
(2-5) MCML including (encoded) MC result
(2-6-1) Encoded MC result
(2-6-2) MC result
(2-7) Decoded MC result
(2-8) MC result
H.625(10)_F6-32

Figure 6-32 – General workflow for modality conversion

Annex A

MCML 1.0 schema

(This annex forms an integral part of this Recommendation.)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:mcml="http://www.itu.int/xml-namespace/itu-t/h.625/MCML/"
   targetNamespace="http://www.itu.int/xml-namespace/itu-t/h.625/MCML/"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified">
   <xs:element name="MCML">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="User"/>
            <xs:element ref="Server" maxOccurs="unbounded"/>
            <xs:element ref="History" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="Version" type="xs:string" use="required"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="User">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Transmitter"/>
            <xs:element ref="Receiver" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="Transmitter">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Device"/>
            <xs:element ref="UserProfile" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="Device">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Location">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="URI"/>
                     <xs:element name="GlobalPosition" minOccurs="0">
                        <xs:complexType>
                           <xs:attributeGroup ref="Location"/>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:attributeGroup name="Location">
      <xs:attribute name="Longitude" type="xs:float" use="required"/>
      <xs:attribute name="Latitude" type="xs:float" use="required"/>
   </xs:attributeGroup>
   <xs:element name="UserProfile">
      <xs:complexType>
         <xs:attributeGroup ref="Profile"/>
      </xs:complexType>
   </xs:element>
   <xs:attributeGroup name="Profile">
      <xs:attribute name="ID" type="xs:string" use="required"/>
      <xs:attribute name="Age" type="xs:int" use="optional"/>
      <xs:attribute name="Gender" type="xs:string" use="optional" fixed='"female",
"male"'/>
   </xs:attributeGroup>
   <xs:element name="Receiver">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Device"/>
            <xs:element ref="UserProfile" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="Server">
      <xs:complexType>
         <xs:choice>
            <xs:element ref="Request" maxOccurs="unbounded">
               <xs:annotation>
                  <xs:documentation> </xs:documentation>
               </xs:annotation>
            </xs:element>
            <xs:element ref="Response">
               <xs:annotation>
                  <xs:documentation> </xs:documentation>
               </xs:annotation>
            </xs:element>
         </xs:choice>
         <xs:attributeGroup ref="ServiceInfo"/>
      </xs:complexType>
   </xs:element>
   <xs:attributeGroup name="ServiceInfo">
      <xs:attribute name="Service" type="xs:string" use="required"/>
      <xs:attribute name="ProcessOrder" type="xs:int" use="required"/>
   </xs:attributeGroup>
   <xs:element name="Request">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Routing"/>
            <xs:element ref="InputUserProfile" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="TargetOutput" minOccurs="0"/>
            <xs:element ref="Input" minOccurs="0"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="Response">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Routing"/>
            <xs:element ref="Output" maxOccurs="unbounded"/>
            <xs:element ref="Error"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="Error">
      <xs:complexType>
         <xs:sequence>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="Code" type="xs:string" use="required"/>
         <xs:attribute name="Service" type="xs:string" use="required"/>
         <xs:attribute name="Message" type="xs:string" use="required"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="Routing">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="From">
               <xs:complexType>
                  <xs:attribute name="URI" type="xs:string" use="required"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="To" maxOccurs="unbounded">
```

```
               <xs:complexType>
                  <xs:attribute name="URI" type="xs:string" use="required"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="InputUserProfile">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="InputModality" minOccurs="0">
               <xs:annotation>
                  <xs:documentation> </xs:documentation>
               </xs:annotation>
            </xs:element>
         </xs:sequence>
         <xs:attributeGroup ref="Profile"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="InputModality">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Speaking" minOccurs="0">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="Language"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="Writing" minOccurs="0">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="Language"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="Signing" minOccurs="0">
               <xs:complexType>
                  <xs:sequence>
                     <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="Language">
      <xs:complexType>
         <xs:attributeGroup ref="LanguageSkill"/>
      </xs:complexType>
   </xs:element>
   <xs:attributeGroup name="LanguageSkill">
      <xs:attribute name="ID" type="xs:string" use="required"/>
      <xs:attribute name="Dialect" type="xs:string" use="optional"/>
      <xs:attribute name="Fluency" type="xs:int" use="required"/>
   </xs:attributeGroup>
   <xs:element name="TargetOutput">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="HypothesisFormat" minOccurs="0">
               <xs:complexType>
                  <xs:attribute name="NofN-best" use="required">
                     <xs:annotation>
                        <xs:documentation> </xs:documentation>
                     </xs:annotation>
                  </xs:attribute>
               </xs:complexType>
            </xs:element>
            <xs:element ref="LanguageType" minOccurs="0"/>
            <xs:element name="Option" minOccurs="0" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:attribute name="Key" use="required"/>
                  <xs:attribute name="Value" use="required"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="LanguageType">
      <xs:complexType>
         <xs:attribute name="ID" type="xs:string" use="required"/>
         <xs:attribute name="Dialect" type="xs:string" use="optional"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="Input">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Data">
               <xs:annotation>
                  <xs:documentation> </xs:documentation>
               </xs:annotation>
            </xs:element>
            <xs:element name="AttachedBinary" minOccurs="0" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:attributeGroup ref="BinaryInfo"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:attributeGroup name="BinaryInfo">
      <xs:attribute name="DataID" use="required"/>
      <xs:attribute name="DataType" type="xs:string" use="optional"/>
      <xs:attribute name="ChannelID" type="xs:int" use="required"/>
   </xs:attributeGroup>
   <xs:element name="Data">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Text" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="Audio" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="Image" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="Video" minOccurs="0" maxOccurs="unbounded"/>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="Text">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="ModelType" minOccurs="0"/>
            <xs:element ref="SentenceSequence" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attributeGroup ref="Channel"/>
      </xs:complexType>
   </xs:element>
   <xs:attributeGroup name="Channel">
      <xs:attribute name="ChannelID" type="xs:int" use="required"/>
      <xs:attribute name="BeginTimestamp" type="xs:dateTime" use="required"/>
      <xs:attribute name="EndTimestamp" type="xs:dateTime" use="required"/>
      <xs:attribute name="EndPoint" type="xs:dateTime" use="optional"/>
   </xs:attributeGroup>
   <xs:element name="ModelType">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Domain" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Personality" minOccurs="0">
               <xs:complexType>
                  <xs:attributeGroup ref="UserModel"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:attributeGroup name="UserModel">
      <xs:attribute name="ID" type="xs:string" use="required"/>
      <xs:attribute name="Age" type="xs:int" use="required"/>
      <xs:attribute name="Gender" type="xs:string" use="required" fixed='"female",
"male"'/>
   </xs:attributeGroup>
   <xs:element name="SentenceSequence">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Sentence" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attributeGroup ref="TextMetaInfo"/>
         <xs:attribute name="N-bestRank" type="xs:int"/>
      </xs:complexType>
   </xs:element>
   <xs:attributeGroup name="TextMetaInfo">
      <xs:attribute name="Order" type="xs:int" use="required"/>
      <xs:attribute name="Score" type="xs:float" use="optional"/>
      <xs:attribute name="TimeStamp" type="xs:dateTime" use="optional"/>
      <xs:attribute name="BeginTime" type="xs:integer" use="optional"/>
      <xs:attribute name="EndTime" type="xs:integer" use="optional"/>
   </xs:attributeGroup>
   <xs:element name="Sentence">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Function" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Surface" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:attribute name="Delimiter" type="xs:string"/>
               </xs:complexType>
            </xs:element>
            <xs:element ref="Chunk" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attributeGroup ref="TextMetaInfo"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="Chunk">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Surface" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:attributeGroup ref="Dictionary"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="POS" minOccurs="0" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:attributeGroup ref="Dictionary"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="Pronunciation" minOccurs="0" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:attributeGroup ref="Dictionary"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="IPA" minOccurs="0" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:attributeGroup ref="Dictionary"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="Accent" minOccurs="0" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:attributeGroup ref="Dictionary"/>
               </xs:complexType>
            </xs:element>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attributeGroup ref="TextMetaInfo"/>
      </xs:complexType>
   </xs:element>
   <xs:attributeGroup name="Dictionary">
      <xs:attribute name="DictionaryID" type="xs:string"/>
      <xs:attribute name="EntryID" type="xs:string"/>
      <xs:anyAttribute namespace="##other"/>
   </xs:attributeGroup>
   <xs:element name="Audio">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="ModelType" minOccurs="0"/>
            <xs:element ref="Signal" maxOccurs="unbounded"/>
            <xs:element name="Feature" minOccurs="0" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:anyAttribute/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         <xs:attributeGroup ref="Channel"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="Signal">
      <xs:complexType>
         <xs:attributeGroup ref="SignalInfo"/>
      </xs:complexType>
   </xs:element>
   <xs:attributeGroup name="SignalInfo">
      <xs:attribute name="AudioFormat" type="xs:string" use="required" fixed='"PCM",
"ADPCM"'/>
      <xs:attribute name="SamplingRate" type="xs:int" use="required"/>
      <xs:attribute name="BitRate" type="xs:int" use="required"/>
      <xs:attribute name="Endian" type="xs:string" use="required" fixed='"big",
"little"'/>
      <xs:attribute name="ChannelQty" type="xs:int" use="required"/>
      <xs:attribute name="ValueType" type="xs:string" use="required" fixed='"float",
"integer"'/>
   </xs:attributeGroup>
   <xs:element name="Image">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="ModelType" minOccurs="0"/>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attributeGroup ref="Channel"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="Video">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="ModelType" minOccurs="0"/>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attributeGroup ref="Channel"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="Output">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Data"/>
            <xs:element name="AttachedBinary" minOccurs="0" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:attributeGroup ref="BinaryInfo"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="History">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Request"/>
            <xs:element ref="Response"/>
         </xs:sequence>
```

```
        <xs:attributeGroup ref="ServiceInfo"/>
      </xs:complexType>
    </xs:element>
</xs:schema>
```

Appendix I

Examples of physical level architecture
Workflow examples of modality conversion using MCP

(This appendix does not form an integral part of this Recommendation.)

This appendix presents three typical examples of modality conversion using MCP defined in [ITU-T F.745]. The application scenarios are as follows.

I.1 Shared S2ST client for two-party communication

S2ST provides two-party communication using a shared application terminal. In this service, speech-to-speech translation can be performed for two-party communication. Although it is also applicable to multi-party communication, processing of multiple translations is not performed in a simultaneous manner, but in sequential steps. Figure I.1 shows the architectural configuration for this example.

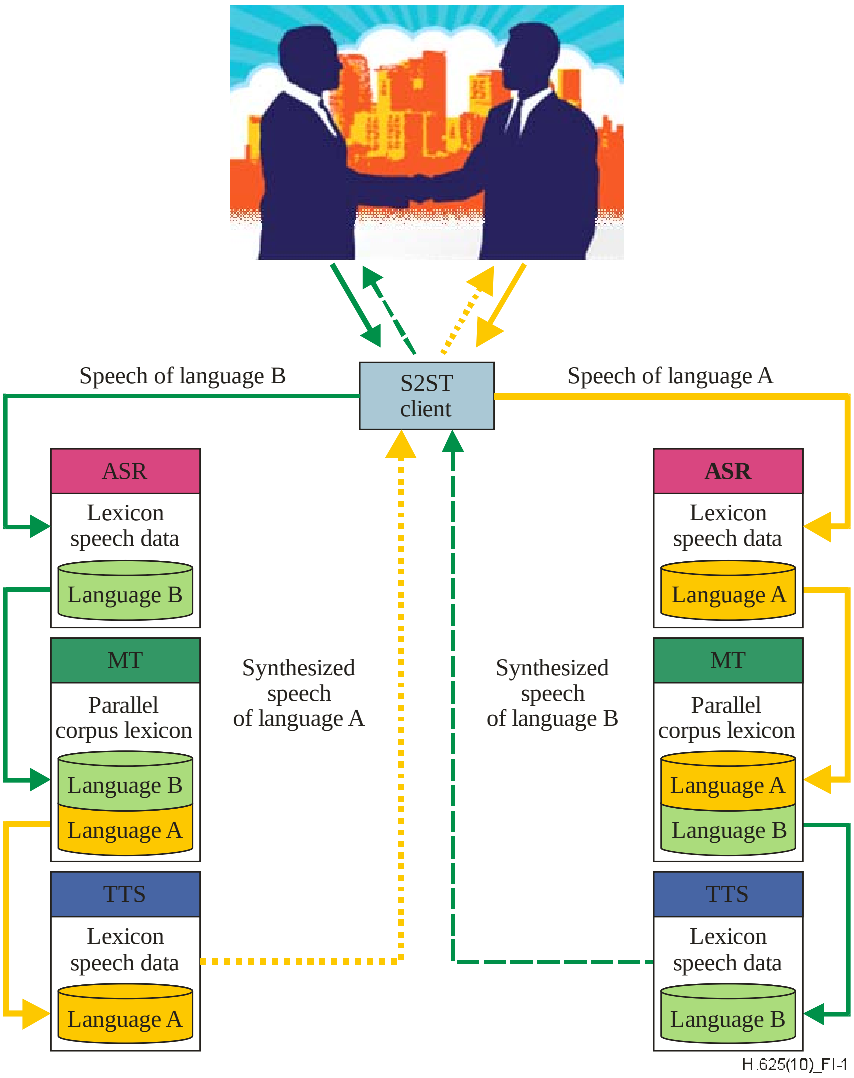

Figure I.1 – Shared S2ST client for two-party communication

I.2 Personal S2ST client communication

S2ST provides users with both two-party and multi-party communications using personal application terminals. This service can also be performed remotely if the application is used in communication between multiple users, where the speech translation results of one user can be distributed to multiple users in multiple languages at the same time. Figure I.2 shows the architectural configuration for this example.

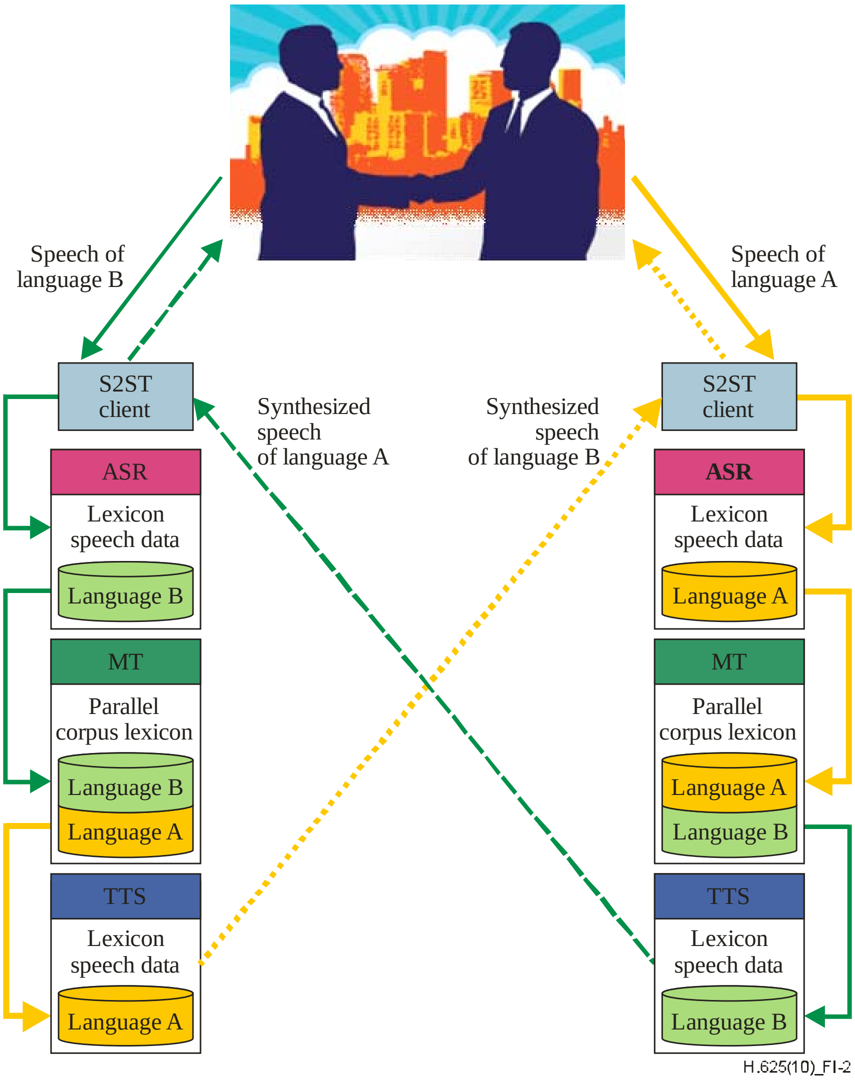

Figure I.2 – Personal S2ST client communication

I.3 Cross-modality communication

S2ST provides different-modality communication between users. For instance, a person who gives spoken utterances can communicate with a counterpart who is using sign language through the S2ST system. Text-input users could also converse with someone who uses sign language. Figure I.3 shows an example between a user who is speaking on a cellular phone and a user who is typing text on a personal computer.

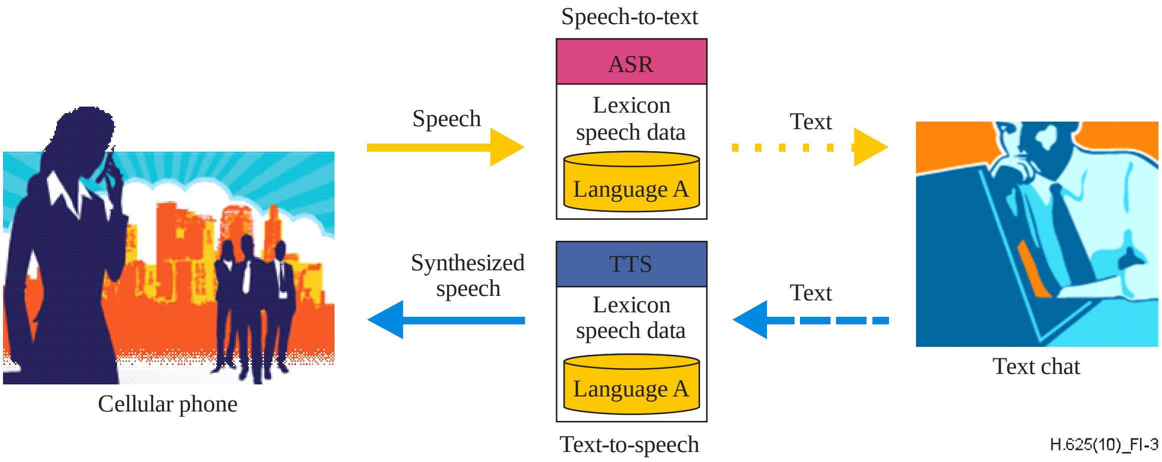

Figure I.3 – Cross-modality communication

Bibliography

[b-ITU-T G.701]	Recommendation ITU-T G.701 (1993), *Vocabulary of digital transmission and multiplexing, and pulse code modulation (PCM) terms*.
[b-ITU-T J.200]	Recommendation ITU-T J.200 (2001), *Worldwide common core – Application environment for digital interactive television services*.
[b-ITU-T P.10]	Recommendation ITU-T P.10 (1998), *Vocabulary of terms on telephone transmission quality and telephone sets.*

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	**Audiovisual and multimedia systems**
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Terminals and subjective and objective assessment methods
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2011