Master in Informatics Engineering
Internship
Intermediary Report

Visual Interactive Voice Response (Visual IVR)

Marcos André Ferreira Calvo

mcalvo@student.dei.uc.pt

DEI Supervisor:

Prof. Dr. Carlos Fonseca

Wit-Software Supervisor:

Eng. João Alves

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Abstract

Since the late 1990's, companies started to introduce interactive voice response (IVR) systems in their call centers. By querying the customer and receiving the option chosen (number pressed), IVR solutions allows these companies to have universal routing of the calling party (customer) to the service that best suits their needs without requiring a live agent, reducing costs and improving customer experience.

Nowadays, with the evolution of mobile technologies, most of the callers use some kind of handheld device such as smartphone or a softphone to call these services which are becoming increasingly more capable of receiving media content such as video or image prior, during or after the call is established. This is where the Visual IVR concept appears as an improved interface for the traditional IVR system by adding rich-media to the user experience, increasing user engagement and proficiency.

With the added capability of displaying information and interactive menus to the user, not only can Visual IVR service provider benefit, but also call experiences can be enhanced and improved.

This internship focuses on the development of a service to share visual enriched content to VoIP capable devices. The main goal is not only to create a way of sharing this content in the Visual IVR context but also in the context of a normal call between two capable devices.

Keywords

"Android", "Application Server", "IP Multimedia Subsystem", "IMS", "Session Initiation Protocol", "SIP", "Interactive Voice Response", "IVR", "MSRP", "Visual IVR", "VoIP", "RCS", "Rich Communication Service".

Table of Contents

List of Tables

List of Figures

Glossary

3G	Third Generation
3GPP	3rd Generation Partnership Project
ACID	Atomicity, Consistency, Isolation, Durability
ACK	Acknowledgment
API	Application Programming Interface
AS	Application Server
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband Integrated Services Digital Network
B2BUA	Back-to-Back User Agent
CCITT	Consultative Committee for International Telephony and Telegraphy
CD	Compact Disk
CDMA	Code Division Multiple Access
CoMP	Coordinated Multi-Point Operation
COMLib	WIT Communication Library
CS	Circuit Switched
CSCF	Call Session Control Function
CSS	Cascading Style Sheets
DB	Data Base
DTMF	Dual Tone Multi Frequency
GPRS	General Packet Radio Service
GSM	Global System for Mobile
GSMA	GSM Association
GW	Gateway
HDTV	High-definition Television
HSPA+	High Speed Packet Access
HSS	Home Subscriber Server
HSUPA	High-Speed Uplink Packet Access
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol

I-CSCF	Interrogating Call Session Control Function
IDC	In-device Co-existence
IDR	Interactive Display Response
IETF	Internet Engineering Task Force
IFC	Initial Filter Criteria
IMR	Interactive Messaging Response
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IPTV	IP Television
ISDN	Integrated Services Digital Network
IVR	Interactive Voice Response
IVVR	Interactive Video and Voice Response
J2EE	Java 2 Platform, Enterprise Edition
JNI	Java Native Interface
LTE	Long-Term Evolution
MBMS	Multimedia Broadcast Multicast Service
MIMO	Multi-Input and Multi-Output
MSRP	Message Session Relay Protocol
NFC	Near Field Communication
OFDMA	Orthogonal Frequency-Division Multiple Access
OSA-GW	Open Service Architecture Gateway
OTT	Overt the Top
OTT-TV	Over the Top Television
P-CSCF	Proxy Call Session Control Function
P2P	Peer-to-Peer
PS	Packet Switched
PSTN	Public Switched Telephone Network
PSTN-TDM	Public Switched Telephone Network Time Division Multiplexing
QoS	Quality of Service
RA	Resource Adapter
RCS	Rich Communication Service

RFC	Request for Comments
RTCP	Real-time Transport Protocol Control Protocol
RTP	Real-Time Transport Protocol
S-CSCF	Serving Call Session Control Function
SBB	Service Building Block
SC-FDMA	Single-carrier frequency-division multiple access
SDP	Session Description Protocol
SER	SIP Express Router
SIP	Session Initiation Protocol
SLEE	Service Logic Execution Environment
SS7	Signaling System no. 7
SSF	Service Switching Function
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UE	User Endpoint
UMTS	Universal Mobile Telecommunications System
VCE	Visual Call Enrichment
VoIP	Voice over Internet Protocol
VXML	Voice Extensible Markup Language
WLAN	Wireless Local Area Network
XML	Extensible Markup Language
XSD	XML Schema Definition
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation

Chapter 1 Introduction

The intent of this document is to summarize the work done by the intern during the first semester of 2014-2015 academic year. The internship is a discipline of the Master's degree in Informatics Engineering course lectured in the Department of Informatics Engineering of the Faculty of Science and Technology of the University of Coimbra (DEI-FCTUC). The internship has a two-semester duration, and is part of the final year of the degree.

An internal supervisor from DEI-FCTUC, Prof. Dr. Carlos Fonseca, and an external supervisor from WIT-Software, Eng. João Alves, supervised the internship.

This chapter provides an introduction to the project theme, some of the historical context regarding the telephony industry and technologies as well as an explanation of the work environment, people involved and planning of both the first and the second semester.

1.1 Work Environment

The internship took place at WIT-Software S.A., a software company located in Coimbra, Portugal. It was founded in 2001, with its origins in the University of Coimbra and Instituto Pedro Nunes (IPN). The main focus of the company is to create advanced solutions and white-label products for the mobile telecommunications industry. In addiction, WIT also develops other products such as Internet Protocol television (IPTV) middleware and TV apps, Over-the-Top television (OTT-TV) solutions, Mobile Banking and Mobile Payment solutions.

This internship in particular is integrated with the telecommunications industry solutions and products. More detail about the context is given in the next section.

1.2 Main Goal

The main goal of this project is to specify the procedures and describe the implementation of Visual Call Enrichment (VCE) share service. This involves developing an Application Server (AS) to be deployed in an IP Multimedia Subsystem (IMS) network, as well as adding the VCE capabilities to an Rich Communication Service (RCS) client application. With this, it should be possible to offer interactive content, such as visual menus and forms to fill in, when calling an IVR system or a Call Centre.

It is noteworthy that this project consists in a proof of concept, so some of the aspects like IVR platform integration can be left aside. Also, WIT Software requested that whenever possible, all of the technologies used should be free or open source.

1.3 Internship Context

Prior to the 20th century, before the invention of the telephone in 1876, the customer service relationship was done face-to-face or by post. When a client had a doubt, problem or complaint they had to go in person to a dedicated customer-service so that the problem could be resolved. With invention of the telephone, at first everything remained the same since the first phones were sold as pairs, to communicate only with each other. With the invention of the switchboard, circa 1894, calls could be made between more than a pair of telephones. At this point, a customer in need could call the store where he bought the product.

In 1960, the call centre emerges. In an effort to increase efficiency, some of the major companies started to create specific departments only to handle customer service. The major game changer for these costumer service centres was the invention of Dual-Tone Multi-Frequency (DTMF) signalling in 1962. The American Bell Telephony System presented DTMF in 62's World Fair in Seattle, USA by unveiling the first phone in history that dialed numbers through audio tones and not with the previous rotary system. This led to the invention of Interactive Voice Response (IVR) systems by the end of the decade.

IVR systems emerge in the early 70's. These systems (computers) could recognize the different tones dialled in DTMF phones. Limited at first, they could recognize small vocabularies and route customers through the options available. In the late 80's, with the evolution of IVR, complex phone trees arose.

In the early 90's, with the emergence of the Internet among customers and companies, e-mail and live chat support started to become a reality. The use of e-mail was of great advantage for the customer because it eliminates the inconvenience and costs of using a phone. For the first time in 40-years, the 1-to-1 interaction with another human was back to customer service instead of a sequence of robotic responses.

All through the 2000's, new ways of giving support to customers were introduced, of which, offering more and more interactivity between clients and service. In the early part of the decade, support software sold with the product as a CD or downloadable, became one of the common practices.

Nowadays, with social networks, the process of answering a question from a client, or helping them in the resolution of a problem is much quicker and more reliable. Also, with the emergence of remote desktop technology a software client can just sit back and watch their tech support solve the problem.

Nevertheless, back to the telecommunications scope, the reality about interactivity in customer support calls at the time of writing, is that, a customer is routed through IVR menus until they have to talk with some attendant, that, at the beginning of the conversation, knows nothing about the problem.

In the past years, due to the fast evolution of Internet, an increasing number of Over-the-top (OTT) solutions for communication have emerged. Applications like Skype, and Whatsapp are a reality and allow clients to share multimedia content such as video, image, files or messages as well as to make voice and video-calls. More about this phenomenon can be read in Chapter 3
State of the Art. This is costing telephony companies money because clients use these services instead of the standard circuit switched networks. In an attempt to reverse this profit loss, telephony companies joined together and released the RCS specification, defining a way to integrate all these features across all networks and devices in a seamless way for the client.

With the effort from telephony companies to keep up with the new solutions available, it is possible to share multimedia content, like video and image. With all these features of today's services and devices, including computers, tablets or smartphones, interactivity between customer and service can be enhanced.

It's in this context that the internship will be developed, trying to achieve a solution to enrich calls with interactive content in the call context, namely, calls to IVR Systems and Call Centres.

1.4 Planning

The planning, made with the help of Eng. João Alves from WIT-Software, for the duration of this internship was divided in two phases, corresponding to the two semesters.

1.4.1 First Semester

In the first phase, the intern should study the available technologies, as well as, study the state of the art for Visual IVR solutions. With this first stage completed, the intern should start using the numerous technologies involved in the development, and finally, some tests and the first development tasks for the AS should be performed.

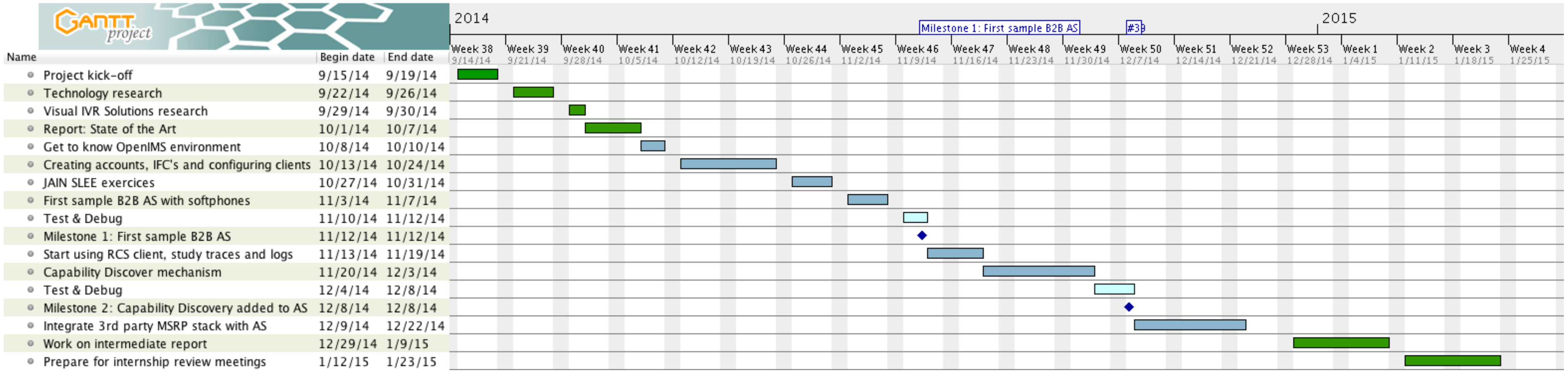
The diagram in Figure 1 shows the plan for the first semester.

1.4.2 Second Semester

For the second phase of the planning, during the second semester, the intern should develop the rest of the AS for it to be used in the various use cases. In addition to this, some customization of the RCS client is needed so the test cases can be properly tested. These tasks should be completed before the end of the development of the AS so the work can be done in an incremental way.

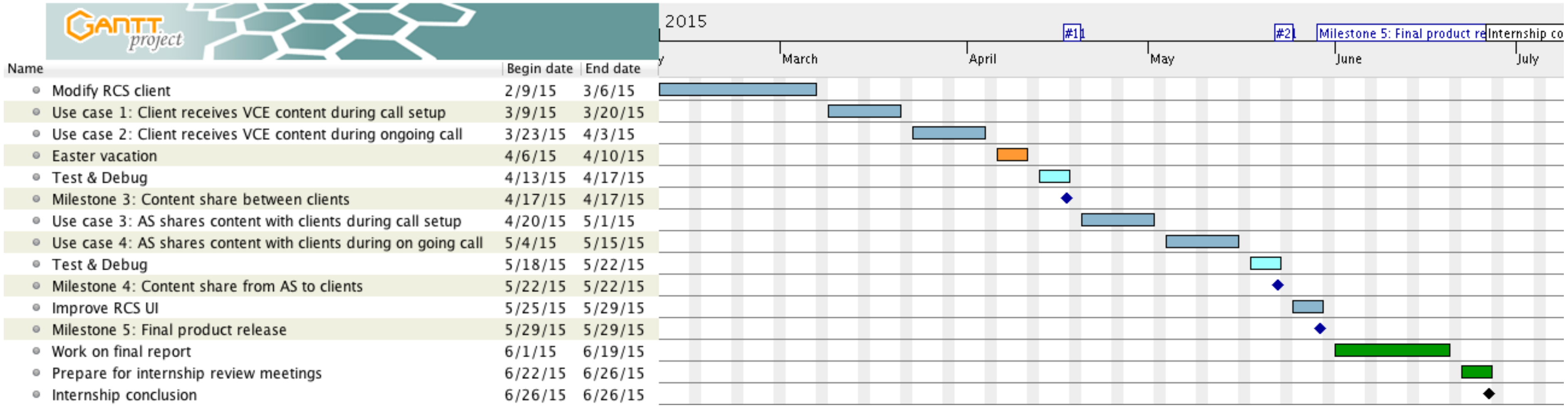
The diagram in Figure 2 shows the plan for the second semester.

It is important to understand that the plan for the second semester suffered some changes due to some delays and decisions. Figure 3 shows the new plan for the second semester after the intermediate report delivery and evaluation.



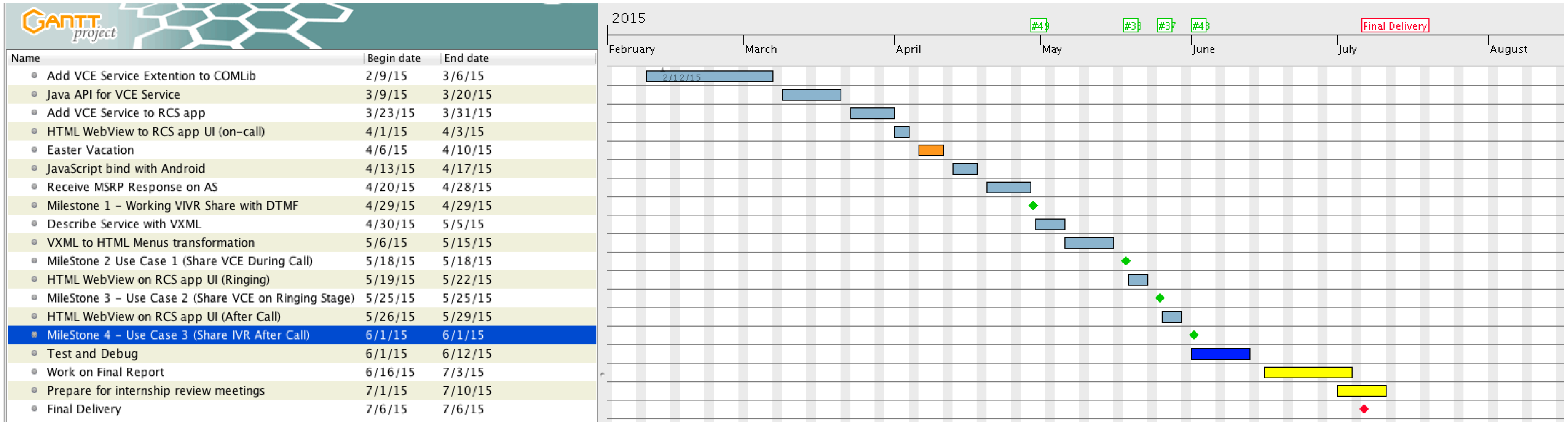
Name	Begin date	End date
Project kick-off	9/15/14	9/19/14
Technology research	9/22/14	9/26/14
Visual IVR Solutions research	9/29/14	9/30/14
Report: State of the Art	10/1/14	10/7/14
Get to know OpenIMS environment	10/8/14	10/10/14
Creating accounts, IFC's and configuring clients	10/13/14	10/24/14
JAIN SLEE exercices	10/27/14	10/31/14
First sample B2B AS with softphones	11/3/14	11/7/14
Test & Debug	11/10/14	11/12/14
Milestone 1: First sample B2B AS	11/12/14	11/12/14
Start using RCS client, study traces and logs	11/13/14	11/19/14
Capability Discover mechanism	11/20/14	12/3/14
Test & Debug	12/4/14	12/8/14
Milestone 2: Capability Discovery added to AS	12/8/14	12/8/14
Integrate 3rd party MSRP stack with AS	12/9/14	12/22/14
Work on intermediate report	12/29/14	1/9/15
Prepare for internship review meetings	1/12/15	1/23/15

Figure 1 - Gantt diagram for first semester planning



Name	Begin date	End date
Modify RCS client	2/9/15	3/6/15
Use case 1: Client receives VCE content during call setup	3/9/15	3/20/15
Use case 2: Client receives VCE content during ongoing call	3/23/15	4/3/15
Easter vacation	4/6/15	4/10/15
Test & Debug	4/13/15	4/17/15
Milestone 3: Content share between clients	4/17/15	4/17/15
Use case 3: AS shares content with clients during call setup	4/20/15	5/1/15
Use case 4: AS shares content with clients during on going call	5/4/15	5/15/15
Test & Debug	5/18/15	5/22/15
Milestone 4: Content share from AS to clients	5/22/15	5/22/15
Improve RCS UI	5/25/15	5/29/15
Milestone 5: Final product release	5/29/15	5/29/15
Work on final report	6/1/15	6/19/15
Prepare for internship review meetings	6/22/15	6/26/15
Internship conclusion	6/26/15	6/26/15

Figure 2 – Original Gantt diagram for second semester planning



Name	Begin date	End date
Add VCE Service Extention to COMLib	2/9/15	3/6/15
Java API for VCE Service	3/9/15	3/20/15
Add VCE Service to RCS app	3/23/15	3/31/15
HTML WebView to RCS app UI (on-call)	4/1/15	4/3/15
Easter Vacation	4/6/15	4/10/15
JavaScript bind with Android	4/13/15	4/17/15
Receive MSRP Response on AS	4/20/15	4/28/15
Milestone 1 – Working VIVR Share with DTMF	4/29/15	4/29/15
Describe Service with VXML	4/30/15	5/5/15
VXML to HTML Menus transformation	5/6/15	5/15/15
MileStone 2 Use Case 1 (Share VCE During Call)	5/18/15	5/18/15
HTML WebView on RCS app UI (Ringing)	5/19/15	5/22/15
MileStone 3 – Use Case 2 (Share VCE on Ringing Stage)	5/25/15	5/25/15
HTML WebView on RCS app UI (After Call)	5/26/15	5/29/15
MileStone 4 – Use Case 3 (Share IVR After Call)	6/1/15	6/1/15
Test and Debug	6/1/15	6/12/15
Work on Final Report	6/16/15	7/3/15
Prepare for internship review meetings	7/1/15	7/10/15
Final Delivery	7/6/15	7/6/15

Figure 3 - New plan for second semester

Chapter 2
Technical Background

2.1 Introduction

In this second chapter, the technical background for the technologies involved with this project is presented. The various protocols, network architecture specifications and concepts relevant for the development of this project are shown in detail. This way the reader can better understand what is described in the chapters to follow. The topics are shown in a top-down approach regarding abstraction from technical jargon, starting with RCS, the specification in which the core of this project's work is based, and going down into less abstract topics and technologies that are core to RCS, such as Voice over IP (VoIP), IMS, Session Initiation Protocol (SIP) or Message Session Relay Protocol (MSRP).

2.2 Rich Communication Services

From the technical point of view, RCS is a GSMA program for the creation of inter-operator communication services, based on IMS (Section 2.4 IP Multimedia Subsystem). The main goals for RCS are to enhance phonebook with the access to presence and service discovery; enhance messaging by enabling a large variety of messaging options like group chat, emoticons, location or file sharing; and enriched calls enabling multimedia content sharing during a voice call, video call and video sharing.

The two main aspects of the RCS specification that are crucial to this project are, capability discovery (section 2.6 of RCS 5.1 specification [18]) and RCS extension service (section 3.12 of RCS 5.1 specification [18])

2.2.1 Capability Discovery

The capability discovery process, or service discovery mechanism is one of the key aspects of RCS. This process enables a RCS user to know and understand the subgroup of services or capabilities that one other user can be part of at one point in time [18]. For example, it is the process that allows the client application to know if a certain contact can be called to join a video call, or share files.

There are two alternative mechanisms to perform the capability discovery: SIP OPTIONS exchange and Presence. They can coexist but SIP OPTIONS exchange is the default and most common practice to be used, and the RCS client provided by Wit-Software (2.2.3 RCS Client (RCS+)) is no exception. This capability discovery mechanism was the only one to be taken into account in the development of the AS relevant to this project.

The process for capability discovery is done according to the flow in the Figure 4 - Capability Discovery Process [18]. User A capability tags are sent in the initial SIP OPTIONS message to user B, in the event of a response to that message, user A shall handle the possible responses as follows:

- **200 OK with capability tags**, updates user B capabilities information if B is a known RCS user. Marks user B as RCS with capabilities information in the response if it is not a known RCS user.

- **200 OK without capability tags** removes user B from RCS known users. No change if is not a known RCS user.
- **480 Unavailable /408 Timeout**, no change to user B either it is a known RCS user or not, only show its capabilities offline.
- **404 Not found /604 Does not exist,** removes user B from RCS known users. No change if is not a known RCS user.

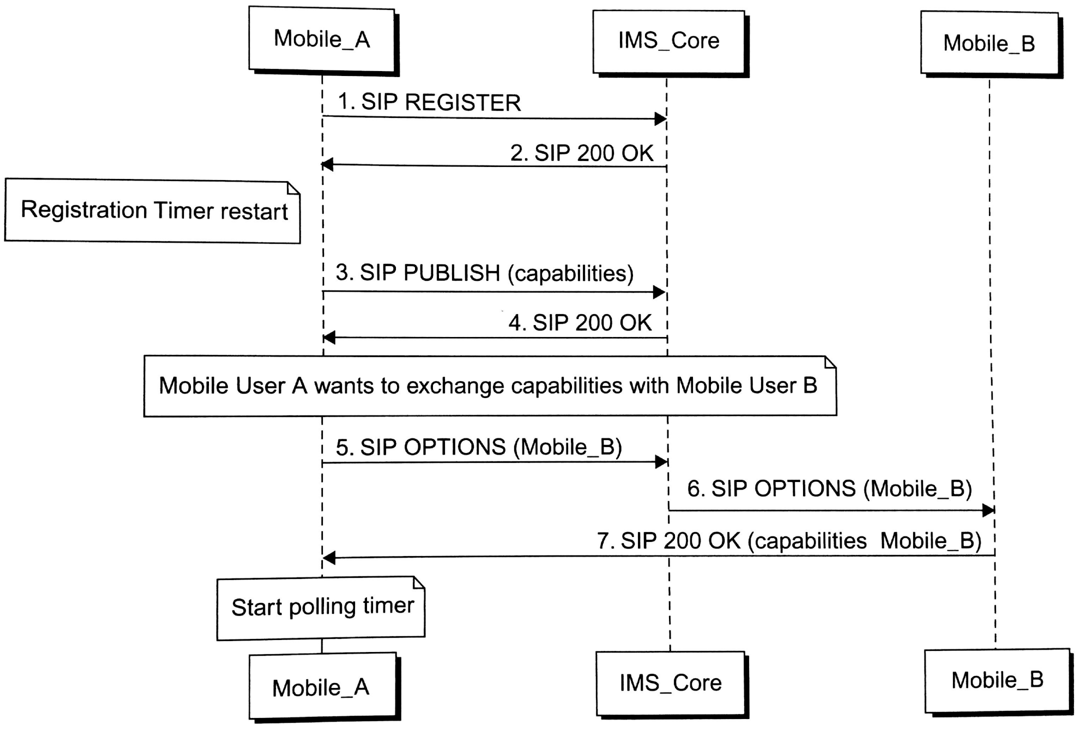

Mobile_A
IMS_Core
Mobile_B
1. SIP REGISTER
2. SIP 200 OK
Registration Timer restart
3. SIP PUBLISH (capabilities)
4. SIP 200 OK
Mobile User A wants to exchange capabilities with Mobile User B
5. SIP OPTIONS (Mobile_B)
6. SIP OPTIONS (Mobile_B)
7. SIP 200 OK (capabilities Mobile_B)
Start polling timer
Mobile_A
IMS_Core
Mobile_B

Figure 4 - Capability Discovery Process [18]

2.2.2 Extension Service for content sharing

Apart from all the other feature services RCS specifies, like IP video call, content sharing, instant messaging or geo-location among others, for the propose of this project, being the intention to create a new capability not yet specified in RCS, the extension feature will be used to announce the required capabilities.

This feature enables an Extension to use the RCS infrastructure to communicate with other RCS entities [18]. Due to its nature, being a general feature that does not specify any of the particular features already specified, it can interact with any other RCS feature, invoking it. With it, the idea is to create other services not yet covered by the current specification, which comes as the right tool for the job in the case of this project.

2.2.3 RCS Client (RCS+)

To test the AS development, and to develop work in a client application that can have the functionalities that are required for this project, a RCS client is needed. Wit-Software as an official GSMA distributor, is responsible for developing a RCS client called RCS+ that already follows RCS specifications and is the right choice to be the test terminal client for the AS development. The ability to show the HyperText Markup Language (HTML) within the RCS+ RCS client will also be developed by the intern.

2.3 Voice over IP

Digital transmission of voice happens in networks such as ISDN and GSM/UMTS, where the voice is transported, in digital form, over digital circuits. Talking about circuits is talking about the virtual connection between two end-points. When this circuit is in place, data can be transported to the remote end of the circuit by sending data to the circuit. This circuit then ensures that this data is delivered to the end-point that is the destination. The user of circuits for transporting voice and other types of media as led to the term "circuit-switched networks" (CS networks). IP networks fall in the category of Packet switched networks (PS networks). Traditional Public Switched Telephone Network (PSTN) phone system uses circuit switching while VoIP uses packet switching.

Essentially, transporting digitized speech through a CS network has much in common with transporting digitized speech through PS network. The "IP" in VoIP refers to the communication network, through which the voice is transported that is PS network. It is important to understand the main differences between circuit switching and packet switching to better comprehend how VoIP works.

- **Circuit-switched networks** require dedicated point-to-point connections during calls. This connection is the circuit referred previously, that guarantees the full bandwidth of the channel and remains connected for the duration of the communication session. It functions much like an electrical circuit as the nodes were physically connected to each other.
- **Packet-switched networks** move data in separate, small blocks, data packets, delivered based on the destination address in each packet header, over a computer network. When received by an end-point, packets are reassembled in the proper sequence to make up the message.

In the next table, the main differences between these two methodologies of implementing telecommunication networks are shown.

Circuit Switched Networks	**Packet Switched Networks**
Low security.	High security.
Dedicated and full bandwidth to each communication.	Bandwidth used to full potential in all communications.
Highly reliable, dedicated circuit.	Low reliable, subject to congestion.
Affected by line failure (call ends).	Not affected by line failure (redirections).
Low availability (busy line).	High availability.
Guaranteed quality of service (QoS).	Protocols are needed for a reliable transfer.
During a crisis or disaster, the network may become unstable or unavailable.	Still working during crisis or disaster.
It was primarily developed for voice traffic rather than data traffic.	Data packets can get lost or become corrupted

Table 1 - Difference between CS Networks and PS Networks

In comparison, it is easy to understand why PS networks are replacing CS networks over time. Although CS networks have some advantages, the main reason behind their replacement with PS networks is that it is easier and cheaper to increase the capacity of a PS network; the alternative of building up telephone network to satisfy the huge demand of nowadays clients is economically out of the question.

The idea of sending voice data over the Internet rather than communication through traditional telephone service dates back to 1995 [29]. The concept allowed computer users to avoid long distance charges. 1995 is also the year of the first release of the firs Internet Software Phone, the VocalTec Internet Phone. It required the same software installed in the two machines, the one calling and the called-party, the sound quality was really poor, but the potential of VoIP was without a doubt proved. In the next few years it evolved gradually, in 1998 PC to phone service was offered by some companies. Since 2000, VoIP usage has expanded dramatically, as the commercial VoIP service providers proliferate. Companies like Skype, the so called second generation providers, changed the business model for VoIP calls, closing networks for private usage basis offering the benefit of free calls and charging for access to other communication networks like PSTN. Third generation providers, such as Google Hangouts, formerly Google Talk, have adopted the concept of federated VoIP which marks the departure from the architecture of legacy networks allowing dynamic interconnection between users on any two domains when a user wishes to make a call.

VoIP systems have session control and signaling protocols to set-up and teardown of calls. These transport audio (and video) streams using special media delivery protocols that encode voice, audio and video. The implementation of VoIP has been achieved using both open standards and proprietary protocols, some examples are: SIP, Session Description Protocol (SDP), Real-Time Transport Protocol (RTP), Real-time Transport Protocol Control Protocol (RTCP) or H.323. The most relevant protocols for this project are described in some of the sections to follow.

2.4 IP Multimedia Subsystem

The RCS communication services are based in IMS networks, this fourth section intent is to introduce the reader to the basics of IMS network architecture describing its components and how communication between them is archived. In addition, the way to include services, e.g. AS, in an IMS network is explained.

2.4.1 Architecture

Figure 5 shows the reader the schematics of IMS networks architecture as defined in 3GPP technical specifications [30]. We can see the network is broken down in its three main layers: Application layer, Session & Control layer (IMS layer) and Transport layer. Each one of the layers is explained in more detail below.

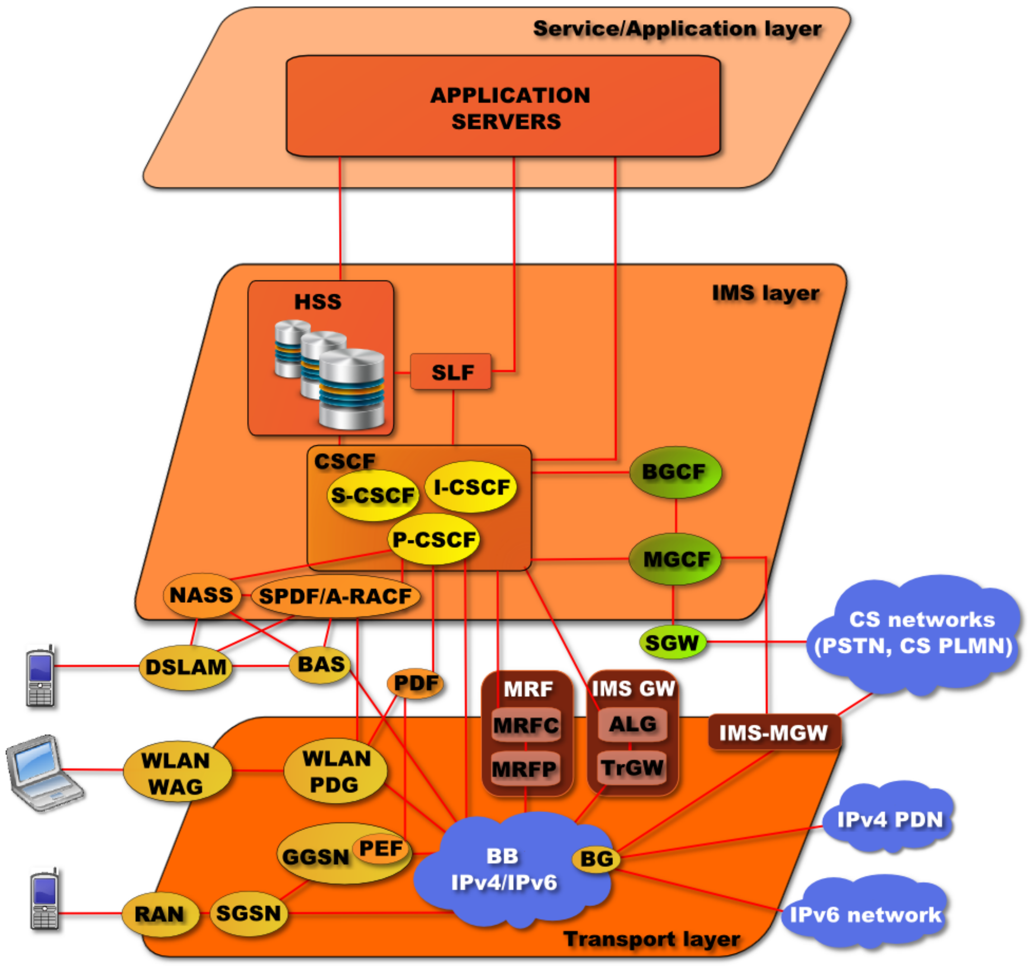


Figure 5 - IMS Architecture [31]

The Application Layer undertakes the control of the end services required by the user. The IMS architecture and SIP signaling has been designed to be flexible and in this way it is possible to support a variety of telephony and non-telephony servers concurrently [31]. Within this layer there are a wide variety of different servers that are supported. This includes a Telephony AS, IP Multimedia - Service Switching Function (IM-SSF), Supplemental Telephony AS, Non-Telephony AS, Open Service Access - Gateway (OSA-GW), among others. This layer is presented in more detail, in section 2.4.3 Applications in IMS regarding application in IMS.

The Session & Control layer (IMS layer) contains what is called the CSCF, which provides the endpoints for the registration and routing for the SIP signaling messages, enabling them to be routed to the correct application servers. The CSCF also enables QoS to be guaranteed. It achieves this by communicating with the transport and endpoint layer. It also includes other elements including the HSS that maintains the user profiles including their registration details as well as preferences and the like. In this layer is located what is called the IMS Core, essential to the IMS functionality and to this project. This is further explained in the next section.

The Transport layer initiates and terminates the SIP signaling, setting up sessions and providing bearer services including the conversion from analogue or digital formats to packets. This IMS layer also contains all of the media processing facilities including media gateways. These can be used to convert VoIP bearer streams to the PSTN TDM format. They can also be used to provide many media-related services such as conferencing, playing announcements, collecting in-band signaling tones, speech recognition, and speech synthesis.

2.4.2 Core Network

IMS Core, presented in Figure 6, refers to the limited number of control components of functional entities of IMS solution. It performs the roles of core control within CSCF and user data storage in HSS [19]. Core network essential functions are to provide network authentication and session control. Greater detail about the four essential core components and the interfaces used to communicate between them is presented in the next paragraphs.

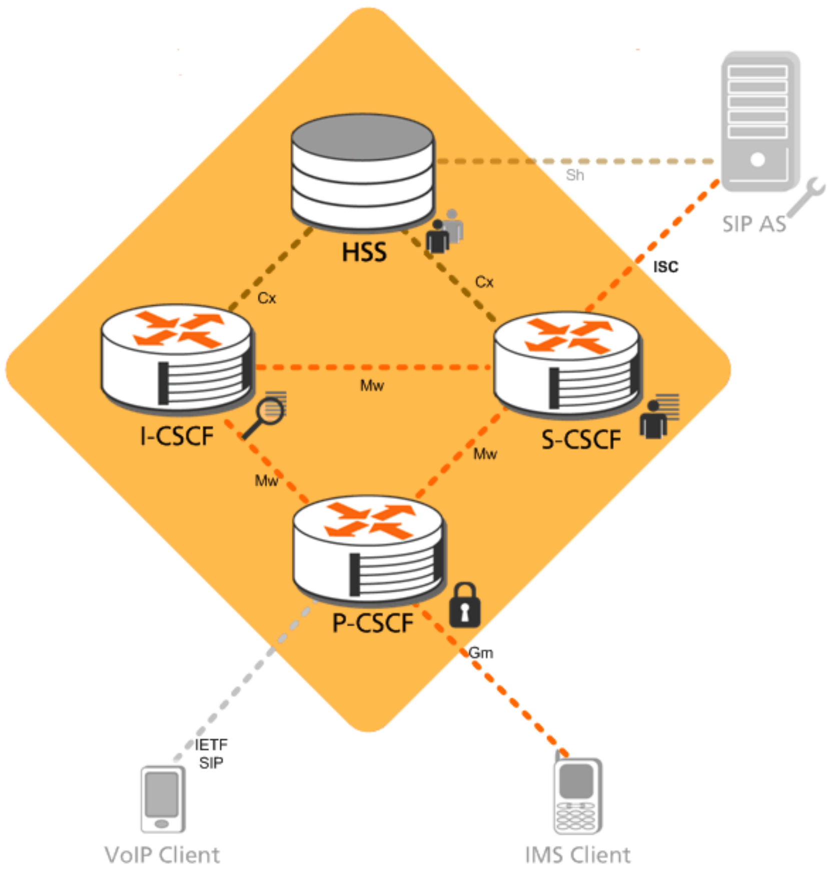


Figure 6 - IMS Core Network [22]

Home Subscriber Server

The HSS is essentially the main database (DB) for subscribers for IMS. It holds the static subscriber data, which is distributed to other core entities (e.g. CSCF's). This static subscriber data refers to: subscriber profiles, location and IP address information. HSS can also preform authentication and authorization for a given subscriber.

The two main subscriber user identities used in IMS context are IP multimedia private identity (IMPI) and IP multimedia public identity (IMPU). Simply put, the IMPI is a unique permanently assigned identity, given by the home subscriber and it's used for registration or authorization purposes. On the other hand, IMPU is the public identity used by any user to contact other users. Each user can have more than one IMPU. These are associated to an IMPI and can be shared across devices.

Serving CSCF

The Serving CSCF (S-CSCF) is the main SIP session control node in an IMS network. The S-CSCF constitutes the registrar for an IMS network subscriber, providing binding between user's IP address and SIP address. There can be multiple S-CSCF's in the network for load balancing and high availability reasons. Each S-CSCF is assigned to a user at the moment of registration and user-S-CSCF correspondence is stored in the HSS.

Proxy CSCF

This CSCF is a SIP Proxy that is the first point of contact for an IMS terminal. It can be described as a specialized Session Border Controller that functions as a user-to-network proxy. All SIP signaling goes through the P-CSCF, which can inspect every signal and ensures that IMS terminals do not misbehave (e.g. change normal signaling routes or not obeying home network's routing policy).

Interrogating CSCF

The Interrogating CSCF (I-CSCF) is used for forwarding an initial SIP request to a S-CSCF when the initiator of the request (e.g. IMS terminal) does not know which S-CSCF should receive such request. It uses the HSS to obtain the correct S-CSCF address and then forwards the SIP request to the correct server.

Interfaces description

Communication between these four IMS components is done through pre-defined interfaces. As an example, as we can see in Figure 6, communication between S-CSCF and HSS is done with a Cx interface. These interfaces are defined in the IMS 3GPP specification and they establish communication between well-known components. The next table (Table 2) shows the several interfaces as well as the components connected by them and the protocol used.

As an important note, many other interfaces exist in IMS environment, but they are not relevant for the scope of the work of this internship, so only the main interfaces, connecting the most relevant components of IMS Core, User Endpoints (UE) and Application Servers are discussed.

Interface	Description	Protocol
Cx	Used to send subscriber data to the S-CSCF; including Filter criteria and their priority.	Diameter
Mw	Used to exchange messages between CSCFs	SIP
ISC	Used to notify the AS of the registered IMPU, registration state and UE capabilities and supply the AS with information to allow it to execute multiple services	SIP
Gm	Used to exchange SIP messages between UE and P-CSCF	SIP
Ma	Used to Forward SIP requests which are destined to a Public Service Identity hosted by the AS or Originate a session on behalf of a user or Public Service Identity, if the AS has no knowledge of a S-CSCF assigned to that user or Public Service Identity.	SIP
Sh	Used to exchange User Profile information between AS and HSS	Diameter

Table 2 - IMS Interfaces

2.4.3 Applications in IMS

Application servers, located in the application layer of IMS, have the purpose of hosting and executing services and interact with the S-CSCF using SIP. Depending on the actual service, the AS can operate in SIP proxy mode (letting through SIP requests and responses), SIP User Agent (UA) mode (behaving as a terminating user) or Back-to-Back User Agent (B2BUA) mode (behaving as terminating UA to the originating UA and as originating UA to the terminating UA).

Back-to-Back User Agent

For the purpose of this project, a B2BUA approach was chosen and can be seen more in detail in Figure 7. According to SIP Request for Comments (RFC) 3261[34] it is a logical entity that receives a SIP request and processes it as User Agent Server (UAS). To determine how this request should be answered, it acts as a User Agent Client (UAC) that generates responses to these requests. Unlike a proxy AS which only forwards the request and responses received adding some content, B2BUA are more versatile and have more control over the flow of SIP messages, which are described in more detail in the next session, protocols.

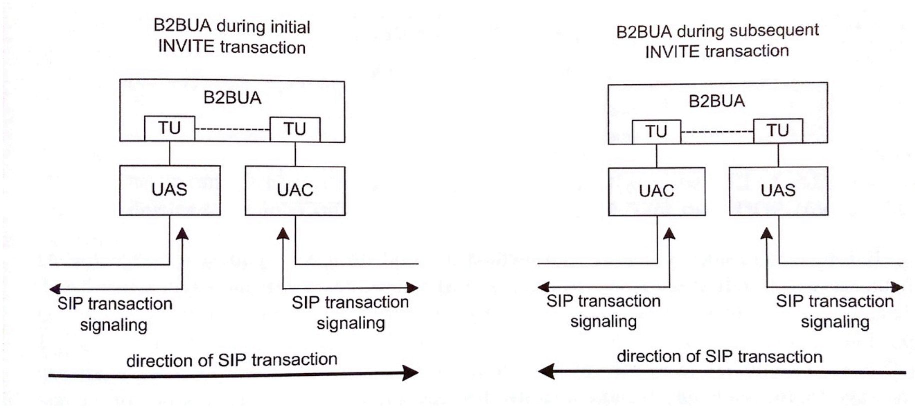

B2BUA during initial INVITE transaction
B2BUA
TU
TU
UAS
UAC
SIP transaction signaling
SIP transaction signaling
direction of SIP transaction
B2BUA during subsequent INVITE transaction
B2BUA
TU
TU
UAC
UAS
SIP transaction signaling
SIP transaction signaling
direction of SIP transaction

Figure 7 - B2BUA during INVITE transaction [35]

2.4.3 Session handling (Initial Filter Criteria)

In IMS, the session handling is one of the key aspects of the system. It allows to dynamically triggering SIP applications. It is implemented as a filter signaling mechanism done in the S-CSCF. This filtering may be used to determine the forwarding of SIP requests to a specific AS. The Initial Filter Criteria (IFC) is an Extensible Markup Language (XML)-based format used by IMS to define this filtering done by the S-CSCF. Stored in the HSS, and downloaded by the S-CSCF upon registration it represents the subscription of a user or set of users to an application. In the context of this project, this was an important feature of IMS since it is important to filter the requests to the AS to be developed.

2.5 Protocols

In this fifth section some details about the relevant protocols in this project are explained, being the most important SIP, RTP, SDP and MSRP.

2.5.1 Session Initiation Protocol

SIP is an application layer signaling protocol used to establish, modify and terminate multimedia sessions such as VoIP calls, video conferencing or file transfer among others. Originally developed in 1996 and standardized in 2002 under the name RFC 3261 [34] by the IETF (Internet Engineering Task Force), it is the control-plane protocol used in the IMS network for registration, session establishment, message routing, capability exchange, etc.

Much like HTTP, SIP employs design elements of request/response transaction model. Most of the header fields, status codes and encoding rules of HTTP are reused in SIP, providing a readable text-based format. The full extent of request and responses available for SIP transactions can be consulted in RFC 3261 [34], the ones relevant for this project are presented in the next tables (Table 3 and Table 4).

Request name	Description
INVITE	Indicates to a end user that he is being invited to a session. It can be a call session, instant messaging, video/image share, etc. The requisites for this session are in the message body using SDP.
ACK	Confirms that a end user as received a final response to one request such as INVITE or OPTIONS
BYE	Terminates a session, either for a call or other, between two end users. Can be sent either by the caller or the one called.
CANCEL	Cancels any pending request.
REFER	Asks end user to re-invite to another end user, used in the situation of a call transfer.
OPTIONS	Queries the capabilities of servers or end users.
REGISTER	Registers the address sent in the To header field in the network.

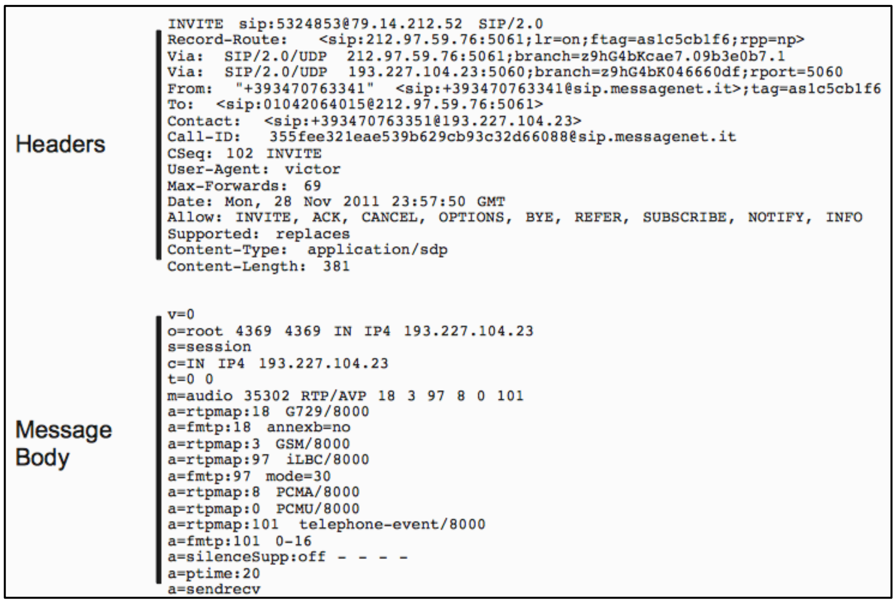
Table 3 – SIP Requests

To this requests, the queried user can give a huge amount of responses. The next table (Table 4) aggregates such responses by type. These responses are much similar to the ones used by HTTP, they specify a three-digit integer response code, which are grouped according to their first digit as "provisional", "success", "redirection", "client error", "server error" or "global failure" codes, corresponding to a first digit of 1–6

Response code	**Description**
1xx – Provisional	Also known as informational responses, they indicate that the queried server is performing some task, has received the request but has no definitive response yet. It is never expected to send an ACK to a 1xx response.
2xx – Success	The request was successful.
3xx – Redirection	Give information about user's new location or about alternative services in the network that can satisfy call session.
4xx – Client error	Definite failure responses indicate that a particular server cannot handle such request. Request should be modified, or sent to another server.
5xx – Server error	Also definite failure messages, they indicate that the server has erred. The request may be ok, but there was a problem in the server handling it.
6xx – Global failure	The request has failed and should not be tried again to any server. The server has definite information about the user but for some reason the contact can't be made. (e.g. busy callee)

Table 4 - SIP Response code types

In Figure 8 we can se a simple example of a SIP message and its three major components. The initial line describes the type of SIP request/response, in this case, an INVITE request. Headers field has the necessary fields for routing, authentication, capabilities, and other transaction related information. The optional message body, separated from headers by a blank line, contains the message to be delivered; it can be like in this example, SDP related.


Headers
Message Body

```
INVITE  sip:5324853@79.14.212.52  SIP/2.0
Record-Route:   <sip:212.97.59.76:5061;lr=on;ftag=as1c5cb1f6;rpp=np>
Via:  SIP/2.0/UDP  212.97.59.76:5061;branch=z9hG4bKcae7.09b3e0b7.1
Via:  SIP/2.0/UDP  193.227.104.23:5060;branch=z9hG4bK046660df;rport=5060
From:  "+393470763341"  <sip:+393470763341@sip.messagenet.it>;tag=as1c5cb1f6
To:  <sip:01042064015@212.97.59.76:5061>
Contact:  <sip:+393470763351@193.227.104.23>
Call-ID:   355fee321eae539b629cb93c32d66088@sip.messagenet.it
CSeq: 102  INVITE
User-Agent:  victor
Max-Forwards:  69
Date: Mon, 28 Nov 2011 23:57:50 GMT
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO
Supported:  replaces
Content-Type:  application/sdp
Content-Length:  381


v=0
o=root 4369 4369 IN IP4 193.227.104.23
s=session
c=IN IP4 193.227.104.23
t=0 0
m=audio 35302 RTP/AVP 18 3 97 8 0 101
a=rtpmap:18  G729/8000
a=fmtp:18  annexb=no
a=rtpmap:3  GSM/8000
a=rtpmap:97  iLBC/8000
a=fmtp:97  mode=30
a=rtpmap:8  PCMA/8000
a=rtpmap:0  PCMU/8000
a=rtpmap:101  telephone-event/8000
a=fmtp:101  0-16
a=silenceSupp:off - - - -
a=ptime:20
a=sendrecv
```

Figure 8 - SIP INVITE message example [36]

SIP is not a vertically integrated communications system, as previously described; it is integrated in IMS with other protocols to build a complete multimedia architecture. As an example of this, the next figure (Figure 9) shows a simple call-flow using SIP between two terminals through an IMS core network. The reader can easily see that other protocols, such as SDP and RTP are used during the call-flow. These protocols are described in the next sections.

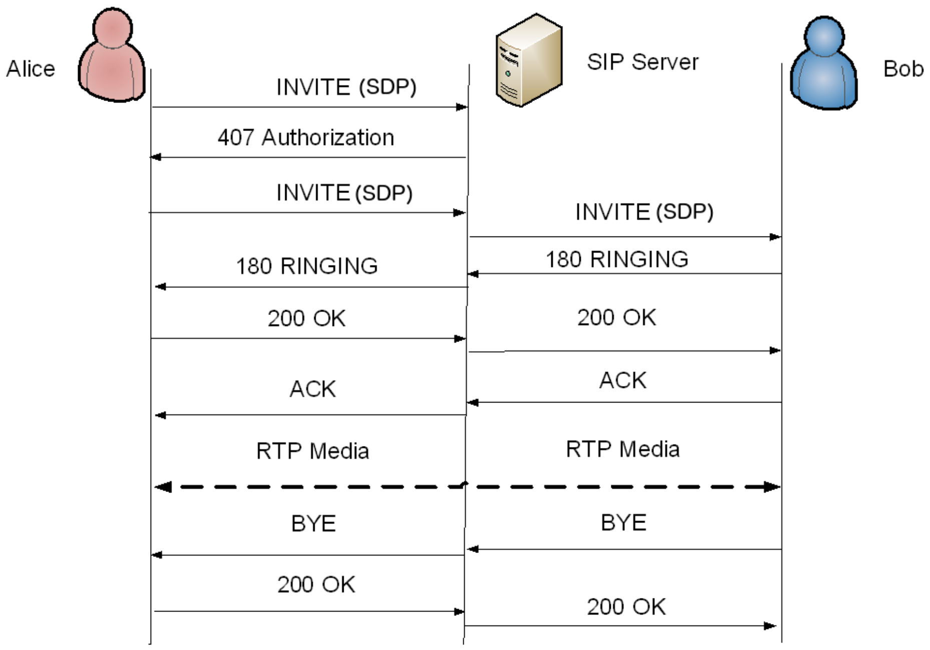


Figure 9 - VoIP call flow using SIP as signaling protocol

2.5.2 Real-Time Transport Protocol

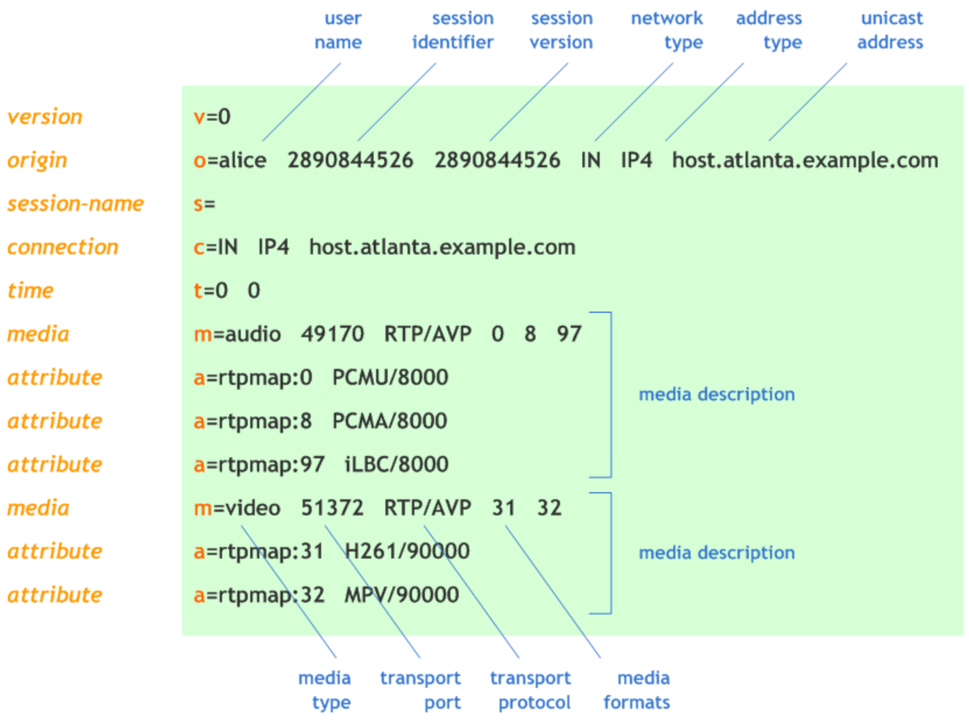
The RTP is a packet format protocol used to deliver audio and video within IP networks, it is used to stream media in communication and entertainment systems such as videoconference, television services and telephony. RTP is designed for end-to-end, real time transfer of stream data; this is achieved because it provides facilities for jitter compensation and detection of out of sequence packets of media data, which are common on any IP network. It is regarded as the primary standard for audio and video transport in IP networks and it is largely used in VoIP services.

In the context of IMS, upon session call establishment, RTP is the protocol used to transfer audio and video between terminals such as mobile phones or softphones. In Figure 9 we can see the voice media being shared end-to-end with a direct connection Bob and Alice

2.5.3 Session Description Protocol

SDP is used to describe streaming media parameters. Standardized in the RFC 4566 [32] by the Internet Engineering Task Force (IETF), it is mainly used upon session announcement, establishment and parameter negotiation to describe multimedia communication sessions. As RTP is used to deliver the media, SDP is used to describe how this media should me shared, negotiating media types, formats, codecs etc.

In the context of IMS it is used to do just what is described above and it is sent between components tunnelled in the message body of SIP messages. In the next figure, a SDP content of an initial SIP INVITE is shown as an example.



```
version        v=0
origin         o=alice  2890844526  2890844526  IN  IP4  host.atlanta.example.com
session-name   s=
connection     c=IN  IP4  host.atlanta.example.com
time           t=0  0
media          m=audio  49170  RTP/AVP  0  8  97
attribute      a=rtpmap:0  PCMU/8000
attribute      a=rtpmap:8  PCMA/8000
attribute      a=rtpmap:97  iLBC/8000
media          m=video  51372  RTP/AVP  31  32
attribute      a=rtpmap:31  H261/90000
attribute      a=rtpmap:32  MPV/90000
```

Figure 10 - Message-body section of SIP containing SDP parameters [37]

2.5.4 Message Session Relay Protocol

Using SDP as a session negotiating protocol and SIP as signaling protocol, MSRP is used to transmit series of instant messages in the context of a communication session. Defined in the RFC 4976 [33], MSRP is used in RCS, thus IMS, for the instant messaging, file transfer and image sharing features.

The design of this protocol is similar to HTTP or SIP being a text-based protocol. It follows the same request/response methodology as those referred, so a session is established through SIP's offer-answer (request-response) model. Unlike SIP, MSRP is much simpler involving the use of much less headers.

Although image sharing, instant messaging and file transfer are not the main objective in this internship, the new capability to be developed will use MSRP as the message (HTML) exchange protocol between end points. An MSRP session will be established for the communication via the HTML forms shown in terminals and received by the AS.

It is important to say that, like RTP, the MSRP session is end-to-end, establishing direct communication between end users, and in this case, between end users and B2BUA AS.

2.6 From voice menus to dynamic content

To finalize this technical background section, the author presents a brief introduction of some of the technologies used to achieve the transformation from voice automated menus to dynamic content.

IVR Systems and Call Centres have a standard to specify the menus presented to the caller. This standard is called Voice XML or VXML.

One of the goals of this internship is to take this specification of a given menu and transform it in an automatic away in dynamic and responsive HTML menus with the service provider branding associated.

2.6.1 Voice XML

VXML[42] is a digital document, very similar to a normal XML but with some special tags that specify audio and voice menus. They are used to specify interactive media and voice dialogs between a human actor and a computer.

These are normally interpreted by a Voice Browser which translates the VXML contents to automated voice outputs so the caller can interact with them with voice or DTMF signals.

```
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
  <form>
    <block>
      <prompt>
        Hello world!
      </prompt>
    </block>
  </form>
</vxml>
```

Figure 11 - VXML Hello World sample[42]

Figure 11 shows a small sample of a very simple VXML file. In this file a very simple Hello World message prompt is present. This file, when interpreted by a Voice Browser, would output the voice message: "Hello World".

VXML has an huge amount of special tags defined in the W3C Recommendation [42], but for the sake of simplicity, only the most common and taken into account in this project are now presented.

The **`<menu>`** tags specify a simple anonymous field that will prompt to the user has a collection of choices. These choices are represented by the **`<choice>`** tag, which defines a simple choice that can be made by the user. These choices have fields like **`dtmf`** or **`next`** that represent the DTMF tone that corresponds to that choice and the next jump, which can be either a menu within the VXML, a URL or a form. Inside these menus the **`<prompt>`** tag can be commonly found to specify the voice output corresponding to that choice.

The next picture shows an example in which the user is prompted to choose between three choices, each corresponding to a different DTMF signal.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd">
<menu>
  <property name="inputmodes" value="dtmf"/>
  <prompt>
   For sports press 1, For weather press 2, For Stargazer
   astrophysics press 3.
  </prompt>
  <choice dtmf="1" next="http://www.sports.example.com/vxml/start.vxml"/>
  <choice dtmf="2" next="http://www.weather.example.com/intro.vxml"/>
  <choice dtmf="3" next="http://www.stargazer.example.com/astronews.vxml"/>
</menu>
</vxml>
```

Figure 12 - VXML Menu example [42]

The **`<form>`** tag specifies a set of data to be collected and summited to a web service in order to retrieve some information to give to the user. The different data is represented by **`<fields>`**, much like HTML inputs. The submission of the form data is done with the **`<submit>`** tag that stores the URL, the dataset and the either GET or POST method.

In the next picture a simple example of the application of these tags can be seen. In this case the user is prompted to give voice information about state and city. The data is submitted to an URL and data is shown/said to the user.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd">
<form id="weather_info">
 <block>Welcome to the weather information service.</block>
 <field name="state">
  <prompt>What state?</prompt>
  <grammar src="state.grxml"  type="application/srgs+xml"/>
  <catch event="help">
     Please speak the state for which you want the weather.
  </catch>
 </field>
 <field name="city">
  <prompt>What city?</prompt>
  <grammar src="city.grxml" type="application/srgs+xml"/>
  <catch event="help">
     Please speak the city for which you want the weather.
  </catch>
 </field>
 <block>
  <submit next="/servlet/weather" namelist="city state"/>
 </block>
</form>
</vxml>
```

Figure 13 - VXML From example [42]

In this project, the author assumes that a VXML is always present to specify a IVR system or a customer service. In order to transform this data into HTML, the following tools were used.

2.6.2 XML Schema Definition

The XSD is used, in this project context, as the recommendation from W3C [42d] to formally specify the rules of how the VXML document is built. It will allow the system to ensure that the VXML for a given service is well built and with the required tags for the system to correctly transform it to HTML.

2.6.3 Extensible Stylesheet Language Transformations

XSLT was the chosen tool to transform the VXML menus and forms into HTML files. This tool allows the transformation of XML like files in HTML, another XML or plain text document such as JSON.

This way, the author can write one ore more XSL files in order to achieve the correct transformation into HTML.

XSL works in a simple way, when matching the XML tags outputs some text or iteration of text for one or more fields. One of the most powerful tools of XSL is the **`<xsl:template>`** which allows the developer to create a template for any tag or field within the XML.

So, in sort, this final picture shows the whole process from VXML to HTML:

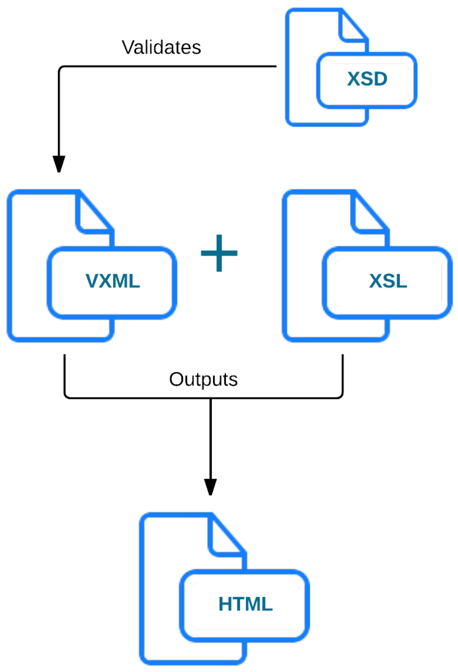


Figure 14 - VXML to HTML transformation process

Chapter 3
State of the Art

3.1 Introduction

In the last two decades customers are used to call customer services and being routed trough the options available to answer their needs by IVR systems, which means listening to the speech presenting the available options and choosing one of them by pressing the corresponding number in the phone dialler. With the increasing capabilities of today's smartphones and softphones, which include receiving multimedia during a call, such as video or images, the idea of presenting these routing menus on the screen of the calling party has emerged. The media that these devices can receive is not interactive, but they are capable of presenting that kind of media (e.g. Web Pages) in other contexts. In this chapter, a brief history in IVR and Visual IVR is presented as well as an overview of the available solutions to integrate interactivity during a phone call.

The scenario of an interactive phone call makes sense in the context of RCS, since the communications are no longer restricted to voice calls, and other media can be shared. Solutions available in this area are also introduced in this chapter.

3.2 Interactive Voice Response

From a general point of view, IVR is a technology that helps humans and machines interact with one another by voice or by DTMF signaling tones[1]. In the telecommunications context, IVR refers to the technology that most of the companies have in their customer services to answer customer incoming calls. During the call, the caller uses the dialler keys or voice commands to navigate through the options menu. IVR systems can follow these commands and direct the customers to the submenus, audio information or a live attendant.

The history of speech and sound recognition research dates back to 1936, when a device called the Voder (voice Operating Demonstrator) was invented at Bell Labs [2]. The purpose of the Voder device was to synthesize human voice, however, the technological breakthrough that would make IVR possible, was a new dial tone methodology introduced by Bell Systems in 1961: the first telephone that could dial area codes using DTMF technology. DTMF devices can transmit audible tones in the 300 Hz to 3.4 kHz frequency range, the same as human voice. The requisites for an IVR system implementation were in place.

During the 1970's and 1980's IVR systems where barely used due to the high maintenance cost they represented, but with the migration to multimedia in the 1990's, there was investment from the companies to integrate their customer services with IVR solutions [1]. From that point on, IVR became mainstream and the standard procedure to make routing decisions based on the input given by the DTMF dial tones sent by the user.

In this decade, IVR can be encountered in other areas outside telecommunications, and evolved to more complex voice recognition systems and applications. IVR applications/systems can be used to control functions in a device or machine, receive information or control entertainment, given that these functions can be performed by inquiring the user, or by receiving voice/DTMF inputs.

For the purpose of this internship, the focus will be more on interacting with IVR systems with DTMF tone decoding, since it is still the main user input mechanism used in todays call centers.

The use of IVR systems offers great advantages to companies, services and clients alike [3]:

- The service can be extended 24/7 without the need to pay extra hours to employees;
- The feeling of instant answer enhances the customer experience;
- High volumes of calls can be answered;
- Calls can be sorted by priority so more urgent issues can be answered quickly;
- No need for a live agent in the inquiring phase.
- Small businesses can look bigger to the outside world if an IVR system is in place to answer the calls;
- Calls can be segmented, and services can be more specialized;
- Location of calls can be identified, and another language can be used to interact with a foreign customer;
- Sensitive information, such as medical test results, can be provided by an IVR system leaving the patient more confortable in his privacy.
- Outbound campaigns can be automated.

3.2.1 Dual-tone multi-frequency signaling

Introduced in 1963 by Western Electric to replace the pulse dialing that was used at the time [4], DTMF sends different frequency tones for each of the digits/characters in the dialler. The tones sent through keypad pressing are decoded by the telephony system, the options are identified and the call can be sent to the number pressed. At the time, the DTMF keypad developed had the 10 digits from 0 to 9, plus the asterisk (*) and the number sign (#) and the letters A, B, C and D. The process of decoding is done by identifying the sinusoidal frequency of the DTMF tone sent by the caller.

Nowadays this type of decoding is used in IVR systems to interpret the calling party choice among the options available [5]. Although the majority of callers use modern smartphones instead of old landline phones with physical keys, this does not pose as a problem for DTMF since these devices can also send the DTMF tone when the virtual keyboard is pressed.

3.2.2 IVR over Internet Protocol

With the growing usage of the Internet Protocol in telecommunications, not only has the use of regular calls or message exchange been affected by it, but IVR services are being affected as well.

The introduction of SIP brings multimedia content sharing point-to-point. Video can be shared by IVR-capable call centers, so, Interactive Video and Voice Response (IVVR) was presented as an extension to IVR [6]. The Full-duplex video capabilities of the communication channel will allow IVR systems to have real-time face-to-face conversations between attendant and customer bringing the advantage of more accurate user identification, preventing fraud and identity theft.

With advances in automation and natural language processing algorithms, another approach is possible, with the exchange of instant messages between customer and an agent. Interactive Messaging Response (IMR) [7] system agents are capable of reading and interpreting the message received, give a response in accordance to pre-written scripts up to 6 customers at a time.

3.3 Visual IVR

The concept behind Visual IVR is similar to the one offered by IVR but with the addition of visual menus so that the user can make their choice without having to hear all the options via pre-recorded voice menus [8]. With the increasing number of handheld devices such as smartphones that support the capabilities of browsing the web or have third party applications installed, this feels like a natural evolution for the IVR technology.

Only in recent years, with the migration of telecommunications to multimedia, and the hardware evolution of smartphones, did the Visual IVR concept begin to emerge. It is a fresh and new concept of customer-business interaction.

With Visual IVR, companies can offer the user a better user experience when contacting a customer service call center, as the visual representation of menus and information makes for a shorter on-hold time because the information can be read instead of listened to. This provides the user the opportunity to skip trough the information/option that they consider irrelevant for the purpose of the call.

More of the attendant work can be done by an IVR system with Visual capabilities. For example, a simple query for some customer info such as name, address or contact number can be filled by the user in a form presented on the smartphone screen, instead of filled by the attendant listening to the user. This enhances also the privacy offered to the user. Passwords can be traded in a system like this, whereas in a voice only system, privacy would be compromised.

Promotions and campaigns can also beneficiate from a Visual IVR system. The customer, when called by an IVR system, answers the phone and listens to a recorded voice whereas in a Visual IVR scenario, the customer could receive the promotion, campaign or issue of the call prior to answering the call.

Finally, summaries and surveys can be presented to the user after the call is done. When the customer disconnects the call, a form to answer simple questions about quality of service or other types of surveys that the company feels the need to have answered, can be presented on the smartphone screen so the caller can do it without having to listen to the questions and ranking system.

In the next section some examples of Visual IVR solutions currently available in the market are presented.

3.3.1 Visual IVR Solutions

In this section, some of the solutions to enrich the call between client and IVR System are presented. A research was made to find the available software that allowed calling customers to have interactive calls, that means that the user is able to interact in some way with the other entity in the other side of the line, with something other than static content like images or video. The main criteria to find the relevant solutions for this study was that each of this software could in some way offer interactivity during call stage.

Some Visual IVR solutions are already emerging and are available to be integrated in business call centers, some with a customer oriented philosophy, and others more invested in creating robust solutions with back office and seamless interaction.

In most of the next examples, some technical aspects of the systems are not reveled by the manufacturers, but the main focus of this stage is to identify the functionalities they offer as well as some of the differences between them.

Some of this solutions where not available to be installed and had no screenshots available, for that reason, there are no image examples for some of the solutions presented below.

CallVU IDR

CallVU IDR [10] is Visual IVR solution patented in 2008 by CallVU LTD. The IDR acronym stands for Interactive Display Response. The work done so far allows companies to offer their calling clients visual menus to choose options during voice call to better communicate with customers. This additional rich-media content may provide additional information to the customer as well as advertise relevant products or services.

It is available in the Android Play Store for Android platforms and Apple Store for iPhone as a standalone application. At first use, the application requires that the user registers their phone number. From that moment on, when calling a call center that provides IVR, the app automatically opens and prompts the user with the visual interaction.

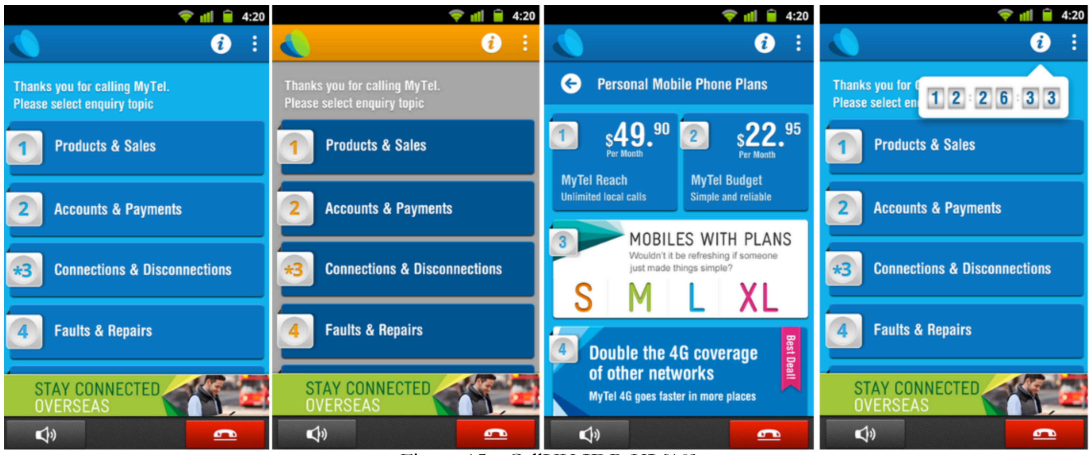

Thanks you for calling MyTel.
Please select enquiry topic
1 Products & Sales
2 Accounts & Payments
*3 Connections & Disconnections
4 Faults & Repairs
STAY CONNECTED OVERSEAS
Personal Mobile Phone Plans
$49.90 Per Month
MyTel Reach
Unlimited local calls
$22.95 Per Month
MyTel Budget
Simple and reliable
MOBILES WITH PLANS
Wouldn't it be refreshing if someone just made things simple?
S M L XL
Double the 4G coverage of other networks
MyTel 4G goes faster in more places
Best Deal!
1 2 2 6 3 3

Figure 15 - CallVU IDR UI [10]

Zappix – Visual IVR Solution

Zappix [11] made a different and simpler approach to the Visual IVR method. It is a standalone application for iOS and Android handheld devices that offers the user the possibility to navigate trough existing IVR menus and directly call the specified service. There is no further interaction with the Zappix application from this point on. A normal phone call is made to a company call center and the DMFT tones required to get to the chosen option are dialed by the application. The Zappix developers add companies and their info to the application. Furthermore, the application can show ads during the time the phone is ringing according to the chosen service.

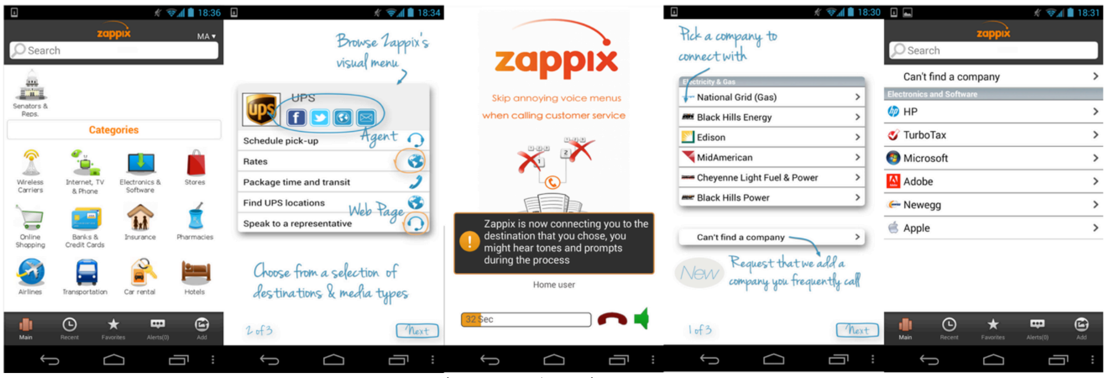

Browse Zappix's visual menu
Agent
Web Page
Choose from a selection of destinations & media types
zappix
Skip annoying voice menus when calling customer service
Zappix is now connecting you to the destination that you chose, you might hear tones and prompts during the process
Pick a company to connect with
Request that we add a company you frequently call
Can't find a company

Figure 16 - Zappix UI [11]

Jacada – VISUALIVR

Jacada VISUALIVR [12] solution is another approach to Visual IVR. It is a web-based solution for companies to present to their customers an alternative to traditional IVR calls and menus. Companies use it to build device-responsive and interactive web menus that can be browsed by the customer on their devices, and ultimately be directed to an attendant that receives the entire path the user followed in the VISUALIVR interface, as well as the corresponding user's account information. For the company using this service, templates are made available to build the menus, and it is possible to reutilize the IVR system scripts to generate user interfaces. The user uses it seamlessly, without the need to install any software on the computer or handheld device.

Radish Systems – ChoiceView

This solution is another stand-alone application for iOS and Android. It is similar to CallVU [10], described above. The process begins with a normal phone call. The user then launches the ChoiceView [13] application, which presents options menus and the ability to send files such as documents, video or images. The navigation is done by touch but a speaker voice guide is offered at the same time. It is possible to share documents, video and images between customer and service. At any moment, a call can be made to an attendant that has all the information on the user's account available.

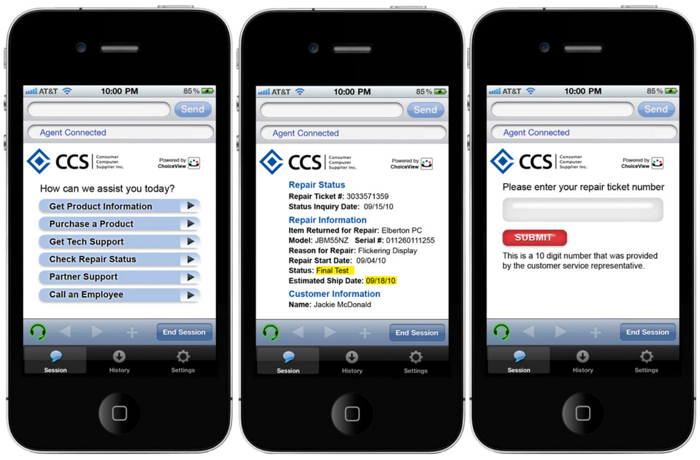


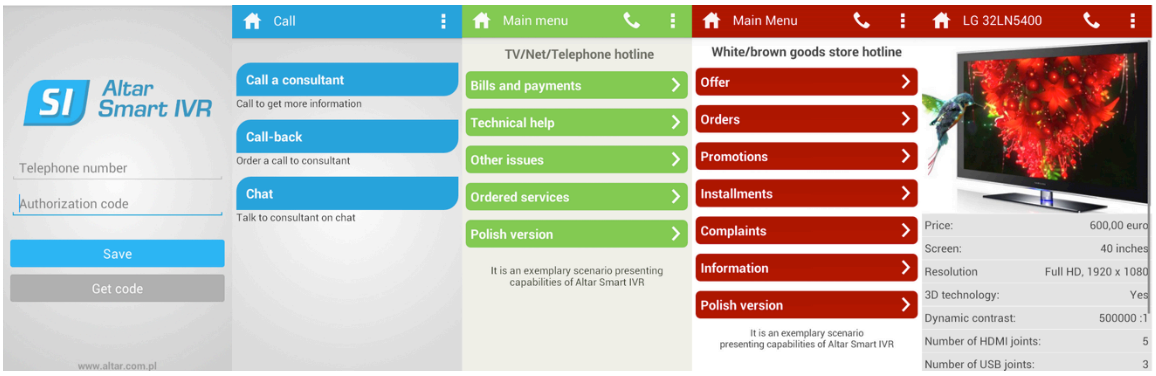
Figure 17 - ChoiceView UI [13]

AT&T – Visual IVR

Developed in the AT&T Labs and Foundry as a proof of concept [14], this solution was presented in 2012. It aims to replace the traditional IVR menus by sending a link to a web page upon the beginning of a call. This web page opens in the device browser and shows the interactive menu where the user can choose the service options as well as change the interaction to a voice call with an attendant. The user identification is done by finding the account associated with the caller's number. This allows the Visual IVR to have the user's data preloaded, thus enhancing the user experience.

Altar – Smart IVR

Like CallVU or Zappix, this Polish solution is a standalone application for Android and iOS that allows users to experience the Visual IVR interface offered by Altar [15]. Available for businesses to add Smart IVR in their already IVR enabled call centers, it offers customer authentication and menu navigating through the options of the service. The information about the client, obtained from the login, is shared with the attendant in the event of a call that can be made at any moment within the Smart IVR mobile application. The option of advertisement placement is available.



Price:	600,00 euro
Screen:	40 inches
Resolution	Full HD, 1920 x 1080
3D technology:	Yes
Dynamic contrast:	500000 :1
Number of HDMI joints:	5
Number of USB joints:	3

Figure 18 - Altar Smart IVR UI [15]

In order to better understand how the above solutions relate to each other a more thorough comparison was made between them. Some of the key aspects that are important to have in the final of the internship were considered:

- **Standalone Application** – Know if the user has to install some software in the phone, and what smartphone operating systems are covered;
- **Web Based** – Understand if the interactivity is done in a web based interface, which means leave the call context to enter the browser application to navigate trough the menus;
- **Send Visual Content** – In which stage of the call is possible to share the visual interactive content. Three stages are defined: Ringing stage, In-call stage, and Off (after the call is disconnected);
- **File Transfer** – The ability to send files in the context of a call;
- **Back Office** - The ability to connect to an external service from the IVR provider to send user inputs and receive context.

The comparison between the solutions above is summed up in the next table (

Visual IVR Solution	Standalone application			Web Based	Send Visual Content			File Transfer	Back Office
	Android	iOS	Windows		Ringing	Call	Off		
CallVU	✔	✔	✗	✗	✔	✔	✔	✗	✗
Jacada	✗	✗	✗	✔	✗	✗	✗	✔	✔
Zappix	✔	✔	✗	✗	✗	✗	✗	✗	✗
Radish	✔	✔	✗	✗	✗	✔	✗	✔	✔
AT&T	✗	✗	✗	✔	✗	✗	✔	✗	✔
Altar	✔	✔	✗	✗	✗	✗	✗	✗	✔

Table 5).

Table 5 – Visual IVR Solutions comparison

There are already some options in the market regarding Visual IVR but they are either standalone mobile apps or web-based solutions that redirect customer to call centers. None of them is integrated with VoIP or RCS (concepts detailed in further sections). The innovation of the solution developed in this internship is the integration of Visual IVR concept as a capability in VoIP calls, more importantly in RCS communications that happen seamlessly to the client. The addition of this capability to the RCS environment could result not only in a better Visual IVR experience but also in the possibility of sharing any kind of interactive content before and during the call.

3.4 Rich Communication Services

The main players in telecommunication networks such as network vendors, operators and device manufacturers got together to create an inter-operator, access-technology independent, rich communication experience to the user [16]. The main focus being the struggle with the OTT applications [17] such as Viber, Skype, Lynk, and many others that offer over-the-top communication channels between users, with the possibility to share multimedia content. Being OTT, these solutions do not bring network providers any profit.

Formerly Rich Communication Suite, this initiative run by GSM Association (GSMA) aims to bring together already defined services into profiles based on global standards. It defines requirements for some service features as well as some architectural facets. The main idea is to facilitate the introduction of commercial, IMS-based communication services for Third Generation (3G), Long-Term Evolution (LTE), Code Division Multiple Access (CDMA) and fixed networks. The functionality within RCS specification [18] include enhanced call trough image and video sharing, enhanced phonebook with presence and service discovery as well as rich messaging including image, video, location, file and emoticons sharing for both mobile and softphone users.

The drive for RCS lies within two main reasons:

- The need to have global interoperability between socially-driven services
- Subscriber-centric social networking (rather than platform-centric)

In the context of this internship these two main features of RCS 5.1 specification [18] are the main focus of study:

- Capability discovery – What the device is capable of receiving and showing (detailed in 2.2.1 Capability Discovery);
- Content sharing – The actual enhancement of the IVR call (detailed in 2.2.2 Extension Service for content sharing).

With that said, it is understandable that an IMS network is needed to integrate and test the AS to be developed. In the next section, a brief introduction to IMS is given and the solutions available are presented and discussed. Furthermore, in the last section, the discussion and decisions made regarding the technologies used to develop the AS to integrate with the IMS network are also presented.

3.4.1 IP Multimedia Subsystem

Described in more detail in 2.4 IP Multimedia Subsystem is a network architectural framework that was designed originally for the evolution of mobile networks with the basic concept behind any mobile network technology: to provide global interoperability between all handsets and all operators worldwide. This purpose was later updated to support other networks and it provides a set of IP-based technologies put in place to allow ubiquitous access to multimedia services from any terminal, mobile, computer or landline [19].

Furthermore, it is designed to offer universal service access, which means that, wherever you are in the world your communication device should provide the same set of services. More importantly, with IMS, this roaming communication happens automatically from the user and developer's point of view. IMS standards handle this complexity.

Other purpose for the creation of IMS networks was the idea of providing a wide range of possibilities for service creation. In other words, two fundamental market needs are addressed with the creation of IMS:

- Standardized services – the idea that the user should be able to reach anyone without having to worry what operator that person subscribes to or the device that they use.
- Innovative/Differentiating Services – Each operator can opt to offer their own services that differentiate the operator in the marketplace, therefore becoming more attractive to the buyer.

IMS History

IMS was originally defined by 3G.IP, a telecommunications forum formed in 1999 [19]. This initial version of the IMS architecture was then carried to the 3rd Generation Partnership Project (3GPP) [16][20]. IMS was included in the 3GPP standardization of mobile 3G systems in Universal Mobile Telecommunications System (UMTS) networks.

The first 3GPP release to introduce IMS was release 5 in the first quarter of 2002 (transition from 2G to 3G networks), which included, for the first time, SIP-based multimedia transactions as well as support for old Global System for Mobile (GSM) and General Packet Radio Service (GPRS) networks.

3GPP standards are structured as releases. The following feature IMS as part of the standardization [16]:

- Release 6 – Enhancements to IMS such as Push To Talk over Cellular (PoC), integrated operation with Wireless Local Area Networks (WLAN), adds High-Speed Uplink Packet Access (HSUPA) and Multimedia Broadcast Multicast Service (MBMS).
- Release 7 – Improvements in Quality of Service (QoS), reduced latency, support for VoIP, High Speed Packet Access (HSPA), Near Field Communication (NFC);
- Release 8 – First LTE release, all IP-network, new radio interface based on Orthogonal Frequency-Division Multiple Access (OFDMA), Single-carrier frequency-division multiple access (SC-FDMA) and Multi-Input and Multi-Output (MIMO);
- Release 9 – WiMAX and LTE/UMTS Interoperability;
- Release 10 – LTE-Advanced with backwards compatibility with LTE (Rel. 8);
- Release 11 – Advanced IP interconnection of services, heterogeneous networks improvements, Coordinated Multi-Point operation (CoMP) and In-device Co-existence (IDC);
- Release 12 – Schedule to be released in March 2015, not yet released at time of writing.

3.4.2 IMS Alternatives

IMS is the standard architectural framework for IP communication services, so it is the obvious choice for the development of this project. Nevertheless, other alternatives are available. Some alternatives existed through the years, but nowadays the only real alternative that can compete with IMS is a peer-to-peer network (P2P).

Peer-to-peer

This is a different approach to the communications architecture. It uses P2P architecture with the use of SIP to control the call session between end points. It is a simpler approach mainly because a central infrastructure between end-points is not needed; users negotiate and communicate directly to each other (in a P2P network). The main advantage is this decentralization, however, telephony companies have to make a profit, and add billing and service severs to the network, which would add a huge bottleneck to the P2P network performance. Another drawback is the complexity in integration with legacy networks. These are the main reasons centralized systems like IMS networks are used.

3.4.3 IMS Solutions

In this section some of the most used and robust, free or open source solutions for implementing a fully functional IMS network are analyzed. When discussing IMS solutions it is important to focus on three core aspects: the presence of IMS core elements such as Call Session Control Functions (CSCF's) and Home Subscriber Server (HSS), the corresponding interfaces and the fact that there is ongoing support and development. In addition it is important to know if there is a supporting community behind the project.

OpenIMS

The Open Source IMS Core [22] is the most used implementation of IMS CSCF's and HSS. It is based upon open source software such as SIP Express Router (SER) or MySQL and delivers interfaces and extra components. The last stable release dates back to 2012, but there is no indication that development has stopped and support is still available.

One other key aspect of OpenIMS is its massive, active, and supportive community.

LittleIMS

This solution [23] provides the IMS core entities CSCF's and HSS and the interfaces needed to interoperate between them. However, neither development nor support are on-going since early 2013, and there is no active community at the time of writing.

Clearwater

Unlike traditional implementations of IMS, project Clearwater [24] was designed with the intention of being deployed in the Cloud. With all core entities available as well as the corresponding interfaces, it as been developed and supported till today.

The big drawback is that it is prepared to run on paid Amazon Cloud servers (Amazon Web Services), as well as the scarce community surrounding this project.

Kamailio

Kamailio [25] is project resulting from OpenSER, SER and SIP Router, in essence this solution can be used as an IMS network, despite the fact that it only provides the CSCF's side and not the HSS. The needed interfaces are provided. It is still in development and there is available support as well as an active community.

IMSZone

It has all the core entities of an IMS network as well as the corresponding interfaces, but development and support stopped in 2009 [26].

The choice of OpenIMS

The table below shows the aspects that were considered at the time a choice had to be made concerning which solution would be used in this project.

	Core Entities		Interfaces	Development	Support	Active Community
	CSCF's	HSS				
OpenIMS	✔	✔	✔	✔	✔	✔
LittleIMS	✔	✔	✔	✗	✗	✗
Clearwater	✔	✔	✔	✔	✔	✗
Kamailio	✔	✗	✔	✔	✔	✔
IMSZone	✔	✔	✔	✗	✗	✗

Table 6 - IMS Solutions comparison

WIT-Software suggested the use of OpenIMS solution for two main reasons: the experience of some colleagues using this tool, which could be of help if problems were encountered in deploying and using it; and, equally important, was the fact that in a previous internship, these various solutions were tested and a machine with the OpenIMS solution deployed and configured was already part of the company's virtual machine catalogue.

All things considered, OpenIMS was the choice made not only because of the reasons above described, but also because of the presence of core entities, interfaces and the large community supporting its use and problem solving.

There is a great experience and tradition of using OpenIMS at WIT-Software. Although it can appear that the choice of using OpenIMS was already made when entering this project, the comparison study between solutions and the justification to use OpenIMS from my part was an important part of the state of the art study of this project.

3.5.1 – Application Server

Among the most important and valuable components of an IMS Network is the AS [27]. They pose as the main difference between IMS and CS Networks. These are used to execute logical IMS services, for example, how services are invoked, how and which services interact with each other or how and when media is delivered.

For the purpose of this internship, a VCE SIP AS should be developed, that will control the flow of a call between two end points and deliver them the HTML files, which will enrich the call.

Several technologies to build telephony and VoIP application servers have been proposed. Regarding SIP Applications Servers stacks, two of them were considered in the preparation stage of this project, so, in the next paragraphs, these solutions are described and compared in order to justify the choice made.

3.5.2 SIP Application Server development tools

The two main solutions referred in the previous section are Sip Servlets and JAIN SLEE. Both provide Java programming language server-side Application Programming Interfaces (API) for deploying network services for SIP communication, but they have very distinct programing models and specifications.

Sip Servlets

Based on the popular HTTP Servlet model but tailored to SIP, a Sip Servlet is a component, normally part of an SIP enabled AS and managed by a Sip Servlet container, as shown in Figure 19 [28]. They interact with clients by receiving requests and sending responses much like HTTP. Currently in version 2.0, it can be integrated with other Java 2 Platform Enterprise Edition (J2EE) components. It is not recommended for more complex services, because it lacks synchronism, with multiple servlets executing request threads accessing the same session object at the same time, the developer has the responsibility for synchronizing the access to those objects as appropriate.

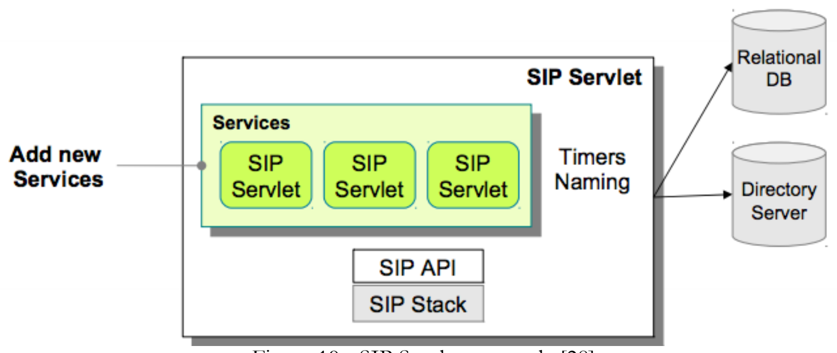


Figure 19 - SIP Servlets example [28]

JAIN SLEE

With a more complex specification than Sip Servlets, JAIN SLEE has standardized a high performing event driven and highly scalable object oriented programming model [28]. It is the specification for a Java Service Logic Execution Environment (SLEE) architecture, currently in version 1.1, it can be the point of integration for multiple network resources and protocols besides SIP. The JAIN SLEE specification allows developers to write robust components as it integrates the well-known ACID (Atomicity, Consistency, Isolation, Durability) properties of transactions into the programming model. Since transactions are handled automatically and concurrency control is part of the specifications, the main focus left for the developer is to handle the events received and produced inside the AS. Figure 20 shows the main components of JAIN SLEE. The most important components to the context of this project are Service Building Blocks (SBB) and Resource Adaptors (RA):

- **Service Building Blocks** are the main components of the SLEE architecture. Each SBB component identifies the event types that are accepted and has handlers or listeners for those events. On the other hand, they can also fire events to other SBB.
- **Resource Adaptors** are used to define how an application running within the SLEE environment interacts with resources. In this specific case, a SIP RA is used. This means that the application is able to interact with the SIP Stack receiving and sending SIP messages.

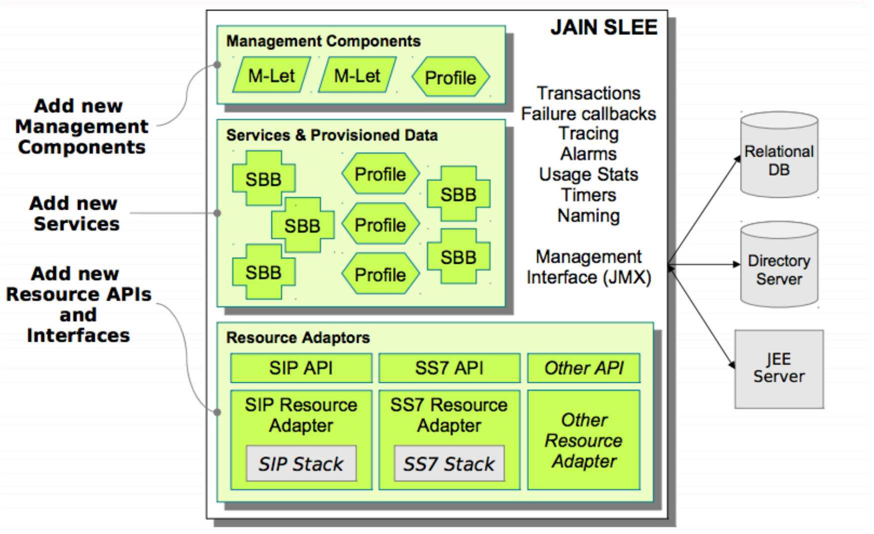


Figure 20 - JAIN SLEE example [28]

Comparison

The following table (Table 7) shows the comparison between Sip Servlets and JAINSLEE frameworks.

	SIP Servlet	**JAIN SLEE**
IMS Related Protocols/Features		
SIP Support	✔	✔
Additional Sip API	✗	✔
Common Sip Components	✔	✔
Service interaction and coordination	✔	✔
Other IMS protocols	✗	✔
Observations & Measurements		
Tracing, Alarms and Usage	✗	✔
O&M interfaces	✗	✔
Extensibility & Migration to IMS		
Non IMS protocols support	✗	✔
No proprietary extensions needed	✗	✔
Service Development		
Promotes good software architecture	✗	✔
Failure model	✗	✔
Concurrency control	✗	✔
Promotes re-use	✗	✔

Table 7 - Comparison between AS development frameworks [28]

Taking all this information in consideration, a choice was made to use JAIN SLEE to develop the SIP AS. Although it has a steep learning curve, and Sip Servlets are recommended for someone with no experience in developing SIP network services the suggestion by WIT to use JAIN SLEE was accepted because it is without a doubt a more robust and flexible tool.

It has all the IMS requirements in place, tracing capabilities, support for other protocols other than SIP and a failure and concurrency control model. Besides that, it promotes good software architecture, promoting reusability of its Object Oriented components. For all these reasons, JAIN SLEE was the choice made because in time, it is more capable of offering scalability and support for adding other components to the service to be developed.

The AS used to deploy the JAIN SLEE application was Red Hat's JBoss. Other alternatives to JBoss exist but they were not considered since Mobicents JAINSLEE was designed to be deployed in JBoss.

Chapter 4
Solution Description

4.1 Objectives

The general objective of this project is to develop a VCE service as well as add an extra capability to the RCS client from WIT-Software (RCS+). This service has the functionality of sharing HTML with the users making a call; this can be done before or during the call (as the use cases detailed in later sections). Besides that it is an objective of this internship to provide VCE capability handling to RCS+. This means changing the application so it can preform the desired actions when receiving HTML through the new capability; showing the HTML, get the values from form fields and be able to send this to the sending party.

This chapter aims to present the methodologies being used to develop the system as well as the requirements; various use cases and the detailed architecture.

4.2 Methodology

For the development of this project a Scrum methodology is being used. Further detail about this methodology is given is this section so the reader can better understand how the workflow of this agile software development methodology is applied.

In order to understand Scrum, the concept of the various roles, events and artefacts has to be explained.

4.2.1 Scrum Roles

The main roles that one can assume during the scrum software development methodology are Product Owner, Scum Master and part of Development Team.

The Project Owner represents the client, ensuring that the development team is adding value to the product.

The Development Team is responsible for the development of the product, preforming tasks for each sprint.

The Scrum Master has the main role of facilitating the development process by removing impediments so the team can meet the goals and product delivery times on schedule.

4.2.2 Scrum Events

Throw-out the process of developing software using Scrum methodology, some events are key to the cohesion of the people involved and quality of the product [38].

Sprint is the basis of the development planning, represents an effort restricted to a time frame. Normally planned from a week to a month, they are decided in team meetings and finalized also in meeting when everything is according to the plan. There are three main types of meetings essential to the Scrum process.

Sprint-planning meetings are done at the beginning of a new sprint, to decide what is to be done, plan the duration and resources needed.

Daily scrum meetings are planned for no more than 15 minutes at the beginning of a work day, are intended to report what was done the day before and potential problems encountered as well as what is the plan for that day. Any impediment that a member of the team sees in achieving a goal must be reported in this daily meeting.

End meetings (Sprint review) are done at the end of each sprint to review the work done, and plan to complete the incomplete tasks if any. The team should reflect about the work done during the sprint and report what went well during the sprint as well as what can be improved in the next ones.

4.2.3 Scrum Artefacts

There many artefacts inherent to the scrum methodology, but two of them are the most important for this project.

Product backlog is a list of requirements for the project/product. This list is maintained and may be altered during the development process It contains the features, bug fixes, non-functional requirements among other project properties. It is, in the simplest manner a list of items to be worked on prioritized by the product owner.

Sprint backlog much like the scrum backlog is a list of items, in this case, tasks that the development team has to accomplish during a sprint.

As a visual aid, the next figure (Figure 21) illustrates the Scrum process for 30 days sprints.

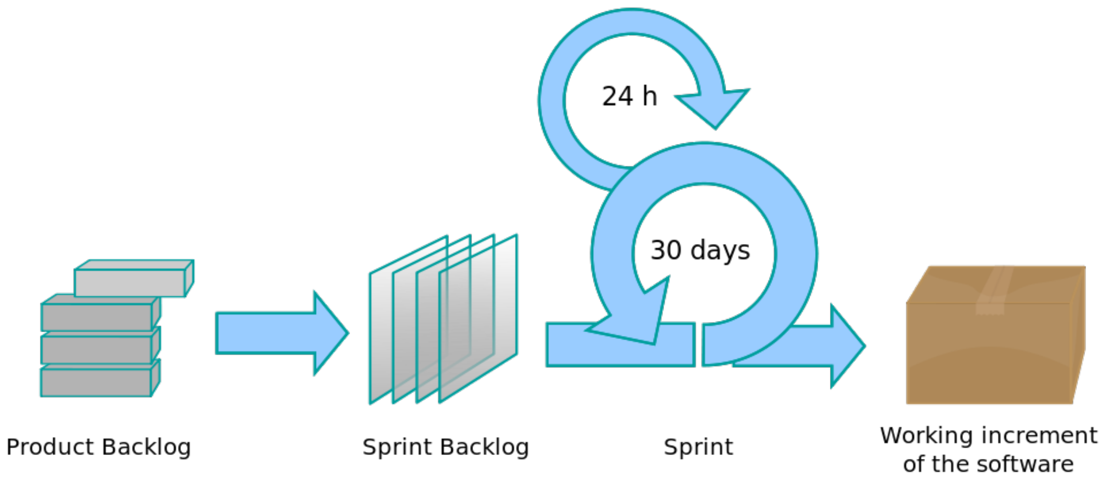


Figure 21 - Scrum process for 30-day sprints [38]

4.2.4 Internship context

For this project, the main roles are assigned to two people. The project owner and scrum master is João Alves and the development team is Marcos Calvo. Since the team is composed of two persons, who work side by side on a daily basis, some adaptation to the Scrum process were made. The daily meeting is done just to report progress on the tasks but trough-out the day, when some problems or doubts are raised, the discussion is prompt, eliminating the need to wait for the next day. Since just one person does the development, the developer does the sprint backlog in the simplest manner as individual notes for each sprint. The duration of the sprints may be different from sprint to sprint, especially in the beginning of the project since some configuration tasks were much simpler than others.

4.3 Solution requirements

In this section, the requirement analysis for this project is presented to the reader. The functional and non-functional requirements, as well as the priority of each one are detailed in Table 8 Table 9 and Table 10. This list can be seen as the scrum backlog of the project since these are the main tasks defined to achieve a working visual call enrichment system. The three tables represent the three main objectives of this internship: environment configuration, AS development and RCS+ client development.

This list of requirements was achieved during project planning and it is the product backlog described earlier. This was achieved by both the development team Marcos Calvo and project owner and scrum master João Alves. Some alterations to this list were made during the development stage, when some of them had to be removed or altered to be more according with the project final objective.

The success or failure of this internship can be well measured by the achievement of this list of requirements. Some are more crucial than others and it is important to have in mind that, during the second semester, if issues are to happen, some of the least important ones can be left aside so the most important ones can be achieved.

Configuration

Description	**Semester**	**Achieved**	**Priority**
Configure OpenIMS	1st	✔	Must-Have
Configure RCS client (RCS+)	1st	✔	Must-Have
Create and configure accounts on HSS	1st	✔	Must-Have
Configure IFC for INVITE Request	1st	✔	Must-Have
Configure IFC for OPTIONS Request	1st	✔	Must-Have

Table 8 - Configuration functional requirements

For the configuration part of the internship planed for the first semester, all the requirements were achieved. The high priority setting of all the requirements are due to the fact that they are crucial to have to start the initial tests with the technologies to use as well as for further development of the AS.

Application Server Development

Each use case shown in the next table is described in the next sections.

Description	**Semester**	**Achieved**	**Priority**
Develop logging system	1st	✔	Nice-to-Have
Call B2BUA system	1st	✔	Must-Have
Capability Discovery System (via	1st	✔	Must-Have

Options Request)			
Integrate 3rd party MSRP stack in AS	1st	✔	Must-Have
Transform VXML to HTML	2nd	✔	Must-Have
HTML share via MSRP	1st	✔	Must-Have
Use case: Originating user receives VCE content from VCE AS during call setup	2nd	✔	Must-Have
Use case: Originating user receives VCE content from VCE AS during on-going call	2nd	✔	Must-Have
Use case: Originating user receives VCE content from VCE AS after the call is terminated	2nd	✔	Must-Have
Receive User interaction with VCE content	2nd	✔	Must-Have
Deliver user input to client's service	2nd	✔	Must-Have
Service configuration and resources upload	2nd	✗	Nice-to-Have

Table 9 - Application Server functional requirements

It is important to notice that some of the requirements are marked as Nice-to-Have, because they are not crucial to the proof of concept that is the objective of this project. Some are merely variants to the share of VCE content at a specified step of the call session.

RCS+ for Android development

All of the RCS+ requirements are high priority requirements, since without accomplishing them, it is not possible to properly demonstrate the service developed. This table compiles the UI alterations to the application as well as the modifications to the communications library from WIT-Software, used in the Android application, COMLib.

Description	**Semester**	**Achieved**	**Priority**
Add new extension tag for VCE capability to COMLib	2st	✔	Must-Have
Develop Java API to user VCE extension service in Android App	2nd	✔	Must-Have
Add new VCE capability functionalities to Android app	2nd	✔	Must-Have
Show HTML within app during call stage (Ringing and In-Call)	2nd	✔	Must-Have
Show HTML within app outside call stage.	2nd	✔	Must-Have

Get information from HTML form	2nd	✔	Must-Have
Compile information in MSRP message	2nd	✔	Must-Have
Send DTMF according to user input	2nd	✔	Must-Have

Table 10 - RCS+ functional requirements

Non-functional Requirements

There are some non-functional requirements that this solution should achieve.

Description	**Achieved**	**Priority**
Standard Protocols use	✔	Must-Have
IMS 11 Compliance	✔	Must-Have
RCS 5.1 Compliance	✔	Must-Have
High-Throughput	✗	Must-Have
All the are free or Open-Source	✔	Must-Have

Table 11 –Non-Functional requirements

4.4 Use Cases

There are three major use cases inherent to this project:

- VCE content shared from AS to calling party during ringing stage;
- VCE content shared from AS to calling party after call is established;
- VCE content shared from AS to calling party after the call ended;

In the further sections, each one of these use cases is described in detail and exemplified with a flow chart between the actors involved.

4.4.1 VCE content during Ringing Stage

One of the use cases of this POC is to share VCE content before the call is established. The idea behind this is to get call context before call is answered, for example, get the user customer number so the call has context.

The user places a call, the VCE AS detect a call to a subscriber of Visual IVR system. The VCE AS generates HTML and sends the VCE Content to the caller before the IVR answers the call.

The next diagrams (Figure 22 and Figure 23) show the flow of requests and responses between the two main actors, calling user and VCE AS.

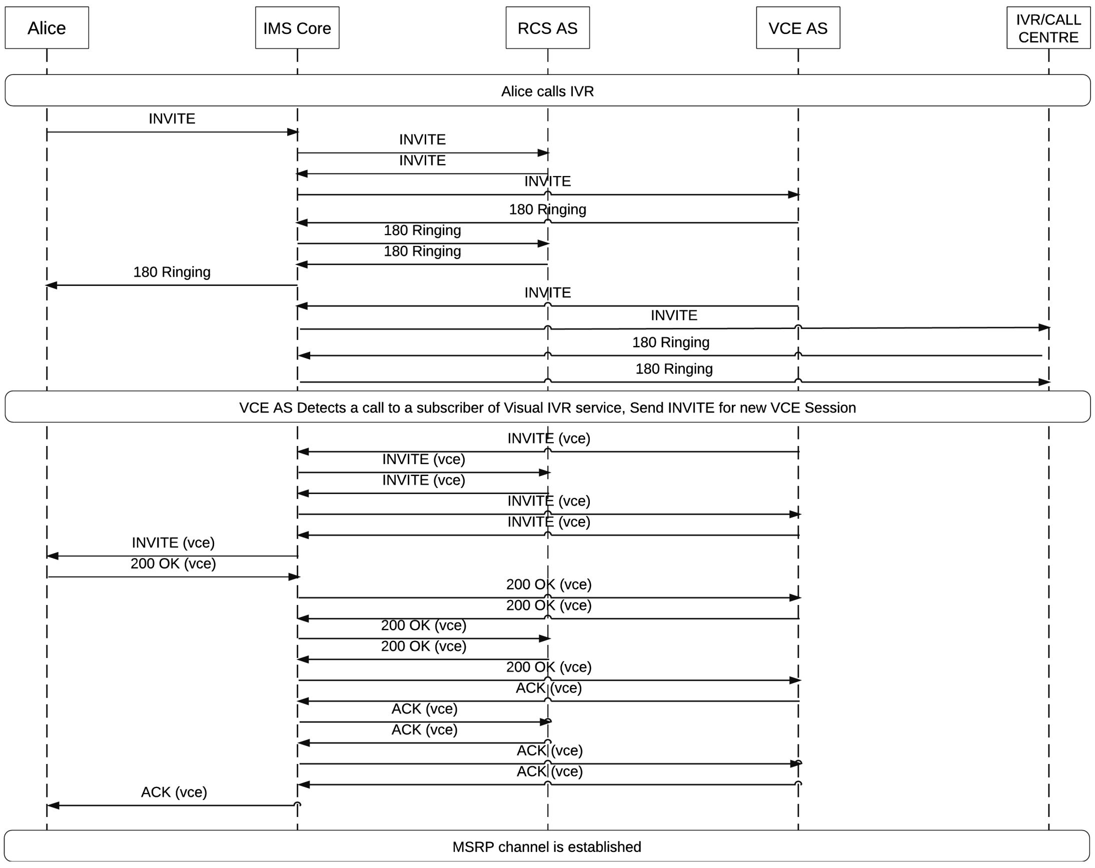


Figure 22 - VCE content share between RCS client and VCE AS during call setup (part 1)

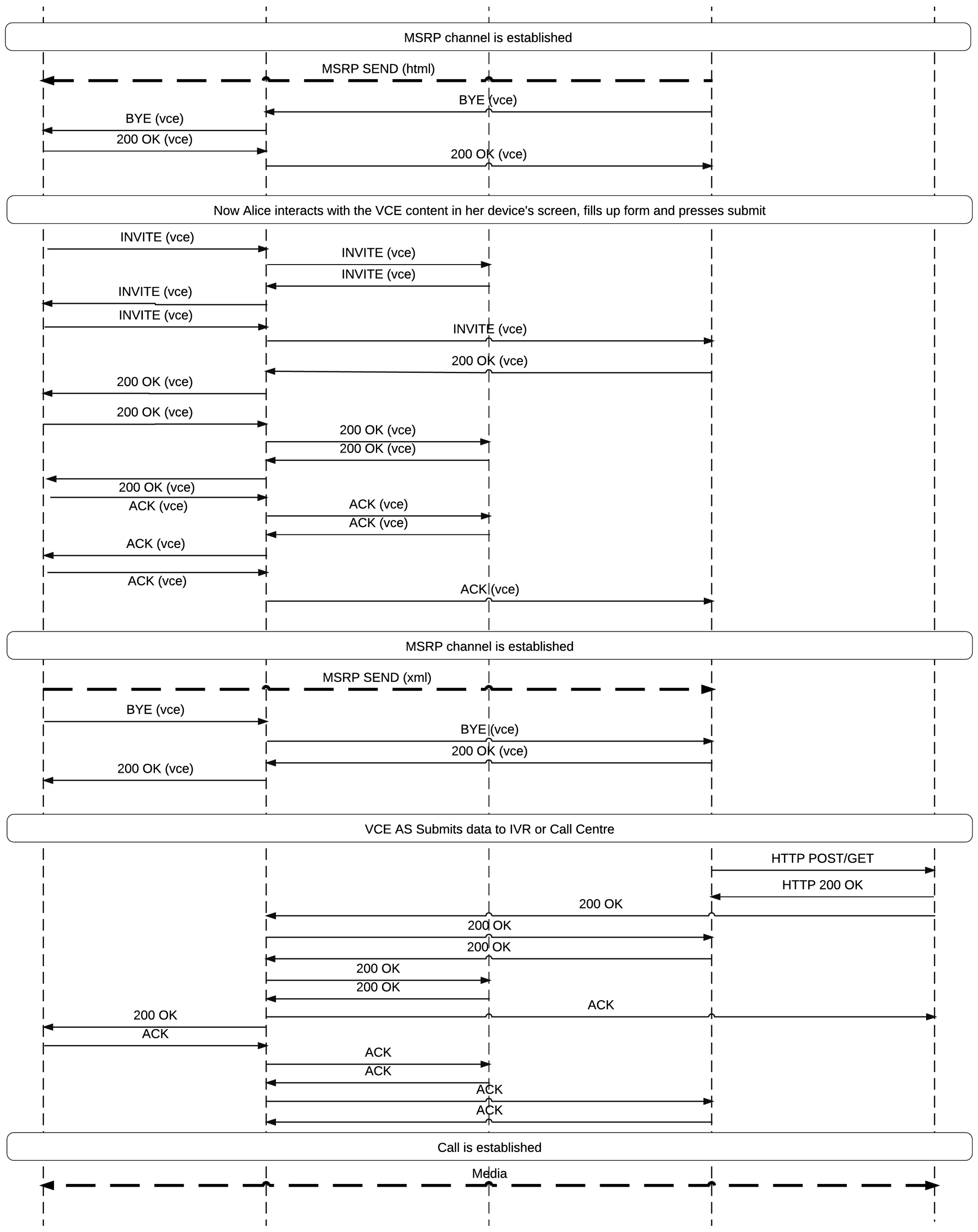


Figure 23 - VCE content share between RCS client and VCE AS during call setup (part 2)

4.4.2 VCE content shared from AS to calling party after call is established

In another embodiment, with VCE AS as producer of the content, a series of options can be sent to the called party as part of an HTML. In this example, without going in too much detail, Alice calls IVR and after call is established, VCE AS shares content with option to Alice through MSRP channel established between VCE AS and Alice. Alice choses the option correspondent to DTMF digit 1 shares this feedback with the AS as a XML. The VCE AS gets next menu and sends it back to Alice. Alice choses another option and this interaction goes on until the Visual IVR flow ends or the service is cancel by Alice.

Figure 24 and Figure 25 show the process of the case described above with more detail, with all the messages exchanged between actors included above.

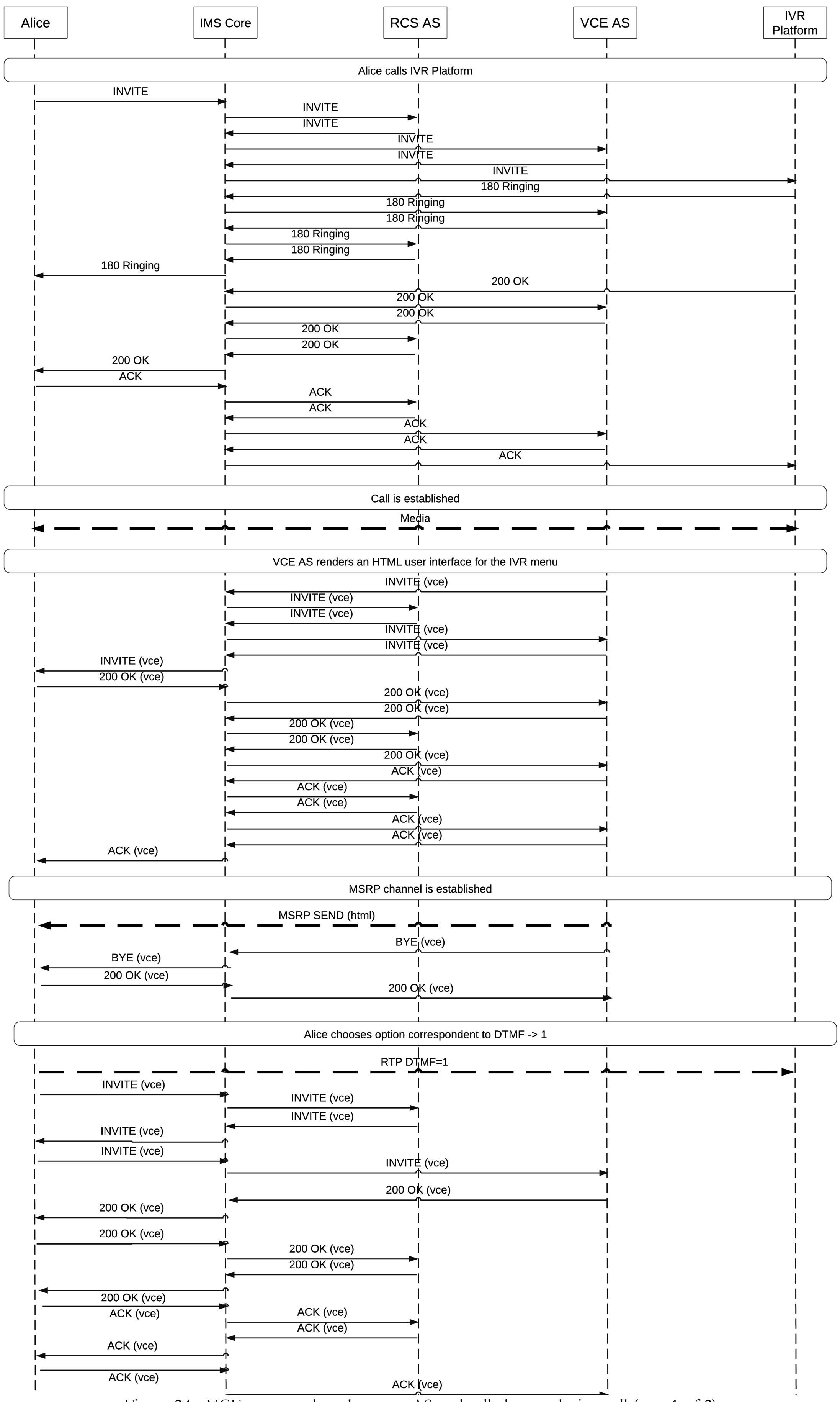


Figure 24 - VCE content share between AS and called party during call (part 1 of 2)

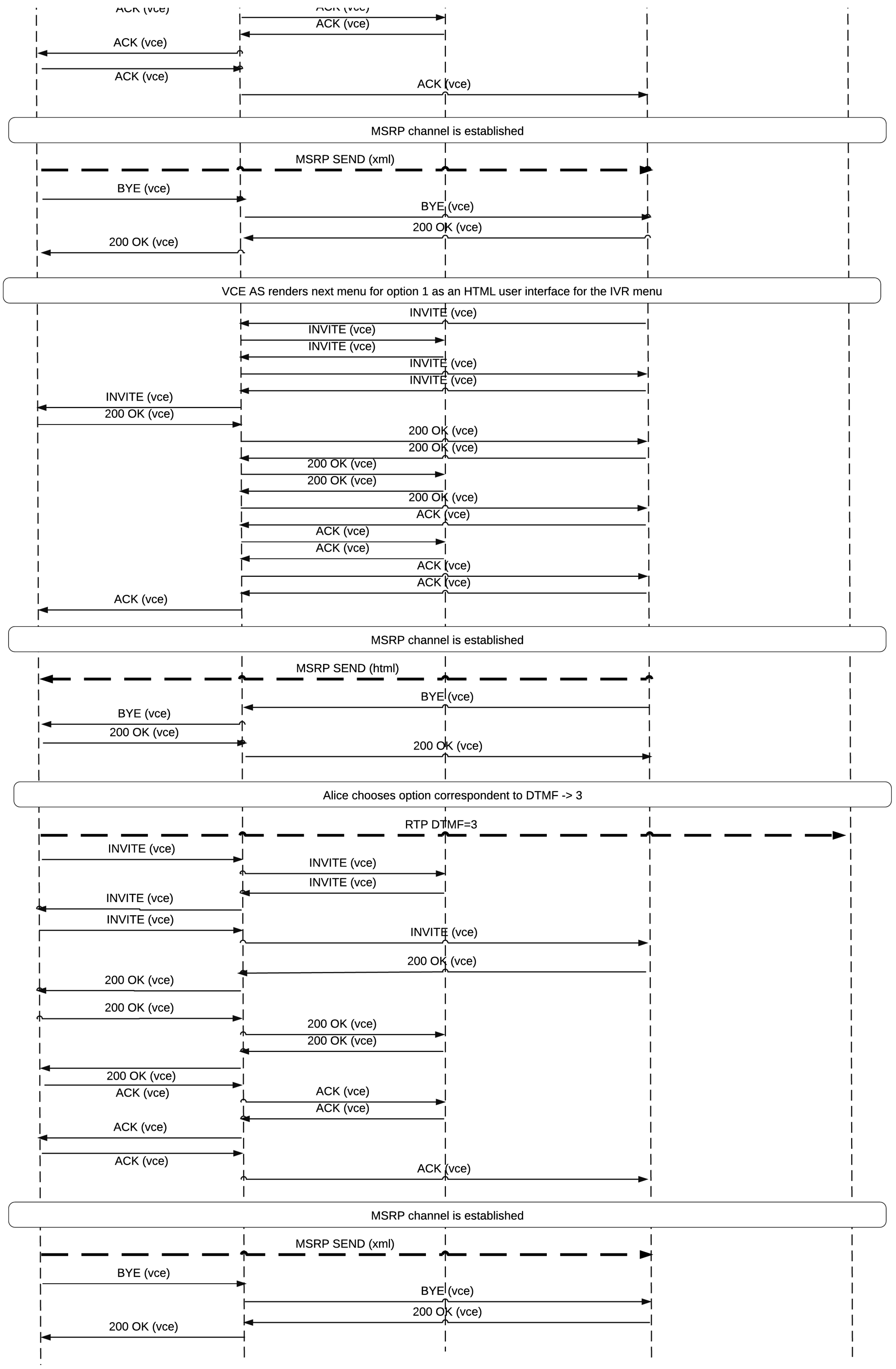


Figure 25 - VCE content share between AS and called party during call (part 2 of 2)

4.4.3 VCE content shared from AS to end-point outside call context

Finally in this final embodiment of the system, the case in which the VCE Content is shared outside call context is presented.

In this case, a user has just finished the call and the IVR Platform, subscriber of the Visual IVR service, has an option to send VCE content after the call is finished. In this case, a form to get information about the quality of the service the user has just experienced in the recently disconnected call.

The VCE As will share as normal the VCE Content and then, upon receiving the form data, submit it to the IVR application service.

The next diagrams (Figure 26 and Figure 27) show the flow of information between Alice, VCE AS and IVR platform for this use case.

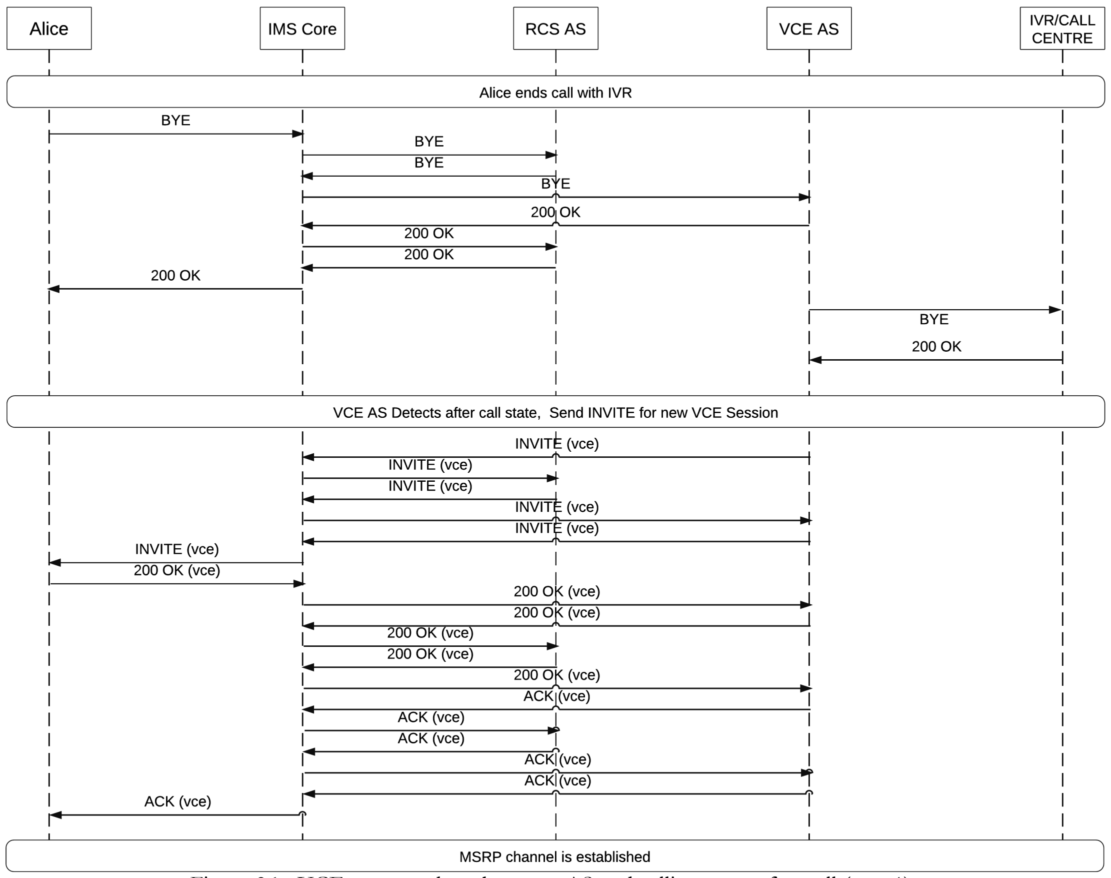


Figure 26 - VCE content share between AS and calling party after call (part 1)

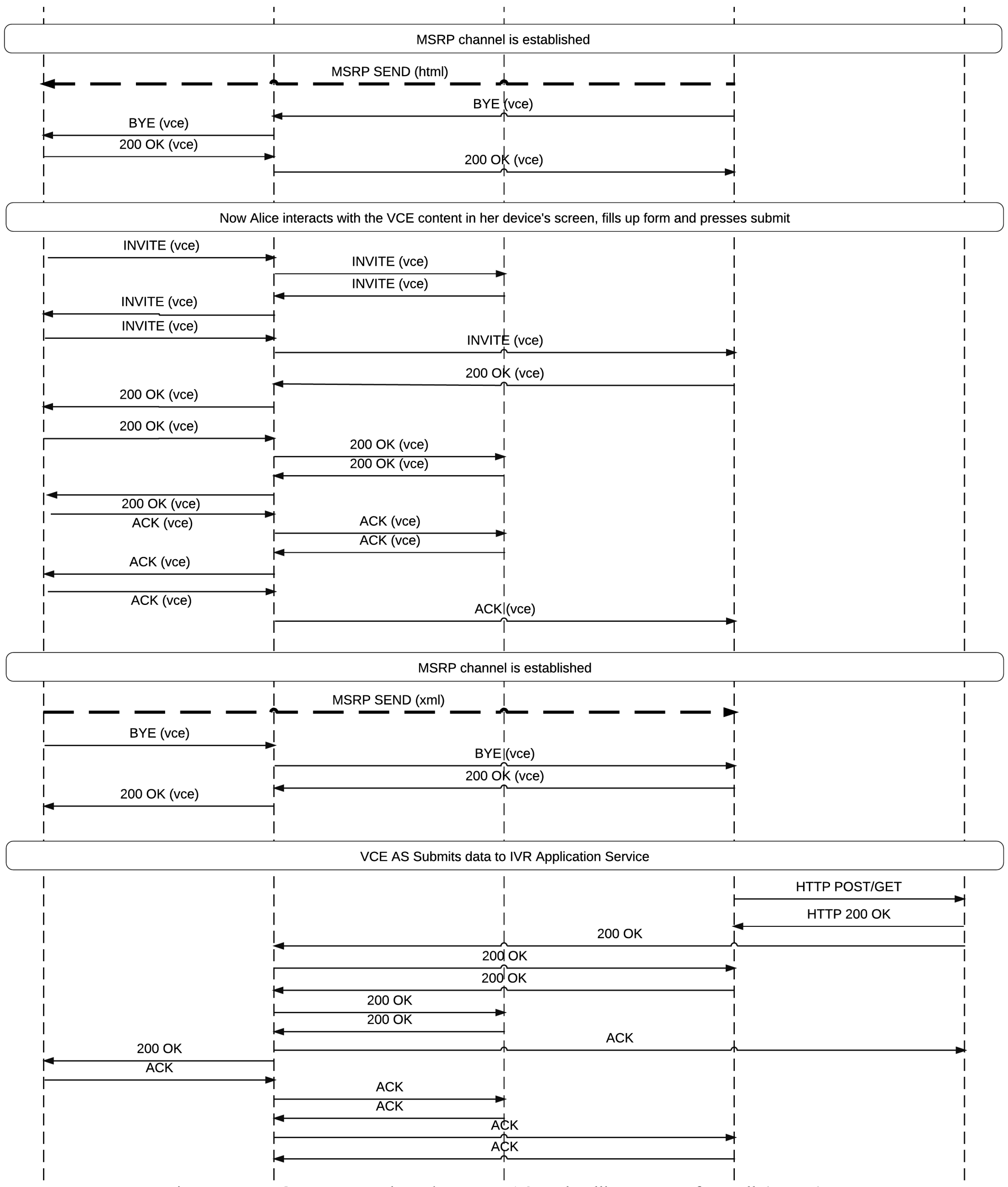
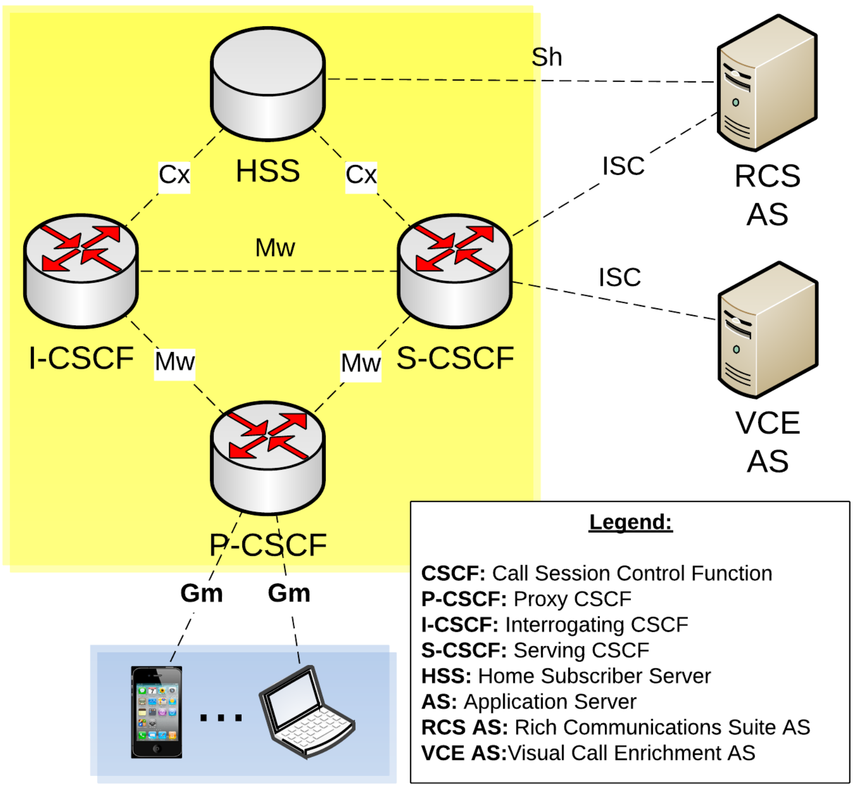


Figure 27 - VCE content share between AS and calling party after call (part 2)

4.5 High Level Architecture

The following diagram (Figure 28) illustrates the required high-level architecture.



Legend:

CSCF: Call Session Control Function
P-CSCF: Proxy CSCF
I-CSCF: Interrogating CSCF
S-CSCF: Serving CSCF
HSS: Home Subscriber Server
AS: Application Server
RCS AS: Rich Communications Suite AS
VCE AS: Visual Call Enrichment AS

Figure 28 - High-level architecture

As described in previous sections, this RCS solution, as most of them, is deployed over IMS architecture. One or more application servers deployed in the IMS Application Layer offer the RCS functionality. For the sake of simplicity, in this case, just one AS (RCS AS) is responsible for all RCS functionalities. All the interfaces present in the diagram were already described in detail in section 2.4.2 Core Network.

The main focus of this project is the development of the VCE AS present in the diagram above, which orchestrates the visual interactive content sharing between end-points and services. This AS architecture and description in presented in the next section.

4.6 Visual Call Enrichment Application Server

Responsible for handling the call flow, VCE capability exchange and VCE content, this AS, to be developed during the course of the internship, will be deployed in the IMS application layer.

In order for it to receive the requests from the client's end-points, it needs to be in the signaling path. The main purpose is for the AS to be aware of the end-points capabilities and control the call being set-up or established by controlling the incoming requests and outgoing responses (INVITES, BYE, CANCEL, 200OK, 180 RINGING, etc.…). As described in section 2.4.3 Applications in IMS of this document, IMS architecture provides the IFC service, and in this case, it was used to add the VCE AS to the signaling path thus redirecting the specific Requests through the IP of the service (VCE AS).

Besides call control, the system is split in two distinct phases, capability discovery and visual content transfer.

The IVR or Call centre menu structure shall be defined in one or more VXML files, these should be then transformed using XSL files into the HTML pages to be shared through the VCE Share service.

4.6.1 Capability Discovery

As earlier described, in section 2.2.1 Capability Discovery, RCS 5.1 specification [18] defines mechanisms to exchange capabilities between RCS enabled end-points. With the process understood, the specific case of the new VCE capability is now explained in detail.

An endpoint, which is able to process interactive visual content, adds the VCE tag when exchanging SIP OPTIONS messages in the capability discovery process. This tag is a new tag, using the extension feature of RCS, and should have the following value:

```
urn%3Aurn-7%3A3gpp-application.ims.iari.rcs.ext.vce
```

The VCE AS should intercept this process of capability exchange. For this to happen, it must be in the signaling path, as part of the capability discovery process, and be aware of end-points VCE capabilities. For this to happen, a triggering point should be configured in the IFC of the User Profile, as described in section 2.4.3 Applications in IMS. This way, during registration, the User Profile is downloaded by the S-CSCF from HSS and becomes responsible of redirecting requests to the appropriate AS.

Figure 29 shows the capability discovery process with the new VCE tag in detail.

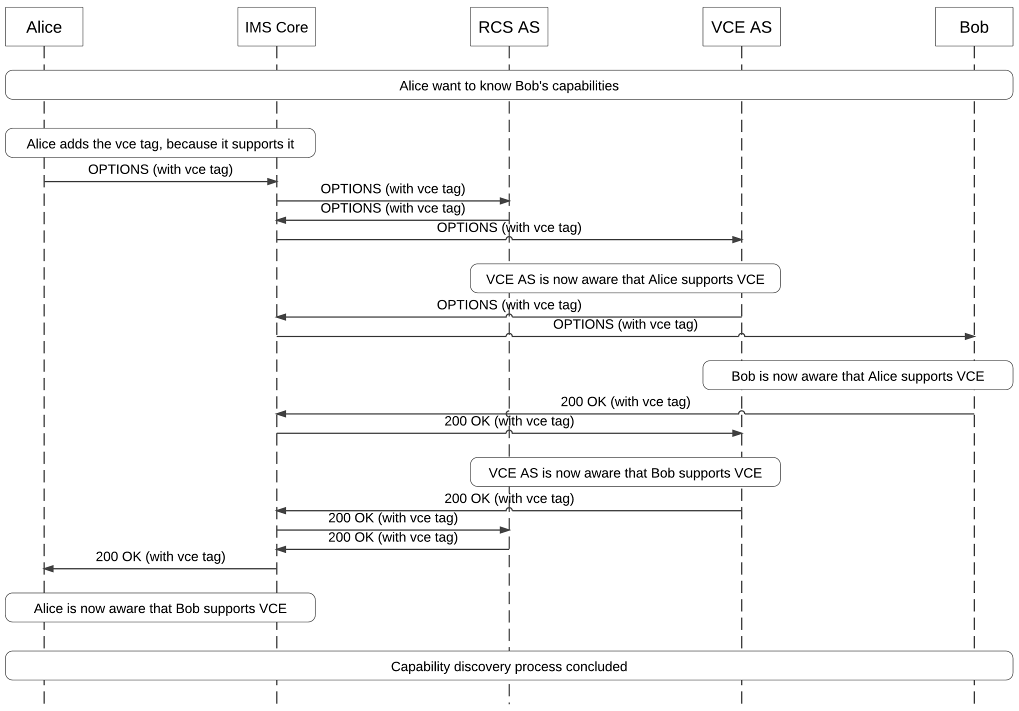


Figure 29 - VCE capability discovery process

4.6.2 Visual content transfer

The second important process is related with the main objective of this internship, to share interactive content between producer and consumer.

This process can only happen after the capability discovery process described in the previous section is complete, this way, both producer (VCE AS) and consumer (RCS+) are aware of each other's capabilities.

This visual call content can be produced either by an end-point (e.g. RCS client application) or by the VCE AS, so for the sake of simplicity, in this process we should only talk about producer and consumer. The process is the same, either it is an AS or an end-point who produces content.

In the case of the AS, the content is produced by transforming the correspondent VXML, with the id of the menu to share, into an HTML file. This transformation is done using a XSL file with the various templates to transform the various menus and forms.

In any phase of a call session, the producer of VCE content can send an SIP INVITE to invite the consumer to a VCE exchange. The SIP INVITE message contains the `vce` tag in the `Accept-Contact` header and the `Content-type` header should have the value `application/sdp`. In this case, SDP will describe a MSRP session with the type `text/html or text/xml` since the transfer type will be HTML or XML through a MSRP session.

Next, if the receiver accepts the session with a 200 OK SIP response, it then requests the VCE content by sending a HTTP GET through the MSRP channel. The producer should then answer with the HTML content that is, upon received, rendered in the consumer RCS application.

The process ends with the receiver interacting with the content and sending XML feedback through a new MSRP Session.

As a visual aid, the process above described, can be seen in the following flow diagram.

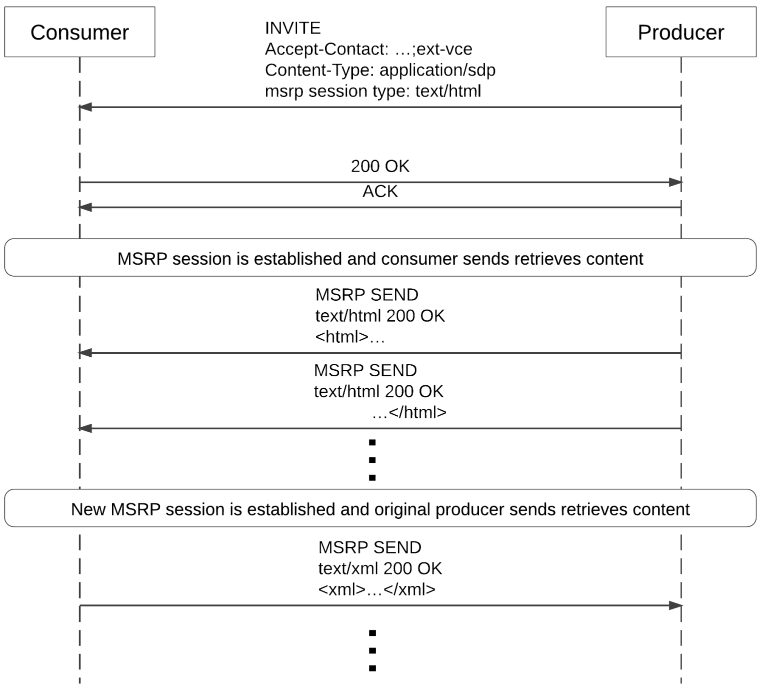

Consumer
Producer
INVITE
Accept-Contact: …;ext-vce
Content-Type: application/sdp
msrp session type: text/html
200 OK
ACK
MSRP session is established and consumer sends retrieves content
MSRP SEND
text/html 200 OK
<html>…
MSRP SEND
text/html 200 OK
…</html>
New MSRP session is established and original producer sends retrieves content
MSRP SEND
text/xml 200 OK
<xml>…</xml>

Figure 30 - VCE content transfer through MSRP channel

4.6.3 Architecture

The architecture overview of the VCE AS can be seen in Figure 31, it was designed so it follows the three main functionalities of the desired final service: capability exchange, call control and visual content transfer.

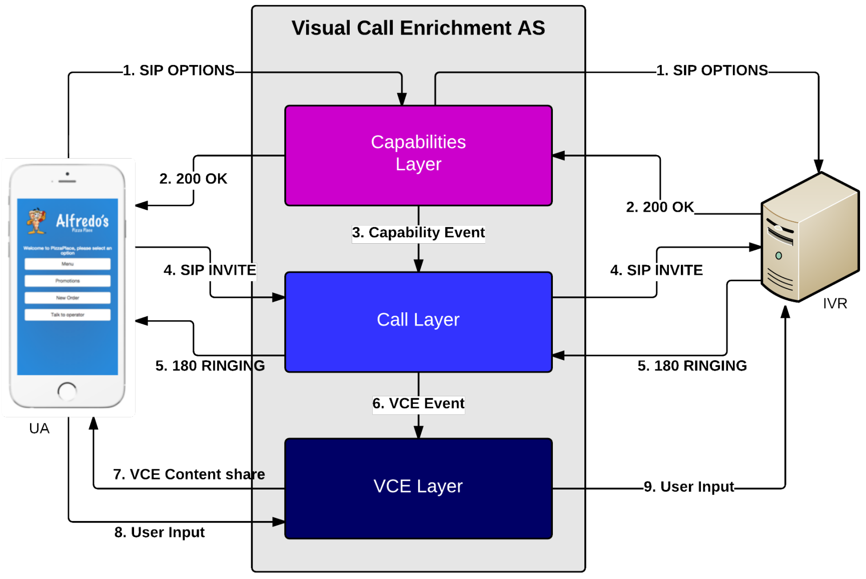


Figure 31 - VCE AS High-Level architecture

As the reader can easily understand by the diagram above there are three major layers in the AS architectural design: Capabilities SBB, Call SBB and MSRP SBB. These SBB's have the following functions:

- **Capabilities Layer** is responsible for the capability discovery process (detailed in 4.5.1). This means the exchange of SIP OPTIONS messages and following responses containing the RCS tags for the capabilities of each end user.
- **Call Layer** is responsible for acting as B2BUA handling SIP requests and responses related to call session establishment and termination. It is also responsible for identifying and separating call events from VCE content events.
- **VCE Layer** is responsible for establishing MSRP session between AS and clients and share the VCE content. It is also responsible for receiving the user input and send it to the IVR System.

The message exchange between end points and VCE AS are numbered from 1 to 9, identifying a possible flow of communication. A brief explanation of each one of the seven steps is given so the reader can better understand what is shown in the diagram (Figure 31):

1. SIP OPTIONS message, part of capability discovery process, is sent from UA to IVR, as the IFC has a trigger for OPTIONS messages to be redirected to the VCE AS so it can store the UA capabilities. Finally Capability layer forwards the request to its destination.
2. IVR responds to OPTIONS message with a 200 OK response message indicating that the capabilities were received and stored. The 200 OK message contains the capabilities of IVR and follows the route stored by the OPTIONS message passing again through VCE AS and Capability layer. The SBB gets the capabilities of IVR and forwards the response to UA.
3. A capability event is fired by the Capability layer and handled by the Call layer. Call layer checks if the capability event refers to the users that this SBB is assigned to and saves/updates the capabilities of each user. Only the VCE capability is taken into account.
4. In this fourth step, UA sends a SIP INVITE request to IVR, thanks to the trigger point in IMS IFC, the message if redirected to VCE AS. Call layer has the INVITE handler and acts as a B2BUA if the purpose of the INVITE is a call establishment handling all the requests and responses of that call.
5. 180 RINGING SIP response code is sent from IVR to AS and forwarded to UA1
6. If the purpose of the INVITE in step 4 is to share VCE content, a VCE event is fired from Call layer and handled in VCE layer that has the listener for this kind of event.
7. Upon a VCE event received, VCE layer establishes the MSRP session with RCS app (depends on the use-case) and shares VCE Content (e.g. HTML) with it.
8. At the RCS+ App the user input is collected and sent, in a form of a VCE Content Share handled by the VCE AS. The Call layer receives the INVITE and fires a new VCE Event to the VCE layer.
9. VCE layer uses this information to make the next step (depends on the use-case) and, it it's the case deliver information to IVR.

4.7 Technologies

All of the technologies chosen until this point are presented in the next table (Table 12). It is important to notice that if some of the possible setbacks are verified, some of the choices can change during the second semester.

Technologies	
Network Architecture	IP Multimedia Subsystem
IMS Core	OpenIMS
IMS Home Service Subscriber	OpenIMS-FhoSS
Application Server Development	Mobicents JAIN SLEE
RCS Client	WIT Software – RCS+ for Android
Communications Library	WIT Software – COMLib 3.6
IVR menus specification	VXML - VoiceXML
VXML validation	XSD - XML Schema
VXML to HTML transform	XSLT
Protocols	
Signaling	Session Initiation Protocol
Session Description	Session Description Protocol
Transport	Real-Time Transport Protocol
VCE Content Share	Message Session Relay Protocol

Table 12 - Technologies and protocols to be used in internship

4.8 Prototyping

The prototyping done in the first semester is shown in this section. The first part of the internship had the objective of studying the state of the art as well as study of available and chosen technologies; the final part was dedicated to exercises, tests and prototyping the AS. The full list of tasks related to testing and prototyping completed in this first semester are shown below:

- Create and deploy a JAIN SLEE Project with all the necessary dependencies and Resource Adaptors (SIP RA and Jain SIP)
- Create an IFC in the HSS on the OpenIMS network to trigger when a new Invite Request is made.
- Change the project in order to handle the SIP INVITE.

- Alter the project, in order to handle all the necessary SIP messages, with the objective of making a call. The AS was made already as a B2BUA AS so the code can be used later. (X-Lite and Zoiper for Mac OS X).
- Start using WIT Software RCS+ for Android RCS client to add the capability discovery process to the AS.
- Configure a new IFC to handle SIP OPTIONS
- Create a new service and SBB in the project, in order to handle SIP OPTIONS Request.
- Retrieve capabilities from OPTIONS Request, and consequent Response (200OK).
- Create and send a Custom Event (Capability Event) to all Call layer responsible for handling call flows.
- Make all the necessary changes in the project, and in the OpenIMS, in order to fully support RCS capabilities.
- Integrate a 3rd party MSRP stack in the project, and all the dependencies associated with it.
- Use 3rd party MSRP stack to establish connection between AS and RCS client (not completed)

Chapter 5
Development

In this fifth chapter it's intended to explain and give detail about the actual technical work that was done, mainly during the second semester of the internship. The main aspects of development presented in this chapter relate to the work environment, and the various areas that the development was made, namely the IMS network, the Application Server, COMLib and RCS+ Android app.

5.1 Development Environment

In order to better understand the conditions in which the work was developed, this subsection presents detailed information about the tools and machines used in the internship.

In regard to the hardware, the author used a laptop, a virtual machine and a smartphone provided by the company. It's important to specify the hardware used in the project so the reader can better understand the conditions in which the development was done and also to understand some of the limitations or constraints that can be a factor in the development stage.

5.1.1 Main machine

The main machine used to develop this project was an Apple MacBook Pro laptop with the specifications described below. This machine was used develop all the components.

Component	Resource
Model	Apple MacBook Pro 13-inch, Late 2011
CPU	2.4 GHz Intel Core i5
Memory	8 GB 1333MHz DDR3
Storage	500 GB SATA Hard Disk Drive
Operating System	Mac OS X Yosemite Version 10.10.3

Table 13 - Main machine specifications

5.1.2 Virtual Machine

Other hardware component given by the company was a virtual machine to with the required configurations and installations of OpenIMS Core and JBoss so the AS could be deployed run and tested by the author. This way, the development of the AS obeys to the initial architecture of the system in which the AS can be deployed and installed in an IMS network as a single machine. The great advantage of this approach is that, in this manner, it's easier to integrate with a IMS network with minor modifications. This machine had the specifications below.

Component	Resource
CPU	2.83 GHz Intel Xeon X3363
Memory	4 GB 1333MHz DDR3
Storage	16 GB
Operating System	Ubuntu Linux 10.04 TLS

Table 14 - Virtual machine specifications

In this machine, the API to develop the AS, described in 3.5.2, had to be installed and the IFC configured as explained in the next section.

5.1.3 Smartphone

The final hardware component used by the author to develop and test this solution was a smartphone. This smartphone served as the main tool to test the Android development done in the WIT-Software's RCS+ application.

Component	Resource
Model	HTC Sensation XL
Chipset	Qualcomm MSM8255 Snapdragon S2
CPU	1.5 GHz Scorpion
Memory	768 MB
Storage	16 GB
Operating System	Android OS, v4.0 (Ice Cream Sandwich)

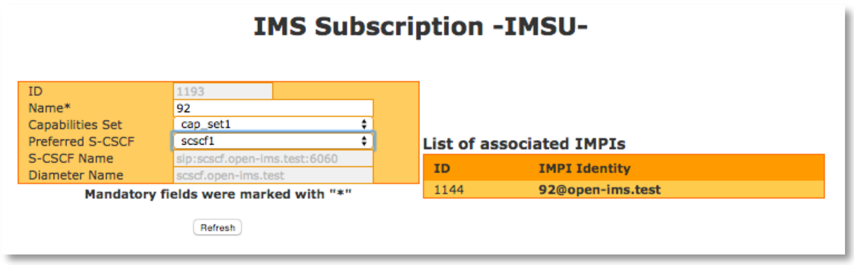
Table 15 - Smartphone specifications

5.2 IP Multimedia Subsystem

Beginning at the most crucial component required for the development of this solution, some work had to be done in the configuration IMS Core (detailed in 2.4), namely the creation of user identities in the HSS and definition of the IFC. As described earlier each user can have various IMPU, which are associated to a IMPI in the HSS. The creation of these user identities is detailed in the next section.

5.2.1 User creation

There was the need of creation of users, in this case two users, one to represent the calling party (client) - 91 - and another to represent the called party (IVR System) - 92. In other to create these, OpenIMS Core has a BackOffice tool to simplify the process. The next image show the UI and configurations of a user subscription (IMSU). Here the name, capabilities set and preferred S-CSCF were defined as can be seen in the image.



IMS Subscription -IMSU-

ID	1193
Name*	92
Capabilities Set	cap_set1
Preferred S-CSCF	scscf1
S-CSCF Name	sip:scscf.open-ims.test:6060
Diameter Name	scscf.open-ims.test

Mandatory fields were marked with "*"

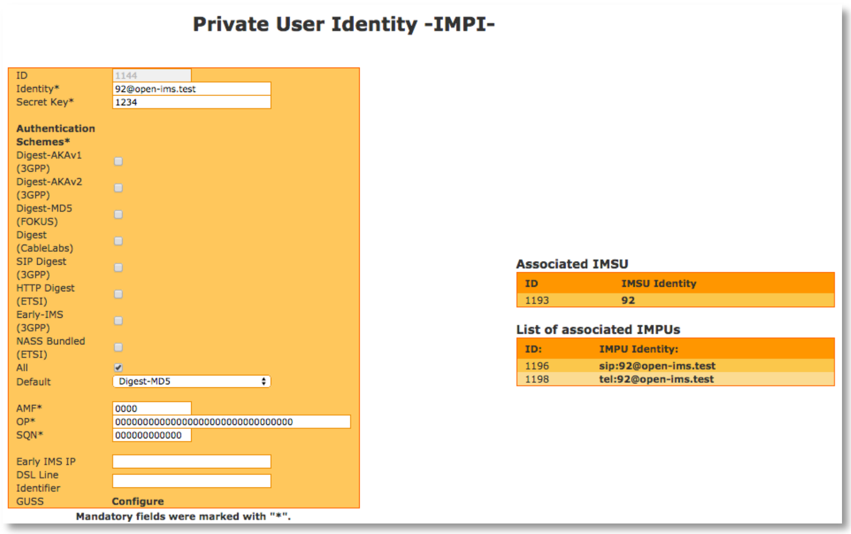
Refresh

List of associated IMPIs

ID	IMPI Identity
1144	92@open-ims.test

Figure 32 - IMS Subscription of user 92

This IMSU is associated to one IMPI, detailing the private information of the user account. Thus IMPI was defined for each user in an UI provided by the OpenIMS Core as can be seen in Figure 33.



Private User Identity -IMPI-

ID	1144
Identity*	92@open-ims.test
Secret Key*	1234
Authentication Schemes*	
Digest-AKAv1 (3GPP)	☐
Digest-AKAv2 (3GPP)	☐
Digest-MD5 (FOKUS)	☐
Digest (CableLabs)	☐
SIP Digest (3GPP)	☐
HTTP Digest (ETSI)	☐
Early-IMS (3GPP)	☐
NASS Bundled (ETSI)	☐
All	✓
Default	Digest-MD5
AMF*	0000
OP*	00000000000000000000000000000000
SQN*	000000000000
Early IMS IP	
DSL Line Identifier	
GUSS	**Configure**

Mandatory fields were marked with "*".

Associated IMSU

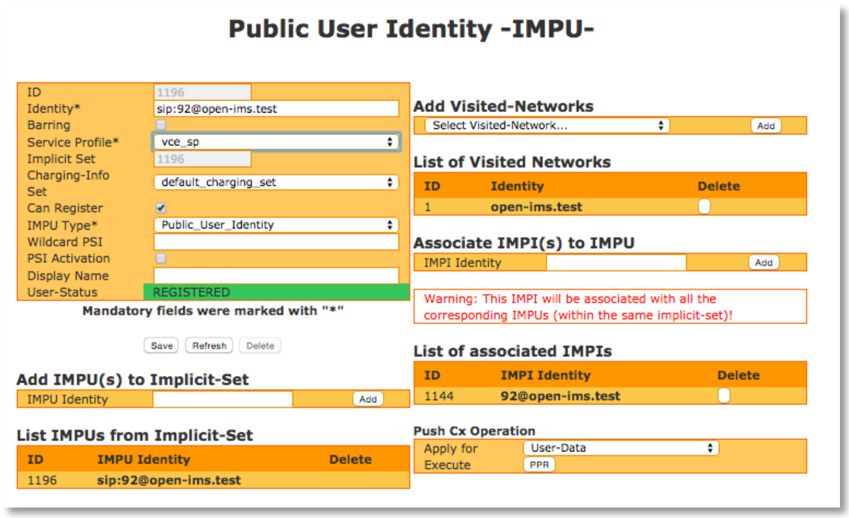
ID	IMSU Identity
1193	92

List of associated IMPUs

ID:	IMPU Identity:
1196	sip:92@open-ims.test
1198	tel:92@open-ims.test

Figure 33 - IMPI definitions for user 92

Here, the main aspects for creating the private identity – `92@open-ims.test` - and secret key were defined. This authentication was used to authenticate users in the RCS+ App. To this IMPI a set of IMPUs are associated, `sip:92@open-ims.test` and `tel:92@open-ims.test`. The process of creating these IMPUs is described below in Figure 34.



Public User Identity -IMPU-

ID	1196
Identity*	sip:92@open-ims.test
Barring	☐
Service Profile*	vce_sp
Implicit Set	1196
Charging-Info Set	default_charging_set
Can Register	✓
IMPU Type*	Public_User_Identity
Wildcard PSI	
PSI Activation	☐
Display Name	
User-Status	REGISTERED

Mandatory fields were marked with "*"

Save Refresh Delete

Add IMPU(s) to Implicit-Set

IMPU Identity Add

List IMPUs from Implicit-Set

ID	IMPU Identity	Delete
1196	sip:92@open-ims.test	

Add Visited-Networks

Select Visited-Network... Add

List of Visited Networks

ID	Identity	Delete
1	open-ims.test	☐

Associate IMPI(s) to IMPU

IMPI Identity Add

Warning: This IMPI will be associated with all the corresponding IMPUs (within the same implicit-set)!

List of associated IMPIs

ID	IMPI Identity	Delete
1144	92@open-ims.test	☐

Push Cx Operation

Apply for	User-Data
Execute	PPR

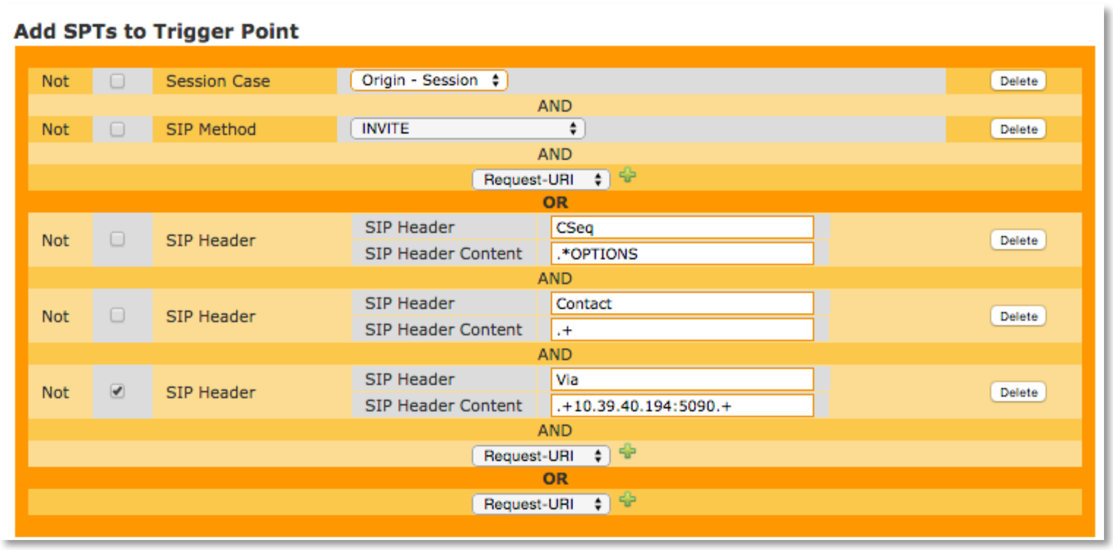
Figure 34 - IMPU definitions for user 92

As can be seen in the figure above, the IMPU defines user identity and the service profile associated with it, in this case, the service profile points to `vce_sp,` the service profile of our AS explained in the next section about the IFC configuration.

5.2.2 Initial Filter Criteria

With users created, the IFC had to be created. As explained in section 2.4.4, IFC is an XML based format to describe the set of rules that represent the provisioned subscription of a user to an application, in this case, the subscription of users 91 and 92 to the VCE AS part of the service profile vce_sp.

The figure below shows the set of rules implemented in this IFC.



Add SPTs to Trigger Point

Not	☐	Session Case	Origin - Session		Delete
AND					
Not	☐	SIP Method	INVITE		Delete
AND					
			Request-URI		
OR					
Not	☐	SIP Header	SIP Header	CSeq	Delete
			SIP Header Content	.*OPTIONS	
AND					
Not	☐	SIP Header	SIP Header	Contact	Delete
			SIP Header Content	.+	
AND					
Not	✓	SIP Header	SIP Header	Via	Delete
			SIP Header Content	.+10.39.40.194:5090.+	
AND					
			Request-URI		
OR					
			Request-URI		

Figure 35 - Trigger point rules in the VCE IFC

Two rules were created, one to select either dialogs relating to new INVITE requests or dialogs relating to OPTIONS requests.

INVITE Request

The AS needs to be in the path of the SIP signalling on order to have control over the call being established as well as know the state in which the call is. The way to do that, is to add a rule that reroutes any new INVITE requests, this way, any INVITE that represents a new session, call or other, is routed trough the VCE AS.

OPTIONS Request

As described earlier, the AS needs to be also in the path of the capability discovery mechanism. This is achieved with the second rule implemented in the IFC. This rule has a more complex set of conjunctive rules due to the fact that OpenIMS has a bug in the creation o trigger point rules. In theory, it should be build just like the previous one, but the SIP Method that is in the options is called OPTION instead of OPTIONS and it does not work. So, a set of rules was created to have the same effect. The first selects requests with the header `CSeq = ".* OPTIONS"` this way, any request that is related to a OPTIONS Request is routed through the VCE AS. But, this is not enough due to the fact that, the Contact header has to be present and with some content as well as the fact that the AS is not in the `Via` header because, if it is, the message is part of a dialog already being routed to the VCE AS. These three rules can be seen in the previous Figure 35.

5.3 Communications Library

WIT Communications Library (COMLib) is a product developed at WIT-Software that handles all related to the communication layer in the RCS+ RCS client for both Android and iOS, as well as other WIT's applications and services that make use of this library to establish communications. The work was done using version 3.6 of this product.

Being a product with confidential and proprietary information protection, any information about the architecture, deploying methods or specifics about the code itself cannot be included in this document. This is not a drawback of this presentation because the only information needed to better understand the work done in this product is that the project has two modules, core and android-api, where the work regarding the VCE share was done by the writer.

In the next section, each one of these sections is explained and presented.

5.3.1 COMLib Core

The COMLib Core is developed in C++ and has all the logic behind the communication itself divided in each of the sections that the library is used by other applications and services. As an example, there are modules for Group Chat, or Conference Calls. COMLib core implements the logic referent to the protocols used to establish this connections.

In other to have the final solutions of this internship working, besides creating the AS and deploying it in an IMS Network, it was essential to add to this core, a new service for the VCE Share and to provide an API for this service.

To achieve this, a new module called VCEShare was added to the COMLib Core, this module developed by the writer has the implementation of the various methods needed by the Android RCS+ client to achieve communication, sharing VCE files through MSRP and creating the correspondent session through SIP signalling.

For this a new vce RCS tag was added and linked to this service:

```
urn%3Aurn-7%3A3gpp-application.ims.iari.rcs.ext.vce
```

This tag is sent as part of the `Accept-Contact` header two times, in the OPTIONS Request and parsed by the COMLib API upon receiving such request from a contact, and also in the INVITE that negotiates the session between two end-points.

With the new tag added it was necessary to handle the file transfer. COMLib already has methods developed to assist new modules with more basic functions such as MSRP File Transfer and File Management. Using this methods provided by this COMLib module, methods where created to provide these main operations in the Android API:

- Accept incoming VCE Share,
- Reject incoming VCE Share,
- Set VCE Share displayed to user,
- Start new VCE Share,
- Terminate running VCE Share.

Furthermore, operations of subscribing and unsubscribing events related to the VCE Share state and process where also developed and provided in the final API:

- Subscribe State change event,
- Unsubscribe State change event,
- Subscribe Progress event,
- Unsubscribe Progress event.

This way, in the Android RCS+ Application can, not only make use of the methods to receive and send VCE but also subscribe to the events of those shares and implement the correct call-backs within the application.

5.3.2 Android API

Once the VCEShare service and API where developed it was necessary to make possible to use its operations by the Android RCS+ application. As the work in the COMLib core was developed in C++, the way to allow the Java code to call and be called by the VCEShare API was using Java Native Interface (JNI).

In the android module of the COMLib project, which handles, among other functionalities, the translation of core API's to Java API's using JNI, two new Classes were added. One written in C++ in order to bind the method developed in the COMLib core and a new Java Class to provide the API methods with the JNI methods being called.

With this it was possible to use these functionalities in the Android application, and successfully send and receive VCE content, as described in the next section.

5.4 Android Client Development

Android RCS Client, RCS+, is a product developed by WIT-Software and used as a competitor by the network operators of the OTT communication applications.

Much like the COMLib project, the RCS+ application has confidential and proprietary information protection, so details about architecture, deployment methods or code cannot be included in this document.

Being one of the objectives of this internship, make use of the AS developed as a new future of this product, some changes had to be made to the Android application in question.

The work done in the Android RCS+ application had two main objectives. To make use of the service API developed in the COMLib and show the AS business logic working with a Android smartphone end-point.

To achieve the firs objective, a VCE Share Worker module was added to the app, so it could subscribe to the communication events, send and receive content. With this achieved, alterations in the user interface were made so the content could be presented to the user, and interacted with. These two main development processes are detailed in the next sections.

5.4.1 VCE Share Worker

One of the main components of the RCS+ application is the content share module. This module, with the assistance of the COMLib methods, handles all the file shares such as Image, Video or generic files. The VCE Share Worker was added to this module.

This class implements the state changes and progress changes call-backs that were described in the previous section (5.3 Communications Library). This way, all the state and progress changes of an incoming or outgoing VCE Share are handled in this class. Also, a Interface was created to be called when some of these changes happen so the UI can be notified by subscribing to this interface call-backs.

The way the UI shows the user the content and how the user feedback is captured is detailed in the next section.

5.4.2 VCE Web View Layout

RCS+ application, in its Android version has very distinct Android Activities and Fragments for each state of the call. The work carried out here was to understand the various activities involved in the UI presented to the user on the three different call states that VCE Content can be shared.

Upon receiving VCE content, this content is automatically accepted by the application. At this point, when the file transfer is finished, the HTML file received has to be shown to the user. To do that a new application view was implemented using Android's Web View that allows developers to open within the application a Web page or local html file.

The way this Web View behaves differs from screen and from call stage moment. The different layouts and the changes they suffer upon receiving content are shown below.

Ringing and On-Call Stage

The moment the RCS+ app places a call, the screens on Figure 36 are shown. A new VCE Web View Layout is added to this screen but hidden. This layout will only be rendered in the case of a successful VCE content transfer.

When the content is successfully transferred, the device vibrates, getting the attention of the user placing the call and renders the local HTML file.

The call controls can be accessed at any time with a long press on the display, and the Visual IVR experience can be cancelled at any time pressing the x white button at the top left.

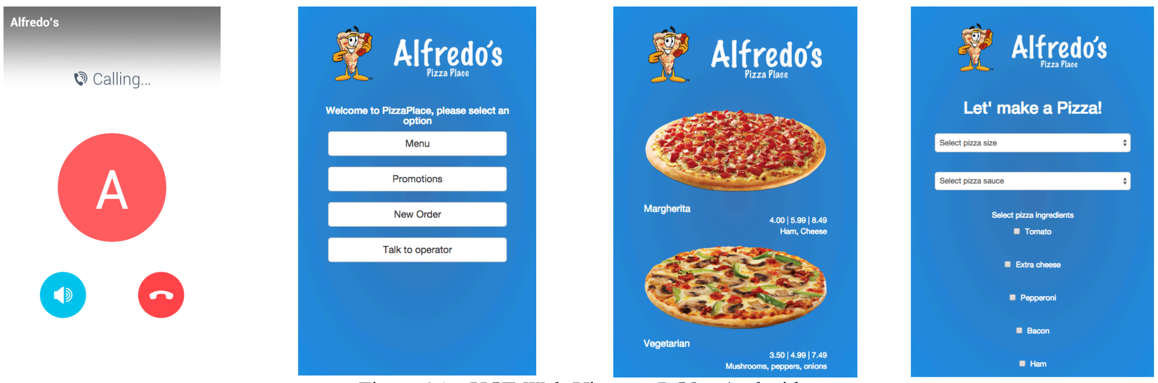


Figure 36 – VCE Web View on RCS+ Android

Call Disconnected Stage

In the case VCE content is shared outside a call, the scenario of an after call share, the user can be interacting with any of the Activities the application has, so, in this case a new Activity is launched in case of a successful VCE file transfer. In practical terms, the UX is the same, with the web page popping into the screen, but the logic behind it is different because it is not a Fragment View integrated with a known Activity screen, but a totally new Activity.

The interaction is the same to cancel the Visual IVR experience, pressing the white button at the top left corner.

5.4.3 JavaScript Binding

When the Web View is launched and becomes the front view of the app, the user has to interact with the content, pressing IVR options buttons or filling up forms.

To achieve this, a JavaScript binding was used. All the buttons in the HTML received have a call to a JavaScript function that calls a Android function in this JavaScript Binding Interface.

There are two different functions that the HTML can call:

- **getOption():** The get option receives the id of the button pressed in the Web View and has two main roles in this interaction. First off all, if the service relates to a legacy IVR, it sends a DTMF with the corresponding digit through the RTP channel using the COMLib tools so the service's system gets the correct feedback from the user. Then, the option chosen is compiled into a XML file and sent to the Visual Call Enrichment AS as a new VCE Share.

- **getInputFromWebViewForm():** This function collects all the information that a form has introduced, and handles the location if the form contains location options. Upon receiving the user input, the location is retrieved as a normal input of Address, City, Country and Postal Code and added to the rest of the form information. The this data is compiled in a new XML file, and sent to the AS as a new VCE Share.§

5.5 Visual Call Enrichment Application Server

Finally, in order to understand the core development of this internship, this chapter shows how the AS was developed.

The AS is the most important component of the work developed, This component has all the business logic of the solution and makes the connection between the service developed and the IMS network enriching the experience of the caller connected to that IMS network when calling a IVR system.

The functions of this AS can be summarized as follows:

- Capability Discovery System,
- Call B2B User Agent,
- Visual IVR,
    - VCE Content Share,
    - VXML to HTML translation.

5.5.1 Architecture

In this section, an overview of the AS architecture is presented. This way the reader can better understand how the previous presented components connect to each other and the relations between them. This can be seen in Figure 37.

Every one of this six different blocks building the AS are referred in the context of JAIN SLEE as Service Building Blocks that can be regarded as 6 different JAVA projects each with a well defined main function. In the next sections each one of these shall be explained in more detail.

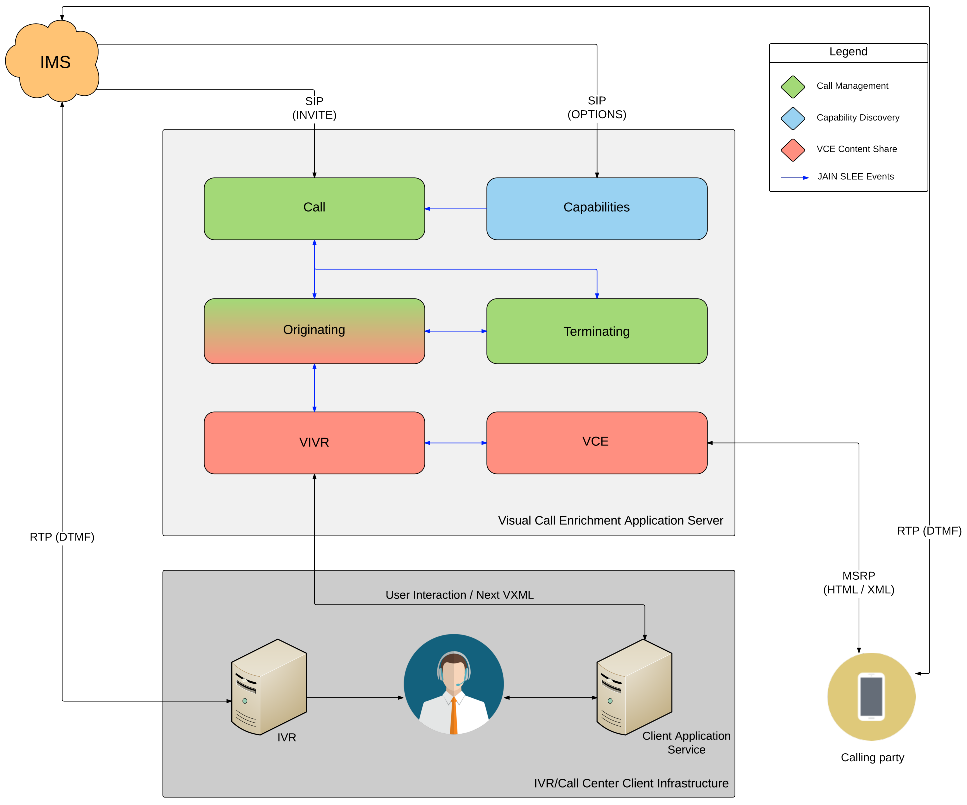


Figure 37 - Application Server Architecture

5.5.2 Capability Discovery

As exposed in 4.6.1 Capability Discovery mechanism relies on the SIP OPTIONS Request to communicate between end-points and share capabilities between them. The Capability Discovery module of the AS handles this communication between end-points communicating in the IMS Network.

Figure 38 shows how the Capability Discovery SBB connects with the other modules in the system. The flow of events and SBB behaviour is detailed next.

As previously shown, the IFC was configured so the all the OPTIONS messages in the open-ims.test realm were to be redirected to the AS. On the other hand, the Capability Discovery SBB has a listener for new OPTIONS messages, or events as there are called in the JAIN SLEE context, to arrive to the AS.

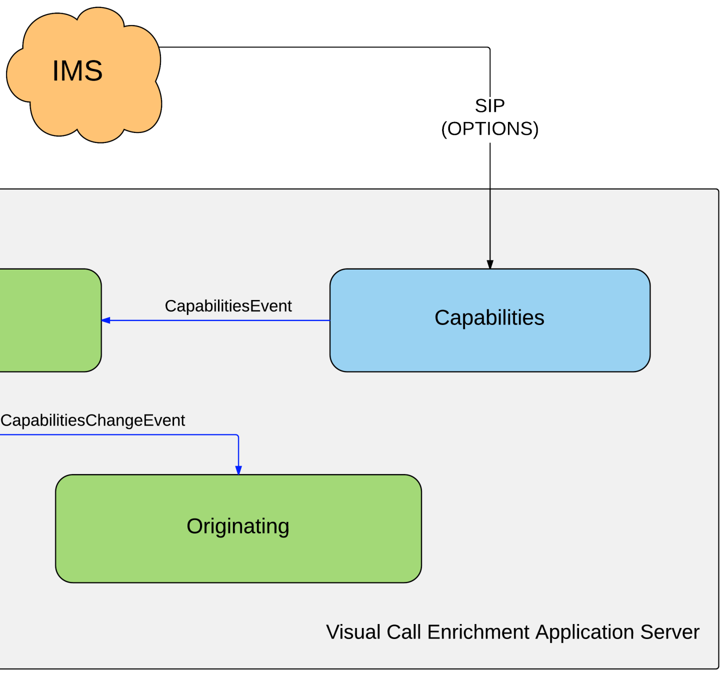


Figure 38 - AS Capabilities Discovery Architecture

This SBB acts then as a Proxy, sending the modified OPTIONS message to the IMS network again so it can be delivered to the end-point it was intended to. In this process, some changes are made to the SIP OPTIONS Request so the VIVR service can be functional from this point on.

First, the SBB adds itself to the `Route` header that contains the route that the message should follow in order to get to its destiny. This way the SIP Dialog between this two end-points are to be redirected to the AS automatically without the intervention of the IFC rule.

Next, the SBB looks at the `Accept-Contact` and `Contact` headers for the vce-share tag, so it can register the sender end-point has a user that can receive vce content.

By this point, the initial request with the needed modifications is sent to the intended next step in the `route` header.

When the response arrives to the SBB, the capabilities of the responding user are checked has the sender was previously. The message is the sent to the next step in the `Via` headers, responsible for indexing the requires jumps to arrive at the dialog initiator user.

At this point the Capabilities SBB is aware of the users participant in the call and the capabilities of sending/receiving vce of both. As this SBB is not the responsible for handling the possible vce share between the end-points, it fires a new `CapabilitiesEvent` to the Call SBB so it can notify the Originating SBB that vce can or cannot be shared with these end-points.

5.5.3 Call B2B User Agent

The Call B2B User Agent is composed by three main SBB's. The Call, Originating and Terminating SBB's. It plays a important role in the VCE session as can be seen in the following figure depicting the Call management architecture.

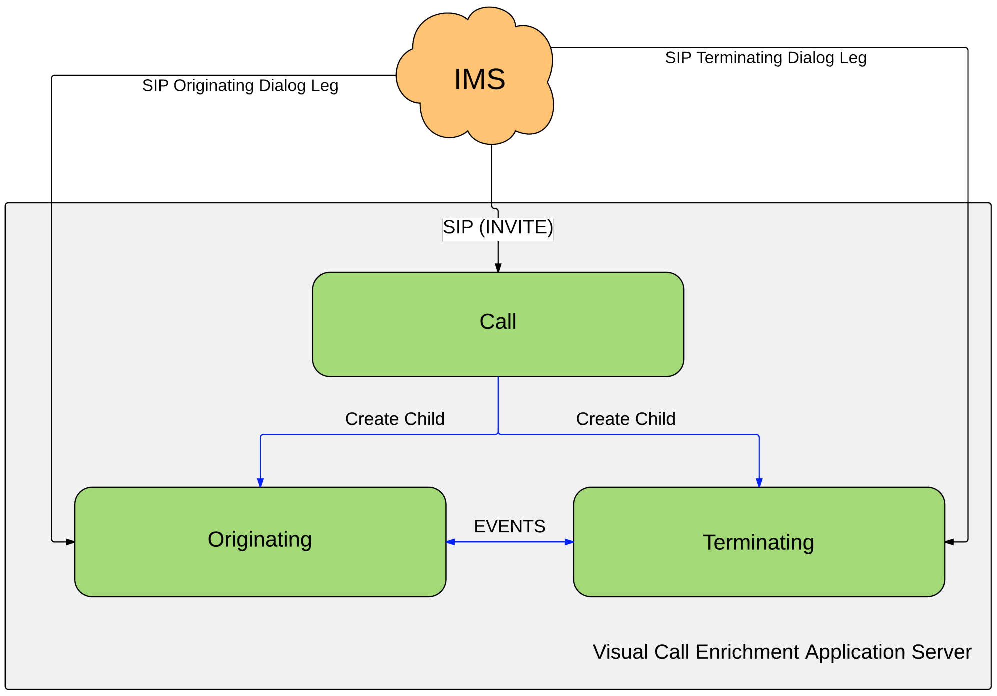

IMS
SIP Originating Dialog Leg
SIP Terminating Dialog Leg
SIP (INVITE)
Call
Create Child
Create Child
Originating
EVENTS
Terminating
Visual Call Enrichment Application Server

Figure 39 – Call B2B UA Architecture

Call SBB's is the entry point of all the INVITE Requests received at the AS, as the IFC re-routes them when obeying the INVITE rule. When a new invite arrives at Call SBB, a verification is made to assure that the INVITE is a initial request from a new dialog relative to a call or other communication that is not from a VCE Share.

After this, if it is from a VCE Share a new `VCESessionEvent` is fired and captured by the Originating SBB for the call in question. The rest of the process of this event is explained in the next section.

In the case that the INVITE is not from a VCE Session, the normal handling of requests and responses in a SIP Voip Call by a B2B User Agent is done by these three SBB's.

The Call SBB creates two new dialogs, one between the calling party and the AS, the originating leg, and another between the AS and called party, the terminating leg. The Originating SBB and Terminating SBB handle each of these dialogs respectively.

From this point on, the dialog between the end points is done exclusively trough this two SBB's, created as child SBB's of the Call SBB.

From the non-VCE Share point of view, this two SBB's have to communicate with each other. This is achieved trough events, much like between Capability Discovery and Call SBB's. As an example, when the Terminating SBB receives a SUCCESS Response (e.g. 200 OK), besides sending the ACK to the called party, it must fire an event to the Originating SBB so it can communicate with the calling party with the correct 200 OK Response.

From the VCE Share point of view, the Originating SBB is the top module to decide when to share the VCE Content with the calling party. More detail on this process is presented in the next section.

5.5.4 Visual IVR

Inside the AS, this is the main achievement of this project. To share, before, after and during the call, Visual Call Enrichment content to the user calling either a IVR System or a Call Centre.

In the following diagram, the most important modules to achieve this are presented, as well as the communication between them.

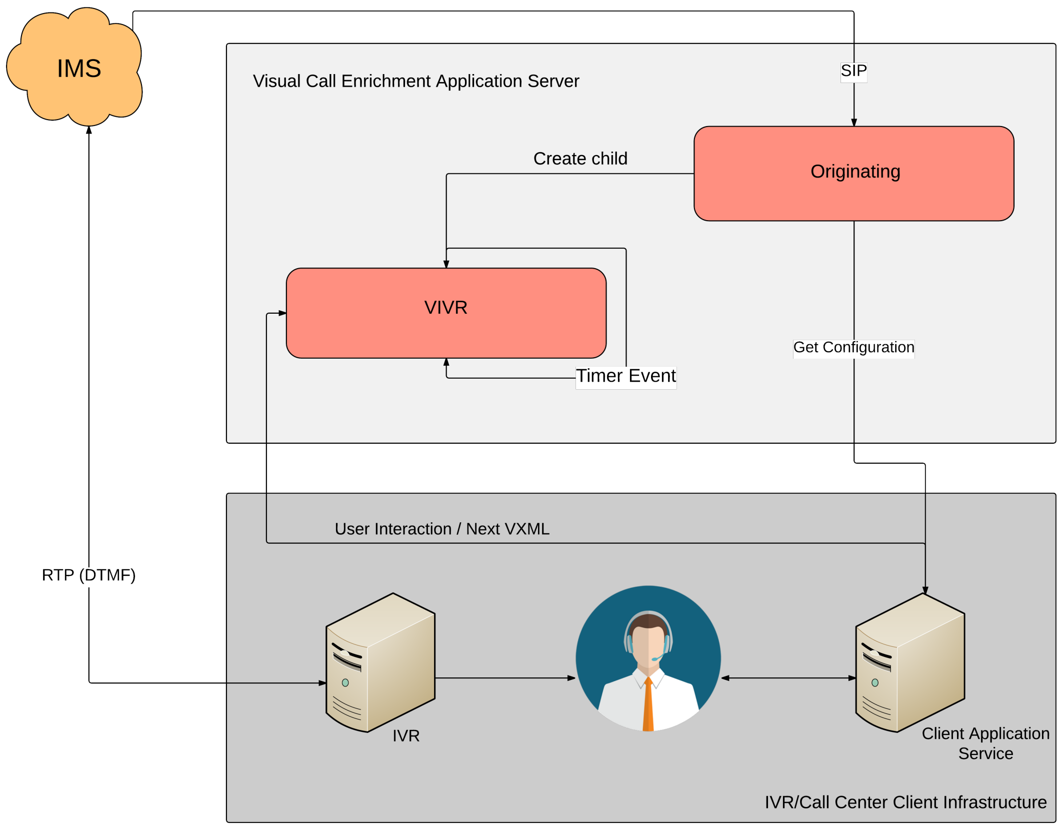

IMS
Visual Call Enrichment Application Server
SIP
Create child
Originating
VIVR
Get Configuration
Timer Event
User Interaction / Next VXML
RTP (DTMF)
IVR
Client Application Service
IVR/Call Center Client Infrastructure

Figure 40 – Visual IVR Architecture

The VCE Share process starts at the Originating SBB. When a new call is placed with the new INVITE Request received by the Call SBB, if the calling party end-point has the vce share capability and the called number is a number registered as a VIVR subscriber. The process of sharing the interactive content starts.

The next step is to get the configuration of the called service. A java properties file is retrieved from the called service application server. This configuration file contains the URL's for the VXML files for the three cases, before, during e after call. According to the existence or not of these paths in the configuration file, the VIVR SBB is created as a child SBB in the correct moment for each case. If the before case is configured for the service, the VIVR service starts upon the reception of the initial INVITE, if it's configured for the during call case, it is initiated after receiving the ACK from the 200 OK answering the call, and in the last case, for the after the call case, the service is initiated when the BYE or the 200 OK from the sent BYE is received.

From this point on a session of VCE Content share is initiated by the VIVR SBB. This SBB is responsible get the service case VXML that was passed as a parameter upon it's creation, and send and receive content according to the contents of that file.

It also has the function of waiting for a share in the case the next menu in the VXML is of the timer type. This means that a new VXML is consulted at a URL in intervals of time and consumed when alterations are made. This way, a Call centre can share VCE content in a middle of a call, without previous input.

VXML to HTML transformation

The first step is to validate the VXML against the local XSD file that contains the schema for the VXML to obey. If the VXML is not valid, the session is not initiated.

Next, the first menu/form form the current use case is translated from VXML to HTML. This is achieved using a XSL file written by the author of this project in order to achieve the final HTML. This XSL contains the various templates that transform each tag of the VXML file in the correspondent HTML tags with the correct id's classes and values, so the user input is read correctly, and the JavaScript Binding with the Android functions is done in the right place. The end result is and HTML file that will be now sent to the calling party end-point, for this a VCE SBB child entity is created.

VCE Content Share

The VCE SBB is present in this project to handle all related to SIP communication between AS and the end-point displaying the VCE content to create the required MSRP Session. This is achieved negotiating the SDP, which describes the session to be established.

First, the initial SIP INVITE Request to negotiate the session has to be built. This is achieved with a copy of the INVITE that was received to initiate the call RTP Session with the correct capability tags for this exchange, as well as the correct SDP content so the session can be established with the RCS+ communications library described in the next sections.

Once the INVITE is created it is sent to the calling party device, this SBB is listening to all responses in this dialog, so the provisional and success messages are received, and it acts accordingly to that response.

When the success message (200 OK) is received, an ACK is sent in response, the MSRP Session is established and the content is sent. Upon reception of the content, a BYE is received at VCE SBB, and the process finishes.

In the eventual response of the calling party, when pressing a button or sending the contents of a form, this SBB is called by the Originating SBB to receive the file with the user input through the receiveVCE() method. The INVITE received is processed and used to create a new MSRP session and receive the user input as a XML file.

At this moment, new event is fired to the VIVR SBB, which interprets the received message and acts according to the usage scenario and current state of the IVR Menu.

The next step can be either get next menu from the current VXML, get next menu from a new VXML or submit the form information to the client's web service and receive further instructions. The logic behind the decisions made according to the received message can be better understood by the analysis of the following diagram.

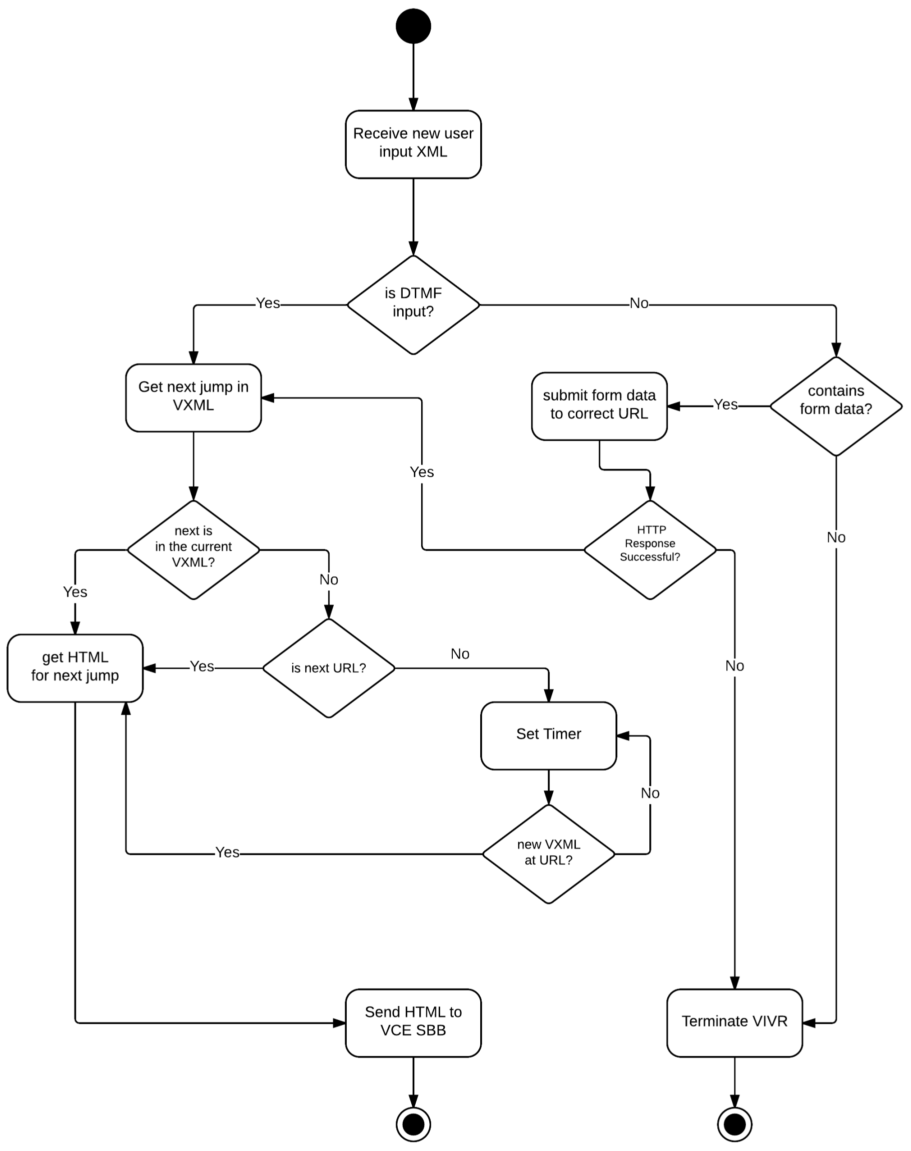


Figure 41 – Visual IVR logic when receiving a new user input

These message exchange and session creations, goes on until reaching a final menu or when the user cancels the visual service.

Message Session Relay Protocol

All the logic described above does not refer how the MSRP Communication and message creation was achieved.

During the project planning it was clear that MSRP Open source library was necessary to use in the JAIN SLEE context to achieve the final result. This was one of the grater worries because only one was found. The problem was the integration with the JAIN SLEE framework, but luckily the person in charge of developing this library also had some experience working with Mobicents and had developed a Recourse Adapter to integrate this library with JAIN SLEE.

This library was used in the project, but because it was still in beta version, and development had stopped since 2013, it has some issues that needed to be resolved.

After some investigation of the source code, it was clear that this library was built to send simple text messages through MSRP. It was not prepared to send large files, and handle the session has a file transfer should behave.

The main issue found was the fact that, for each thread created to handle a message sent, after the message is delivered, the thread is never closed or killed and the library enter a infinite loop if a new session is created.

This issue was resolved with a rewriting of the source code, notifying the thread responsible for the session that receives the notification from the AS.

Two methods were added to inform if all messages were sent and another to kill the thread responsible.

5.5.5 Mock Back End

As a final note of the development chapter, it is important to refer that the author developed a simple Web Application to serve as a mock back end. This Web Application simulates the client side communication allowing the VCE AS to retrieve the configuration for the service subscribers, has well as the VXML for each service or step.

These application also served to simulate the Call Centre were the attendant would send the VCE content to the caller in the middle of the call.

Lastly, it was also used to simulate the Web service that would receive the data submission from forms by a HTTP Get or POST and respond with next menu/form in the form of a VXML file.

This was not part of the development or the plan, but it was essential to ensure the VCE As was capable of fulfilling the client side communications requirements.

Chapter 6
Requirement Validation and Testing

6.1 Introduction

Although the development is the most important part of the internship, testing the validity and reliability of the features of the service created is also of great importance.

This chapter intends to introduce the reader to all the testing and requirement validation that was carried out by the author during the internship and after the development phase.

These tests are not refereeing to development tests, that allow the developer to check the reliability of the various methods and services developed, but the final testing phase, which was important to understand if the final product had met al the initial requirements.

Software testing is defined has the phase to evaluate if a product is able to meet the expectations and deliver the expected results.

6.2 Methodology

There are several methodologies of conducting software testing. Nowadays, a huge number of them are available, and different sets of test cases different test strategies. But almost all of them seem to have these levels of testing in common:

- **Unit**: At this level, individual units of source code are tested. Normally done by white-box testers or the developer itself because a detailed knowledge of the code is needed. These tests intend to find out if that bit of code, or function, works has it was designed to work. To achieve this, test cases have to be written that have a set of entry parameters and expect a solid result. This level of testing was done during the development by the author, with the objective of validating each module of the project in development.

- **Integration:** In this phase, and after unit testing, the modules are tested combined has a group. The purpose of these tests is to verify if the various modules can communicate with no problems. They are normally conducted by a test team, using a black-box approach. In this internship, tests were made by the author of this project whenever a new module was introduced.

- **System**: Normally carried out after integration tests, these have the purpose of evaluating if the system is fully compliant with the requirements. Is usually done by a test team with a black-box approach. In the case of this internship, the tests were done by the author and carried out prior to the development.

- **Acceptance:** Finally the tests that are usually by the buyer/client. They intend to evaluate if the product/software meets the customer specified requirements. As in this POC internship, there wasn't a predefined customer, these tests were also done by the author after the development stage.

During the development, a scrum-like methodology was adopted (4.2 Methodology). The approach for the testing was a little different. As detailed above, not all tests were done during and after the sprints. Some were done during the development (unit and integration) has they were essential to maintain the quality and validity of the work, and others were done in the end (system and acceptance) to validate the requirements. This approach is similar to the V-Model testing methodology. Although it is an extension of the waterfall method, it was well integrated in the scrum development process, has the acceptance and system tests were done prior to the development. In the Figure 42 - V-Model Development methodology [40] all the steps of the testing phase can be identified.

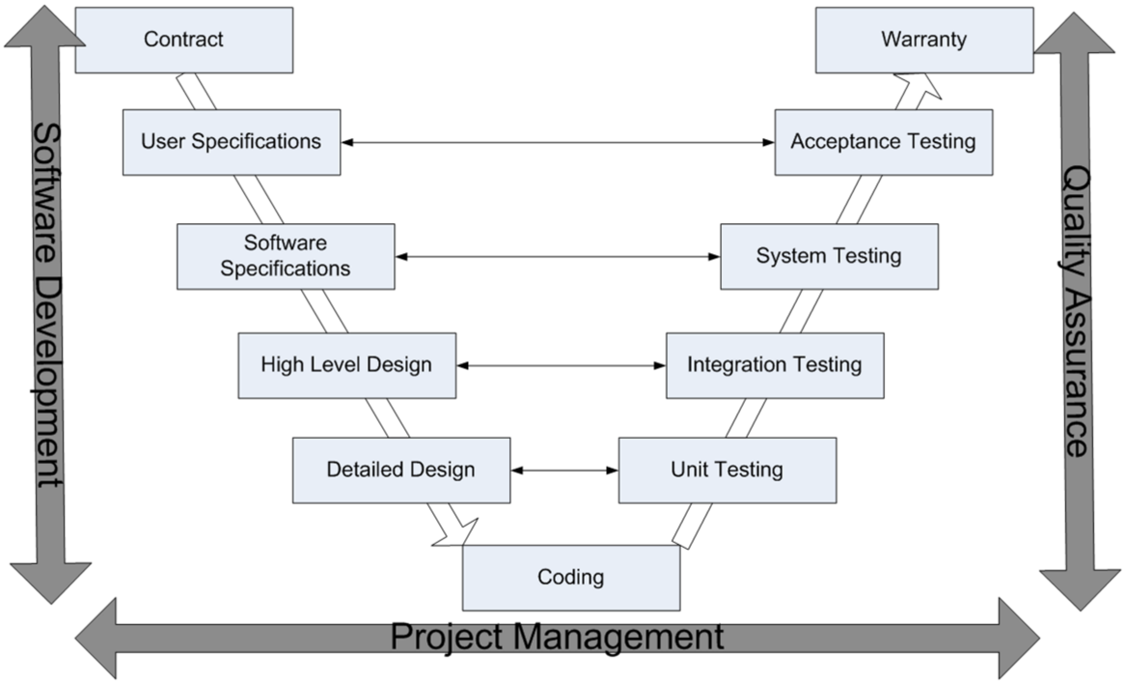


Figure 42 - V-Model Development methodology [40]

6.3 Test Results - Functional Tests

In order to retrieve conclusions from each test, a new test model was designed to conduct the various tests carried out by the author. The Table 16 - Sample test show an example of what each cell in the test model means.

Requirement: Name of the requirement		**Test Level:** Integration/System/Acceptance	
Description: A small introduction to the test			
Input	**Expected**	**Result**	**Pass/Fail**
The premise of the test.	The expected result	The obtained result	Yes/No

Table 16 - Sample test

As described above, the intention of this chapter is not to show the results of the unit tests but the results of the integration, system and acceptance tests. The problem with the unit tests in a document like this is that is impracticable to present them all, and impracticable to keep track of all that were done during the development. It's my belief that it is of grater importance to present the results of the final alpha product, than the results during the development.

6.3.1 Application Server

The core of whole system, and the main focus of development was the AS. For this fact, testing this module was very important. The next sections show the integration tests and their results.

6.3.2 Capability Discovery

OPTIONS IFC		**Integration Test**	
OPTIONS Request is triggered to AS `ip:port` at the IFC			
Input	**Expected**	**Result**	**Pass/Fail**
OPTIONS Request is sent by end-point	The Request is sent to the AS	Same as expected	**Yes**
Get VCE Capabilities from Request		**Integration Test**	
OPTIONS Request is used to get end-point capabilities			
Input	**Expected**	**Result**	**Pass/Fail**
OPTIONS Request is received at the AS	The VCE tag is correctly recognized	Same as expected	**Yes**
Get VCE Capabilities from Response		**Integration Test**	
200 OK Response is used to get second end-point capabilities			
Input	**Expected**	**Result**	**Pass/Fail**
200 OK Response is received at the AS	The VCE tag is correctly recognized	Same as expected	**Yes**
Fire Capabilities Event		**Integration Test**	
New Capabilities Event is fired			
Input	**Expected**	**Result**	**Pass/Fail**
VCE tag and user URI are retrieved	New custom JAIN SLEE Capabilities Event is sent	Same as expected	**Yes**
Listen to Capabilities Event		**Integration Test**	
Capabilities Event is received at SBB			

Input	Expected	Result	Pass/Fail
The new Capabilities event was fired	SBB receives the new Event through the added listener	Same as expected	Yes

Table 16 - Capability Discovery Tests

6.3.3 Call B2B User Agent

INVITE IFC		**Integration Test**	
INVITE Request is triggered to AS `ip:port` at the IFC			
Input	**Expected**	**Result**	**Pass/Fail**
INVITE Request is sent by end-point	The Request is sent to the AS	Same as expected	Yes
Forward INVITE		**Integration Test**	
INVITE Request is sent to the end-point			
Input	**Expected**	**Result**	**Pass/Fail**
INVITE arrives at the AS	The Request is sent to the end-point	Same as expected	Yes
Act as B2B		**Integration Test**	
The AS acts as a B2B User agent			
Input	**Expected**	**Result**	**Pass/Fail**
Dialogs are established	Requests and responses are sent to the AS and correctly forwarded to the intended end-point	Same as expected	Yes
Call is terminated		**Integration Test**	
The call is terminated by one of the end-points			
Input	**Expected**	**Result**	**Pass/Fail**
Call is established, a BYE is received at the AS	The BYE Request is correctly forwarded and the 200 OK response delivered	Same as expected	Yes
Call is declined/cancelled		**Integration Test**	
The call is declined by one of the end-points			

Input	Expected	Result	Pass/Fail
End-points are ringing, one of them rejects the call	The call ends for both end-points	Cancel call works, Declined works only first time due to a OpenIMS known issue	No

Table 17 - Call B2B UA Tests

6.3.3 Visual IVR

Getting client's remote configuration		**Integration Test**	
A Call is placed to a Visual IVR Client, configuration is retrieved			
Input	**Expected**	**Result**	**Pass/Fail**
The call arriving to the AS is destined to a Visual IVR Client	The AS gets the configuration properties file from remote server.	Same as expected	Yes
Getting client's VXML		**Integration Test**	
Depending on the call state (Ringing, In-Call, Disconnected) AS gets correspondent VXML			
Input	**Expected**	**Result**	**Pass/Fail**
Configuration file for Visual IVR client was retrieved	The AS gets the correct VXML according to the call state	Same as expected	Yes
Interpreting caller's input		**Integration Test**	
Depending on the caller input, the correct decision is made to get next step in VIVR service			
Input	**Expected**	**Result**	**Pass/Fail**
MSRP VCE File transfer arrives at AS during VIVR session	The AS retrieves next step according to the current menu, sends form data to remote server or ends session.	Same as expected	Yes
Form submission		**Integration Test**	
Form data is submitted to correct Web Service			
Input	**Expected**	**Result**	**Pass/Fail**

Form data arrives as a MSRP VCE File transfer	The AS submits as a GET or POST the data to remote Web Service	Same as expected	Yes

Table 18 – Visual IVR Tests

6.3.4 VCE Share

INVITE with SDP		**Integration Test**	
INVITE Request is sent to caller end-point			
Input	**Expected**	**Result**	**Pass/Fail**
New HTML menu is created.	The AS creates the INVITE Request with correct SDP to initiate session and sends it to caller.	Same as expected	Yes
Outgoing MSRP Session		**Integration Test**	
MSRP Session establishment with caller end-point			
Input	**Expected**	**Result**	**Pass/Fail**
MSRP Session is successfully negotiated. The 200 OK arrives to the AS	The AS sends ACK and initiates new MSRP Session and new VCE Transfer	Same as expected	Yes
200 OK with SDP		Integration Test	
200 OK Response is sent to caller end-point			
Input	**Expected**	**Result**	**Pass/Fail**
INVITE for a new MSRP Session arrives at the AS	The AS, responds with a 200 OK with the correct SDP	Same as expected	Yes
Incoming MSRP Session		**Integration Test**	
MSRP Session establishment with caller end-point			
Input	**Expected**	**Result**	**Pass/Fail**
MSRP Session is successfully negotiated	The AS gets new MSRP Session and accepts File transfer	Same as expected	Yes

Table 19 – VCE Share Tests

6.4 Android RCS+ Application

This section shows the Integration tests that were conducted in the RSC+ app, these tests are relative to the work done in the COMLib which is the module that handles the communication of the application.

VCE tag at Register		**Integration Test**	
REGISTER Request with VCE tag sent to IMS Network			
Input	**Expected**	**Result**	**Pass/Fail**
RCS+ Application tries to register in an IMS Network	COMLib correctly adds the VCE extension tag to the REGISTER Request	Same as expected	**Yes**
VCE tag at Capability Exchange		**Integration Test**	
OPTIONS Request with VCE tag			
Input	**Expected**	**Result**	**Pass/Fail**
RCS+ Application exchanges capabilities with another end-point	COMLib correctly adds the VCE extension tag to the OPTIONS Request	Same as expected	**Yes**
Accept new VCE Share		**Integration Test**	
Session negotiation with another end-point			
Input	**Expected**	**Result**	**Pass/Fail**
RCS+ Application receives INVITE for new VCE Share session	COMLib correctly accepts and gets new MSRP session	Same as expected	**Yes**
Initiate new VCE Share		**Integration Test**	
Session negotiation with another end-point			
Input	**Expected**	**Result**	**Pass/Fail**
RCS+ Application asks to send initiate new VCE Share	COMLib correctly creates and sends new INVITE with correct SDP	Same as expected	**Yes**
Reject new VCE Share		**Integration Test**	
Session negotiation with another end-point			

Input	Expected	Result	Pass/Fail
RCS+ Application receives new INVITE with VCE Share tag.	COMLib correctly rejects new session if SDP is not correct.	Same as expected	Yes

Table 20 – VCE Share Tests

6.5 System Tests

With the integration tests done, it's important to show the results of the system tests. These were done with an alpha version of the final product, with the intention to test the initial requirements for the final interaction of the user with the end-point.

Visual IVR on ringing stage		**System Test**	
Receiving VCE Content during ringing stage			
Input	**Expected**	**Result**	**Pass/Fail**
User places a call to a IVR / Call centre subscriber of the Visual IVR service	During the ringing stage, the app shows the content intended for the ringing stage	Same as expected	**Yes**
Visual IVR during call		**System Test**	
Receiving VCE Content during call			
Input	**Expected**	**Result**	**Pass/Fail**
IVR / Call centre subscriber of the Visual IVR service answers call.	The application vibrates and shows the initial menu of the IVR	Same as expected	**Yes**
Visual IVR outside call		**System Test**	
Receiving VCE Content outside the call context			
Input	**Expected**	**Result**	**Pass/Fail**
User is using the RCS+ application in any screen, IVR/Call Centre sends VCE content	The application shows the content received.	Same as expected	**Yes**
Menu interaction		**System Test**	
The user interacts with the menus and forms			

Input	Expected	Result	Pass/Fail
RCS+ App is showing VCE Content. User presses buttons on the VCE content	The application correctly collects the user interaction and data.	Same as expected	**Yes**
DTMF dial		**System Test**	
Correct DTMF is sent when pressing a button related to a legacy IVR			
Input	**Expected**	**Result**	**Pass/Fail**
RCS+ App is showing VCE Content. User presses a button.	The application shows dials the correspondent DTMF sound.	Same as expected	**Yes**
Visual IVR Cancel		**System Test**	
Correct DTMF is sent when pressing a button related to a legacy IVR			
Input	**Expected**	**Result**	**Pass/Fail**
RCS+ App is showing VCE Content. User presses X button.	The application shows previous activity screen. Visual IVR session ends.	Same as expected	**Yes**

Table 21 – System Tests

6.6 Acceptance tests

10. Finally, in the realm of the functional tests, the acceptance test results are presented. The client normally does these tests, however, being this project a POC, these were performed by the author. They were done in accordance with the use cases presented in

, and try to assert if the final product requirements were achieved.

These must be viewed as the most definitive tests to know if the internship was a success or a failure.

Capability Discovery Mechanism		Acceptance Test	
AS must be aware of the capabilities of the caller.			
Input	**Expected**	**Result**	**Pass/Fail**
AS Receives SIP Request	AS updates user's capabilities	Same as expected	**Yes**

VCE Share during ringing stage		**Acceptance Test**	
RCS capable device receives VCE Content before call is answered			
Input	**Expected**	**Result**	**Pass/Fail**
User places a call to the IVR / Call Centre with RCS+	RCS+ Application receives and shows VCE Content	Same as expected	**Yes**

VCE Share during call		**Acceptance Test**	
RCS capable device receives VCE Content during call			
Input	**Expected**	**Result**	**Pass/Fail**
User is on call with the IVR / Call Centre with RCS+	RCS+ Application receives and shows VCE Content	Same as expected	**Yes**

VCE Share outside call		**Acceptance Test**	
RCS capable device receives VCE Content outside call context			
Input	**Expected**	**Result**	**Pass/Fail**
User ends call with the IVR / Call Centre with RCS+	RCS+ Application receives and shows VCE Content	Same as expected	**Yes**

VCE Share from Call Centre		**Acceptance Test**	
VCE Content is shared when Call Centre operator sends it			
Input	**Expected**	**Result**	**Pass/Fail**
User is on call with the Call Centre with RCS+. Call Centre operator sends VCE content	RCS+ Application receives and shows VCE Content	Same as expected	**Yes**

VCE Share Cancel		**Acceptance Test**	
VCE Content is shared when Call Centre operator sends it			
Input	**Expected**	**Result**	**Pass/Fail**
User is on call with the Call Centre with RCS+. Call Centre operator sends VCE content	RCS+ Application receives and shows VCE Content	Same as expected	**Yes**

Client uploads configuration		**Acceptance Test**	
Visual IVR Client uploads new configuration with a VXML			
Input	**Expected**	**Result**	**Pass/Fail**
Client send VXML describing a Visual IVR service	Configuration is added to system.	This part of the project was not completed	**No**

Table 22 – Acceptance Tests

6.7 Non-functional Tests

As stated in chapter 4, there are a few non-functional requirements that this solution should achieve. This section analyses if these objectives were accomplished.

6.7.1 State of the art compliance

The final product should be compliant with standard protocols for these type of communications, IMS 11[30] and RCS 5.1[18] Specifications.

To make these guaranties, the author follower RCS 5.1 Specification and used the standard protocols for IMS Communications such as SIP, MSRP or RTP.

Following the RCS 5.1 Specification, the author followed correctly how to implement and achieve the correct flow of communications for the capability discovery mechanism as well as the correct flow of communications on how to place a call and share media.

In the fact that a new RCS extension was developed, the RCS 5.1 Specification was important to understand the correct way to implement a ne service within the RCS context. The correct tag creation and service invocation also followed the specification recommendations.

The IMS 11 Specification was followed to understand how the various components of a IMS network connect to each other, from what component the AS should expect to receive messages, to whom it should send responses and also to correctly configure OpenIMS Core and it's IFC.

6.7.2 High-Throughput

This requirement was impossible to achieve due to the lack of time and excessive complexity of the project. The idea was to use SIPp tool [41] to test the high-throughput of the system.

SIPp is a open source tool to generate SIP traffic. Basically works as a unit test tool that allows the user to write multiple XML scenarios and the output of such scenarios.

The user can set some parameters to better test a real scenario such as number of users, maximum connections or retransmissions. These data can then be used to give some insight about the performance of the system.

Unfortunately these tests were not completed due to lack of time, but the SIPp tool was used during the development to simulate a user.

Although the validity of the tests, with regard to SIP, is undeniable, SIPp does not test MSRP traffic which plays a massive role in the VCE Share.

6.7.3 Open-Source Software

Only open-source and free tools or frameworks were used in the development of this project. Although it is important to refer that, if this POC evolves into a product, the developers should conduct a more extensive study of the state of the art. Only Open Source or free tools and frameworks were taken into account in the preparation for this project and some of them are not reliable for a final product, namely, the MSRP Lib used in the AS.

Chapter 7
Demonstration Examples

These final chapter intents to show the reader how this POC can evolve to a real-world product and some of the usage scenarios it can be applied.

As the integration with IVR Systems or Call centres can be a little confusing. Although they are two completely different concepts, they are often confused as the same, and sometimes used in the same customer service as a whole.

These three scenarios try to separate each of this concepts and show the application of the result of this internship in real-world applications.

7.1 Legacy IVR integration

In this first instalment, the Visual IVR AS is integrated with a legacy IVR System. This IVR system only responds to DTMF signalling and the menus are classic 1-9, * and # inputs.

The Visual IVR AS acts a s middle man, with the service configured with a VXML and the assets (Images + CSS) needed to render correctly the generated HTML.

The HTML files make use of the modified RCS+ application to respond with the correct DTMF signals as the user choses options in the interactive menus.

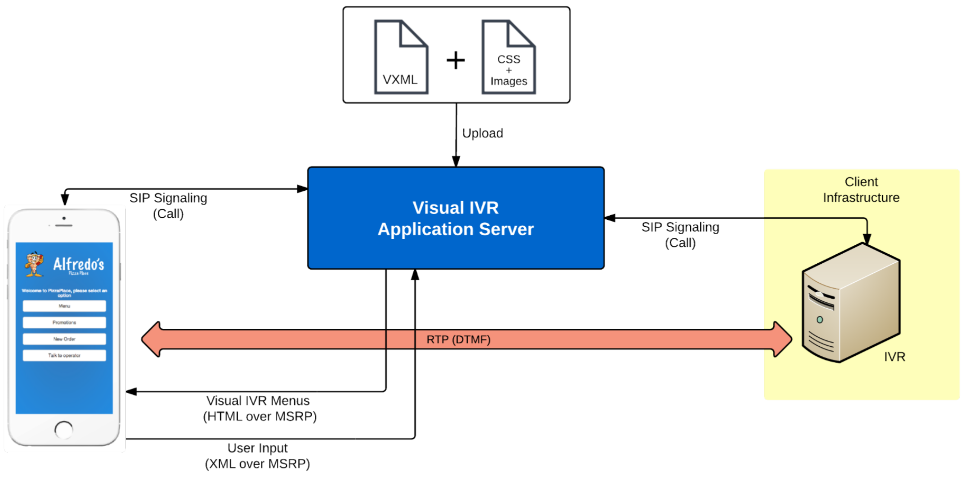


Figure 43 - Legacy IVR Visual IVR Integration

7.2 Enhanced IVR

Secondly, in the Figure 44, another scenario can be viewed with detail. In this case, a IVR System much like the previous one has a IVR service available, and a VXML describing it. The difference in this case is that the client's service can have a application service deployed in the network and receive the submission of forms.

Businesses such as restaurants, food chains, hotels, transportations services or any automated customer service could take advantage of this solution, enhancing the IVR System already in place.

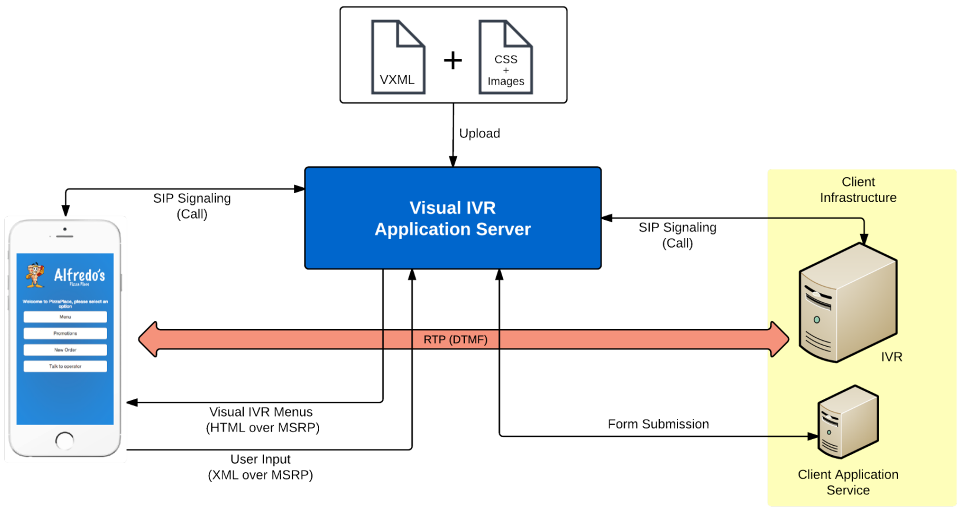


Figure 44 - Enhanced IVR Visual IVR Integration

7.3 User Customized – Web-like experience

Finally, the last usage scenario, where the call centre takes the front stage as the main actor in the Visual Call Enrichment content share.

In this instalment, the user calling the call centre has a web like experience within the call context. Here, the call centre attendant, when talking to the user, has the ability to send content during the call, for example, a walkthrough on how to reset the home WiFi Router or a copy of the last bill.

Services like customer services could take advantage of this solution to offer their clients a visual experience of within the call context.

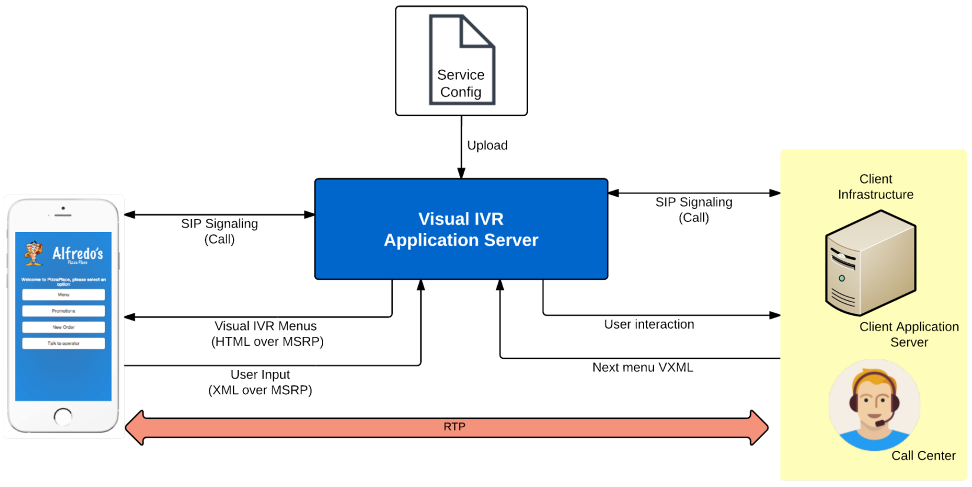


Figure 45 - Web Like experience Visual IVR Integration

Chapter 8 Conclusions

The main objective of this internship was to develop an AS in an IMS network. This AS should add the value of VCE content sharing during call setup, during an ongoing call and outside call context. In addition, it is also an objective to modify WIT's RCS+ RCS client and it's communications library, so it can handle the new VCE content. In this first semester, the main objective was to research what technologies to use, how to use them and plan the development for the second semester. With this first step accomplished, the main focus was on testing the technologies to get the know-how in AS development. Some of the development of the AS started at the end of the semester achieving the majority of tasks planned.

In order to integrate this new service with the RCS client from WIT, which is already in the market, a decision was made to use this client as the test application. This means developing new features in an Android application and in the COMLib which are being developed by a large number of people, during a long period of time. But being the purpose of this internship to learn and develop new capabilities, as well as experience the work environment of a software company this is the path that should be followed and not the simpler one.

During this first semester not the entire plan was fulfilled, this was due to some problems. First of all, the internship was scheduled to start on September 15, but because of some bureaucracies, it only started in the 1st of October. Half a month was lost in this process. In those days, company supervisor guided the intern in what could be studied about some of the tools that could be used. Another problem occurred when integrating with RCS client, during the process of developing Capability Discovery mechanism; the intern was unable to get consistency in which messages were redirected from AS to client. The problem was with the company's network and the solution was simple but it took one week to understand what the problem was.

After the intermediate evaluation by the University of Coimbra representatives, some changes where mad to the scope of the project. As it was well observed by the jury, the focus was a little bit in a grey area, as the final objective was not so clear as the author thought is would be. There were some confusions has if this system was as advantage for IVR Systems or call Centres. With this in mind, the objective of having two RCS clients sharing VCE content was abandoned in favour of a more to the point approach. The idea of as AS that could benefit both IVR systems and Call Centres was the main focus.

With the architecture reviewed, with little to no changes has the initial architecture was already ambitious; the development stage with some changes to the initial plan could fully begin.

One of the ideas of WIT-Software was to have a Back office, so the clients could submit their VXML files and CSS files, but the idea was abandoned in a early stage because the project was a POC, and the innovation part of building the AS and adding the new capability to the RCS app were favoured. The time was not enough to develop all the features to achieve the requirements and also have a good period to conduct tests. These test were specially important to validate the final quality and achievements of the project, so the idea of the Back End was abandoned. To simulate this, the author developed a Web Service that had static configuration files and VXML files for each mock subscriber of the service. Doing this, allowed the author to save time and had the normal interaction and integration with a mock back end.

The development, after the referred restructuration of the plan, followed as normal, within the normal two to three days mismatches, by being late or ahead of plan. The major delay occurred wen the MSRP Lib issue was detected because no time was planed to solve this problem with the open source library.

The work carried out in this internship was, without a doubt, the biggest and hardest academic challenge the author has ever took. It was with great pleasure and joy that the author saw the main objectives achieved, and the final product being regarded as a possible future product of WIT-Software.

On a personal level, this experience contributed greatly to the author's confidence as a future software developer, allowing a more close to reality experience of work and was the perfect step into professional life.

From the technical point of view it was a great challenge to work in such a exigent company and to deal with so many different technologies. It was an enriching experience working in various programing languages such as Java, C++ or XSL, different frameworks like Android SDK or Mobicents JAIN SLEE, so many and different protocols like SIP and MSRP or so recent and innovative areas of knowledge like IMS or RCS.

Looking back to the work developed, it is he author belief that the main objectives were all accomplished, and a functional and valuable POR was produced.

References

[1] *Interactive Voice Response.* (n.d.). Retrieved 10 10, 2014, from Wikipedia: http://en.wikipedia.org/wiki/Interactive_voice_response

[2] *Voder.* (n.d.). Retrieved 10 10, 2014, from Wikipedia: http://en.wikipedia.org/wiki/Voder

[3] *What is an IVR and 6 Benefits of Using One.* (n.d.). Retrieved 10 12, 2014, from Talkdesk.com: http://blog.talkdesk.com/what-is-an-ivr-and-6-benefits-of-using-one

[4] *Dual-Tone Multifrequency Signaling.* (n.d.). Retrieved 10 13, 2014, from Wikipedia: http://en.wikipedia.org/wiki/Dual-tone_multi-frequency_signaling

[5] *DTMF.* (n.d.). Retrieved 10 13, 2014, from Call-center-tech.com: http://www.call-center-tech.com/dtmf.htm

[6] *IVVR.* (n.d.). Retrieved 10 16, 2014, from Apexcomm: http://www.apexcomm.com/index.php/ivr-products-and-services/platforms/interactive-voice-and-video

[7] *Interactive Messaging Response.* (n.d.). Retrieved 10 21, 2014, from Wikipedia: http://en.wikipedia.org/wiki/Interactive_voice_response#Interactive_messaging_response_.28IMR.29

[8] *Visual IVR.* (n.d.). Retrieved 10 12, 2014, from Wikipedia: http://en.wikipedia.org/wiki/Visual_Interactive_Voice_Response

[9] *Visual IVR Improves Contact Center Customer Experience.* (n.d.). (PSSHELP, Producer) Retrieved 10 21, 2014, from http://www.psshelp.com/visual-ivr-improves-contact-center-customer-experience/

[10] *Call VU.* (n.d.). Retrieved 09 20, 2014, from Call VU: http://www.callvu.com/

[11] *Visual IVR Solution - Zappix.* (2014, 10 28). Retrieved from Zappix.com: http://www.zappix.com/zappix-visual-ivr-solution/

[12] *Jcada VISUALIVR.* (n.d.). Retrieved 10 27, 2014, from Visual IVR from Jacada: http://www.jacada.com/products/jacada-visual-ivr-plus

[13] *The ChoiceView Visual IVR.* (n.d.). Retrieved 10 29, 2014, from Radish, The ChoiceView Company: http://www.radishsystems.com/products/choiceview-visual-ivr/

[14] *Ideas abound at the AT&T Innovation Show.* (n.d.). Retrieved 11 2, 2014, from TechHive: http://www.techhive.com/article/2010680/ideas-abound-at-the-atandt-innovation-show.html

[15] *Altar Smart IVR.* (n.d.). Retrieved 10 30, 2014, from Altar - We provide good contact: http://www.altar.com.pl/en/oferta/systemy-call-contact-center/altar-smart-ivr-2/

[16] *Rich Communication Services.* (n.d.). Retrieved 10 3, 2014, from Wikipedia: http://en.wikipedia.org/wiki/Rich_Communication_Service

[17] *Over the top content.* (n.d.). Retrieved 10 18, 2014, from Wikipedia: http://en.wikipedia.org/wiki/Over-the-top_content

[18]*Specs and Product Docs.* (n.d.). Retrieved 09 24, 2014, from GSMA: http://www.gsma.com/network2020/rcs/specs-and-product-docs/

[19]*IMS Tutorial.* (n.d.). Retrieved 1 8, 2015, from Radio-Electronics.com: http://www.radio-electronics.com/info/telecommunications_networks/ims-ip-multimedia-subsystem/tutorial-basics.php

[20]*3GPP.* (n.d.). Retrieved 10 10, 2014, from Wikipedia: http://en.wikipedia.org/wiki/3GPP

[21]*Broadband Integrated Services Digital Network.* (n.d.). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Broadband_Integrated_Services_Digital_Networ

[22]*OpenIMSCore.* (n.d.). Retrieved 09 29, 2014, from OpenIMSCore: http://www.openimscore.org/

[23]*littleIMS 2.0 Documentation.* (n.d.). Retrieved 10 24, 2014, from LittleIMS: https://cipango.atlassian.net/wiki/display/LITTLEIMS/Home

[24]*Project Clearwater.* (n.d.). Retrieved 10 25, 2014, from Project Clearwater: http://www.projectclearwater.org/

[25]*Kamailio SIP Server.* (n.d.). Retrieved from Kamailio.org: http://www.kamailio.org/w/

[26]*IMSZone.* (n.d.). Retrieved 10 31, 2014, from IMSZone.org: http://www.imszone.org/

[27]*Telephony Application Server.* (n.d.). Retrieved 09 17, 2014, from Wikipedia: http://en.wikipedia.org/wiki/Telephony_application_server

[28]Long, D. (n.d.). *JAIN SLEE vs. SIP Servlet.* Retrieved 12 10, 2014, from OpenCloug.org: https://developer.opencloud.com/devportal/download/attachments/29818915/atnac2007_OC_JAIN_SLEE_vs_SIP_Servlet_f.pdf

[29]*The History of VoIP.* (n.d.). Retrieved 01 10, 2015, from Mainline Communications: http://www.mainlinecom.com/voiphistory.html

[30]*3GPP Specification Detail.* (n.d.). Retrieved 12 14, 2014, from 3gpp.org: http://www.3gpp.org/DynaReport/23228.htm

[31]*IP Multimedia Subsystem.* (n.d.). Retrieved 09 23, 2014, from Wikipedia: http://en.wikipedia.org/wiki/IP_Multimedia_Subsystem

[32]*SDP: Session Description Protocol.* (2006, 07). Retrieved 11 2, 2014, from Network Working Group Request for Comments: 4566 : http://tools.ietf.org/html/rfc4566

[33]*Relay Extensions for the Message Session Relay Protocol (MSRP).* (2007, 09). Retrieved 11 5, 2014, from Network Working Group Request for Comments: 4976: http://tools.ietf.org/html/rfc4976

[34]*SIP: Session Initiation Protocol.* (2002, 06). Retrieved 11 4, 2014, from Network Working Group Request for Comments: 3261: http://tools.ietf.org/html/rfc3261

[35]Noldus, R., Olsson, U., Mulligan, C., Fikouras, I., Ryde, A., & Stille, M. (2011). *IMS Applcications Developer's Handbook.* Academic Press.

[36]*From Sip to RTP (Part 3).* (2012, 06 02). Retrieved 01 10, 2015, from Informatica Pressapochista: http://www.informaticapressapochista.com/asterisk/from-sip-to-rtp-part-3/

[37]*SIP – Session Initiation Protocol.* (n.d.). Retrieved from in2EPS into the Evolved Packet System: http://www.in2eps.com/fo-abnf/tk-fo-abnf-sdp.html

[38]*Scrum (software development).* (n.d.). Retrieved 10 30, 2014, from Wikipedia: http://en.wikipedia.org/wiki/Scrum_%28software_development%29

[39]*SIP INFO DTMF.* (n.d.). Retrieved 01 16, 2015, from Voip-info.org: http://www.voip-info.org/wiki/view/SIP+Info+DTMF

[40]*Diference between acceptance and functional tests.* (n.d.) Retrieved 12 08 2015 from Stack Overflow: http://stackoverflow.com/questions/3370334/difference-between-acceptance-test-and-functional-test

[41]*SIPp Tool* (n.d.) Retreived 25 08 2015 from SIPp: http://sipp.sourceforge.net/

[42]*Voice Extensible MArkup Language (Voice XML) Version 2.0* (n.d.) Retrieved 12 08 2015, from W3.org: http://www.w3.org/TR/voicexml20/