UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

Implementación de un software para el servicio de portabilidad numérica de telefonía fija/móvil usando protocolo SOAP/ WSDL para la empresa AMITEL S.A.C. Puno 2015

TESIS

PRESENTADA POR:

Bach. JOSE PIO CHAYÑA BURGOS

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ESTADÍSTICO E INFORMÁTICO

PUNO – PERÚ

2017

UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

Implementación de un software para el servicio de portabilidad numérica de telefonía fija/móvil usando protocolo SOAP/ WSDL para la empresa AMITEL S.A.C. Puno 2015

TESIS

PRESENTADA POR:

Bach. JOSE PIO CHAYÑA BURGOS

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ESTADÍSTICO E INFORMÁTICO

APROBADA POR:

PRESIDENTE : _____
Dr. Edgar Eloy Carpio Vargas

PRIMER MIEMBRO : _____
Dr. Vladimiro Ibañez Quispe

SEGUNDO MIEMBRO : _____
M.Sc. Leonid Alemán Gonzales

DIRECTOR : _____
M.Sc. Ernesto Nayer Tumi Figueroa

ASESOR : _____
Dr. Juan Reynaldo Paredes Quispe

ÁREA : Informática
TEMA : Ingeniería de Software

DEDICATORIAS

"A Dios todo poderoso, a mis queridos padres Quintín y Paula a mi amada esposa Gloria Luz y mis queridos hijos, Camila, Caleb, Bianka".

AGRADECIMIENTOS

A la Escuela Profesional de Ingeniería Estadística e Informática de la Universidad Nacional del Altiplano - Puno, por cobijarnos en sus aulas durante el tiempo de mi formación.

A los Catedráticos de la Escuela Profesional de Ingeniería Estadística e Informática, por compartir sus conocimientos y contribuir con mi formación profesional, por absolver cada uno de mis dudas, con esmero y dedicación en las sesiones de aprendizaje, mi cariño, respeto y admiración por cada uno de ellos.

ÍNDICE GENERAL

CAPÍTULO IV RESULTADOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo de investigación titulado **Implementación de un software para el servicio de portabilidad numérica de telefonía fija/móvil usando protocolo SOAP/ WSDL para la empresa AMITEL S.A.C. Puno 2015**, tiene como propósito estudiar la estructura del protocolo de comunicación SOAP/WSDL y aplicarlo en un software de portabilidad numérica para la empresa AMITEL S.A.C., el software implementado se desarrolló en tres módulos: El módulo de servidor permitirá intercambiar mensajes de portabilidad numérica con el administrador de la base de datos centralizada (ABDCP), usando el protocolo Simple Object Access Protocol (SOAP), el modulo cliente se conecta con el administrador centralizado ABDCP usando el protocolo SOAP para enviar mensajes de requerimiento de portabilidad numérica a su vez deberá atender y procesar los mensajes que nos envía el administrador centralizado hacia nuestro sistema; el módulo de administración se implementa para facilitar la interacción con el usuario final. La hipótesis de la investigación es la implementación del software usando el protocolo SOAP/WSDL para la portabilidad numérica cumple con los requerimientos del estándar de calidad de producto ISO 9126. La población de estudio estuvo constituido por todo el personal que está a cargo del sistema de la empresa AMITEL S.A.C. La metodología de desarrollo de software fue la XP (Extreme Programing). Del desarrollo de la investigación se concluye que la implementación del software cumple los requerimientos del estándar de calidad de producto ISO 9126.

Palabras clave: Comunicación, Telefonía, Protocolo, Portabilidad, Servicio.

ABSTRACT

The present research work titled Implementation of a software for the number portability service of fixed telephony / mobile using SOAP / WSDL protocol for the company AMITEL S.A.C. Puno 2015, aims to study the structure of the SOAP / WSDL communication protocol and to apply it in a number portability software for the company AMITEL SAC, the implemented software was developed in three modules: The server module will allow to exchange numerical portability messages with The centralized database administrator (ABDCP), using the Simple Object Access Protocol (SOAP) protocol, the client module connects with the centralized ABDCP administrator using the SOAP protocol to send messages of numerical portability requirement, And process the messages sent by the centralized administrator to our system; The management module is implemented to facilitate interaction with the end user. The research hypothesis is the software implementation using the SOAP / WSDL protocol for numerical portability meets the requirements of the ISO 9126 product quality standard. The study population consisted of all personnel who are in charge of the system of The company AMITEL SAC The methodology of software development was XP (Extreme Programing). The development of the research concludes that the implementation of the software meets the requirements of the ISO 9126 product quality standard.

Keywords: Communication, Telephony, Protocol, Portability, Service.

INTRODUCCIÓN

El software es una herramienta tecnológica que ayuda automatizar las diferentes tareas y procesos, tareas que manualmente sería imposibles o lenta realizarlo, por este motivo la gran mayoría de empresas privadas y públicas han desarrollado diferentes soluciones de software para ello han usado hardware y lenguajes de programación diversos, con estas implementaciones han resuelto sus necesidades particulares, pero hoy en día con la globalización de las comunicaciones se requiere un intercambio de información entre empresas e instituciones que faciliten el intercambio de información sin modificar su software ya implementado.

En el presente trabajo de investigación estudiamos el protocolo SOAP[1]/WSDL[2] que es un protocolo de la tecnología de servicios web que nos brinda muchas bondades para el intercambio de mensajes entre empresas o instituciones sin la necesidad de conocer el software en los puntos extremos teniendo su ventaja más significativa el uso de XML[3] como formato para el intercambio de mensajes y puede ser implementados sobre servicios como HTTP, HTTPS, FTP, SSH lo cual ayuda enormemente su implementación.

El desarrollo de la presente investigación se divide como sigue:

CAPITULO I: Se plantea los motivos por los cuales se realiza el presente trabajo de investigación, para lo cual se plantea como principal problema "Como implementar un software usando el protocolo SOAP/WSDL para la portabilidad numérica de la empresa AMITEL S.A.C".

[1] SOAP: Simple Object Access Protocol
[2] WSDL: Web Services Description Language
[3] XML: Xtensible Markup Language (Lenguaje de Marcas Extensible),

CAPITULO II: Se estudia los antecedentes del problema en el ámbito nacional e internacional que nos sirvieron como base para la implementación del software, también se estudió detalladamente la documentación del protocolo SOAP/WSDL, los métodos de implementación de software, los pasos para el desarrollo de software.

CAPITULO III: Se establece el tipo de diseño de la investigación el cual se determina como investigación aplicada debido que se tiene un producto final y se aplica los conocimientos adquiridos durante nuestra formación, además se establece como muestra para la medición de la calidad del software al personal de la empresa AMITEL S.A.C en donde se realiza el estudio e implementación del software y la medición del producto se realizara usando el estándar ISO-9126.

CAPITULO IV: Se define el ámbito de la investigación y se especifica los requerimientos funcionales del sistema implementado, se detalla el modelamiento del software así como los diagramas de uso, caso, diagrama de secuencia y se detalla los resultados de la aplicación de la métrica del proyecto usando el estándar ISO 9126, el cual nos da como resultado que el software cumple los requisitos exigidos según dicho estándar y puede ser usado para la gestión de portabilidad numérica.

CAPÍTULO I

PLAN DE INVESTIGACIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

La portabilidad numérica en las redes de telecomunicaciones es considerada un factor esencial que contribuye al desarrollo de los servicios de telecomunicaciones, en nuestro país se implementó este servicio a razón de la resolución de concejo directivo No 166-2013-CD/OSIPTEL que reglamenta el servicio de portabilidad numérica y que beneficia a los usuarios de telefonía escoger el operador que le ofrezca mejores servicios y/o mejores tarifas.

Para suplir este servicio el organismo supervisor de las telecomunicaciones(OSIPTEL) adjudicó la administración del sistema de portabilidad numérica a la empresa ICONECTIV[4] (ABDCP), quien se encarga de centralizar toda la información referente a la portabilidad numérica y sirve como nodo central para la interconexión de todo los operadores fijos y móviles y entre sus principales funciones está el de centralizar los números portados, definir los parámetros de intercambio de información, validar la información remitida por los operadores referente a la portabilidad numérica. Por otro lado

[4] ICONECTIV : Empresa administradora de la base de datos centralizada de portabilidad numérica

los operadores de telefonía fija como Amitel tienen la responsabilidad de implementar un software que permita interactuar con el sistema ABDCP respondiendo y enviando mensajes de portabilidad entiempo real adaptándose a los requerimientos del sistema centralizado ABDCP.

Por este motivo la operadora de telefonía fija AMITEL S.A.C. requiere la implementación de un software confiable y estandarizado que permita cumplir los parámetros para el intercambio de mensajes que deben entregarse y procesarse en un formato correcto y en un tiempo determinado ya establecido de tal forma asegurar la correcta interoperabilidad entre el sistema centralizado y los operadores, dichos protocolos y formatos son establecidos por el administrador centralizado de portabilidad numérica.

Por ello nace la necesidad de implementar un software que cumpla el requerimiento de intercambio de mensajes de comunicación entre el administrador centralizado de portabilidad y el operador de telefonía. En el presente trabajo de investigación se opta por la implementación de un software que use como soporte de comunicación el protocolo SOAP/WSDL.

1.2 FORMULACIÓN DEL PROBLEMA

¿Cómo *implementar un software usando el protocolo SOAP/WSDL para portabilidad numérica?*

1.3 JUSTIFICACIÓN DE LA INVESTIGACIÓN

La evolución de los sistemas de información tubo inicio con la implementación de sistemas mono usuario (un solo usuario) que podían resolver una tarea específica y mantener su información solo para un usuario, con el pasar del

tiempo el avance de la tecnología puso al alcance las redes de comunicación, con la llegada de esta tecnología se implementaron sistemas de información capaces de compartir información pero se tenía la limitante de que estos sistemas de información usaban gran ancho de banda y muchas de ellas su actualización no era en tiempo real para resolver este inconveniente nació la tecnología cliente servidor que su filosofía principal es tener un servidor central que atiende a los clientes y mantiene la data actualizada en tiempo real, hoy en día existe una gran cantidad servicios y sistemas de información implementados con esta filosofía sistemas con web, ftp, e-mail, que son muy eficientes, pero la necesidad de intercambiar solo información ya no es suficiente actualmente la filosofía es que todo los sistemas ejecutar procedimientos fuera de su máquina local, en esta necesidad nace los sistemas distribuidos como CORBA, RMI, JSON, XML-RCP que como filosofía central tiene la ejecución de procedimientos remotos e intercambiar información a nivel de programación esto nos trae nuevas posibilidades de brindar soluciones para resolver necesidades y problemas de programación e implementar nuevos servicios para los consumidores, pero esta tecnología presenta el inconveniente de requerir un sistema intermedio para el formateo de información cuando las plataformas de comunicación son diferentes cuando se trata de integrar dos plataformas con diferentes sistemas operativos como Linux y Windows, para resolver este inconveniente nace la tecnología de servicios web, que usa como formato para intercambio de información al estándar ya conocido XML y usa los servicios ya implementados como HTTP, HTTPS, SSH, Telnet como medio de transporte.

El software implementado servirá como base a futuras aplicaciones que se requiera implementar con características de servicio web usando el protocolo SOAP/WSDL a su vez se cumplirá el requerimiento de la empresa Amitel S.A.C. en la implementación de un sistema que le permita implementar el servicio de portabilidad numérica fija y móvil.

1.4 OBJETIVOS DE LA INVESTIGACIÓN

1.4.1 Objetivo general

Implementar un software usando el protocolo SOAP/WSDL para la servicio de portabilidad numérica de telefonía fija/móvil en la empresa AMITEL S.A.C.

1.4.2 Objetivo especifico

- Implementar el módulo software servidor para responder mensajes de portabilidad numérica fija/móvil usando protocolo SOAP/WSDL.
- Implementar módulo software cliente para generar mensajes de consulta de portabilidad numérica fija/móvil usando protocolo SOAP/WSDL.
- Implementar la interfaz de usuario para los centros de atención de usuarios.

1.5 HIPÓTESIS DE LA INVESTIGACION

1.5.1 Hipótesis general

La implementación del software usando el protocolo SOAP/WSDL para el servicio de portabilidad numérica de la empresa AMITEL S.A.C cumple con los requerimientos del sistema ADBCP.

CAPÍTULO II

MARCO TEÓRICO

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

2.1.1. Internacional

Los sistemas de portabilidad están siendo implementados en todo el mundo para lo cual se tomó la decisión de un modelo de comunicación conformado por hardware y software y un sistema centralizado (SCP) que deberá soportar mecanismos de comunicación con base a estándares abiertos así como prever el respaldo de las comunicaciones, para lo cual deberá contemplar conectividad a través de Internet y/o enlaces dedicados, a su vez, Deberá soportar al menos los siguientes protocolos abiertos: SOAP/XML (Simple Object Access Protocol)/(Extended Markup Language), HTTP´s (HyperText Transfer Protocol) y SFTP (Secure Shell File Transfer Protocol). Cada una de las interfaces deberá incluir mecanismos de respuestas de confirmación por parte del receptor del mensaje quedando asentado en los registros de auditoría. (Kershaw, 2014)

Actualmente la portabilidad numérica existe como realidad en más de 50 países, con diferentes características y niveles de éxito en cada uno, siendo Singapur en 1997 el primer país en implementarlo. En la mayoría de los países usan el protocolo SOAP/WDSL para la implementación de portabilidad numérica.

a) En Alemania, la portabilidad numérica de telefonía fija fue introducida el 1 de Enero de 1998, para número geográficos y números que contienen servicios de valor agregado, como números gratis, números de gasto compartido y servicios Premium (cada uno con su código de encabezado definido de encaminamiento). La portabilidad numérica en éste país no tiene cargos para el usuario. (OECD, 2004).

b) En Bélgica, la portabilidad numérica en telefonía fija fue implementada en el año 2000, con una solución centralizada e independiente para numeración fija y móvil. El sistema fue desarrollado bajo la dirección de la Asociación Temporal para la Portabilidad Numérica (TVN), un consorcio de 11 empresas. (Asociados, 2002)

c) En Brasil la portabilidad del número en telefonía fija está disponible desde el 1 de Marzo de 2009 para todo el territorio. Los usuarios tienen a su disposición un afiche que explica claramente las características del servicio, respondiendo las preguntas más frecuentes, como los cobros que se hacen y en qué casos es posible portar el número. (ANATEL, 2009).

d) En Canadá, la portabilidad fue introducida de manera lenta ya que en un principio no estuvo disponible en la mayoría de los centros

mayoritarios de población. De acuerdo al regulador, para Marzo de 2000, sólo un 37% de los números tenían acceso a ejercer su derecho de portabilidad. Cabe destacar que en éste país la numeración es manejada por un ente privado, la Canadian Numbering Administration Consortium, que es una asociación explícita de operadores, la cual tiene poderes para administrar a conveniencia el recurso que significan los números telefónicos.(OECD, 2002).

2.1.2. Nacional

En nuestro País se estableció la implementación de la portabilidad numérica para el servicio de telefonía móvil a partir del año 2008, el año 2012 se establece la ley para permitir la portabilidad de números fijos, el 13 de diciembre del 2013 se aprobó las condiciones para la implementación de la portabilidad numérica, casi todo los operadores implementaron sus sistemas usando el protocolo SOAP/WDSL.

2.2. SUSTENTO TEORICO

2.2.1. Web services

Un servicio web (en inglés, Web Service o Web services) describe una forma estandarizada de integrar aplicaciones WEB mediante el uso de XML, SOAP, WSDL y UDDI sobre los protocolos de la Internet. XML es usado para describir los datos, SOAP se ocupa para la transferencia de los datos, WSDL se emplea para describir los servicios disponibles y UDDI se ocupa para conocer cuáles son los servicios disponibles. Uno de los usos principales es permitir la comunicación entre las empresas y

entre las empresas y sus clientes. Los Web Services permiten a las organizaciones intercambiar datos sin necesidad de conocer los detalles de sus respectivos Sistemas de Información.

A diferencia de los modelos Cliente/Servidor, tales como un servidor de páginas Web, los Web Services no proveen al usuario una interfaz gráfica (GUI). En vez de ello, los Web Services comparten la lógica del negocio, los datos y los procesos, por medio de una interfaz de programas a través de la red. Es decir conectan programas, por tanto son programas que no interactúan directamente con los usuarios. Los desarrolladores pueden por consiguiente agregar a los Web Services la interfaz para usuarios mediante una página Web o un programa ejecutable, tal de entregarles a los usuarios una funcionalidad específica que provee un determinado Web Service.

Los Web Services permiten a distintas aplicaciones, de diferentes orígenes, comunicarse entre ellos sin necesidad de escribir programas costosos, esto porque la comunicación se hace con XML. Los Web Services no están ligados a ningún Sistema Operativo o Lenguaje de Programación. Un programa escrito en Java puede conversar con otro escrito en Pearl; Aplicaciones Windows puede conversar con aplicaciones Unix. Por otra parte los Web Services no necesitan usar browsers (Explorer) ni el lenguaje de especificación HTML.

El modelo de computación distribuida de los Web Services permite la comunicación de aplicación a aplicación. Un ejemplo seria, la aplicación que procesa las órdenes de compra se puede comunicar con el sistema

de inventarios, tal que este último le puede informar a la aplicación de compras cuales ítems deben comprarse por estar bajo su nivel mínimo. Dado el nivel integración que proveen para las aplicaciones, Los Web Services han crecido en popularidad y han comenzado a mejorar los procesos de negocios. De hecho, algunos postulan que los Web Services están generando la próxima evolución de la Web.

Los Web Services están construidos con varias tecnologías que trabajan conjuntamente con los estándares que están emergiendo para asegurar la seguridad y operatividad, de modo de hacer realidad que el uso combinado de varios Web Services, independiente de la o las empresas que los proveen, este garantizado. A continuación se describen brevemente los estándares que están ocupando los Web Services.

a) XML

Abreviación de Extensible Markup Language. El XML es una especificación desarrollada por W3C. Permite a los desarrolladores crear sus propios tags, que les permiten habilitar definiciones, transmisiones, validaciones, e interpretación de los datos entre aplicaciones y entre organizaciones.

b) WSDL

Abreviación de Web Services Description Language, es un lenguaje especificado en XML que se ocupa para definir los Web Service como colecciones de punto de comunicación capaces de intercambiar mensajes. El WSDL es parte integral de UDDI y parte del registro global

de XML, en otras palabras es un estándar de uso público (no se requiere pagar licencias).

c) UDDI

Abreviación de Universal Description, Discovery and Integration. Es un directorio distribuido que opera en la Web que permite a las empresas publicar sus Web Services, para que otras empresas conozcan y utilicen los Web Services que publican, opera de manera análoga a las páginas amarillas.

2.2.2. SOAP (Simple Object Access Protocol)

Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por Dave Winer en 1998, llamado XML-RPC. **(Moscatelli, 2013)**

La especificación del protocolo SOAP, básicamente, describe un formato de mensajes para comunicar aplicaciones. La filosofía de SOAP, para lograr dicha comunicación, es no inventar una nueva tecnología, sino combinar tecnologías existentes y de amplia aceptación en la industria de software. En particular, combina XML para la codificación de los mensajes y HTTP como protocolo de transporte (aunque no se excluye el uso de otros protocolos de transporte). (Moscatelli, 2013)

SOAP define un mecanismo simple y liviano (en contraposición a sofisticado) para la comunicación, en un entorno distribuido o descentralizado, entre componentes de software o aplicaciones. La

comunicación se realiza mediante mensajes codificados en XML y transportados por un protocolo de transporte (SOAP no mandata el uso de un protocolo de transporte en particular, aunque si define como es el transporte en caso de usar HTTP). En definitiva SOAP define un mecanismo para el intercambio de información, estructurada y tipeada, entre pares de aplicaciones en un entorno distribuido, teniendo como objetivos de diseño la simplicidad y la extensibilidad.

SOAP no define por sí mismo la semántica de las aplicaciones, como ser un modelo de programación o algún tipo de semántica específica de una implementación, sino que provee un mecanismo simple para expresar la semántica de las aplicaciones, mediante un modelo modular de empaquetado de mensajes y la definición de como codificar los datos de las aplicaciones en dichos módulos, es posible ver a SOAP desde distintos puntos de vista:

- Como un mecanismo para invocar métodos en servidores, servicios, o componentes, para lo cual se define en la especificación una metodología para encapsular e intercambiar invocaciones RPC, en los mensajes, usando la extensibilidad y flexibilidad que proporciona XML.

- Como un protocolo para intercambio de mensajes (sincrónicos o asincrónicos).

- Como un formato para intercambio de documentos XML.

a) Mensajes de SOAP

La especificación de SOAP define el formato de los mensajes para comunicar aplicaciones, normalmente dichos mensajes son una forma de comunicación de una única vía entre un emisor y un receptor (mensajes asincrónicos), sin embargo pueden ser combinados de manera de implementar patterns de requerimiento/respuesta (mensajes sincrónicos).

En la versión 1.0 de la especificación de SOAP se establecía como mandatorio el uso de HTTP como protocolo de transporte, sin embargo a partir de la versión 1.1 se desacopla SOAP del uso de HTTP, lo cual permite una mayor variedad de protocolos de transporte como FTP, SMTP. En definitiva la especificación no establece un protocolo de transporte particular pero si específica cómo se realiza el transporte en caso de usar HTTP.

Además de lo anterior, el hecho de usar HTTP tiene las siguientes ventajas:

- Es el protocolo estándar para la comunicación en Internet.
- Está disponible para todas las plataformas.
- Requiere muy poco soporte en tiempo de ejecución para funcionar adecuadamente.
- La seguridad de HTTP es simple y efectiva.
- Es un protocolo que permite atravesar firewalls.
- No es un protocolo orientado a la conexión (para establecer o mantener una sesión se deben intercambiar pocos o ningún paquete).

La especificación de SOAP establece que los mensajes sean codificados en XML. El uso de XML tiene las siguientes ventajas:

- XML es un protocolo para representar datos independientemente de plataformas y lenguajes.
- Se está transformando rápidamente en un estándar al ser ampliamente aceptado por la industria de software.
- Está disponible en todas las plataformas.
- Es adecuado para manipular datos estructurados y tipiados
- Es fácil de analizar (parsing) y entender (tanto para máquinas como por personas) por ser texto.

b) Arquitectura básica de SOAP

Desde el punto de vista de la presente tesis interesa usar SOAP para comunicar aplicaciones, realizando invocaciones RPC e implementando el pattern requerimiento/respuesta, por lo cual el comportamiento es similar al de una arquitectura cliente servidor, donde el proceso es iniciado por el cliente y el servidor responde al requerimiento del mismo.

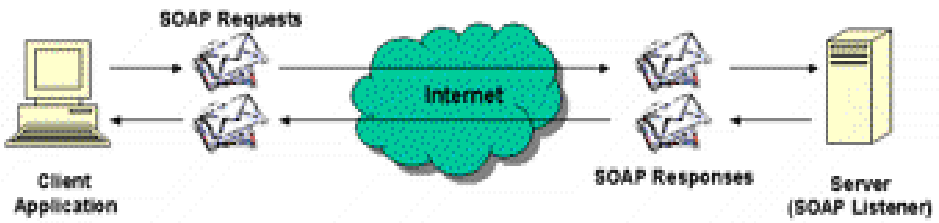


Figura 1 Arquitectura SOAP

Este tipo de comunicación se basa en un sistema de mensajes SOAP sincrónicos, codificados en XML que son transportados por HTTP.

Puede pensarse en cuál es la necesidad de un protocolo de las características de SOAP, si actualmente existen tecnologías de procesamiento distribuido que funcionan correctamente y ampliamente difundidas como ser DCOM o CORBA, por mencionar solamente a dos de las más conocidas, cada una de las cuales provee su propio protocolo de comunicación.

Si se piensa en la evolución de los sistemas distribuidos hacia los WEB Services, puede verse que las distintas tecnologías de procesamiento distribuido existentes presentan una serie de limitaciones debido a:

- Son dependientes de la plataforma y de fabricantes. DCOM está ligado a Windows NT y CORBA, a pesar de ser una arquitectura abierta, está controlado por determinados vendedores. Por lo tanto hacer que los sistemas distribuidos, construidos usando DCOM o CORBA, sean accesibles a otros sistemas heterogéneos (escritos o funcionando en otra plataforma) es un problema bastante serio, a menos que ambas partes tengan la misma plataforma y sistemas similares, lo cual implica que las aplicaciones tengan un alto acoplamiento.

- Presentan problemas con los firewalls. Tanto DCOM como CORBA asignan puertos en forma dinámica, lo cual dificulta su uso en internet, donde los firewalls corporativos bloquean normalmente el acceso, a no ser por puertos específicos (por ejemplo puerto HTTP).

- Hacen uso de protocolos sofisticados. Esto implica la instalación de grandes y complejas librerías del lado del cliente, además de implicar

que el proceso de desarrollo de los clientes DCOM y CORBA es complejo. Esto contradice la idea de los WEB services cuya idea base es publicar servicios para que sea accesibles a otras aplicaciones. Además requieren un gran soporte en tiempo de ejecución para funcionar adecuadamente.

- Ambas tecnologías utilizan formatos propietarios de representación de los datos. DCOM utiliza un formato llamado Network Data Representation (NDR) y CORBA utiliza un esquema similar pero incompatible llamado Common Data Representation (CDR). Además dichos formatos son binarios y muy ligados a la arquitectura de los modelos de componentes respectivos.

En definitiva en el entorno de sistemas distribuidos funcionando sobre internet, las tecnologías existentes presentan serias limitaciones debido a la heterogeneidad del entorno, la presencia de firewalls y al uso de formatos propietarios para representar datos. Bajo estas condiciones SOAP es una alternativa sumamente útil para comunicar aplicaciones heterogéneas (interoperabilidad), fundamentalmente por el uso de XML que es un estándar basado en texto e independiente de plataformas y lenguajes y por el uso de HTTP como transporte, el cual normalmente atraviesa los firewalls dado que estos no bloquean el puerto HTTP. Además al no ser un protocolo sofisticado y al no definir modelos de programación permite que las aplicaciones tengan un bajo acoplamiento, lo cual es ideal en el entorno de internet.

De todas maneras puede verse que SOAP no es una tecnología que reemplace a las existentes sino una tecnología que permite la interoperabilidad de aplicaciones heterogéneas.

i. Protocolo SOAP

La especificación del protocolo SOAP indica que consiste de tres partes:

- *El constructor SOAP ENVELOPE:* que define un framework para expresar qué hay en un mensaje, a quién está dirigido el mensaje y cuando es opcional o mandatorio.
- *Las reglas de codificación: que definen un mecanismo de serialización para ser usado para intercambiar instancias de tipos de datos.*
- *La representación SOAP RPC:* que define una metodología que puede ser usada para representar invocaciones a procedimientos remotos y sus respuestas.

ii. Mensaje SOAP

Tanto los mensajes REQUEST como RESPONSE consisten en mensajes HTTP, pero es de hacer notar que en caso de usar cualquier otro protocolo de transporte no cambia el contenido del mensaje, el cual está codificado en XML. La parte XML de los mensajes tiene la estructura que se muestra en la siguiente figura:

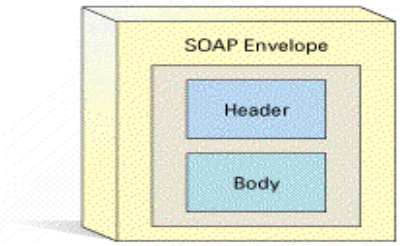


Figura 2 SOAP ENVELOPE

Los mensajes SOAP están codificados en XML y consisten de una sección denominada ENVELOPE (obligatoria), la cual está compuesta de una sección denominada HEADER (opcional) y de una sección denominada BODY (obligatoria).

iii. SOAP ENVELOPE

Esta construcción sintáctica de nombre ENVELOPE contiene el resto del documento XML y debe estar presente siempre y ser la primera sección del mensaje. Define los distintos NAMESPACES que son usados en el resto del mensaje.

Tabla 1 SOAP - ENVELOPE

```
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope"
      SOAP-PENV:encodingStyle="http://schemas.xmlsoap.org/soap/
      encoding/">
</SOAP-ENV:Envelope>
```

Los NAMESPACES se utilizan para garantizar la unicidad de los elementos y evitar ambigüedades.

iv. SOAP HEADER

Esta sección sintáctica de nombre HEADER es opcional y es un mecanismo genérico para extender las características de los mensajes SOAP de una manera descentralizada y sin un acuerdo previo entre las partes que se comunican. En caso de estar presente debe ser el primer hijo de la construcción ENVELOPE.

A modo de ejemplo algunas extensiones que pueden ser implementadas mediante esta construcción son transportar información auxiliar para la autenticación, manejo de transacciones.

Tabla 2 SOAP cabecera

```
<SOAP-ENV:Header>
          <User Information>
</SOAP-ENV:Header>
<SOAP-ENV:Header>
          <Transaction Information>
</SOAP-ENV:Header>
```

v. SOAP BODY

Es una sección sintáctica de nombre BODY que actúa como contenedor para la información que se envía al receptor del mensaje. Esta sección debe estar siempre en los mensajes SOAP y debe estar a continuación del HEADER, si está presente, o ser el primer hijo de ENVELPE si el HEADER no está presente.

Los usos típicos de esta construcción son proveer un mecanismo simple de intercambiar información con el receptor del mensaje SOAP. En esta parte del mensaje es donde se encuentran las invocaciones RPC o bien el resultado de la invocación.

Tabla 3 CUERPO

```
<SOAP-ENV:Body>
  <m1:RemoteMethodName   xmlns:m1="some URI">
    <arg1>value1</arg1>
    <arg2>value2</arg2>
  </m1:RemoteMethodName>
    <m2:RemoteMethodName  xmlns:m2="some URI">
    <arg1>value1</arg1>
  </m2:RemoteMethodName>
</SOAP-ENV:Body>
```

vi. Mensajes

Cada una de las construcciones sintácticas HEADER y BODY pueden ser organizadas en bloques, es decir unidades sintácticas usadas para delimitar información que lógicamente constituye una unidad computacional para un emisor o receptor. La siguiente figura muestra lo explicado anteriormente:

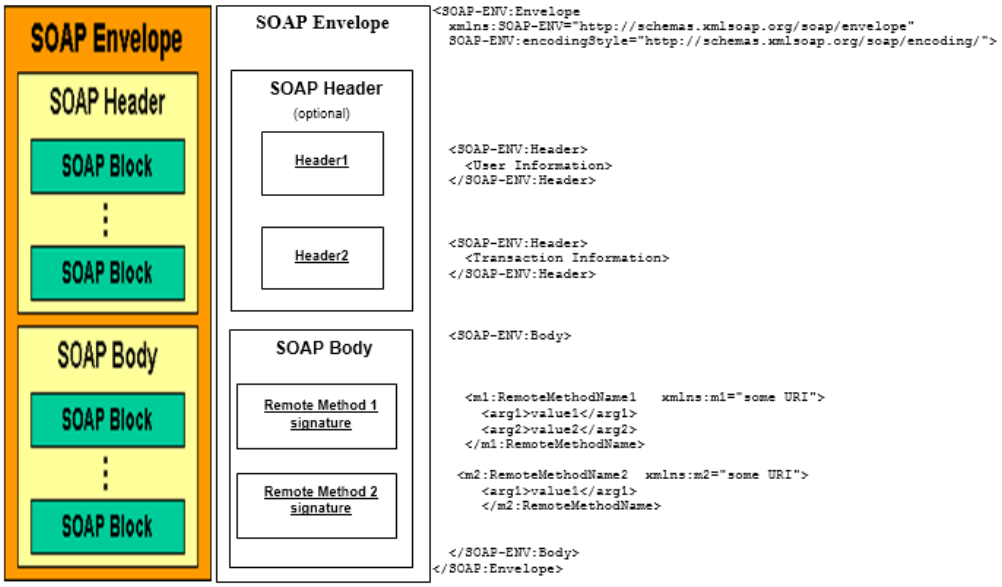


Figura 3 ESTRUCTURA MENSAJE SOAP

vii. SOAP Y HTTP

Como se mencionó los mensajes SOAP son transportados generalmente por HTTP. En este caso pueden usarse distintos métodos para el REQUEST pero el más común es el POST.

Por lo tanto un mensaje SOAP consiste en un HEADER HTTP que se encuentra antes del mensaje SOAP, este HEADER HTTP difiere poco de un cabezal HTTP típico como se detalla a continuación:

En la primera línea se especifica el método de envío, la URI requerida y la versión del protocolo usada:

Tabla 4 Método de envió URL

```
POST /ProductCatalog HTTP/1.1
```

En la siguiente línea se específica el destino:

Tabla 5 Destino URL

```
HOST: www.mywebsite.com
```

Las siguientes tres líneas son usadas para definir el formato MIME para desplegar el mensaje, la codificación HTTP y el largo del mensaje:

Tabla 6 Formato MIME

```
Content-Type:text/xml;
charset="utf-8"
Content-Length: nn
```

A continuación se agregan cabezales SOAPACTION, esta parte del HEADER HTTP es la que difiere de otros cabezales típicos HTTP, que indican la intención del mensaje y son usados en el REQUEST. En particular una posibilidad es indicar el método a ser invocado y el valor es una URI identificando el método (formada por el nombre del método precedida por la URI del HOST y usando el carácter # como separador)

Tabla 7 Identificador URL

```
SOAPAction:"http://www.mywebsite.com#NombreMetodo”
```

Otros posibles valores para el cabezal SOAPACTION son el string vacío (“”) que indica que se intenta acceder a la URI especificada en la primer línea o bien sin valor que indica que no se tiene información sobre la intención del mensaje

Tabla 8 Valor SOAPACTION

```
SOAPAction:””
SOAPAction:
```

- Los firewalls y otros filtros de seguridad pueden usar este campo (SOAPACTION) para conocer el requerimiento que llega sin necesidad de parsear el mensaje.
- El identificador que sigue al carácter # debe machear con los nombres de las tags en el BODY.

A continuación se presenta el mensaje SOAP completo:

Tabla 9 Mensaje completo SOAP

```
POST /ProductCatalog HTTP/1.1
HOST: www.mywebsite.com
Content-Type: text/xml;
charset="utf-8"
Content-Length: nn

     SOAPAction:http://www.mywebsite.com#RemoteMethodName1
     SOAPAction:http://www.mywebsite.com#RemoteMethodName2

<SOAP-ENV:Envelope

     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope"

          SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/
          soap/encoding/">
          <SOAP-ENV:Header>

              <User Information>

          </S°OAP-ENV:Header>

          <SOAP-ENV:Header>

              <Transaction Information>

          </SOAP-ENV:Header>
          <SOAP-ENV:Body>

            <m1:RemoteMethodName    xmlns:m1="http://www.mywebsite.com#">

                <arg1>value1</arg1>

                <arg2>value2</arg2>

            </m1:RemoteMethodName>

           <m2:RemoteMethodName  xmlns:m2="http://www.mywebsite.com#">
               <arg1>value1</arg1>
               </m2:RemoteMethodName>

            </SOAP-ENV:Body>
</SOAP:Envelope>
```

2.2.3. Reglas de codificación (Encoding rules)

Las Reglas de Codificación Básicas son las reglas definidas originalmente en el estándar ASN.1 para codificar información abstracta en un flujo de bits único, esto es, que pueda ser interpretado en cualquier máquina de la misma manera. Las reglas, denominadas sintaxis de transferencia en el contexto de ASN.1, especifican las secuencias de octetos exactas para codificar un elemento de datos dado. La sintaxis define elementos como: las representaciones para tipos de datos básicos, la estructura de la información longitud, y los medios para definir tipos complejos o compuestos basados en más tipos primitivos. La sintaxis BER, junto con dos subconjuntos de BER (Canonical Encoding Rules-CER- y Distinguished Encoding Rules-DER-), están definidas por el documento de estándares X.690 de ITU-T, el cual es parte de las series de documentos ASN.1.

2.2.4. SOAP RPC

Uno de los aspectos fundamentales de SOAP definir una metodología para encapsular e intercambiar invocaciones RPC usando la extensibilidad y flexibilidad que proporciona XML.

El uso de SOAP para realizar RPC es ortogonal al protocolo usado para el transporte del mensaje SOAP. En el caso de usar HTTP, la invocación RPC se mapea directamente al request de HTTP y la respuesta RPC se mapea al response de HTTP.

Para realizar una invocación RPC es necesaria la siguiente información:

- La ubicación del objeto remoto
- El nombre del objeto remoto
- El nombre del método
- Los parámetros del método

Tanto las invocaciones RPC como las respuestas son transportadas en el elemento BODY del mensaje SOAP y son modeladas como structs. Es posible usar el cabezal SOAPACTION para especificar el nombre del objeto remoto y su ubicación.

En el caso de un request, la tag inicial de cada entrada en el elemento BODY es usada para especificar el nombre del método (<NombreMetodo>) y los elementos hijos de esta tag son usados para especificar los parámetros de entrada y/o entrada/salida del método, en el mismo orden que en la signatura del método remoto.

A continuación se muestra un ejemplo de un request SOAP

Tabla 10 Solicitud de petición SOAP

```
<SOAP-ENV:Body>
        <m:addProduct xmlns:m="MY_XML_SCHEMA/IProductCatalog">
            <name>Sony TV</name>
            <desc>17 inch color TV</desc>
            <price>255.99</price>
            <cateogryId>101</categoryId>
            </m:addProduct>

</SOAP-ENV:Body>
```

En el caso de un response, la tag inicial de cada entrada del elemento BODY es el nombre el método seguido por la palabra “response” (<NombreMetodoResponse>) y los elementos hijos de esta tag

representan los parámetros de salida y/o entrada/salida del método, en el mismo orden que en la signatura del método remoto.

En el caso que el método retorne un valor este es el primer hijo de la tag y a continuación se encuentra los parámetros de salida y/o entrada/salida.

Tabla 11 Ejemplo salida y entrada SOAP

```
<SOAP-ENV:Body>
        <m:addProductResponse xmlns:m="MY_XML-
                SCHEMA/IProductCatalog">
            <productId>P00145TV</productId>
        </m:addproductResponse>
<SOAP-ENV:Body>
```

2.2.5. Arquitectura

a) Funcionamiento genérico de RPC en SOAP

La siguiente figura muestra la arquitectura general de un sistema distribuido basado en SOAP:

Figura 4 FUNCIONAMIENTO RPC- SOAP

El siguiente diagrama de secuencia explica en detalle la figura anterior:

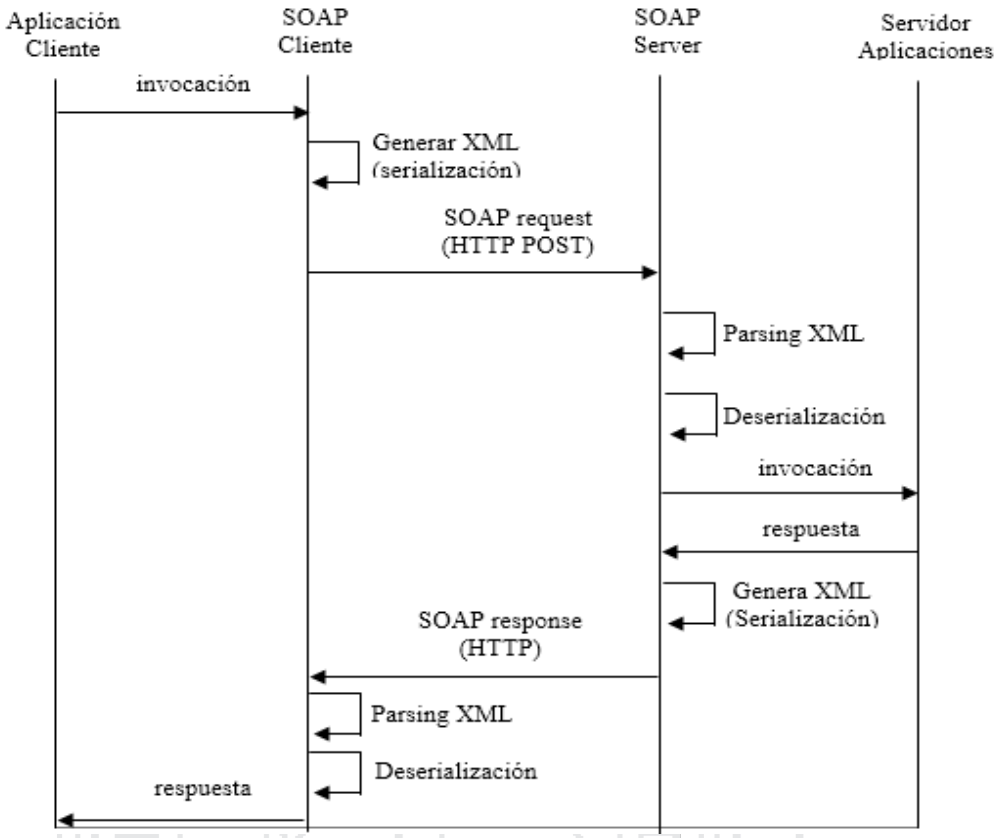


Figura 5 DIAGRAMA DE SECUNENCIA SOAP

La aplicación cliente formula un requerimiento de servicio (invocación), utilizando las funcionalidades de un Cliente SOAP se crea un mensaje en formato XML en el cual se serializa el requerimiento.

El mensaje XML conteniendo el requerimiento del cliente es enviado mediante un protocolo de transporte (en el caso de HTTP mediante un POST) a un servidor SOAP

- El servidor SOAP parsea el mensaje XML recibido.
- El servidor SOAP deserializa el mensaje XML.
- El servidor SOAP realiza la invocación al servidor de aplicaciones apropiado.

- El servidor SOAP recibe el resultado de la invocación y genera un mensaje en formato XML (serialización de la respuesta)
- El servidor SOAP envía el mensaje XML al cliente SOAP utilizando un protocolo de transporte (HTTP).
- El cliente SOAP al recibir la respuesta parsea el mensaje XML.
- El cliente SOAP deserializa el mensaje XML y pasa el resultado a la aplicación cliente.

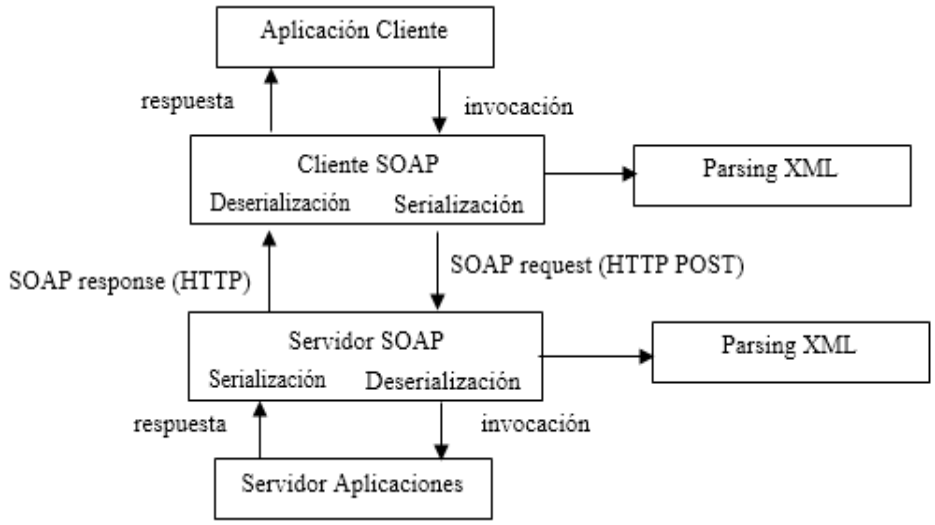


Figura 6 PARSE XML SOAP

b) Ventajas

i. Protocolo abierto

SOAP es una especificación abierta, construido sobre tecnologías también abiertas como XML y HTTP. Por estas razones ha sido aceptado uniformemente por la industria, lo cual incrementa sus posibilidades de transformarse en estándar.

ii. Simplicidad

La especificación de SOAP está bien definida y es sumamente simple.

iii. Independiente de plataformas y lenguajes

SOAP propone el uso de XML y HTTP para acceder objetos, servicios y servidores. Hoy día HTTP es un protocolo de transporte aceptado por la industria y es usado en forma general sobre todas las plataformas. XML está siguiendo un camino parecido, dado que actualmente existen versiones de XML para casi cualquier plataforma o lenguaje de uso común. El uso de XML y HTTP hace a SOAP completamente independiente de plataformas, sistemas operativos o lenguajes de programación.

iv. Interoperabilidad

El mundo de la computación distribuida se encuentra dividido en grandes grupos dependiendo de la tecnología usada, cada uno de los cuales adhiere a su propio protocolo: Microsoft con DCOM, CORBA o Java RMI/IIOP. Es un hecho que la interacción entre sistemas basados en diferentes tecnologías no es algo simple por más que existan bridges entre DCOM y CORBA.

Cabe la pregunta de porque si los distintos browser existentes pueden acceder a los servidores WEB sin tener en cuenta la plataforma del mismo, porque los programadores no pueden hacer lo mismo, o sea invocar servicios remotos de una forma independiente de la plataforma?

Uno de los objetivos de SOAP es eliminar las dificultades que separan las plataformas de la programación distribuida. Para esto SOAP se basa en atributos de otros protocolos exitosos para la WEB: simplicidad, flexibilidad, independencia de plataforma y basado en texto. Puede afirmarse que SOAP no es una nueva tecnología sino la utilización de tecnologías existentes para internet para estandarizar la comunicación entre aplicaciones distribuidas a través de la WEB.

En el nivel más bajo (core) SOAP es una manera estándar de serializar la información necesaria para invocar servicios remotos en un formato que puede ser transportado a través de la red e interpretado por el servicio remoto independientemente de su plataforma.

En el caso de DCOM se usa un esquema de serialización propietario llamado Network Data Representación (NDR), Corba utiliza un esquema similar pero incompatible llamado Common Data Representación (CDR). Por este motivo no es posible hacer invocaciones entre uno y otro en forma nativa y solamente es posible si se dispone de un bridge, el cual agrega complejidad y disminuye la performance.

SOAP enfoca la interoperabilidad entre los sistemas distribuidos en el nivel de serialización de datos, introduciendo XML en reemplazo de los formatos propietarios de serialización, lo cual permite adoptar una posición neutral entre las distintas plataformas.

En el área de interoperabilidad SOAP tiene ventajas con respecto a otros protocolos como ser IIOP DCOM, RMI, etc. Fundamentalmente si se piensa en la evolución de los sistemas distribuidos hacia los WEB

Services, en donde las aplicaciones distribuidas son accesibles a otros sistemas (escritos y funcionando en otra plataforma) a través de internet, SOAP representa un mecanismo sumamente útil a tales efectos, mientras que otras tecnologías no logran un buen desempeño debido a los formatos propietarios de codificación de la interacción y la dependencia de las plataformas.

v. Firewalls

Al usar HTTP como transporte puede pasar fácilmente a través de los firewalls, lo cuales dejan pasar habitualmente el tráfico que les llega por el puerto HTTP.

Generalmente los desarrolladores se deben esforzar mucho para hacer que sus aplicaciones distribuidas trabajen en internet debido a la presencia firewalls dado que los mismos bloquean comúnmente casi todos los puertos excepto algunos, entre los cuales se encuentra el puerto HTTP.

Otras tecnologías distribuidas, como DCOM y CORBA se basan en la asignación dinámica de puertos para realizar invocaciones remotas. Esto obviamente implica convencer a los administradores del lugar donde reside el servidor que abran los puertos necesarios en el firewall. Sin embargo los clientes de las aplicaciones distribuidas que están del otro lado de otro firewall corporativo tiene el mismo problema, si no configuran su firewall para abrir el mismo puerto no pueden hacer uso de la aplicación. Obviamente esta reconfiguración del firewall donde reside el cliente no es algo práctico.

Pero como SOAP utiliza HTTP como mecanismo de transporte y muchos firewalls permiten que HTTP los atraviese, no existen problemas de invocaciones de ambos lados de un firewall.

Es de destacar que SOAP permite que los administradores configuren los firewalls para bloquear selectivamente los mensajes SOAP usando SOAP headers específicos (el header SOAPACTION, puede ser usado por los firewalls y otros filtros para conocer el requerimiento que llega sin necesidad de parsear el mensaje).

vi. Transporte

La versión 1.0 la especificación de SOAP mandataba el uso de HTTP como transporte, en la versión 1.1 de la especificación se flexibilizó esta posición permitiendo el uso de protocolos de transporte como FTP, SMTP, además de HTTP y sus variantes (HTTPS).

Puede pensarse que en algún momento se usen otros protocolos como ser IIOP, RMI, etc. y de esta manera aumentar su usabilidad e interoperabilidad.

vii. Bajo acoplamiento

Los sistemas distribuidos basados en SOAP tienen un bajo acoplamiento, por lo cual pueden ser fácilmente mantenidos dado que pueden ser modificados independientemente de otros sistemas.

En particular SOAP presenta un bajo acoplamiento con los sistemas y las aplicaciones internas de una organización, situación que no se da con

otras tecnologías distribuidas (CORBA, DCOM, etc.) donde existe un alto acoplamiento con la arquitectura subyacente de las aplicaciones lo que implica que tanto el emisor como el receptor deben usar el mismo sistema o por lo menos sistemas compatibles.

viii. Escalabilidad

Al usar el protocolo HTTP como transporte, los sistemas distribuidos construidos sobre SOAP son sumamente escalables.

c) Desventajas

i. Performance

No es una forma de comunicación entre aplicaciones compacta, dado que los mensajes están codificados en XML, o sea en un formato ASCII. Esto hace que el transporte sea ineficiente a través de la red, en particular en los casos de grandes conjuntos de datos. Lo contrario sucede con los formatos binarios de datos como mecanismo de comunicación entre aplicaciones, que son usados por CORBA (CDR), DCOM (NDR).

Frente a esto puede argumentarse que a pesar que los mensajes SOAP sean mucho más grandes que sus equivalentes en formato NDR o CDR, por estar en formato XML o sea ASCII, que esta desmejora de la performance es insignificante en el caso de aplicaciones sobre internet donde la latencia inherente a la propia red es mucho más significativa.

Lo que si puede decirse es que para una LAN, o lo que es lo mismo dentro de una empresa, DCOM, CORBA o Java RMI tienen una

performance mejor y por lo tanto estos protocolos son preferidos para desarrollar en el ámbito de una LAN (recordar que además no existen los problemas inherentes a los firewalls). Existen estudios publicados en http://www.tkim.net/paper/soap.html que comparan la performance de SOAP e IIOP. En el caso local, usando CORBA/IIOP se logra una mejora de 15 veces que usando SOAP. Esto se da aún en el caso de usar IIOP sobre HTTP

ii. Parsing

Como está basado en XML (formato ASCII) el parsing requiere más recursos de CPU, que otros protocolos basados en formato binario.

iii. Marhalling/Unmarshalling

Como está basado en XML (formato ASCII) el marshalling / unmarshalling requiere más recursos de CPU, que otros protocolos basados en formatos binarios.

iv. Serialización

Todos los datos son serializados y pasados por valor y no por referencia, esto puede provocar problemas de sincronización si múltiples copias de un objeto fuesen pasadas en el mismo momento.

v. Semántica

SOAP no define la semántica de los mensajes, lo cual significa que la aplicación cliente y la aplicación servidor deben acordar a la semántica de los mensajes.

2.2.6. SOAP y las Tecnologías distribuidas

SOAP no es comparable a las tecnologías de procesamiento distribuido existentes, dado que SOAP es un protocolo de comunicación entre aplicaciones y no es una arquitectura distribuida completa, por lo cual varias características de estas arquitecturas quedan fuera del alcance de SOAP, como forma de cumplir con los objetivos de simplicidad y extensibilidad de su diseño.

La siguiente tabla muestra cómo se compara SOAP con las características relevantes de las arquitecturas distribuidas y como estas se comparan entre sí.

Tabla 12 Comparativa de tecnología Web Services

	CORBA	DCOM	JAVA-RMI	SOAP
Performance	Una vez obtenida la referencia a un objeto, CORBA permite una comunicación directa, por lo cual la comunicación es muy rápida.	Requiere mucho intercambio inicial para activar y usar el objeto remoto. Una vez obtenida la referencia al objeto remoto, es posible acceder en forma directa al objeto.	Buena performance. Funciona únicamente para el lenguaje Java y por lo tanto la performance está optimizada.	La performance es baja, debido a que se debe extraer el mensaje SOAP, hacer el parsing de XML, crear los objetos apropiados y convertir los parámetros.
Manejo del Estado	Orientado a la conexión y steteful.	Steteful.	Muy flexible, provee sub-protocolos stateless y stateful.	La especificación de SOAP no establece nada al respecto, sin embargo si se utiliza HTTP como protocolo de transporte, este requiere una arquitectura stateless, la cual puede no ser apropiada en todas las circustancias.
Garbage collection	CORBA no establece el manejo distribuido de memoria, aunque existen implementaciones dependiendo del fabricante.	Provee un mecanismo automático.	Provee de un excelente mecanismo.	La especificación de SOAP no establece nada al respecto.
Seguridad	No provee un soporte intrínseco para autentificación, autorización o identidad.	Provee soporte para autentificación, autorización e identidad. Los usuarios pueden definir el nivel apropiado de seguridad.	Como Java RMI trabaja con el lenguaje de programación Java, hereda los mecanismos de seguridad existntes en Java. El uso del RMI Security Manager habilita la carg dinámica de clases y por lo tanto provee seguridad adicional.	La especificación de SOAP no define ninguna característica de seguridad propia del protocolo SOAP. La seguridad, por lo menos en alguno de sus aspectos, queda determinada por el protocolo de transporte que se utilice. Teniendo en cuenta que los mensajes está codificados en XML y por lo tanto en texto, cualquiera podría alterar dichos mensajes. Como SOAP utiliza HTTP como transporte puede utilizar cualquier característica de seguridad estándard de HTTP. Pueden usarse los mecanismos de autenticación o bien canales seguros de comunicaciones (SSL y HTTPS). Otros aspectos de seguridad pueden implementarse haciendo uso de usa de headers específicos de los

En conclusión SOAP no define o deja expresamente de lado ciertas características de las arquitecturas distribuidas con la finalidad de mantener los objetivos de su diseño de ser un protocolo simple y extensible. Sin embargo puede observarse que SOAP no tiene por finalidad sustituir las tecnologías de procesamiento distribuido existentes sino brindar interoperabilidad en un entorno heterogéneo, donde presenta características que lo hacen sumamente útil frente a las limitaciones de otras tecnologías. Por lo tanto es posible que los sistemas distribuidos existentes de una organización o los desarrollos de nuevos sistemas sean expuestos a todos aquellos clientes que soporten SOAP.

2.2.7. Portabilidad Numérica

Derecho de los usuarios de los servicios de telecomunicaciones de poder conservar su número de teléfono, aún en el evento que cambie de empresa operadora, servicio o ubicación (TRANSPORTES, 2002).

Es un mecanismo que permite a los usuarios mantener su número telefónico cuando éstos cambian de proveedor de servicio (Portabilidad de Proveedor de Servicios), ubicación geográfica (Portabilidad Geográfica) o de servicios de telecomunicaciones (Portabilidad del Servicio). (Piccini, 2014)

En nuestro caso cuando se hable de Portabilidad Numérica nos referimos a la Portabilidad Proveedor de Servicio.

2.2.8. Agentes involucrados

En el proceso de implementación de la portabilidad numérica intervienen 3 agentes: los usuarios, las empresas operadoras y las autoridades regulatorias. (Piccini, 2014)

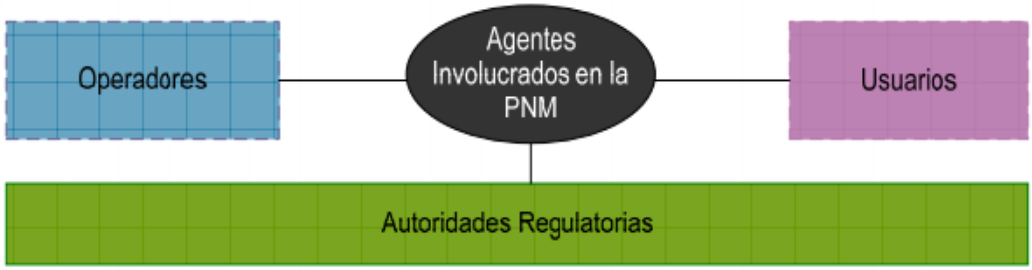


Figura 7 AGENTES PORTABILIDAD NÜMERICA

Los beneficios de la implementación y puesta en marcha de la portabilidad numérica se encuentran distribuidos entre los 3 agentes:

a) Usuarios

Mejores ofertas, reducción de tarifas, reducción de precios de los equipos, mejoras en la calidad, incremento en la gama de servicios, mejor servicio a clientes, mayor cobertura de red, entre otros.

Les permite elegir al operador que les prestará el servicio, sobre la base de sus propios criterios de elección (comparaciones de precios, atención al cliente y/o nivel de calidad de los servicios) seleccionen libremente a su proveedor de servicio.

A los usuarios que llaman a abonados que han portado su número telefónico les brinda una mayor comodidad ya que se evitan el costo de

mantener actualizada sus agendas o la no realización de las llamadas.

b) Empresas:

Representa una mayor exigencia competitiva incentivándolas a prestar una mayor atención respecto de las apreciaciones y necesidades de sus clientes.

c) Administración:

Contribuye a una mejor administración de la numeración y favorece la competencia ya que facilita la entrada de nuevos operadores.

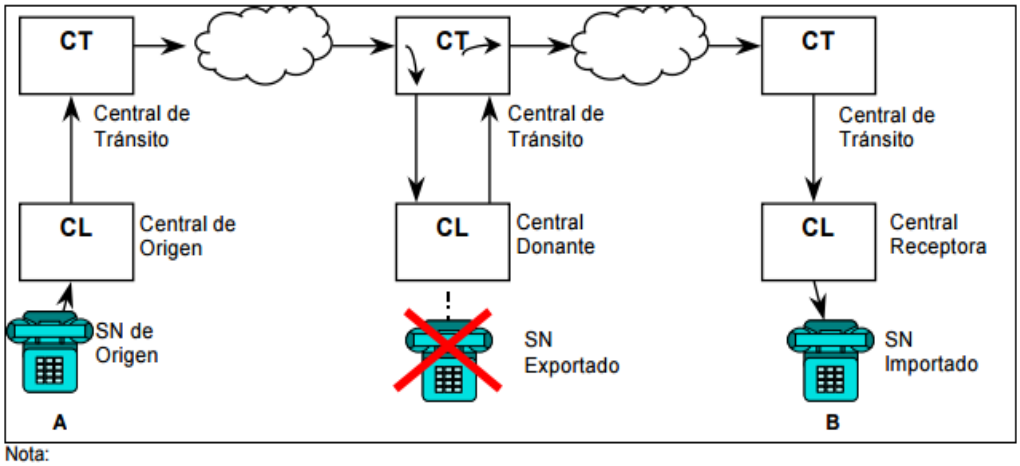
2.2.9. Técnicas de implementación más comunes

a) Técnica de *reenvió de llamada:*

Esta técnica utiliza las funcionalidades ya existentes en las centrales de conmutación, mediante las que es posible reencaminar una llamada entrante hacia otro destino preseleccionado. De modo genérico, la red origen enruta la llamada hacia la red donante (aquella a la que originalmente pertenecía el número) como consecuencia del análisis de los dígitos, y en la central donde antiguamente se ubicaba el número se desencadena el proceso de desvío hacia la nueva posición del abonado. Este método implica un uso intensivo de los recursos de la red donante y su implementación es sencilla, a corto plazo.

Modalidades: Remote Call Forwarding Se caracteriza porque la red donante puede comunicar a la red receptora la información relativa a la portabilidad, lo que permite el desvío transparente de servicios suplementarios hasta el abonado. Red en la cual se localiza un número

luego de ser “portado”, consideremos la configuración de red que se muestra a continuación:



Nota:
- CL: Central Telefónica Local
- CT: Central Telefónica de Tránsito
- SN: Número de Abonado (suscriber number)
- SN Exportado: Número de abonado enviado de una central a otra
- SN Importado: Número de abonado introducido en una central

Figura 8 TECNICAS DE REENVIO DE LLAMADAS

El **Abonado A** origina una llamada al **Abonado B**, quien originalmente estaba en la central que se muestra en el centro del diagrama, ahora llamada “Central Donante”. Al llegar la llamada a la central “donante”, el conmutador “donante” buscará información contenida en el conmutador para determinar si dicho número ha sido “portado”. Dado que efectivamente ha sido “portado”, la llamada será encaminada al conmutador receptor en base a la información de encaminamiento que está prefijada a la información de la parte llamada, mediante el uso del reenvío de llamadas. Para cada llamada que entra al conmutador “donante”, se debe revisar primero los datos de la base del conmutador para ver si el número ha sido o no “portado” a otro portador. Estos datos se guardan en los datos de la línea de cada número como información de reenvío de llamadas. Todas las centrales que tienen números portables

requerirán que el conmutador cuente con el software para el reenvío de llamadas. La llamada se reencamina a través de la red de telefonía pública en base al prefijo de la dirección de encaminamiento en la red + el número de abonado. No obstante que el cliente B ha mantenido su mismo número, la red reconoce a B a través de un segundo número de abonado que no puede ser discado por los usuarios llamantes; este número sólo lo conocen los elementos de la red. Además, los enlaces hacia y desde la central “donante” se mantendrán mientras dura la llamada.

b) Call forwarding unconditional

La red donante puede desviar la llamada a la red receptora sin procesar la información recibida desde la red origen.

2.2.10. Técnica de consulta a base de datos

Técnica de implementación basada en red inteligente, que consiste en consultar la información sobre el número portado a bases de datos, permitiendo que el control del establecimiento de la llamada se mantenga por la red de origen. Al cursar una llamada, la central origen (a través del Punto de Transferencia de Señalización), se dirigirá hacia la base de datos en la cual recogerá la información correspondiente al referido número, enviando información de encaminamiento nuevamente hacia la central origen. Dicha información le permitirá encaminar adecuadamente la llamada hacia la central receptora. En el caso de ser un número no portado, la llamada se encaminará de manera convencional.

Modalidades:

a) Consulta en Liberación (Query on Release)

Método de portabilidad de números en el cual la llamada es encaminada a la central "donante" por defecto; si ésta es rechazada (debido a que el número no corresponde a dicha central) la llamada es devuelta a la central de tránsito (equipada con red inteligente) para que efectúe la consulta a la base de datos, con dicha información se encamina correctamente la llamada hacia la red receptora. Este método centraliza la información de encaminamiento para los números "portados", lo cual facilita la actualización instantánea y la capacidad de compartir la base de datos con otros operadores. Además minimiza el número de consultas a realizar. Aumenta el tiempo necesario para establecer la llamada "portada" en comparación de las llamadas "no portadas".

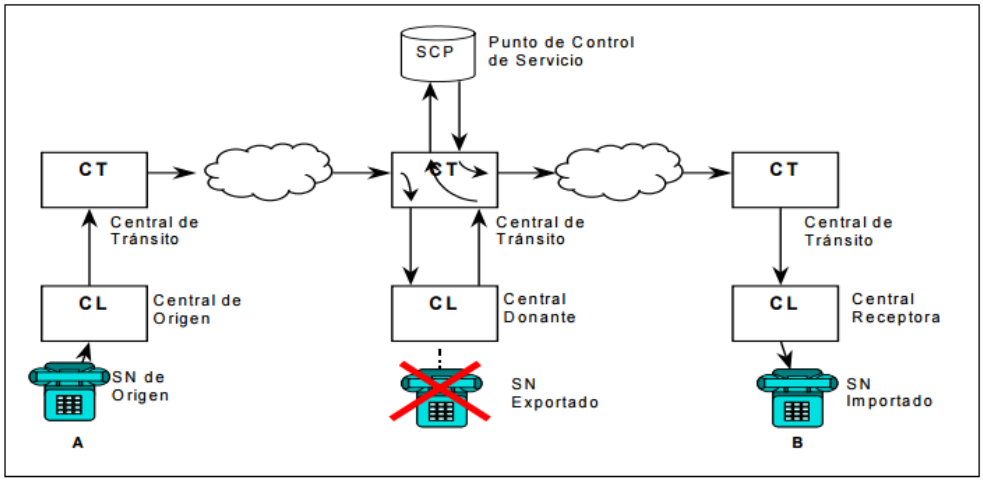


Figura 9 DIAGRAMA DE CONSULTA EN LIBERACIÓN

b) Return to Pivot

La consulta a la base de datos tiene lugar en caso de rechazo de llamada, por parte de la red donante al identificar el número como portado. Difiere

del QoR porque un número es portado desde el conmutador donante. El mensaje de retorno contiene la información relativa al correcto encaminamiento.

c) Consulta de Todas las Llamadas (All Call Query)

En este caso, todas las llamadas generan una consulta a la Red Inteligente para determinar si el número está portado, y de estarlo, bajo qué condiciones y hacia qué operador, central y número de abonado. La consulta a la base de datos se realiza directamente por parte de la red de origen antes del encaminamiento de cualquier llamada, de modo que en el caso de que el número haya sido portado, se enrute directamente, la llamada a la red receptora, sin que la red donante intervenga en la gestión.

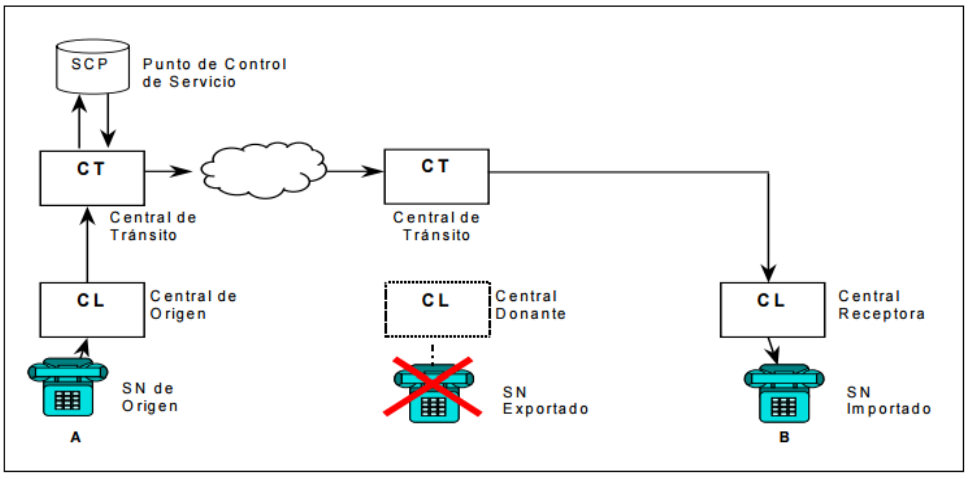


Figura 10 DIAGRAMA DE CONSULTA DE TODAS LAS LLAMADAS

Tabla 13 Ventajas y desventajas de las técnicas de portabilidad más comunes

Métodos de Portabilidad	REENVIO DE LLAMADA (Remote Call Forwarding / Call Forwarding Unconditional)	CONSULTA A BD Query on Release	CONSULTA A BD All Call Query
Ventajas	Implementación rápida. Capacidad de enviar el prefijo de encaminamiento más allá de las fronteras de la red. Uso de funcionalidades ya existentes en las centrales de conmutación.	Método óptimo en calidad ofrecida para poco tráfico de portabilidad. Minimiza el número de consultas necesarias a ser enviadas. Facilita la actualización instantánea de la información de números portados.	Administración de base de datos más sencilla. Método óptimo en cuanto a calidad ofrecida para mucho tráfico de portabilidad. Mayor independencia de recursos de terceros. Permite que el control del establecimiento de la llamada se mantenga por la red de origen siendo más eficiente y flexible.
Desventajas	Dependencia de recursos de terceros. La gestión de fallas es compleja. Resulta costoso para el conmutador donante debido al enlace deficiente a través de dicha central mientras dure la llamada. Uso ineficiente de recursos numéricos, ya que cada usuario "portado" emplea dos números de abonado: uno que es el marcado por el que llama y el otro es usado por la red para el encaminamiento de la llamada.	Implementación no inmediata. Aumento del tiempo necesario para establecer la llamada "portada" en comparación con las llamadas "no portadas	Se hacen consultas para números que finalmente no serán portados.

2.3. SISTEMA

Es un conjunto de partes o elementos, organizados y relacionados que interactúan entre sí para lograr un objetivo. Los sistemas reciben (entrada) datos, energía o materia del ambiente y proveen (salida) información, energía o materia.

En informática existen gran cantidad de sistemas:

- Sistema operativo.
- Sistema experto.
- Sistema informático.
- Aplicación o software.
- Computadora.

2.3.1 Software

El software son los programas y la documentación asociada tal como requisitos, modelos de diseño y manuales de usuario. **(**INTECO**, 2009)**

Los productos software pueden desarrollarse para un cliente particular o se pueden desarrollar para un mercado general. Por lo tanto, los productos software pueden ser:

Genéricos: desarrollados para ser vendidos a un ámbito de clientes diferentes.

Hechos **a medida (personalizados):** desarrollados para un cliente individual de acuerdo a su especificación.

Se puede crear software nuevo desarrollando nuevos programas, configurando sistemas de software genéricos o reutilizando software existente.

a) Atributos de un buen software

El software debe proporcionar la funcionalidad y el rendimiento requeridos a los usuarios y debe ser sostenible, fiable y aceptable. (INTECO, 2009)

i. **Mantenibilidad (capacidad de poder mantenerse):** el software debe evolucionar para cumplir con las necesidades de cambio.

Fiabilidad: el software deber ser digno de confianza.

Eficiencia: el software no debe hacer un uso derrochador de los recursos del sistema.

ii. **Aceptabilidad:** el software debe ser aceptado por los usuarios para los que se diseñó. Esto significa que ha de ser entendible, usable y compatible con otros sistemas.

El software se puede definir como el conjunto de tres componentes:

1. **Programas (instrucciones):** los programas son conjuntos de instrucciones que proporcionan la funcionalidad deseada y el rendimiento cuando se ejecutan. Están escritos usando lenguajes específicos que los ordenadores pueden leer y ejecutar, tales como lenguaje ensamblador, Basic, FORTRAN, COBOL, C… Los programas también pueden ser generados usando generadores de programas.

2. **Datos:** este componente incluye los datos necesarios para manejar y probar los programas y las estructuras requeridas para mantener y manipular estos datos. Los programas proporcionan la funcionalidad requerida manipulando datos. Usan datos para ejercer el control apropiado en lo que hacen. El mantenimiento y las pruebas de los programas también necesitan datos. El diseño del programa asume la disponibilidad de las estructuras de datos tales como bases de datos y archivos que contienen datos.

3. **Documentos:** este componente describe la operación y uso del programa. Además de los programas y los datos, los usuarios necesitan también una explicación de cómo usar el programa. Documentos como manuales de usuario y de operación son necesarios para permitir a los usuarios operar con el sistema. Los documentos también son requeridos por las personas encargadas de mantener el software para entender el interior del software y modificarlo, en el caso en que sea necesario.

Figura 11 Elementos de Software

Es importante contar con una definición exhaustiva del software ya que de otra manera se podrían olvidar algunos componentes. Una percepción común es que el software sólo consiste en programas, sin embargo, los programas no son los únicos componentes del software. **(INTECO, 2009).**

2.3.2 Características del software

Hoy en día, casi todo el mundo se ve afectado por el amplio uso del software, bien como usuario o bien como profesional encargado de construirlo. Los usuarios han de apreciar las ventajas de usar software, mientras que los profesionales necesitan entender las características únicas del software de forma que puedan construir software de alta calidad y realizar proyectos software de forma satisfactoria. Para poder comprender lo que es el software (y consecuentemente la ingeniería del software), es importante examinar las características del software que lo diferencian de otras cosas que el hombre puede construir.

El software es un elemento lógico y se diferencia del hardware, un elemento físico, en sus características.

El software se desarrolla No se fabrica en el sentido clásico. Aunque existen similitudes entre el desarrollo del software y la construcción del hardware, ambas actividades son fundamentalmente distintas. Cada producto software es diferente porque se construye para cumplir los requisitos únicos de un cliente. Cada software necesita, por lo tanto, ser construido usando un enfoque de ingeniería. Construir un producto software implica entender qué es necesario, diseñar el producto para que cumpla los requisitos, implementar el diseño usando un lenguaje de programación y comprobar que el producto cumple con los requisitos.

Todas estas actividades se llevan a cabo mediante la ejecución de un proyecto software y requiere un equipo trabajando de una forma coordinada. En el software, el recurso principal son las personas. No

siempre es posible acelerar la construcción de software añadiendo personas porque el desarrollo de software requiere un esfuerzo en equipo. El equipo tiene que trabajar de forma coordinada y compartir un objetivo de proyecto común. Se necesita comunicación efectiva dentro del equipo. Un nuevo miembro del equipo no es inmediatamente productivo y necesita la iniciación adecuada al equipo y la formación para realizar el trabajo. Esto requiere una inversión de tiempo y esfuerzo por parte de los miembros del equipo existentes y les puede distraer de su propio trabajo.

El ***software no se estropea*** Los defectos no detectados harán que falle el programa durante las primeras etapas de su vida. Sin embargo, una vez que se corrigen (suponiendo que no se introducen nuevos errores) los fallos disminuyen. El software no se estropea, pero se deteriora. Durante su vida, el software sufre cambios (mantenimiento). Conforme se hacen los cambios, es bastante probable que se introduzcan nuevos defectos, lo que hace que el software se vaya deteriorando debido a los cambios.

El software ***se construye a medida*** Otro aspecto del software es que, debido a que la industria del software es nueva, el software se diferencia del hardware en el aspecto relacionado con el uso de componentes. Aunque la mayoría de la industria tiende a ensamblar componentes, en el caso del software, la mayoría **se construye a medida**. Aunque la reutilización y ensamblaje de componentes está aumentando, el software con frecuencia se construye de acuerdo a los requisitos específicos de un cliente. Los componentes reutilizables se han creado para que el ingeniero pueda concentrarse en elementos verdaderamente innovadores

de un diseño. El componente software debería diseñarse e implementarse para que pueda volver a ser reutilizado en muchos programas diferentes. Hoy en día, se ha extendido la visión de la reutilización para abarcar tanto algoritmos como estructuras de datos, permitiendo al ingeniero del software crear nuevas aplicaciones a partir de las partes reutilizables.

El hardware usa componentes estándar con funciones e interfaces bien definidas. La fase de diseño en el ciclo de vida de un producto hardware implica seleccionar los componentes disponibles más adecuados y decidir el enfoque para montarlos. Los componentes de hardware estándar son útiles porque conducen a:

- Reducir el coste y el tiempo de lanzamiento al mercado
- Buena calidad
- Ingeniería rápida
- Fácil mantenimiento
- Fácil mejora

Como la industria del hardware, la industria del software está intentando adoptar el mecanismo de reutilizar para hacer más fácil y más rápida la construcción. Las ventajas de la reutilización de software están siendo entendidas y apreciadas. Existen algunos elementos reutilizables a través de librerías de funciones y objetos reutilizables que combinan funciones y datos. Mientras que la reutilización y el montaje basado en componentes se están incrementando, la mayoría del software continua siendo construido de forma personalizada, y los niveles de reutilización actuales están lejos de los que deberían ser. Además, la tarea de identificar

componentes reutilizables potenciales es difícil porque cada producto software es único y distinto. La industria del software tiene procesos bien definidos para la reutilización de componentes. Esto incluye procesos para la construcción de componentes, almacenamiento de los mismos en librerías de donde se pueden extraer para su reutilización. Con el paso de los años, la industria del software espera crear componentes reutilizables específicos a dominios de aplicación particulares.

2.3.3 Aplicaciones del software

El software puede aplicarse en cualquier situación en la que se haya definido previamente un conjunto específico de pasos procedimentales, es decir, un algoritmo (excepciones notables a esta regla son el software de los sistemas expertos y de redes neuronales). **(INTECO, 2009)**

Algunas veces es difícil establecer categorías genéricas para las aplicaciones del software que sean significativas. Conforme aumenta la complejidad del software, es más difícil establecer compartimentos nítidamente separados. Las siguientes áreas del software indican la amplitud de las aplicaciones potenciales:

Software de sistemas: engloba el conjunto de programas que han sido escritos para servir a otros programas. Algunos programas de sistemas (compiladores, editores y utilidades de gestión de archivos) procesan estructuras de información compleja pero determinada. Otras aplicaciones de sistemas (ciertos componentes del sistema operativo, utilidades de manejo de periféricos, procesadores de telecomunicaciones) procesan datos en gran medida indeterminados. En cualquier caso, el área del

software de sistemas se caracteriza por una fuerte interacción con el hardware de la computadora; una gran utilización por múltiples usuarios; una operación concurrente que requiere una planificación, una compartición de recursos y una sofisticada gestión de procesos; unas estructuras de datos complejas y múltiples interfaces externas.

Software de tiempo real: coordina/analiza/controla sucesos del mundo real conforme ocurren. Entre los elementos del software de tiempo real se incluyen: un componente de adquisición de datos que recolecta y da formato a la información recibida del entorno externo, un componente de análisis que transforma la información según lo requiera la aplicación, un componente de control/salida que responda al entorno externo y un componente de monitorización que coordina todos los demás componentes, de forma que pueda mantenerse el respuesta en tiempo real.

Software de gestión: se ocupa del tratamiento de la información comercial y constituye la mayor de las áreas de aplicación del software. Los sistemas discretos (nóminas, cuentas de haberes-débitos, inventarios) han evolucionado hacia el software de sistemas de información de gestión (SIG) que accede a una o más bases de datos que contienen información comercial. Las aplicaciones en esta área reestructuran los datos existentes para facilitar las operaciones comerciales o gestionar la toma de decisiones. Además de las tareas convencionales de procesamiento de datos, las aplicaciones de software

de gestión también realizan cálculo interactivo (El procesamiento de transacciones en puntos de venta).

Software de ingeniería y científico: se caracteriza por los algoritmos de manejo de números. Las aplicaciones van desde la astronomía a la vulcanología, desde el análisis de la presión de los automotores a la dinámica orbital de las lanzaderas espaciales y desde la biología molecular a la fabricación automática. Sin embargo las nuevas aplicaciones del área de ingeniería/ciencia se han alejado de los algoritmos convencionales numéricos. El diseño asistido por computadora (CAD), la simulación de sistemas y otras aplicaciones interactivas, han comenzado a coger características del software de tiempo real e incluso de software de sistemas.

Software empotrado: los productos inteligentes se han convertido en algo común en casi todos los mercados de consumo e industriales. El software empotrado reside en memoria de sólo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumo. El software empotrado puede ejecutar funciones muy limitadas y curiosas (El control de las teclas de un horno microondas) o suministrar una función significativa y con capacidad de control (Funciones digitales en un automóvil, tales como control de la gasolina, indicadores en el salpicadero, sistemas de frenado)

Software de computadoras personales: el mercado del software de computadoras personales ha germinado en las pasadas décadas. El procesamiento de textos, las hojas de cálculo, los gráficos por

computadora, multimedia, entretenimiento, gestión de bases de datos, aplicaciones financieras, de negocios y personales y redes o acceso a bases de datos externas son algunas de los cientos de aplicaciones.

Software basado en web: las páginas web buscadas por un explorador son software que incorpora instrucciones ejecutables y datos.

Software de inteligencia artificial: hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados el cálculo o el análisis directo. Los sistemas expertos, también llamados sistemas basados en el conocimiento, reconocimiento de patrones (imágenes y voz), redes neuronales artificiales, prueba de teoremas y los juegos son representativos de las aplicaciones de esta categoría.

2.3.4 Ingeniería del software.

La ingeniería del software es una disciplina de ingeniería preocupada por todos los aspectos de la producción de software desde las primeras etapas de especificación del sistema hasta el mantenimiento del sistema después de que éste se haya puesto en uso. Se preocupa de las teorías, métodos y herramientas para el desarrollo profesional de software. La ingeniería del software se preocupa del desarrollo de software rentable.

La ingeniería del software debería adoptar un enfoque sistemático y organizado para su trabajo y usar las herramientas y técnicas apropiadas dependiendo del problema a solucionar, las restricciones de desarrollo y los recursos disponibles. **(INTECO, 2009)**

La ingeniería del software es la aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software, que es la aplicación de la ingeniería del software. **(IEEE, 1990)**

El enfoque sistemático, disciplinado y cuantificable es con frecuencia calificado de modelo de proceso de software (en el sentido general) o de proceso de desarrollo de software (en el sentido específico). El proceso de desarrollo de software específico consiste en un conjunto particular de prácticas de desarrollo de software que son realizadas por el ingeniero de software en un orden predeterminado.

Los ingenieros adoptan un enfoque sistemático y organizacional para su trabajo. Cuando se habla de prácticas de desarrollo de software se hace referencia a un requisito empleado para recomendar un enfoque disciplinado y uniforme del proceso de desarrollo de software, es decir, una actividad bien definida que contribuye a la satisfacción de los objetivos del proyecto; generalmente la salida de una práctica se convierte en la entrada de otra. Entre las prácticas de desarrollo de software se encuentran las siguientes (las prácticas pueden depender en base al proceso y la terminología asociada al mismo):

- Ingeniería de requisitos
- Análisis de sistemas
- Diseño/arquitectura a alto nivel
- Diseño a bajo nivel
- Codificación

- Integración
- Diseño y revisiones de código
- Pruebas
- Mantenimiento
- Gestión de proyectos
- Gestión de la configuración

La mayoría de las disciplinas reconocen algunas prácticas como mejores prácticas. Una mejor práctica es una práctica que, a través de la experiencia e investigación, se ha probado que lleva al resultado deseado fiablemente y se considera prudente y recomendable hacerla en una variedad de contextos

a) Desafíos de la Ingeniería del software

Los desafíos clave con los que se enfrenta la ingeniería del software son:

Heterogeneidad: desarrollando técnicas para construir software que puedan utilizar plataformas y entornos de ejecución heterogéneos.

Entrega: desarrollando técnicas que lleven a una entrega de software más rápida.

Confianza: desarrollando técnicas que demuestren que los usuarios pueden tener confianza en el software.

b) Capas de la Ingeniería del Software

El enfoque de ingeniería del software cuenta con un compromiso organizacional con la calidad porque no es posible incorporar la ingeniería

del software en una organización que no está centrada en conseguir calidad. La ingeniería del software es una tecnología multicapa. Se puede ver como un conjunto de componentes estratificados, que reposan sobre ese enfoque de calidad.

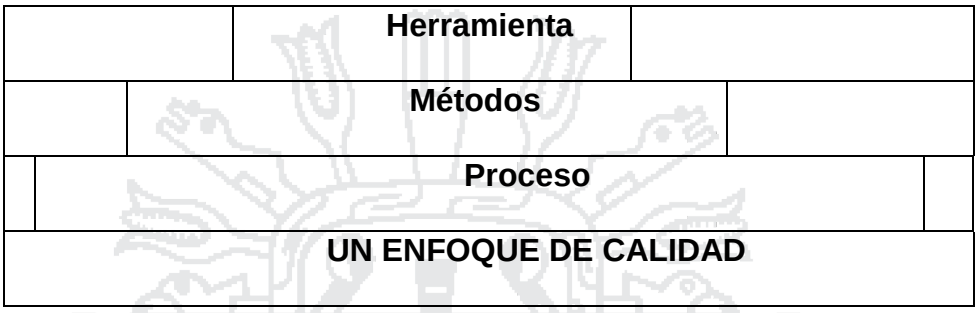


Figura 12 PROCESO, MÉTODOS Y HERRAMIENTAS.

Estos componentes que forman parte de la ingeniería del software son:

- Procesos: un marco de trabajo que ayuda al jefe de proyecto a controlar la gestión del proyecto y las actividades de ingeniería.
- Métodos: las actividades técnicas requeridas para la creación de productos de trabajo.
- Herramientas: la ayuda automatizada para los procesos y métodos.

Procesos: El fundamento de la ingeniería del software es la capa de proceso. El proceso define un marco de trabajo para un conjunto de áreas clave de proceso que se deben establecer para la entrega efectiva de la tecnología de la ingeniería del software. La capa de proceso define el proceso que se usará para construir el software y las actividades y tareas que un jefe de proyecto tiene que gestionar. Por lo tanto, las áreas claves del proceso forman la base del control de gestión de proyectos del software y establecen el contexto en el que se aplican los métodos técnicos, se obtienen productos de trabajo (modelos, documentos, datos, informes, formularios), se establecen hitos, se asegura la calidad y el

cambio se gestiona adecuadamente. El proceso de la ingeniería del software es la unión que mantiene juntas las capas de tecnologías y que permite un desarrollo racional y oportuno de la ingeniería del software. Se pueden ver todas las actividades, incluyendo las actividades técnicas, como parte del proceso. Además, cualquier recurso, incluyendo herramientas usadas para construir el software también encajan en el proceso. La capa de proceso es, por lo tanto, el fundamento de la ingeniería del software y da soporte a las capas de métodos y herramientas. Todos los enfoques de la construcción de software tienen un proceso, pero en muchos casos, son ad hoc, invisibles y caóticos. Una buena ingeniería de software hace que el proceso de software sea más visible, predecible y más útil para aquellos que construyen software.

Métodos: La capa de métodos se centra en las actividades técnicas que se deben realizar para conseguir las tareas de ingeniería. Proporciona el "cómo" y cubre las actividades de ingeniería fundamentales. Los métodos abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento. Los métodos de la ingeniería del software dependen de un conjunto de principios básicos que gobiernan cada una de las áreas de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas. La construcción de software implica una amplia colección de actividades técnicas. La capa de métodos contiene los métodos definidos para realizar esas actividades de forma eficiente. Se centra en cómo se han de realizar las actividades técnicas. Los personas involucradas usan los métodos para realizar las actividades de ingeniería fundamentales

necesarias para construir el software. Para varias actividades de proceso, la capa de métodos contiene el correspondiente conjunto de métodos técnicos para usar. Esto abarca un conjunto de reglas, los modos de representación gráficos o basados en texto, y las guías relacionadas para la evaluación de la calidad de la información representada.

Para definir la capa de métodos, es necesario seleccionar un método adecuado de un amplio rango de métodos disponibles.

Herramientas: La capa de herramientas proporciona soporte a las capas de proceso y métodos centrándose en el significado de la automatización de algunas de las actividades manuales. Las herramientas se pueden utilizar para automatizar las siguientes actividades:

- Actividades de gestión de proyectos.
- Métodos técnicos usados en la ingeniería del software.
- Soporte de sistemas general.
- Marcos de trabajo para otras herramientas.

La automatización ayuda a eliminar el tedio del trabajo, reduce las posibilidades de errores, y hace más fácil usar buenas prácticas de ingeniería del software. Cuando se usan herramientas, la documentación se convierte en una parte integral del trabajo hecho, en vez de ser una actividad adicional. De ahí que la documentación no se tenga que realizar como actividad adicional. Las herramientas se pueden utilizar para realizar actividades de gestión de proyecto así como para actividades técnicas. Existen una gran variedad de herramientas para múltiples actividades. Entre ellas se pueden destacar las siguientes:

- Herramientas de gestión de proyectos.
- Herramientas de control de cambios.
- Herramientas de análisis y diseño.
- Herramientas de generación de código.
- Herramientas de pruebas.
- Herramientas de reingeniería.
- Herramientas de documentación.
- Herramientas de prototipos.

Estas herramientas soportan las capas de proceso y de métodos en varias actividades.

c) El ciclo de vida de desarrollo del software

El ciclo de vida es el conjunto de fases por las que pasa el sistema que se está desarrollando desde que nace la idea inicial hasta que el software es retirado o remplazado (muere). También se denomina a veces paradigma. Entre las funciones que debe tener un ciclo de vida se pueden destacar:

- Determinar el orden de las fases del proceso de software.
- Establecer los criterios de transición para pasar de una fase a la siguiente.
- Definir las entradas y salidas de cada fase.
- Describir los estados por los que pasa el producto.
- Describir las actividades a realizar para transformar el producto.
- Definir un esquema que sirve como base para planificar, organizar, coordinar, desarrollar…

Un ciclo de vida para un proyecto se compone de fases sucesivas compuestas por tareas que se pueden planificar. Según el modelo de ciclo de vida, la sucesión de fases puede ampliarse con bucles de realimentación, de manera que lo que conceptualmente se considera una misma fase se pueda ejecutar más de una vez a lo largo de un proyecto, recibiendo en cada pasada de ejecución aportaciones a los resultados intermedios que se van produciendo (realimentación).

Fases: una fase es un conjunto de actividades relacionadas con un objetivo en el desarrollo del proyecto. Se construye agrupando tareas (actividades elementales) que pueden compartir un tramo determinado del tiempo de vida de un proyecto. La agrupación temporal de tareas impone requisitos temporales correspondientes a la asignación de recursos (humanos, financieros o materiales).

Entregables: son los productos intermedios que generan las fases. Pueden ser materiales o inmateriales (documentos, software). Los entregables permiten evaluar la marcha del proyecto mediante comprobaciones de su adecuación o no a los requisitos funcionales y de condiciones de realización previamente establecidos.

Las actividades genéricas del ciclo de vida del desarrollo del software son:

- ***Especificación:*** lo que el sistema debería hacer y sus restricciones de desarrollo.
- ***Desarrollo:*** producción del sistema software.
- ***Validación:*** comprobar que el sistema es lo que el cliente quiere.

- ***Evolución:*** cambiar el software en respuesta a las demandas de cambio.

d) Modelos de ciclo de vida del software

La ingeniería del software se vale de una serie de modelos que establecen y muestran las distintas etapas y estados por los que pasa un producto software, desde su concepción inicial, pasando por su desarrollo, puesta en marcha y posterior mantenimiento, hasta la retirada del producto.

Los modelos de ciclo de vida del software describen las fases del ciclo de software y el orden en que se ejecutan las fases. Un modelo de ciclo de vida de software es una vista de las actividades que ocurren durante el desarrollo de software, intenta determinar el orden de las etapas involucradas y los criterios de transición asociados entre estas etapas.

A estos modelos se les denomina “Modelos de ciclo de vida del software”. El primer modelo concebido fue el de Royce, más comúnmente conocido como “Cascada” o “Lineal Secuencial”. Este modelo establece que las diversas actividades que se van realizando al desarrollar un producto software, se suceden de forma lineal.

Un modelo de ciclo de vida del software:

- Describe las fases principales de desarrollo de software.
- Define las fases primarias esperadas de ser ejecutadas durante esas fases.
- Ayuda a administrar el progreso del desarrollo.

- Provee un espacio de trabajo para la definición de un proceso detallado de desarrollo de software.

Las principales diferencias entre distintos modelos de ciclo de vida están en:

- El alcance del ciclo dependiendo de hasta dónde llegue el proyecto correspondiente. Un proyecto puede comprender un simple estudio de viabilidad del desarrollo de un producto, o su desarrollo completo o en el extremo, toda la historia del producto con su desarrollo, fabricación y modificaciones posteriores hasta su retirada del mercado.
- Las características (contenidos) de las fases en que dividen el ciclo. Esto puede depender del propio tema al que se refiere el proyecto, o de la organización.
- La estructura y la sucesión de las etapas, si hay realimentación entre ellas, y si tenemos libertad de repetirlas (iterar).

i) Modelo en cascada

Es un enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de forma que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.

El modelo en cascada es un proceso de desarrollo secuencial, en el que el desarrollo se ve fluyendo hacia abajo (como una cascada) sobre las fases que componen el ciclo de vida.

En el modelo original de Royce, existían las siguientes fases:

1. Especificación de requisitos
2. Diseño
3. Construcción (Implementación o codificación)
4. Integración
5. Pruebas
6. Instalación
7. Mantenimiento

Para seguir el modelo en cascada, se avanza de una fase a la siguiente en una forma puramente secuencial.

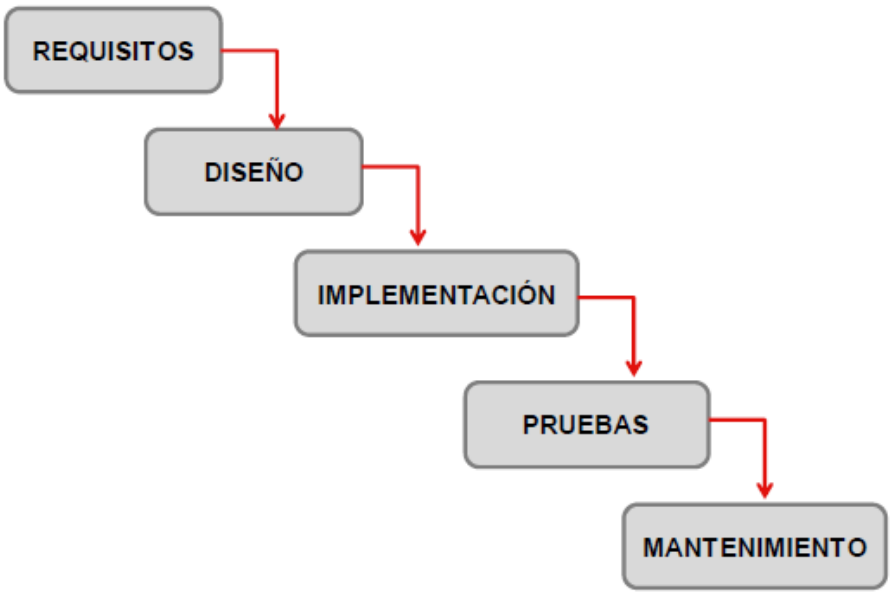


Figura 13 DIAGRAMA MODELO EN CASCADA

ii). Modelo en V

El modelo en v se desarrolló para terminar con algunos de los problemas que se vieron utilizando el enfoque de cascada tradicional. Los defectos estaban siendo encontrados demasiado tarde en el ciclo de vida, ya que las pruebas no se introducían hasta el final del proyecto. El modelo en v

dice que las pruebas necesitan empezarse lo más pronto posible en el ciclo de vida. También muestra que las pruebas no son sólo una actividad basada en la ejecución. Hay una variedad de actividades que se han de realizar antes del fin de la fase de codificación. Estas actividades deberían ser llevadas a cabo en paralelo con las actividades de desarrollo, y los técnicos de pruebas necesitan trabajar con los desarrolladores y analistas de negocio de tal forma que puedan realizar estas actividades y tareas y producir una serie de entregables de pruebas.

El modelo en v es un proceso que representa la secuencia de pasos en el desarrollo del ciclo de vida de un proyecto. Describe las actividades y resultados que han de ser producidos durante el desarrollo del producto. La parte izquierda de la v representa la descomposición de los requisitos y la creación de las especificaciones del sistema. El lado derecho de la v representa la integración de partes y su verificación. V significa "Validación y Verificación".

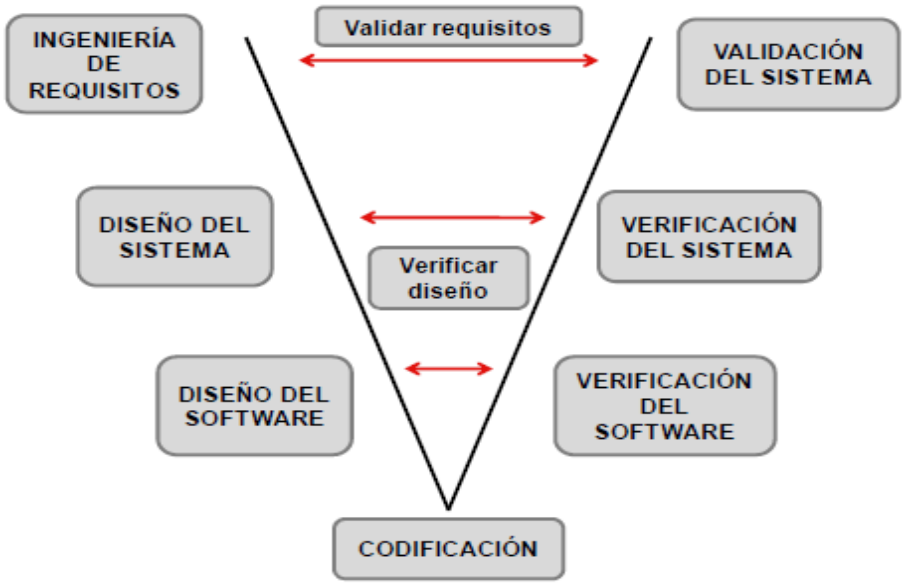


Figura 14 DIAGRAMA MODELO EN V

Realmente las etapas individuales del proceso pueden ser casi las mismas que las del modelo en cascada. Sin embargo hay una gran diferencia. En vez de ir para abajo de una forma lineal las fases del proceso vuelven hacia arriba tras la fase de codificación, formando una v. La razón de esto es que para cada una de las fases de diseño se ha encontrado que hay un homólogo en las fases de pruebas que se correlacionan.

iii). Modelo iterativo

Es un modelo derivado del ciclo de vida en cascada. Este modelo busca reducir el riesgo que surge entre las necesidades del usuario y el producto final por malos entendidos durante la etapa de recogida de requisitos.

Consiste en la iteración de varios ciclos de vida en cascada. Al final de cada iteración se le entrega al cliente una versión mejorada o con mayores funcionalidades del producto. El cliente es quien, después de cada iteración, evalúa el producto y lo corrige o propone mejoras. Estas iteraciones se repetirán hasta obtener un producto que satisfaga las necesidades del cliente.

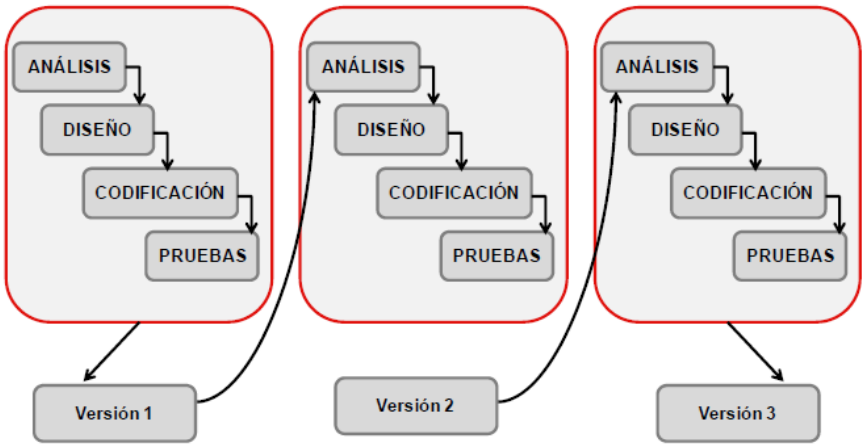


Figura 15 DIAGRAMA MODELO INTERATIVO

Este modelo se suele utilizar en proyectos en los que los requisitos no están claros por parte del usuario, por lo que se hace necesaria la creación de distintos prototipos para presentarlos y conseguir la conformidad del cliente.

iv). Modelo de desarrollo incremental

El modelo incremental combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos. Se basa en la filosofía de construir incrementando las funcionalidades del programa. Este modelo aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal produce un incremento del software.

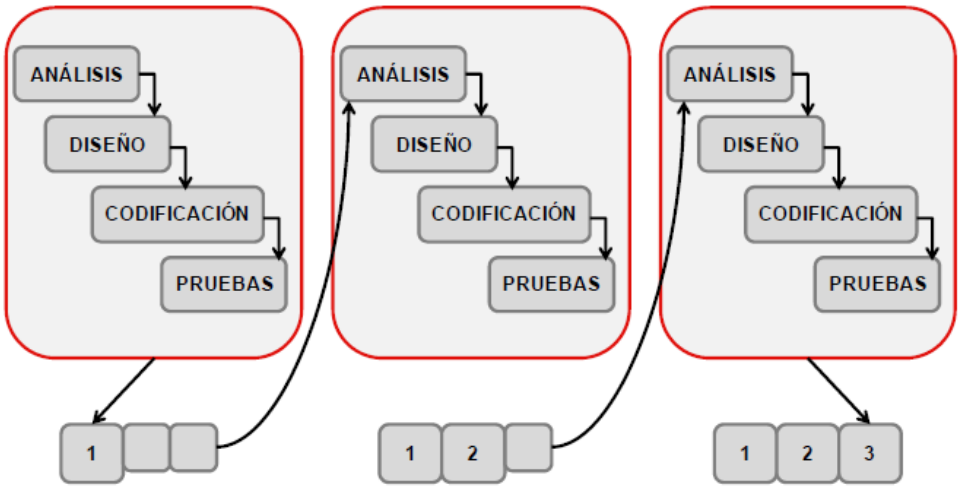


Figura 16 DIAGRAMA DESARROLLO INCREMENTAL

Cuando se utiliza un modelo incremental, el primer incremento es a menudo un producto esencial, sólo con los requisitos básicos. Este modelo se centra en la entrega de un producto operativo con cada incremento. Los primeros incrementos son versiones incompletas del

producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación.

v). Modelo en espiral

El desarrollo en espiral es un modelo de ciclo de vida desarrollado por Barry Boehm en 1985, utilizado de forma generalizada en la ingeniería del software. Las actividades de este modelo se conforman en una espiral, cada bucle representa un conjunto de actividades. Las actividades no están fijadas a priori, sino que las siguientes se eligen en función del análisis de riesgos, comenzando por el bucle anterior.

Al ser un modelo de ciclo de vida orientado a la gestión de riesgos se dice que uno de los aspectos fundamentales de su éxito radica en que el equipo que lo aplique tenga la necesaria experiencia y habilidad para detectar y catalogar correctamente riesgos.

Tareas:

Para cada ciclo habrá cuatro actividades:

1. Determinar o fijar objetivos:
    - Fijar también los productos definidos a obtener: requerimientos, especificación, manual de usuario.
    - Fijar las restricciones.
    - Identificar riesgos del proyecto y estrategias alternativas para evitarlos.
    - Hay una cosa que solo se hace una vez: planificación inicial o previa.

2. Análisis del riesgo:

   - Estudiar todos los riesgos potenciales y se seleccionan una o varias alternativas propuestas para reducir o eliminar los riesgos

3. Desarrollar, verificar y validar (probar):

   - Tareas de la actividad propia y de prueba.
   - Análisis de alternativas e identificación de resolución de riesgos.
   - Dependiendo del resultado de la evaluación de riesgos, se elige un modelo para el desarrollo, que puede ser cualquiera de los otros existentes, como formal, evolutivo, cascada. Así, si los riesgos de la interfaz de usuario son dominantes, un modelo de desarrollo apropiado podría ser la construcción de prototipos evolutivos.

4. Planificar:

   - Revisar todo lo que se ha llevado a cabo, evaluándolo y decidiendo si se continua con las fases siguientes y planificando la próxima actividad.

El proceso empieza en la posición central. Desde allí se mueve en el sentido de las agujas del reloj.

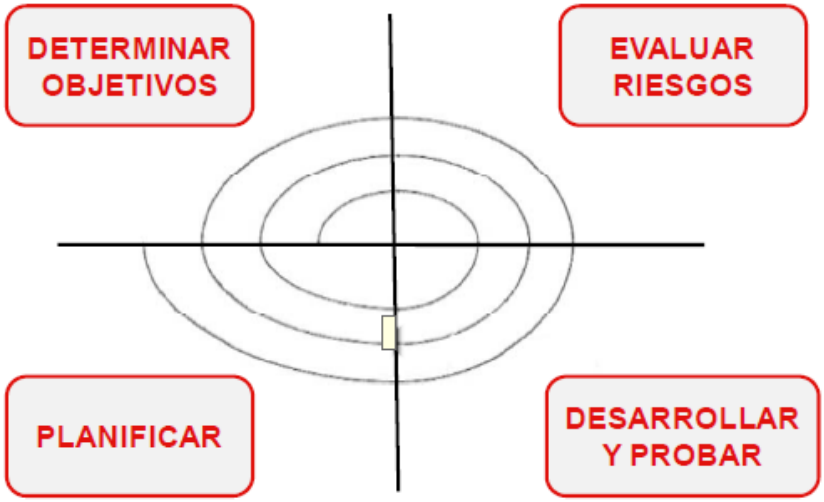


Figura 17 DIAGRAMA MODELO ESPIRAL

vi). Modelo de prototipos

El paradigma de construcción de prototipos comienza con la recolección de requisitos. El desarrollador y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas del esquema en donde es obligatoria más definición. Entonces aparece un diseño rápido. El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente. El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

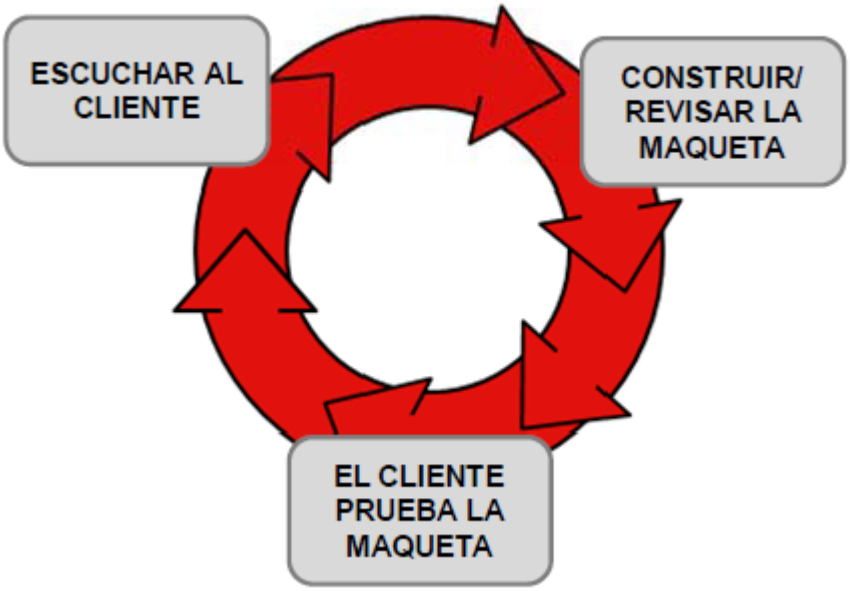


Figura 18 DIAGRAMA MODELO PROTOTIPOS

vii). ISO/IEC 12207

Esta norma establece un marco de referencia común para los procesos del ciclo de vida del software, con una terminología bien definida a la que puede hacer referencia la industria del software. Contiene procesos, actividades y tareas para aplicar durante la adquisición de un sistema que contiene software, un producto software puro o un servicio software, y durante el suministro, desarrollo, operación y mantenimiento de productos software. El software incluye la parte software del firmware.

Esta norma incluye también un proceso que puede emplearse para definir, controlar y mejorar los procesos del ciclo de vida del software.

La ISO 12207 define **un modelo de ciclo de vida** como un marco de referencia que contiene los procesos, actividades y tareas involucradas en el desarrollo, operación y mantenimiento de un producto software, y que

abarca toda la vida del sistema, desde la definición de sus requisitos hasta el final del uso.

Esta norma agrupa las actividades que pueden llevarse a cabo durante el ciclo de vida del software en cinco procesos principales, ocho procesos de apoyo y cuatro procesos organizativos. Cada proceso del ciclo de vida está dividido en un conjunto de actividades; cada actividad se subdivide a su vez en un conjunto de tareas.

- Procesos principales del ciclo de vida: son cinco procesos que dan servicio a las partes principales durante el ciclo de vida del software. Una parte principal es la que inicia o lleva a cabo el desarrollo, operación y mantenimiento de productos software. Los procesos principales son:

  1. Proceso de adquisición
  2. Proceso de suministro
  3. Proceso de desarrollo
  4. Proceso de operación
  5. Proceso de mantenimiento

- Procesos de apoyo al ciclo de vida: son procesos que apoyan a otros procesos como parte esencial de los mismos, con un propósito bien definido, y contribuyen al éxito y calidad del proyecto software. Un proceso de apoyo se emplea y ejecuta por otro proceso según sus necesidades.

Los procesos de apoyo son:

- Proceso de documentación
- Proceso de gestión de la configuración
- Proceso de verificación
- Proceso de validación
- Proceso de revisiones conjuntas
- Proceso de auditoría
- Proceso de solución de problemas

- Procesos organizativos del ciclo de vida: se emplean por una organización para establecer e implementar una infraestructura construida por procesos y personal asociado al ciclo de vida, y para mejorar continuamente esta estructura y procesos.

- Proceso de gestión
- Proceso de infraestructura
- Proceso de mejora
- Proceso de formación

2.3.5 Modelamiento UML

a) Lenguaje de Modelamiento Unificado

UML (Unifed Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar

Jacobson y JimRumbaugh. Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que basar la construcción de sus herramientas (ComputerAided Software Engineering (Ingeniería de Software Asistido por Computadora)). En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores, en un trabajo conjunto por la estandarización de una metodología y simbología de modelamiento.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa (principalmente Booch, OMT y OOSE). UML ha puesto fin a las llamadas "guerras de métodos" que se han mantenido a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los Ingenieros de Software que trabajan en el desarrollo orientado a objetos.

Uno de los objetivos principales de la creación de UML era posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado. Para ello era necesario definir una notación y semántica común. En la figura superior se puede ver cuál ha sido la evolución de UML hasta la creación de UML 1.3, en el que se basa este documento. Hay que tener en cuenta que el estándar UML no define un

proceso de desarrollo específico, tan solo se trata de una notación.(Craig, 2011).

i. Diagramas de Casos de Uso

El Caso de Uso es una excelente herramienta para estimular a que los usuarios hablen, de un sistema, desde sus propios puntos de vista. No siempre es fácil para los usuarios explicar cómo pretenden utilizar un sistema.

La idea es involucrar a los usuarios en las etapas iníciales del análisis y diseño del sistema. Esto aumenta la probabilidad de que el sistema sea de mayor provecho para la gente a la que ayudara, en lugar de ser un manojo de expresiones de computación incomprensible e inmanejable por los usuarios finales.

ii. Diagramas de Secuencia

Permite plasmar es una secuencia grafica los diferentes pasos a realizar para la obtención de un proyecto, no tendremos mayores problemas con el nuestro puesto que la mayoría de estos casos tiene que ser documentada.

iii. Diagramas de Flujo

Aquí podremos graficar la secuencia de acciones y sub-acciones requeridas bajo condiciones o bucles de iteración repetitiva, en fin una secuencia grafica de ejecución, para nuestro caso particular de la automatización de la remasterización de la Imagen ISO original hasta la

instalación de nuevos paquetes, configuraciones varias y finalmente la generación de la Imagen ISO final y la puesta en prueba de esta.

2.3.6 Metodologías de desarrollo de software

Las metodologías han evolucionado de manera significativa en las últimas décadas. Permitiendo así el éxito o el fracaso de muchos de los sistemas desarrollados para distintas áreas. **(**Valdéz**, 2014)**

Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo. **(**INTECO**, 2009)**

Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, incremental…). Definen artefactos, roles y actividades, junto con prácticas y técnicas recomendadas.

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo por el cual fue creado.

Una definición estándar de metodología puede ser el conjunto de métodos que se utilizan en una determinada actividad con el fin de formalizarla y

optimizarla. Determina los pasos a seguir y cómo realizarlos para finalizar una tarea.

Metodologías tradicionales y ágiles

Desarrollar un buen software depende de un gran número de actividades y etapas, donde el impacto de elegir la metodología para un equipo en un determinado proyecto es trascendental para el éxito del producto.

Según la filosofía de desarrollo se pueden clasificar las metodologías en dos grupos. Las metodologías tradicionales, que se basan en una fuerte planificación durante todo el desarrollo, y las metodologías ágiles, en las que el desarrollo de software es incremental, cooperativo, sencillo y adaptado.

a) *Metodologías tradicionales* Las metodologías tradicionales son denominadas, a veces, de forma peyorativa, como metodologías pesadas. Centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto.

Otra de las características importantes dentro de este enfoque, son los altos costes al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil.

Las metodologías tradicionales (formales) se focalizan en la documentación, planificación y procesos (plantillas, técnicas de administración, revisiones.)

b) *Metodologías ágiles* Este enfoque nace como respuesta a los problemas que puedan ocasionar las metodologías tradicionales y se basa en dos aspectos fundamentales, retrasar las decisiones y la planificación adaptativa. Basan su fundamento en la adaptabilidad de los procesos de desarrollo.

Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan.

2.3.7 Programación Extrema (XP)

La Programación Extrema (XP) es una metodología ágil basada en una serie de valores y de prácticas de buenas maneras que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas.

Este modelo de programación se basa en una serie de metodologías de desarrollo de software en la que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación.

Una de las características principales de este método de programación, es que sus ingredientes son conocidos desde el principio de la informática. Los autores de XP han seleccionado aquellos que han considerado mejores y han profundizado en sus relaciones y en cómo se refuerzan los unos con los otros. El resultado de esta selección ha sido esta metodología única y compacta. Por esto, aunque no está basada en principios nuevos, sí que el resultado es una nueva manera de ver el desarrollo de software.

El objetivo que se perseguía en el momento de crear esta metodología era la búsqueda de un método que hiciera que los desarrollos fueran más sencillos. Aplicando el sentido común.

XP propone un ciclo de vida dinámico, donde se admite expresamente que, en muchos casos, los clientes no son capaces de especificar sus requerimientos al comienzo de un proyecto.

Por esto, se trata de realizar ciclos de desarrollo cortos (llamados iteraciones), con entregables funcionales al finalizar cada ciclo. En cada iteración se realiza un ciclo completo de análisis, diseño, desarrollo y pruebas, pero utilizando un conjunto de reglas y prácticas que caracterizan a XP.

Típicamente un proyecto con XP lleva 10 a 15 ciclos o iteraciones. La Figura N° XX esquematiza los ciclos de desarrollo en cascada e iterativos tradicionales (incremental o espiral), comparados con el de XP.

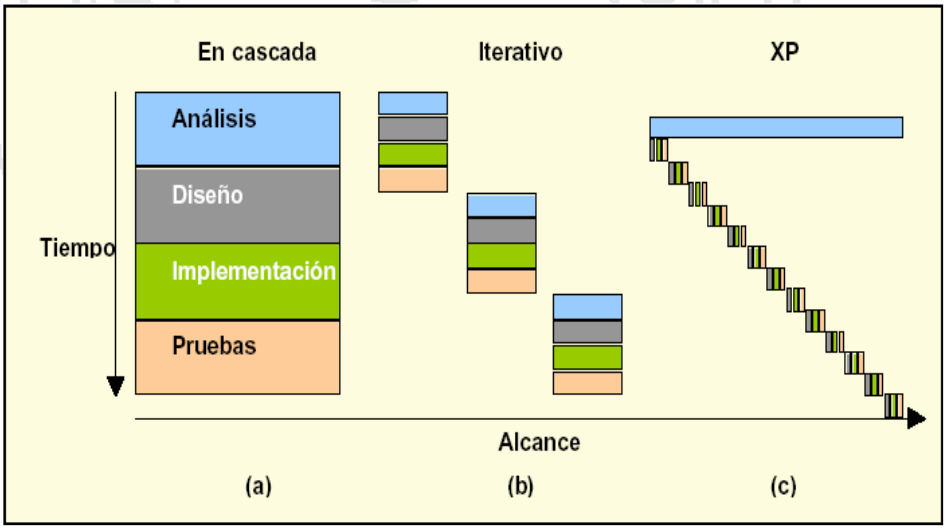


Figura 19 CICLO DE VIDA DE LA METODOLOGÍA XP

Análisis

La Metodología XP plantea en análisis como un permanente diálogo entre la parte empresarial y técnica del proyecto, en la que los primeros decidirán el alcance ¿Qué es lo realmente necesario del proyecto?, la prioridad qué debe ser hecho en primer lugar, la composición de las versiones que debería incluir.

Diseño

El propósito del diseño es de crear una arquitectura para la naciente implementación, el diseño arquitectural sólo puede comenzar una vez que el equipo tenga un entendimiento razonable de los requerimientos del sistema. El diseño, como el análisis, nunca termina realmente hasta que el sistema final es entregado.

Desarrollo

- Esta etapa debe reunir las siguientes características o cualidades.
- El cliente está siempre disponible
- Se debe escribir código de acuerdo a los estándares
- Desarrollar la unidad de pruebas primero
- Todo el código debe programarse por parejas
- Integrar frecuentemente
- Todo el código es común a todos

Prueba

Todo el código debe ir acompañando, Los casos de prueba se escriben antes que el código. Los desarrolladores escriben pruebas unitarias y los clientes especifican pruebas funcionales.

2.3.8 Métricas del proyecto

Se aplica las métricas para valorar la calidad de los productos de ingeniería o los sistemas que se construyen.

Proporcionan una manera sistemática de valorar la calidad basándose en un conjunto de reglas claramente definidas. Se aplican a todo el ciclo de vida permitiendo descubrir y corregir problemas potenciales.

La utilización de métricas para el proyecto tiene 2 aspectos fundamentales:

- Estas métricas se utilizan para minimizar la planificación de desarrollo haciendo los ajustes necesarios que eviten retrasos y reduzcan problemas y riesgos potenciales.
- Las métricas para el proyecto se utilizan para evaluar la calidad de los productos en el momento actual y cuando sea necesario, modificando el enfoque técnico que mejore la calidad.

2.3.9 Mediciones del software

Se pueden categorizar dos medidas:

• Medidas Directas: Como costo y esfuerzo aplicado, líneas de código (LDC) producidas, velocidad de ejecución, tamaño de memoria y los defectos durante un periodo de tiempo establecido.

• Medidas Indirectas: incluyen funcionalidad, calidad, complejidad, eficiencia, fiabilidad, facilidad de mantenimiento, y muchas otras "capacidades".

2.3.10 Métricas de calidad del software

- Los requisitos del Software son la base de las medidas de calidad. La falta de concordancia con los requisitos es una falta de calidad.
- Unos estándares específicos definen un conjunto de criterios de desarrollo que guían la manera en que se hace la ingeniería del Software. Si no se siguen los criterios, habrá seguramente poca calidad.
- Existe un conjunto de requisitos implícitos que a menudo no se nombran. Si el software cumple con sus requisitos explícitos pero falla en los implícitos, la calidad del software no será fiable.

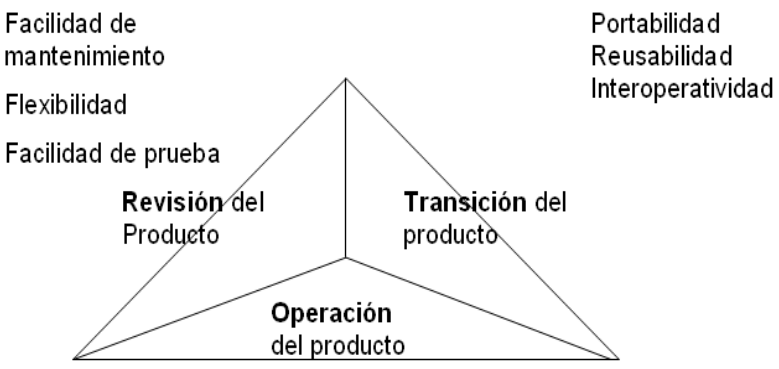


Figura 20 MÉTRICAS DE CALIDAD DEL SOFTWARE.

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1 TIPO Y DISEÑO DE LA INVESTIGACIÓN

El presente trabajo de investigación, de acuerdo con las características del problema, los objetivos y la hipótesis se enmarca dentro del tipo aplicado.

3.1.1 Población

La población de estudio está formada por el personal que usara el sistema de la empresa AMITEL S.A.C. (nueve usuarios).

3.1.2 Muestra

La Muestra de estudio está formada por todo el personal que usará el sistema de la empresa AMITEL S.A.C. nueve usuarios.

3.1.3 Localización

El presente trabajo de investigación se realizó en el área de Desarrollo de sistemas de la empresa AMITEL S.A.C.

Dirección: Jr. Arequipa 1024

Coordenadas: Latitud: -15.84313; **Longitud:** -70.02656

Figura 21 LOCALIZACIÓN DELA INVESTIGACIÓN

3.2 MÉTODOS

3.2.1 Método de recolección de datos

Para la evaluación de la calidad del producto del software, las repuestas fueron recopiladas en la ficha de evaluación de la calidad de producto de software con el ISO-9126, los cuales se encuentran en los Anexos N° I

3.2.2 Desarrollo del sistema

El desarrollo del sistema implementación de un software para el servicio de portabilidad numérica de telefonía fija/móvil usando el protocolo SOAP/WSDL para la empresa AMITEL S.A.C – PUNO, 2015, está basado de acuerdo a los procedimientos establecidos por la metodología de desarrollo ágil XP (Programación Extrema), debido que esta metodología propone un ciclo de vida dinámico y se tiene las especificaciones del requerimiento de la empresa por etapas, esto nos permite realizar ciclos de desarrollo cortos (llamados iteraciones), con entregas funcionales de software al finalizar cada ciclo. En cada iteración se realizó el ciclo completo de análisis, diseño, desarrollo y pruebas, utilizando un conjunto

de reglas y prácticas que recomienda la modelo de desarrollo de software ágil XP.

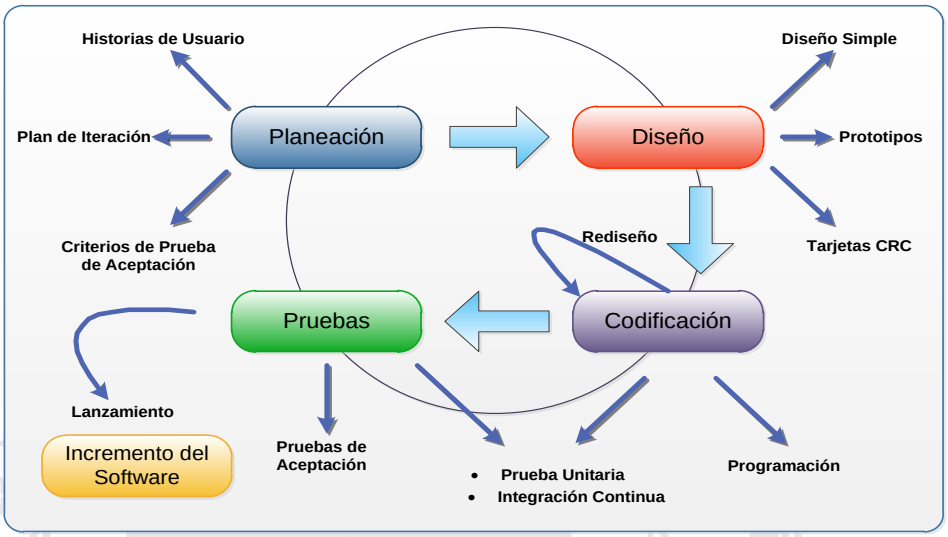


Figura 22 PROGRAMACIÓN EXTREMA XP

3.2.3 Metodología de desarrollo

Se utilizó la Metodología de desarrollo de software Ágil Programación Extrema XP por la adaptabilidad del requerimiento y para el modelado se utilizó UML por ser de uso casi universal.

3.2.4 Proceso de implementación XP

a) Planeación:

Se utiliza historias de usuario: Se analiza el requerimiento para lo cual se toma como base el documento oficial NPC-PER-INTSPEC-001 Issue 1 Rev 1-spa titulado *iconectiv Number Portability Clearinghouse Especificación de Interfaces para el ABDC,* donde se encuentra toda la especificación técnica para el desarrollo del software.

Se crean los Planes de Entrega, se define plazos de entrega como lo recomienda la metodología.

Se llevan a cabo la Planificación de Iteración: se evalúa las iteraciones prioritarias para la implementación.

Se desarrollan Reuniones Diarias, con el fin de facilitar la comunicación entre el grupo de trabajo de portabilidad numérica de la empresa y la exposición de los diferentes problemas.

b) Diseño:

- Se genera diagramas de uso
- Se genera diagramas de secuencia
- Se genera Diagrama de eventos
- Se utiliza UML para modelar objetos
- Se ignoran las funcionalidades extra que podrían incorporarse al proyecto, se centra en lo principal.
- Se remueve la redundancia, se eliminan las funcionalidades no necesarias y se renuevan los diseños obsoletos.

c) Codificación:

- Se utilizan estándares para escribir el código CAMELCASE.
- Se crean las pruebas antes de empezar a codificar, lo cual hará más sencillas y efectivas las pruebas.
- Se realizó en equipos de trabajo y luego se llevó a cabo una integración paralela (debido a esta integración no se garantiza la consistencia y la calidad a necesidad de hacer pruebas exhaustivas)
- Se deja la optimización para el final, una vez que el código requerido este completo.

d) Pruebas:

- Se crean pruebas de aceptación a partir de los resultados obtenidos en la ficha de evaluación ISO 9126

CAPÍTULO IV

RESULTADOS

4.1 RESULTADOS

4.1.1 Ámbito del problema

El software para el servicio de portabilidad numérica de telefonía fija/móvil usando el protocolo SOAP/WSDL para la empresa AMITEL S.A.C – PUNO, 2015 se desarrolló para cubrir la necesidad contar con el servicio de portabilidad numérica de la empresa.

4.1.2 Especificación de requerimientos del problema

a) Requerimientos Funcionales

Los requerimientos funcionales más relevantes que debe cubrir el sistema son los siguientes:

i) Se debe restringir el acceso al sistema, usando niveles de acceso y contraseñas cifradas.

ii) Se debe contar con la facilidad para mantenimiento de las tablas maestras.

iii) Se debe contar con la facilidad de ejecutar la aplicación de servidor SOAP

iv) Se debe verificar la comunicación con el sistema centralizado.

v) Se debe contar con módulo de respaldo de información automático periódico.

b) Requerimientos no funcionales:

i) Aplicación multiplataforma.

ii) Integración con los sistemas actuales

iii) Alta disponibilidad.

iv) Uso de sistemas GNU.

v) Uso de librerías modulares

c) Requerimientos técnicos:

i) Hardware

- Microprocesador INTEL 2.4Ghz.
- RAM de 2Gb
- Tarjeta de video 128 Mb (16Mb)*
- Disco Duro con espacio disponible de 500 Mb
- Monitor LED 15”

ii) Software

- Plataforma Ubuntu 14.0
- Java SE 8.00
- Servidor Web: Glasfish
- Netbeans 8.0.2
- MySQL
- Rational Rose 2007 Enterprise Edition
- Office 2010
- Notepad++

- Mozilla Firefox
- Google Chrome
- MySql-Workbench

iii) Resultados Esperados

- Se debe asegurar tener el escenario de comunicación funcionando con el ABDCP.
- Se debe asegurar la correcta comunicación Cliente al ABDCP.
- Se debe asegurar el correcto funcionamiento del sistema servidor SOAP.
- Se debe registrar como usuario con la clave de autorización otorgado por el ABDCP al servidor ABDCP.
- Se debe generara usuarios y claves con roles según el nivel de acceso.
- Solo se podrán acceder al sistema centros de gestión registrados en la DB local del sistema central de AMITEL S.A.C.
- Los centros de gestión podrán realizar las operaciones de portabilidad numérica como:

  Consulta previa.

  Solicitud de portabilidad.

  Solicitud de retorno.

- El acceso a las operaciones de portabilidad estará sujeto al nivel de acceso del usuario

- Debe estar disponible 24x7.
- Debe procesar los mensajes recibidos por el sistema ABDCP según los parámetros definidos.
- Se debe comprobar la integridad de datos recibidos.
- Se debe registrar en la base de datos local los mensajes recibidos
- Se debe generar parámetros del sistema basados en el sistema WSDL
- Debe registrase como usuario al sistema central ABDCP.
- Debe dar formato según los parámetros establecidos por el ABDCP de las comunicaciones de: Consulta previa, Solicitud de portabilidad, Solicitud de retorno.
- Debe registrar todo los registros enviados en la base de datos local

4.1.3 Modelamiento del software

a) Identificación de actores

- Clientes (abonados)
- Gestor de base de datos Centralizado (ABDCP)
- Centro de gestión
- Operador beneficiario
- Operador Donante

b) Identificación de casos de uso del negocio

- Procesar peticiones centro de gestión
- Procesar consulta previa
- Procesar solicitud de portabilidad.
- Procesar solicitud de retorno

4.1.4 Diagramas de uso

El presente diagrama de uso representa la funcionalidad general del módulo de centro de atención de usuarios.

a) Funcionalidad General

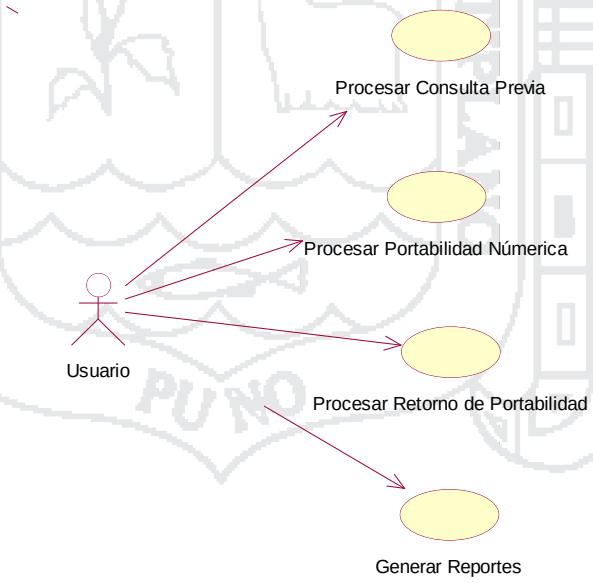


Figura 23 DIAGRAMA DE USO FUNCIONALIDAD GENERAL

b) Acceso al sistema

El siguiente diagrama muestra los procesos y agentes que intervienen en el acceso al módulo de atención al usuario

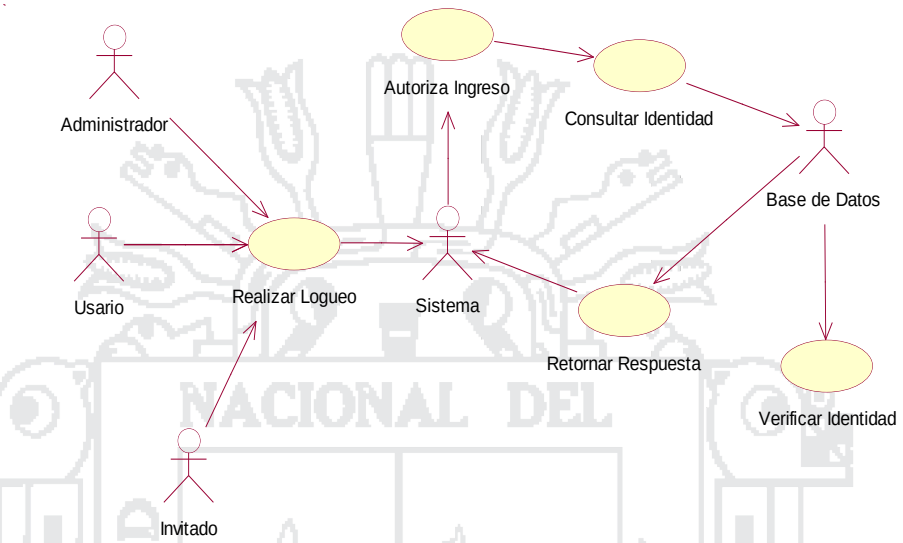


Figura 24 DIAGRAMA DE USO ACCESO AL SISTEMA

c) Otorgamiento de roles

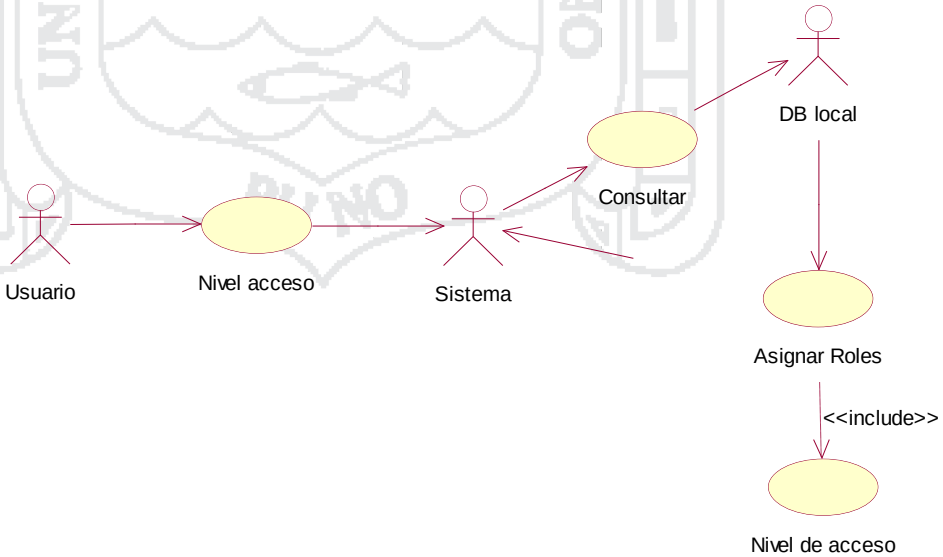


Figura 25 DIAGRAMA DE USO OTORGAMIENTO DE ROLES

d) Consulta Previa

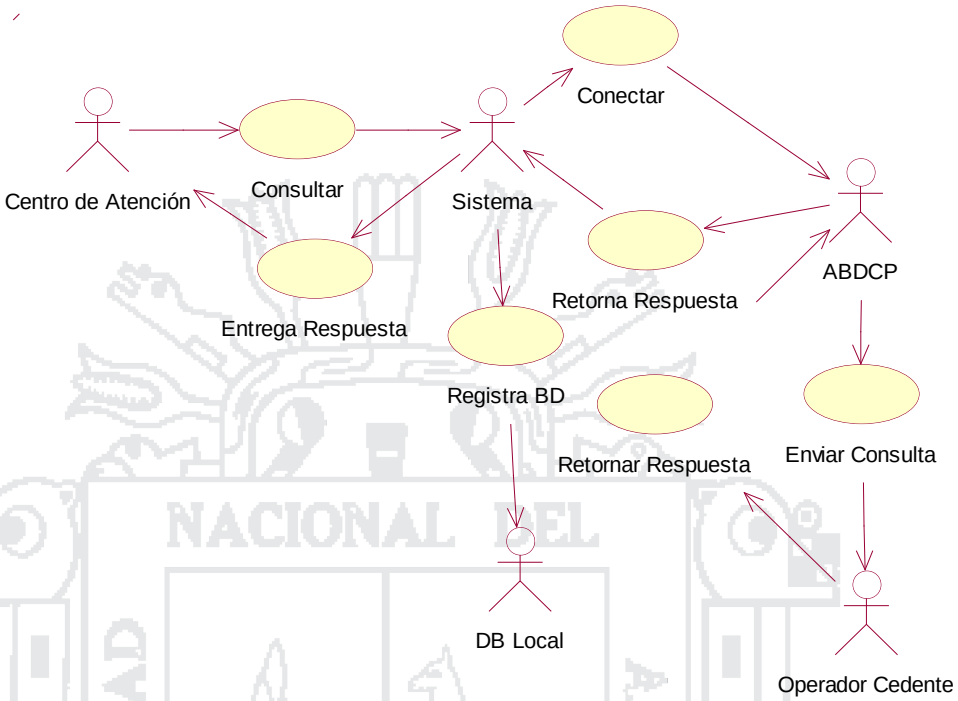


Figura 26 DIAGRAMA DE USO CONSULTA PREVIA

e) Portabilidad Numérica / Retorno

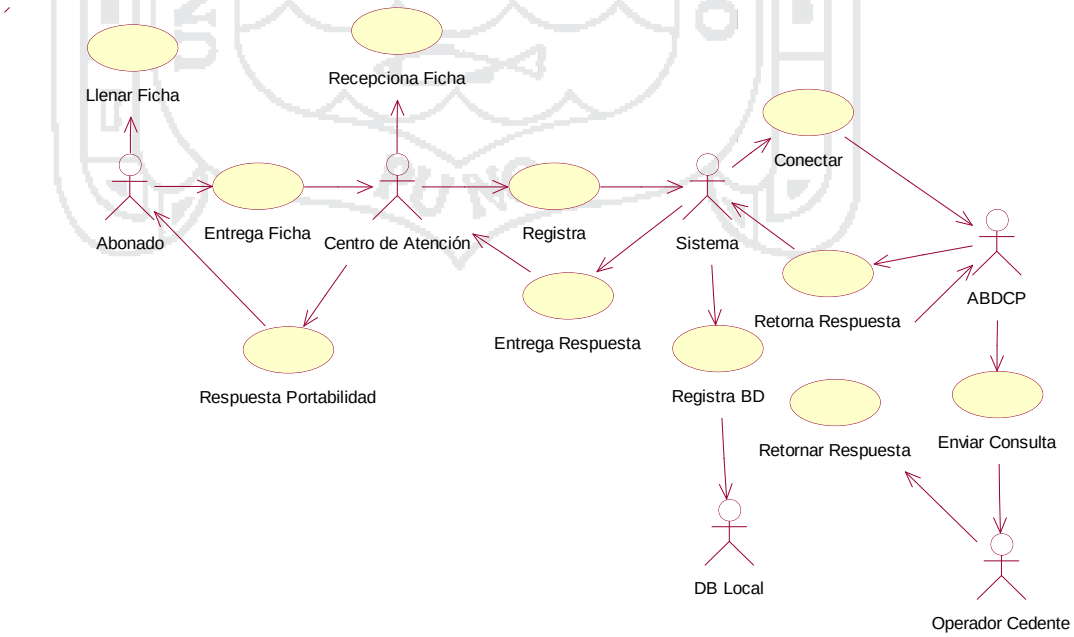


Figura 27 DIAGRAMA DE USO PORTABILIDAD NUMERICA RETORNO

4.1.5 Servidor SOAP

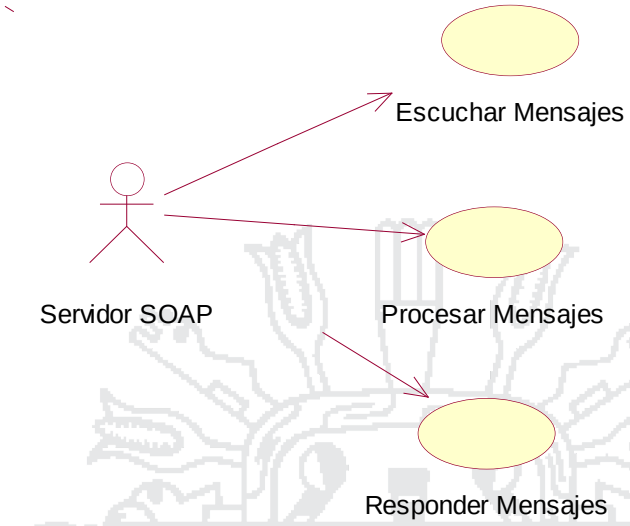


Figura 28 DIAGRAMA DE USO SERVIDOR SOAP

a) Funcionamiento General

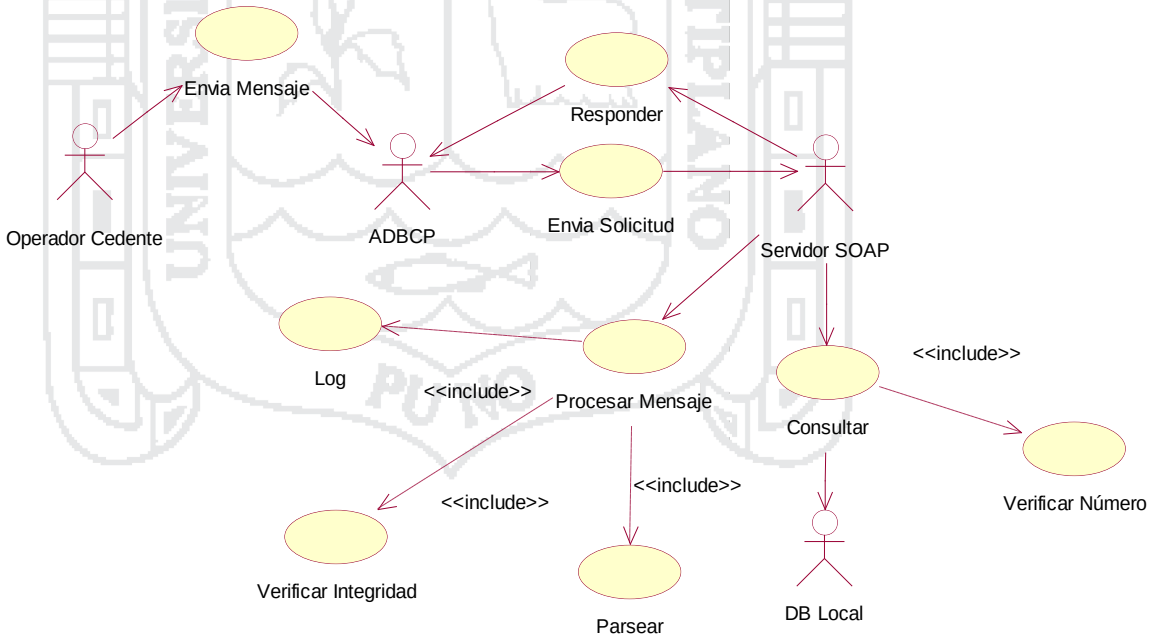


Figura 29 DIAGRAMA DE USO FUNCIONALIDAD GENERAL SERVIDOR SOAP

b) SOAP Comunicación HTTP

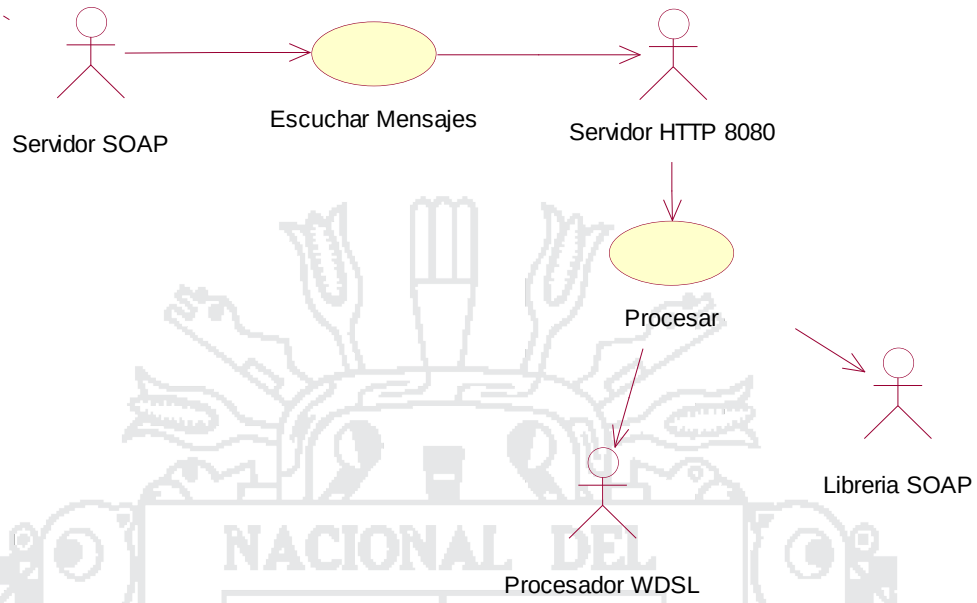


Figura 30 DIAGRAMA DE USO SOAP COMUNICACIÓN HTTP

4.1.6 Cliente SOAP

a) Funcionamiento General

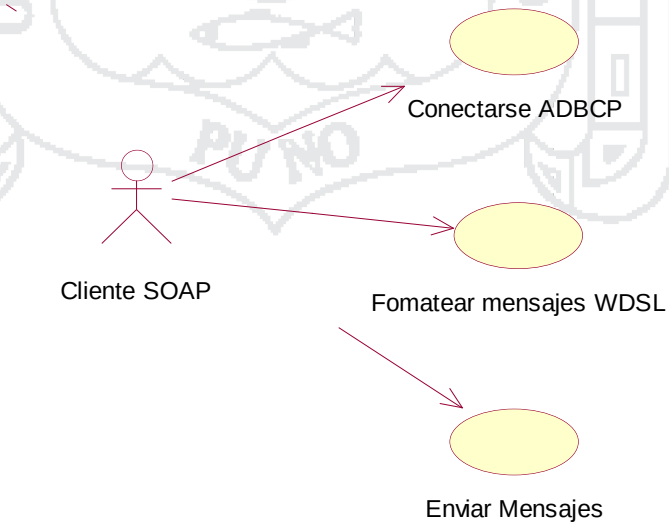


Figura 31 DIAGRAMA DE USO SOAP FUNCIONAMIENTO GENERAL

b) Autenticación

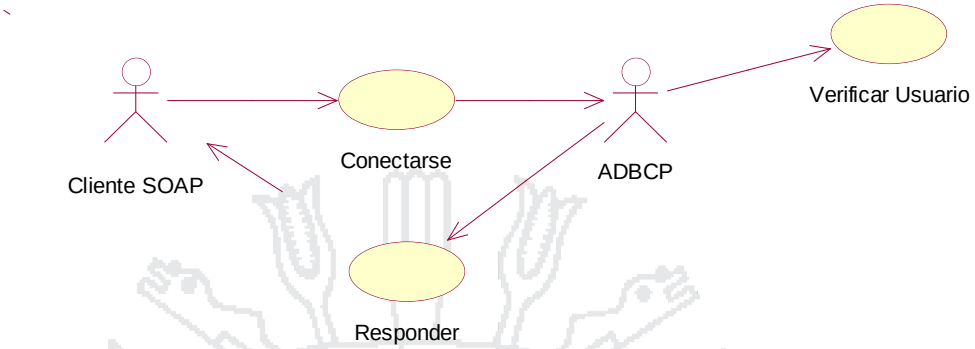


Figura 32 DIAGRAMA DE USO SOAP AUTENTICACIÓN

c) Diagrama de comunicación

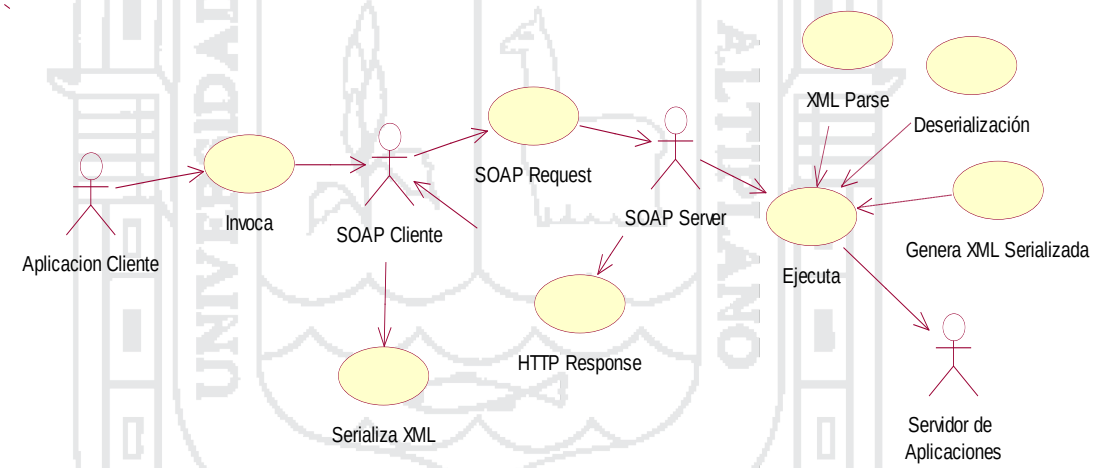


Figura 33 DIAGRAMA DE USO COMUNICACIÖN

4.1.7 Diagrama de secuencia

a) Acceso al sistema

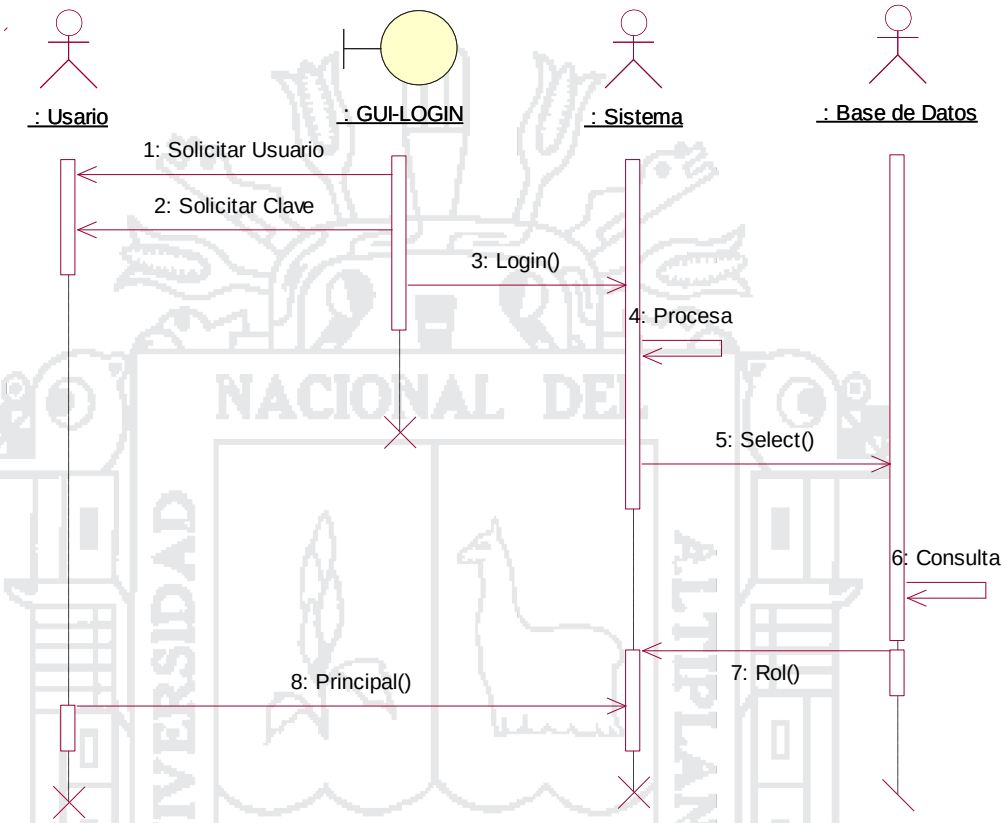


Figura 34 DIAGRAMA DE SECUENCIA ACCESO AL SISTEMA

b) Diagrama de secuencia SOAP

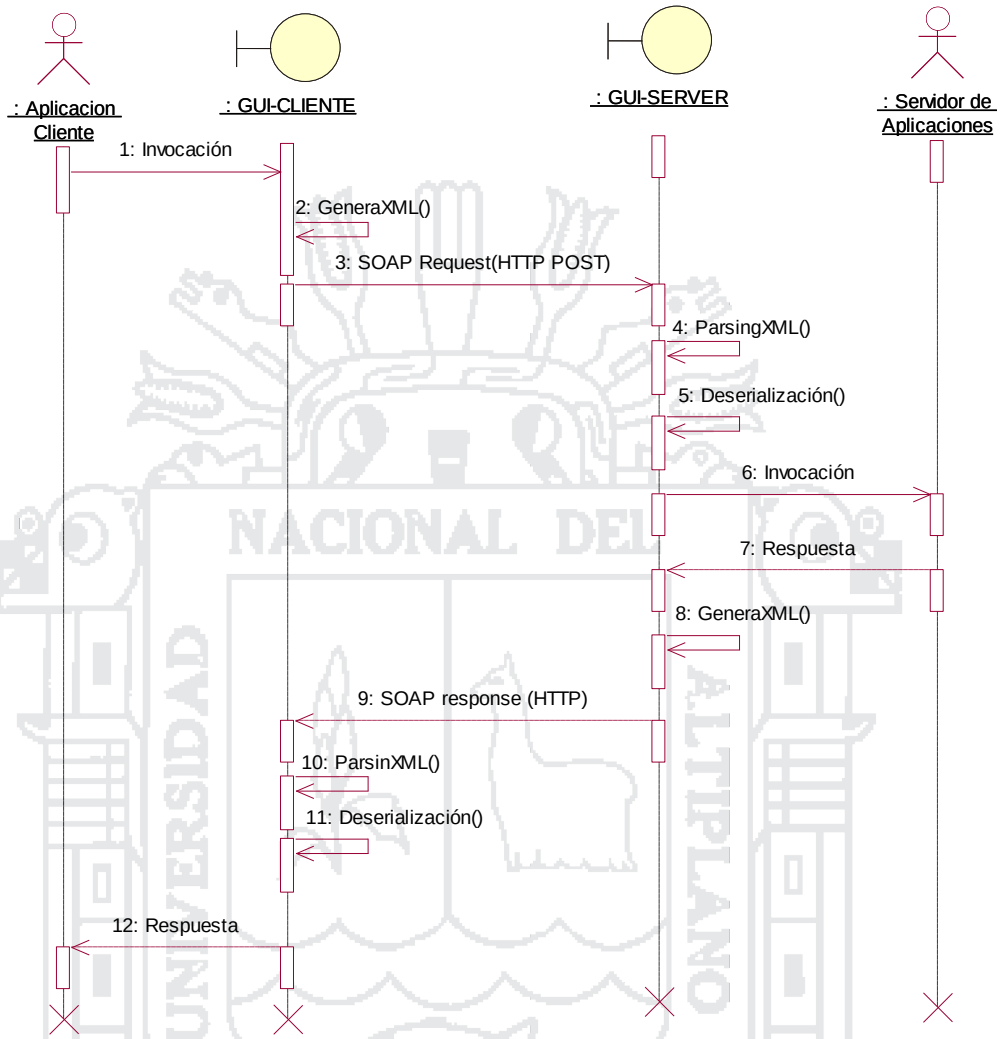


Figura 35 DIAGRAMA DE SECUENCIA SOAP

c) Consulta previa / portabilidad

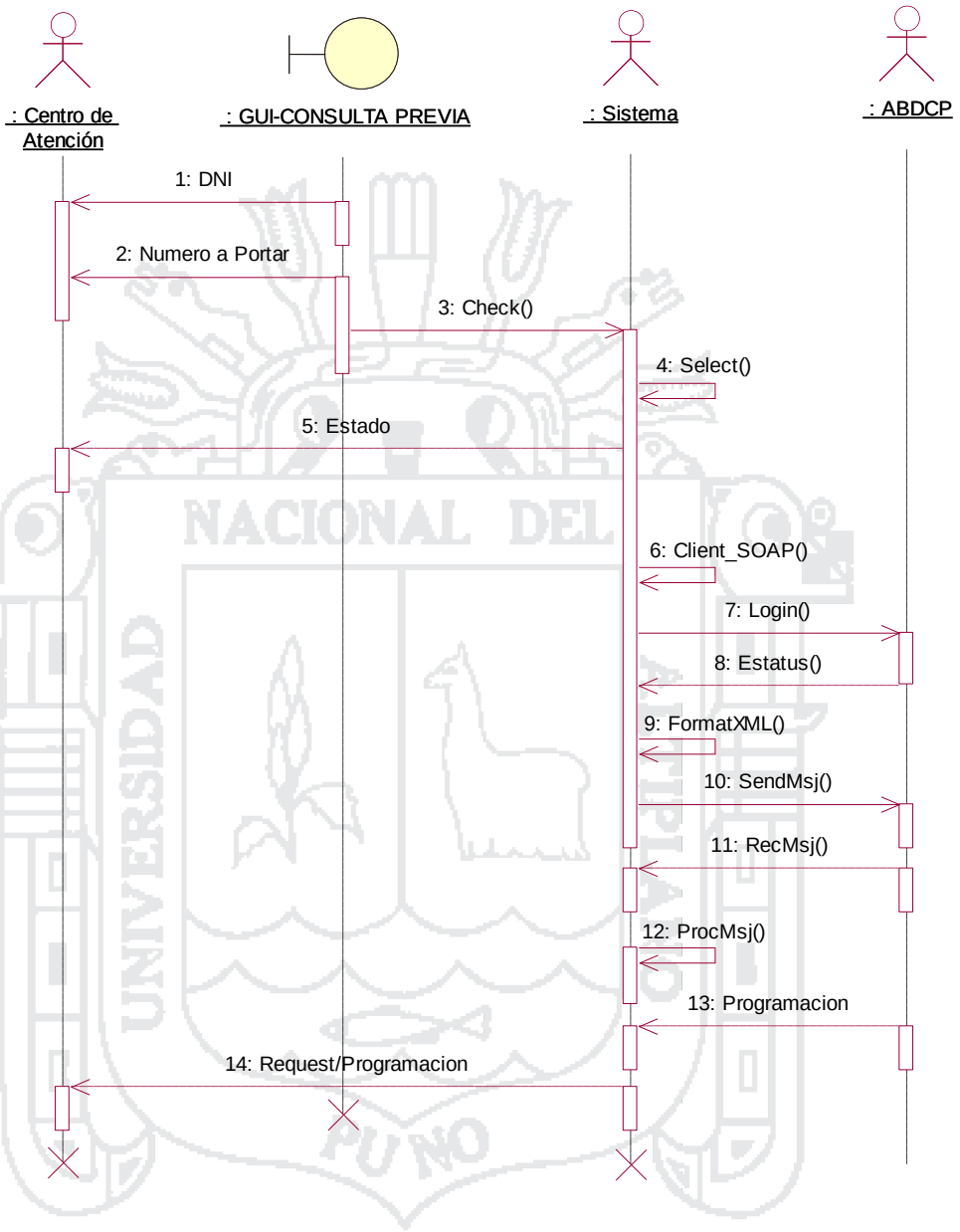


Figura 36 DIAGRAMA DE SECUENCIA CONSULTA PREVIA

d) Secuencia operador cedente

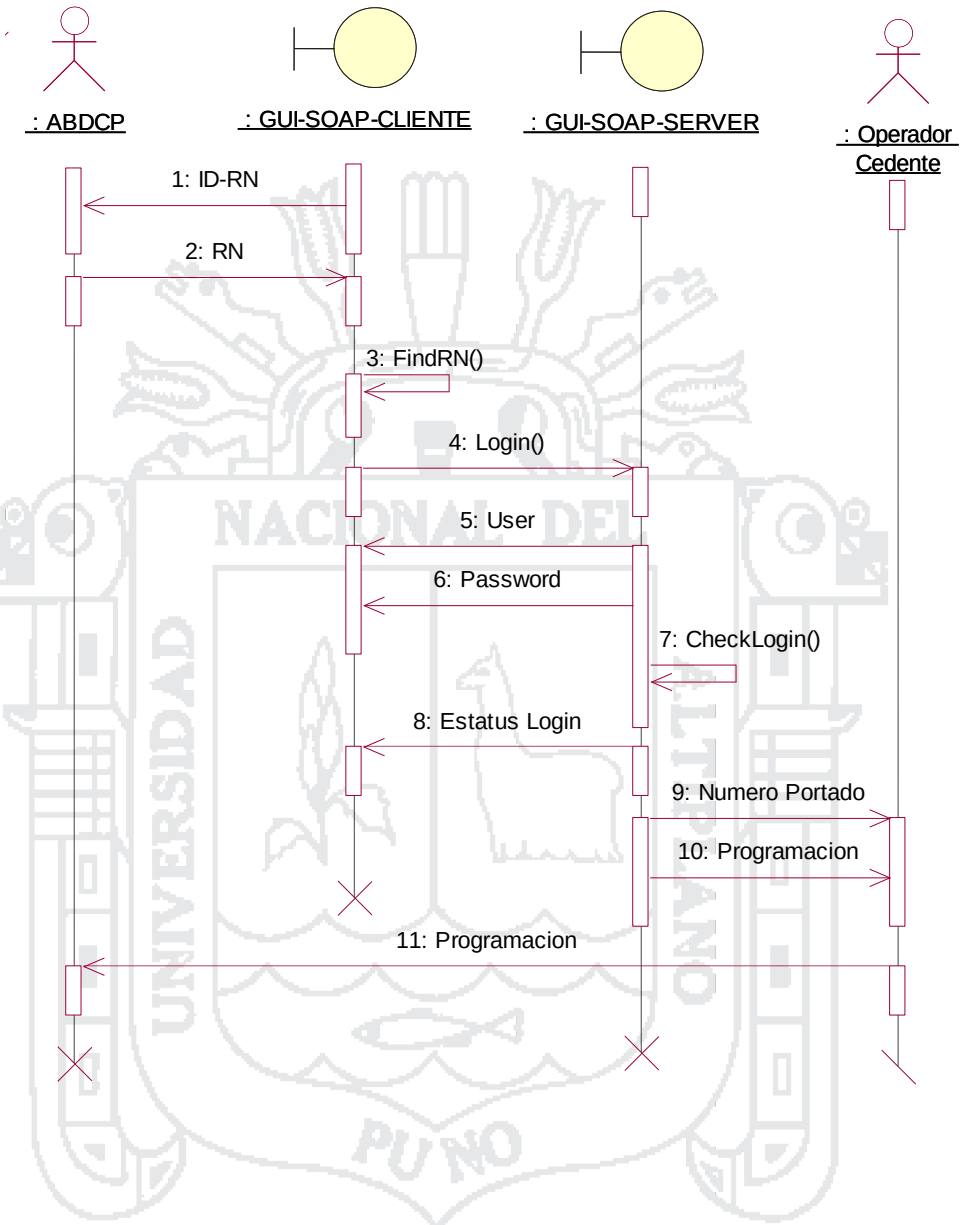


Figura 37 DIAGRAMA DE SECUENCIA OPERADOR CEDENTE

e) Secuencia de intercambios de mensajes de consulta previa

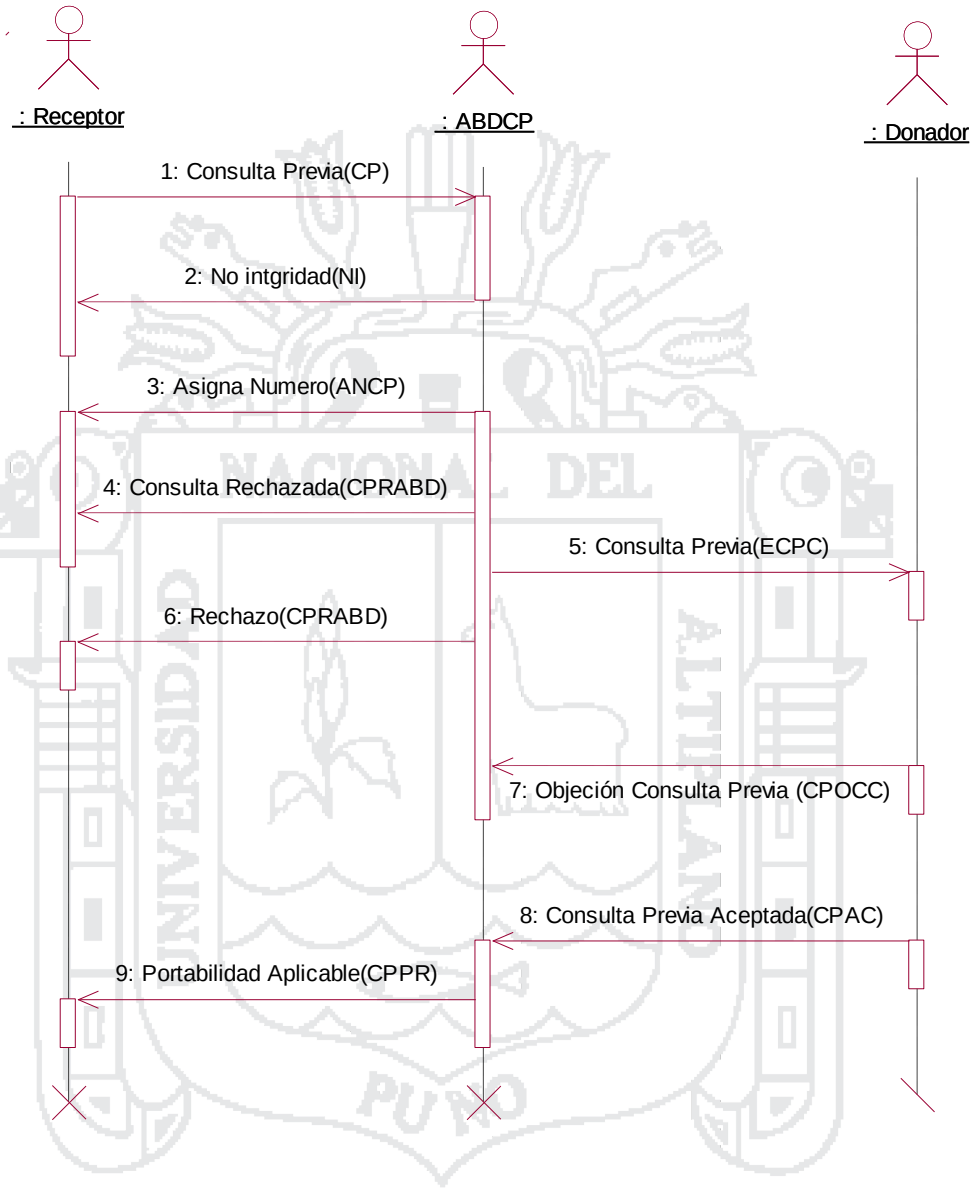


Figura 38 DIAGRAMA INTERCAMBIO DE MENSAJES CONSULTA PREVIA

f) Portabilidad

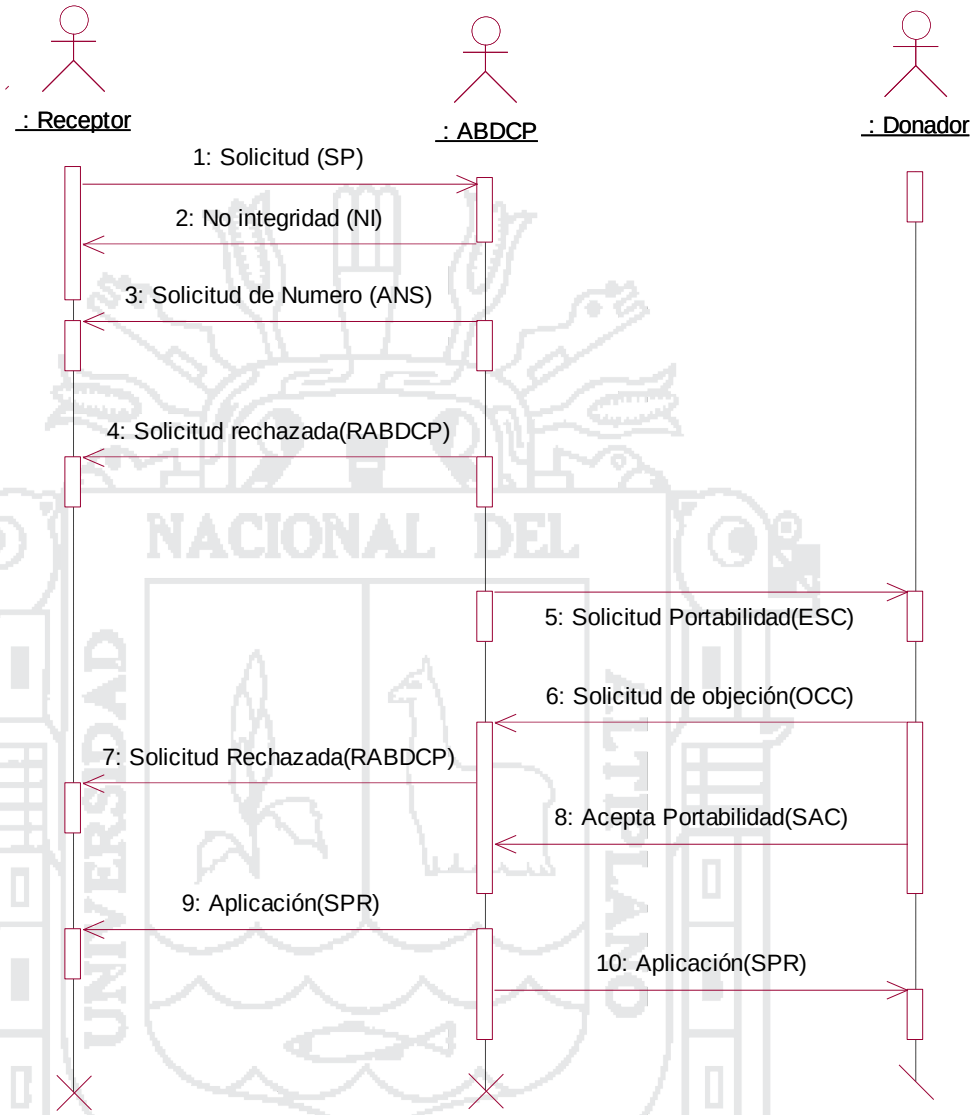


Figura 39 DIAGRAMA INTERCAMBIO DE MENSAJES PORTABILIDAD

4.1.8 Implementación cliente SOAP

Para la implementación del cliente SOAP se usa el WSDL del servidor ABDCP que se obtiene accediendo a su dirección URI el cual se presenta en el ANEXO III, y se puede apreciar los atributos y los métodos.

```
▼<xs:complexType>
  ▼<xs:sequence>
     <xs:element minOccurs="1" name="userID" nillable="false" type="xs:string"/>
     <xs:element minOccurs="1" name="password" nillable="false" type="xs:string"/>
     <xs:element minOccurs="1" name="xmlMsg" nillable="false" type="xs:string"/>
     <xs:element minOccurs="0" name="attachedDoc" nillable="false" type="xs:base64Binary"/>
   </xs:sequence>
 </xs:complexType>
```

Figura 40 ATRIBUTOS SERVIDOR SOAP

```
<wsdl:operation name="receiveMessage">
  <wsdl:input message="ns:receiveMessageRequest" wsaw:Action="urn:receiveMessageRequest"></wsdl:input>
  <wsdl:output message="ns:receiveMessageResponse" wsaw:Action="urn:receiveMessageResponse"></wsdl:output>
</wsdl:operation>
```

Figura 41 ATRIBUTOS SERVIDOR PROCEDIMIENTO SOAP

De estos atributos generamos la siguiente clase

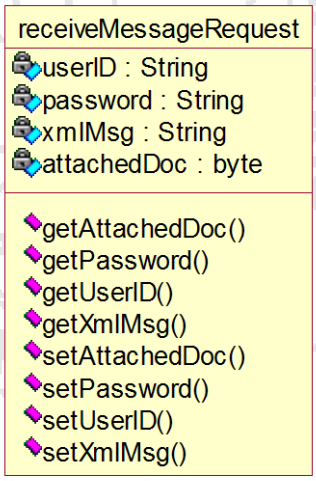


Figura 42 CLASE GENERADA A PARTIR DE WSDL DEL SERVIDOR SOAP

```
▼<wsdl:operation name="receiveMessage">
   <wsdl:input message="ns:receiveMessageRequest" wsaw:Action="urn:receiveMessageRequest"></wsdl:input>
   <wsdl:output message="ns:receiveMessageResponse" wsaw:Action="urn:receiveMessageResponse"></wsdl:output>
 </wsdl:operation>
```

Figura 43 ATRIBUTOS SERVIDOR SOAP

```
private static ReceiveMessageResponse receiveMessage(ReceiveMessageRequest parameters) {
    ABDCPWebService service = new ABDCPWebService();
    ABDCPWebServicePortType port = service.getABDCPWebServiceHttpSoap12Endpoint();

    System.out.println(port.toString());
    return port.receiveMessage(parameters);
}
```

Figura 44 ENVIO DE MENSAJE AL SERVIDOR SOAP

4.1.9 Implementación servidor SOAP

Para la implementación del servidor SOAP se utilizó NetBeans que tiene implementado librerías para aligerar la implementación de servicios web, a continuación presentamos parte del código principal.

```
public ReceiveMessageResponse receiveMessage(ReceiveMessageRequest parameters) {
    ReceiveMessageResponse responder = new ReceiveMessageResponse();
    ReceiveMessageRequest para;
    session = HibernateUtil.getSessionFactory().openSession();
    String xml;
    String idMensaje = "XX";
    String user;
    String password;
    user = parameters.getUserID();
    password = parameters.getPassword();
    xml = parameters.getXmlMsg().toString();
    System.out.println(xml);
    //----------------- ENCABEZADO
    Tmensaje tmensaje = new Tmensaje();
    tmensaje = XMLMensaje.get(xml);
    System.out.println("Destinatario:" + tmensaje.getDestinatario());
    System.out.println("Remitente:" + tmensaje.getRemitente());
    System.out.println("Identificador Mensaje:" + tmensaje.getIdentificadorMensaje());
    System.out.println("Identificador Proceso:" + tmensaje.getIdentificadorProceso());
    System.out.println("Fecha de Creacion:" + tmensaje.getFechaCreacionMensaje());
    //-----------------SAVE MENSAJES
```

Figura 45 PARTE DE CODIGO FUENTE SERVIDOR SOAP

4.1.10 Análisis de resultados

En cada etapa del proceso de desarrollo del software según el modelo XP se fue midiendo los resultados en base a las recomendaciones de la métrica ISO 9126 cuyos resultados son como sigue:

a) Funcionalidad

Tabla 14: Puntajes de funcionalidad según el usuario respecto a la calidad del producto estándar ISO 9126

	Adecuación	Exactitud	Interoperabilidad	Seguridad	Conformidad
Usuario1	3	5	5	5	5
Usuario2	4	4	3	4	5
Usuario3	2	4	4	5	5
Usuario4	4	4	3	5	5
Usuario5	5	5	4	4	5
Usuario6	4	5	5	5	4
Usuario7	3	5	4	4	5
Usuario8	4	5	4	4	5
Usuario9	5	5	3	4	5
Promedio	3.78	4.67	3.89	4.44	4.89

Tabla 15: Cuadro porcentual de funcionalidad a la calidad del producto estándar ISO 9126

	Promedio	%
Adecuación	3.78	76%
Exactitud	4.67	93%
Interoperabilidad	3.89	78%
Seguridad	4.44	89%
Conformidad	4.89	98%

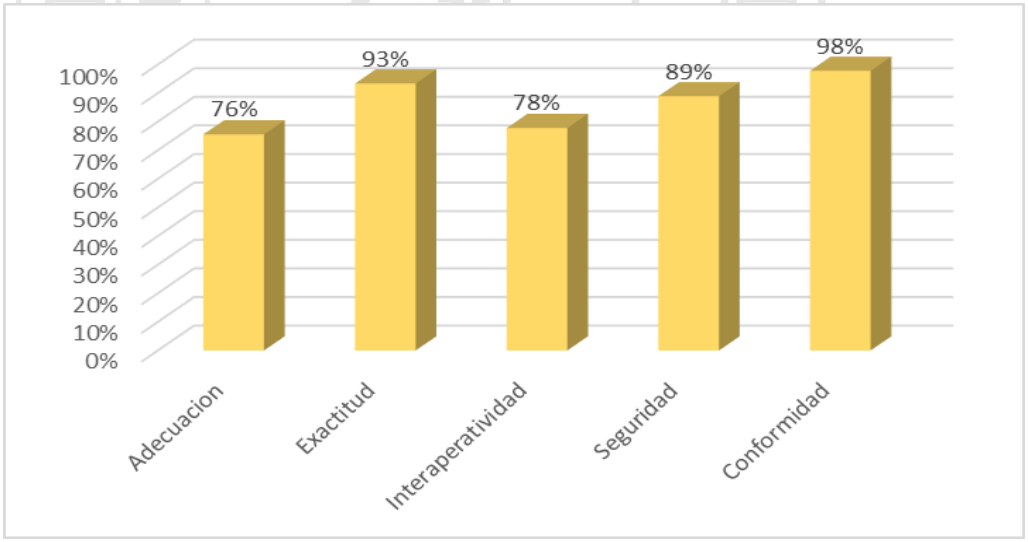


Figura 46 Funcionalidad según el usuario a la calidad del producto estándar ISO 9126

De los resultados obtenidos del análisis del software funcionalmente se tiene que el 76% de los usuarios califica al software como adecuado, el 93% califica como exacto, 78% califica como de buena interoperabilidad, el 89% califica como seguro, el 98% indica que el software se adapta el requerimiento de la portabilidad.

b) Fiabilidad

Tabla 16: Puntajes de fiabilidad según el usuario respecto a la calidad del producto estándar ISO 9126

	Madurez	Tolerancia a Fallos	Recuperabilidad	Conformidad
Usuario1	3	4	4	4
Usuario2	4	5	5	5
Usuario3	4	4	4	4
Usuario4	4	3	3	4
Usuario5	5	4	4	5
Usuario6	5	4	5	4
Usuario7	4	3	4	4
Usuario8	4	4	4	5
Usuario9	4	5	5	4
Promedio	4.11	4.00	4.22	4.33

Tabla 17: Cuadro porcentual de fiabilidad a la calidad del producto estándar ISO 9126

	Promedio	%
Madurez	4.11	82%
Tolerancia a Fallos	4	80%
Recuperabilidad	4.22	84%
Conformidad	4.33	87%

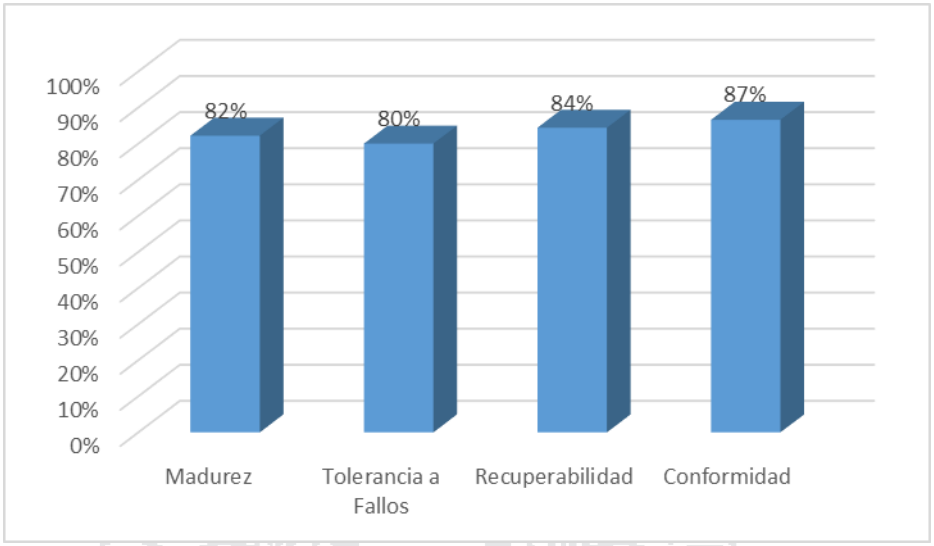


Figura 47 Fiabilidad según el usuario a la calidad del producto estándar ISO 9126

De los resultados obtenidos del análisis del software respecto a la fiabilidad se tiene que el 82% de los usuarios califica al software como que no tubo errores en su funcionamiento, el 80% califica como preparado para la tolerancia de fallas, 84% califica con buena capacidad para la recuperación ante fallas, el 87% indica que el software se adapta el requerimiento de fiabilidad.

c) Usabilidad

Tabla 18: Puntajes de usabilidad según el usuario respecto a la calidad del producto estándar ISO 9126

	Comprensibilidad	Facilidad de Aprendizaje	Atracción	Conformidad	Operatividad
Usuario1	4	5	4	5	4
usuario2	5	5	4	5	5
Usuario3	4	5	5	5	5
Usuario4	4	5	5	4	4
Usuario5	5	4	5	5	5
Usuario6	5	4	4	5	5
Usuario7	4	4	5	4	4
Usuario8	5	5	5	4	5
Usuario9	5	3	4	5	4
Promedio	4.56	4.44	4.56	4.67	4.56

Tabla 19: Cuadro porcentual de usabilidad a la calidad del producto estándar ISO 9126

	Promedio	%
Comprensibilidad	4.56	91%
Facilidad de Aprendizaje	4.44	89%
Atracción	4.56	91%
Conformidad	4.67	93%
Operatividad	4.56	91%

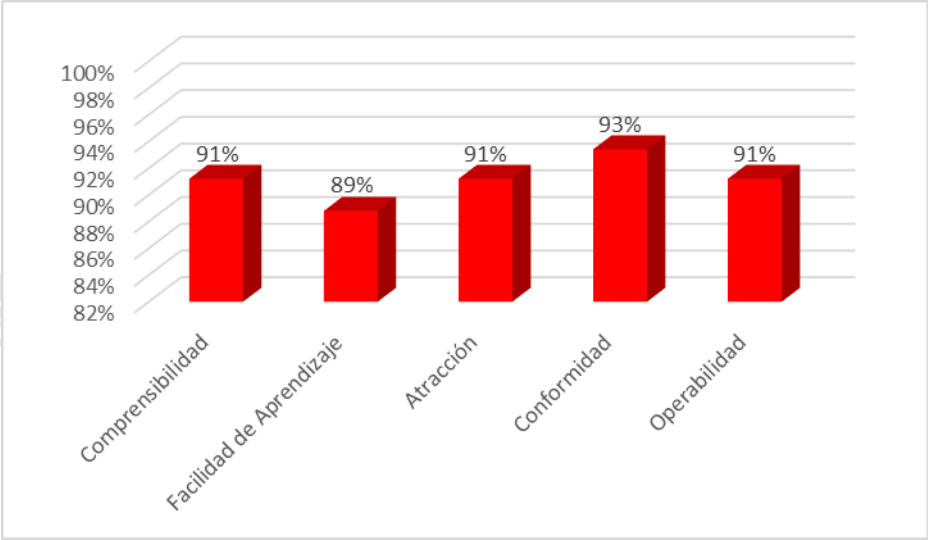


Figura 48 Usabilidad según el usuario a la calidad del producto estándar ISO 9126

De los resultados obtenidos del análisis del software respecto a la usabilidad se tiene que el 91% de los usuarios califica al software como comprensible, el 86% califica como fácil de aprender, 91% califica como atractivo la interfaz gráfica, el 93% indica que el software se adapta el requerimiento de uso, el 91% califica como operativa comprensible.

d) Eficiencia

Tabla 20: Puntajes de eficiencia según el usuario respecto a la calidad del producto estándar ISO 9126

	Comportamiento Temporal	Utilización de Recursos	Conformidad
Usuario1	4	4	5
usuario2	4	5	5
Usuario3	5	3	4
Usuario4	5	3	3
Usuario5	4	4	4
Usuario6	4	3	4
Usuario7	5	4	5
Usuario8	4	4	4
Usuario9	5	3	4
Promedio	4.44	3.67	4.22

Tabla 21: Cuadro porcentual de eficiencia a la calidad del producto estándar ISO 9126

	Promedio	%
Comportamiento Temporal	4.44	89%
Utilización de Recursos	3.67	73%
Conformidad	4.22	84%

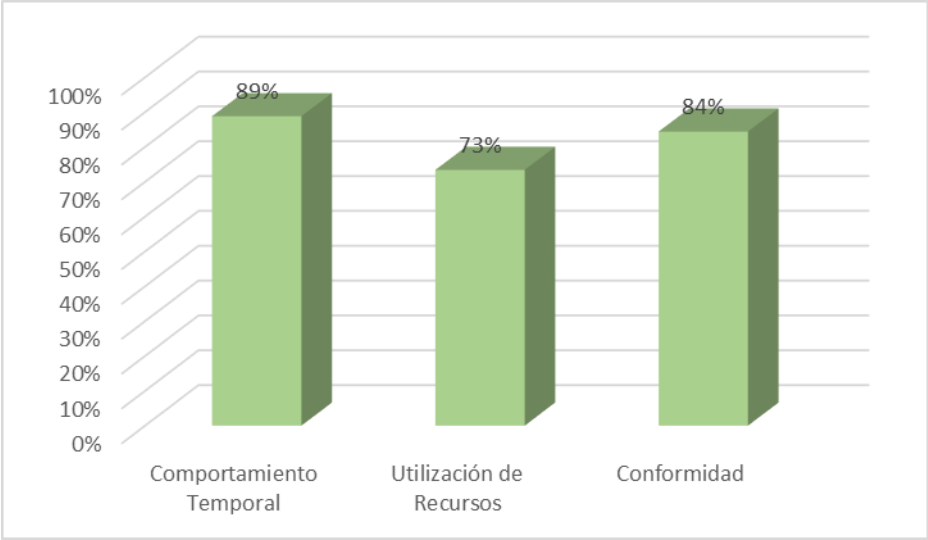


Figura 49 Eficiencia según el usuario a la calidad del producto estándar ISO 9126

De los resultados obtenidos del análisis del software respecto a la eficiencia se tiene que el 89% de los usuarios califica al software como

rápido en las transacciones realizadas, el 73% califica como que usa pocos recursos en la computadora, el 84% indica que el software se adapta el requerimiento de uso.

e) Mantenibilidad

Tabla 22: Puntajes de mantenibilidad según el usuario respecto a la calidad del producto estándar ISO 9126

	Analizabilidad	Confiabilidad	Estabilidad	Facilidad de Prueba	Conformidad
Usuario1	4	5	5	5	5
Usuario2	5	5	5	5	5
Usuario3	4	5	5	4	4
Usuario4	4	4	4	4	4
Usuario5	3	4	5	4	5
Usuario6	4	5	5	4	5
Usuario7	5	4	4	5	5
Usuario8	5	5	5	5	4
Usuario9	5	5	4	4	4
Promedio	4.33	4.67	4.67	4.44	4.56

Tabla 23: Cuadro porcentual de mantenibilidad a la calidad del producto estándar ISO 9126

	Promedio	%
Analizabilidad	4.33	87%
Confiabilidad	4.67	93%
Estabilidad	4.67	93%
Facilidad de Prueba	4.44	89%
Conformidad	4.56	91%

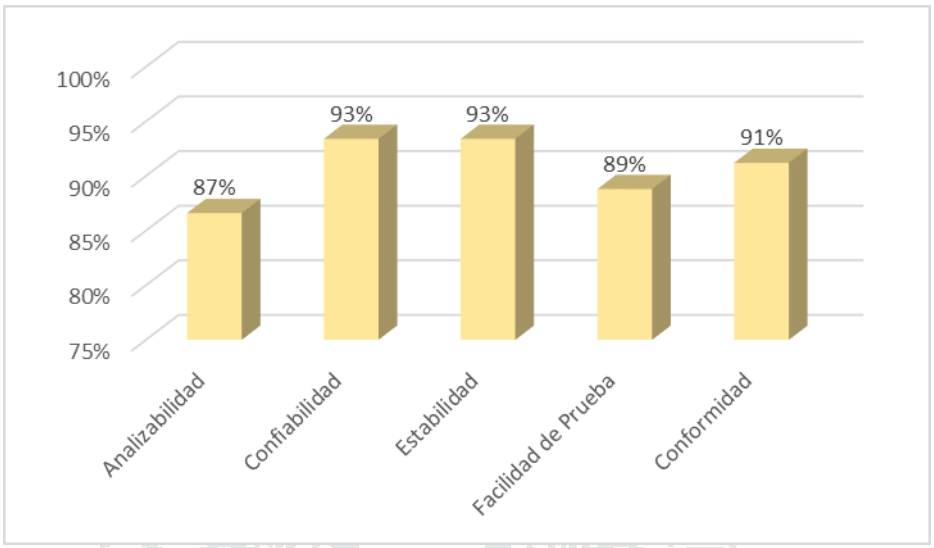


Figura 50 Mantenibilidad según el usuario a la calidad del producto estándar ISO 9126

De los resultados obtenidos del análisis del software respecto a la mantenibilidad se tiene que el 87% de los usuarios califica al software como fácil de detectar los errores, el 93% califica como confiable, 93% califica como estable, el 89% indica que el software es fácil de probar, el 91% califica como operativa comprensible y escalable.

f) Portabilidad

Tabla 24: Puntajes de portabilidad según el usuario respecto a la calidad del producto estándar ISO 9126

	Adaptabilidad	Facilidad de Instalación	Coexistencia	Reemplazabilidad	Conformidad
Usuario1	4	5	5	4	5
Usuario2	5	5	5	4	5
Usuario3	4	5	5	4	4
Usuario4	4	5	5	5	5
Usuario5	5	5	5	4	5
Usuario6	5	5	5	4	5
Usuario7	4	4	5	5	4
Usuario8	5	4	4	5	4
Usuario9	5	5	5	5	5
Promedio	4.56	4.78	4.89	4.44	4.67

Tabla 25: Cuadro porcentual de portabilidad a la calidad del producto estándar ISO 9126

	Promedio	%
Adaptabilidad	4.56	91%
Facilidad de Instalación:	4.78	96%
Coexistencia	4.89	98%
Reemplazabilidad	4.44	89%
Conformidad	4.67	93%

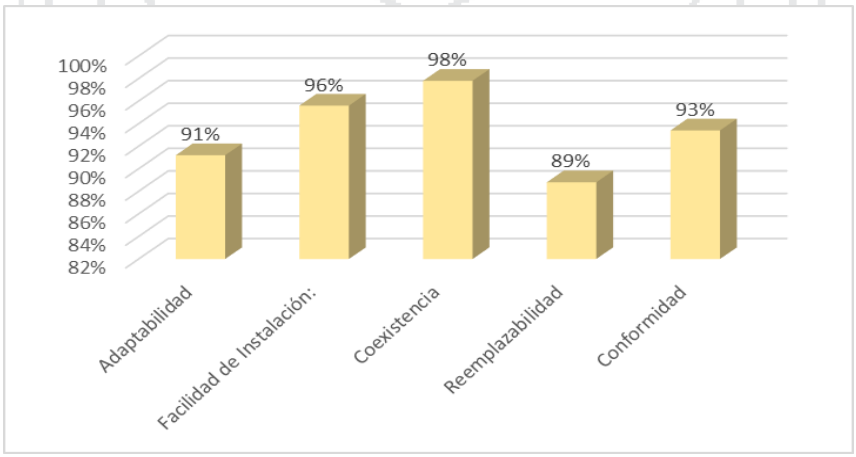


Figura 51 Portabilidad según el usuario a la calidad del producto estándar ISO 9126

De los resultados obtenidos del análisis del software respecto a la portabilidad se tiene que el 91% de los usuarios califica al software como adaptable, el 96% califica como fácil de instalar, 98% califica como que puede coexistir con otros aplicativos, el 98% indica que el software puede remplazar a software de versiones anteriores, el 93% califica como operativa comprensible y escalable.

4.1.11 Métrica general del software

La sumatoria de la ficha de evaluación de la calidad de software es como sigue:

Tabla 26: Sumatoria de promedios de calidad de software

FUNCIONALIDAD	FIABILIDAD	USABILIDAD	EFICIENCIA	MANTENIBILIDAD	PORTABILIDAD
21.67	16.66	22.79	12.33	22.67	23.34
TOTAL	119.46				

Y el cuadro de decisiones del ISO 9126 está dado por:

Tabla 27: Intervalo de calificación de calidad de software ISO 9126

Clasificación	Intervalo	Decisión
A) Inaceptable	[27 – 54 >	
B) Mínimamente Aceptable	[54 – 81 >	
C) Aceptable	[81 – 95 >	
D) Cumple los Requisitos	[95 – 122 >	Resultado
E) Excede los Requisitos	[122 – 135]	

De los resultados se determina que el SOFTWARE PARA EL SERVICIO DE PORTABILIDAD NUMÉRICA DE TELEFONIA FIJA/MOVIL USANDO PROTOCOLO SOAP/WSDL PARA LA EMPRESA AMITEL S.A.C. – PUNO cumple los requisitos exigidos según el estándar ISO 9126 y puede ser usado como software de gestión de sistemas de portabilidad, esta

determinación lo respalda las pruebas en conjunto realizadas con el sistema ABDCP que se encuentra en el anexo I, VII

También se hizo pruebas de validación y comparación con el sistema de la empresa operadora de telefonía fija FRAVATEL sistema desarrollado en el lenguaje de programación PHP teniendo similares resultados.

CONCLUSIONES

- La implementación del módulo software servidor usando el protocolo SOAP/WSDL cumple los requerimientos para el servicio de portabilidad numérica y del estándar ISO 9126, que se verifico la implementación tiene la capacidad de recibir los mensajes, procesarlos y responder al cliente ABDCP.

- La implementación del módulo software cliente usando el protocolo SOAP/WSDL cumple los requerimientos para el servicio de portabilidad numérica y del estándar ISO 9126, se pudo verificar que la implementación tiene la capacidad de conectarse al servidor ABDCP (sistema de portabilidad numérica) autenticarse y hacer requerimientos de portabilidad.

- La implementación del módulo software para los centros de atención de cliente de la empresa AMITEL S.A.C, cumple los requerimientos de interfaz de administración para el servicio de portabilidad numérica y vez cumple con el estándar ISO 9126, lo que se verifico que la implementación tiene la aceptación de los usuarios en funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad usabilidad.

RECOMENDACIONES Y SUGERENCIAS

- El protocolo SOAP/WSDL es un protocolo abierto y es implementado en casi todo los lenguajes de programación, se recomienda usar el lenguaje de programación Java para un mejor performance y fácil implementación, debido que el lenguaje JAVA incorpora librerías nativas y muchas funciones para la implementación.

- Se recomienda usar la metodología de desarrollo de software XP (programación extrema) como modelo de proceso de software por su adaptabilidad natural a este tipo de proyectos.

- Se recomienda implementar los módulos de software iniciando la implementación de prototipo y pruebas como lo recomienda el modelo XP.

- Se sugiere usar Java EE para la implementación de la interfaz, debido a su amplia gama de herramientas y gran desempeño en la implementación.

- Se sugiere usar un sistema de control de versiones GLT en el momento de desarrollo para tener las versiones de nuestra aplicación según se avanza.

BIBLIOGRAFÍA

Cabrera, C. (1998). *"Algunos conceptos básicos de programación orientada a objetos", primera edición.*

Dubois, P. (2005). *"MySQL", Editorial Prentice-Hall, Edición Especial.*

Evans T. (1997). "Construya su propia Intranet", Prentice.

HANSEN, G. (2007). "*Diseño y administración de base de datos", Editorial Madrid Prentice-Hall, Edición II.*

HAHN H. (2008). *"Internet: Manual de referencia", Editorial Madrid: McGraw-Hill, Segunda Edición*.

Heather W. (2005). *"XML: Manual de referencia", Editorial McGraw-Hill, Segunda Edición.*

James A. (1992). *"Análisis y diseño de sistemas de información", Mc. Graw Hill segunda edición*.

Kendall, J. (2005), "Análisis y diseño de sistemas", Editorial Pearson, Sexta Edición.

Navate, E. (1995) *"Sistema de Base de Datos", Editorial Iberoamericana, Edición Addison Wesley.*

OROS CABELLO J. (2007). *"Diseño de páginas web interactivas con JavaScript y CSS", Editorial Alfa Omega, Tercera Edición.*

Pérez, L. A. (2000*). "Estadística básica para ciencias sociales y educación", Editorial San Marcos.*

Pressman, R. (2002). *"Ingeniería del Software un Enfoque Práctico", Editorial. Mc Graw-Hill, Quinta Edición.*

Rodríguez G. Denis E. (2009). *"Adobe Flash CS4 Animación y publicaciones Web", Ediciones .Macro E.I.R.L.*

Rumbaugh, J., Jacobson, I. y Booch, G. (1999). *"El Lenguaje Unificado de Modelado, Manual de Referencia", España: Editorial Addison Wesley.*

Romero, G. (2004). *"UML con Rational Rose". Lima, Perú: Grupo Editorial Megabyte.*

Schmuller, J. (2000) *"Aprendiendo UML en 24 Horas", Primera Edición Prentice Hall.*

Solomon, P. R. (1998). *"Guía para redactar informes de investigación". Editorial Trillas.*

STOUD, R. (2005). *"World Wide Web: Manual de referencia", Editorial Madrid Osborne, Primera Edition*.

Weitzenfeld, A. (2002). *"Ingeniería de software orientado a objetos: Teoría y práctica con UML y Java", México: Editorial ITAM Departamento Académico de Computación.*

WEBGRAFÍA

Diseño de Bases de Datos Relacionales-Una extensión informal de UML. [En línea]. Consultado: (08, Enero, 2011). Disponible en: es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x332.html.

Ejemplo completo sobre la utilización del método RUP. [En línea]. Consultado: (12, Enero, 2011). Disponible en:www.dsic.upv.es/asignaturas/facultad/lsi/ejemplorup/index.html.

Calero Muñoz, Coral. Métricas de calidad del software. (En línea). Consultado: (04, Febrero, 2011). Disponible en: Alarcos.inf-cr.uclm.es/doc/calidadSI/Metodos%20De%20Calidad.ppt.

Computación e Informática

http://www.rodolfoquispe.org/blog/que-es-la-ingenieria-de-software.php; accedido 08 de Abril del 2011

ANEXOS

ANEXO I
Ficha de evaluación de la calidad del producto estándar ISO – 9126.

INDICADORES	PUNTUACION				
1. FUNCIONALIDAD					
Adecuación: la capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas específicas y objetivos de los usuarios.					
Exactitud: la capacidad del producto software para proporcionar los resultados o efectos correctos y con el grado de precisión acordado.					
Interoperabilidad: la capacidad del producto software para interactuar con uno o más sistemas específicos.					
Seguridad: referido a la capacidad del producto software para proteger la información y los datos.					
Conformidad: la capacidad del producto software para adaptarse a los estándares, convenciones o regulaciones en leyes y prescripciones relativos a la funcionalidad.					
2. FIABILIDAD					
Madurez: la capacidad del producto software para evitar fallos provocados por errores en el software.					
Tolerancia a Fallos: la capacidad del producto software para mantener un nivel de rendimiento determinado en caso de defectos en el software o incumplimiento de su interfaz.					
Recuperabilidad: la capacidad del producto software para restablecer un determinado nivel de rendimiento y recuperar los datos afectados directamente en caso de ocurrir un fallo.					
Conformidad: la capacidad del producto software para adaptarse a los estándares, convenciones o regulaciones referidas a la fiabilidad.					
3. USABILIDAD					
Comprensibilidad: la capacidad del producto software para permitir al usuario que entienda si el software es adecuado, y como debe utilizarse para determinadas tareas y bajo ciertas condiciones de uso.					
Facilidad de Aprendizaje: la capacidad del producto software para permitir al usuario aprender su aplicación.					
Atracción: la capacidad del software para atraer al usuario.					
Conformidad: la capacidad del producto software para adaptarse a estándares, convenciones, guías de estilo y regulaciones relacionadas con la usabilidad.					
Operabilidad: la capacidad del producto software para permitir que el usuario lo opere y lo controle					
4. EFICIENCIA					
Comportamiento Temporal: la capacidad del producto software para proporcionar tiempos de respuesta y de procesamiento apropiados cuando realiza sus funciones bajo condiciones determinadas.					
Utilización de Recursos: la capacidad del producto software para utilizar cantidades y tipos de recursos apropiados cuando el software realiza su función bajo determinadas condiciones.					
Conformidad: la capacidad del producto software para adaptarse a estándares o convenciones relacionadas con la eficiencia.					
5. MANTENIBILIDAD					
Analizabilidad: capacidad del producto software de diagnosticar sus deficiencias o causas de fallos, o de identificar las partes que deben ser modificadas.					
Confiabilidad: capacidad del producto software de permitir implementar una modificación especifica. La implementación incluye los cambios en el diseño, el código y la documentación.					
Estabilidad: capacidad del producto software de evitar los defectos inesperados de las modificaciones.					
Facilidad de Prueba: capacidad del producto software de permitir validar las partes modificadas.					
Conformidad: capacidad del producto software de cumplir los estándares o convenciones relativas a la mantenibilidad.					
6. PORTABILIDAD					
Adaptabilidad: la capacidad del producto software para ser adaptado para ambientes determinados sin realizar acciones o aplicar medios, más que los proporcionados para este propósito para el software considerado					
Facilidad de Instalación: la capacidad del producto software para ser instalado en un ambiente determinado.					
Coexistencia: la capacidad del producto software para coexistir con otro software independiente en un ambiente común compartiendo recursos.					
Reemplazabilidad: la capacidad del producto software para ser utilizado en lugar de otro producto de software para el mismo propósito en el mismo ambiente.					
Conformidad: la capacidad del producto software para adaptarse a estándares relacionados con la portabilidad.					
SUB TOTALES					
TOTAL					

Fuente: Ficha de Evaluación ISO – 9126

Indicador Cualitativo	Valor
Deficiente	1
Malo	2
Regular	3
Bueno	4
Muy bueno	5

Fuente: Escala valorativa. (Escala de Likert)

Clasificación	Intervalo	Decisión
A) Inaceptable	[27 – 54 >	
B) Mínimamente Aceptable	[54 – 81 >	
C) Aceptable	[81 – 95 >	
D) Cumple los Requisitos	[95 – 122 >	
E) Excede los Requisitos	[122 – 135]	

Fuente: Cuadro de decisiones ISO 9126.

ANEXO II

Pruebas de conformidad al servicio de portabilidad AMITEL – ABDCP

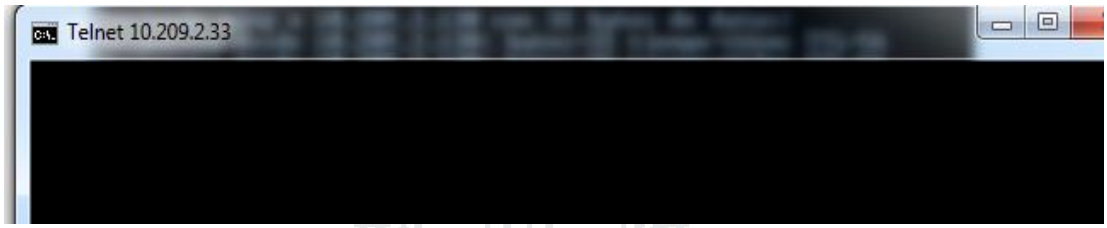
Prueba de conectividad

Paso 1: EXITO

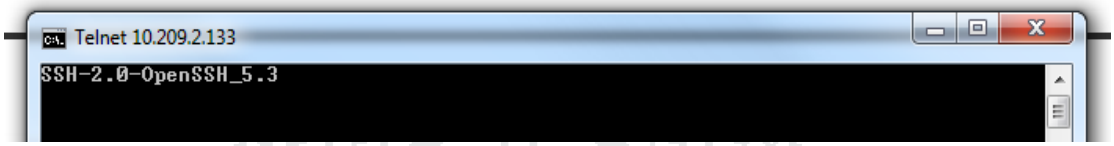
```
C:\Users\jburgos>ping 10.209.2.33

Haciendo ping a 10.209.2.33 con 32 bytes de datos:
Respuesta desde 10.209.2.33: bytes=32 tiempo=166ms TTL=247
Respuesta desde 10.209.2.33: bytes=32 tiempo=166ms TTL=247
Respuesta desde 10.209.2.33: bytes=32 tiempo=166ms TTL=247
Respuesta desde 10.209.2.33: bytes=32 tiempo=166ms TTL=247

Estadísticas de ping para 10.209.2.33:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 166ms, Máximo = 166ms, Media = 166ms

C:\Users\jburgos>ping 10.209.2.133

Haciendo ping a 10.209.2.133 con 32 bytes de datos:
Respuesta desde 10.209.2.133: bytes=32 tiempo=166ms TTL=247
Respuesta desde 10.209.2.133: bytes=32 tiempo=166ms TTL=247
Respuesta desde 10.209.2.133: bytes=32 tiempo=166ms TTL=247
Respuesta desde 10.209.2.133: bytes=32 tiempo=166ms TTL=247

Estadísticas de ping para 10.209.2.133:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 166ms, Máximo = 166ms, Media = 166ms

C:\Users\jburgos>ping 10.209.3.130

Haciendo ping a 10.209.3.130 con 32 bytes de datos:
Respuesta desde 10.209.3.130: bytes=32 tiempo=166ms TTL=56
Respuesta desde 10.209.3.130: bytes=32 tiempo=167ms TTL=56
Respuesta desde 10.209.3.130: bytes=32 tiempo=166ms TTL=56
Respuesta desde 10.209.3.130: bytes=32 tiempo=166ms TTL=56

Estadísticas de ping para 10.209.3.130:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 166ms, Máximo = 167ms, Media = 166ms

C:\Users\jburgos>
```

PASO 2: EXITO

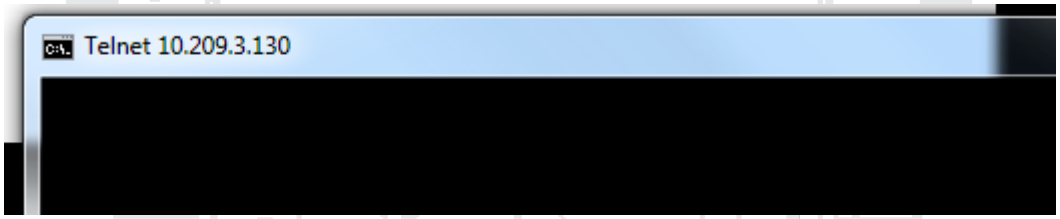
telnet 10.209.2.33 443



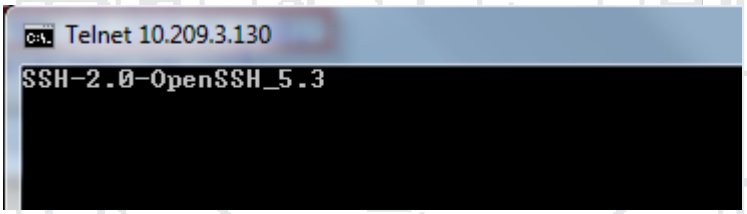
telnet 10.209.2.133 22

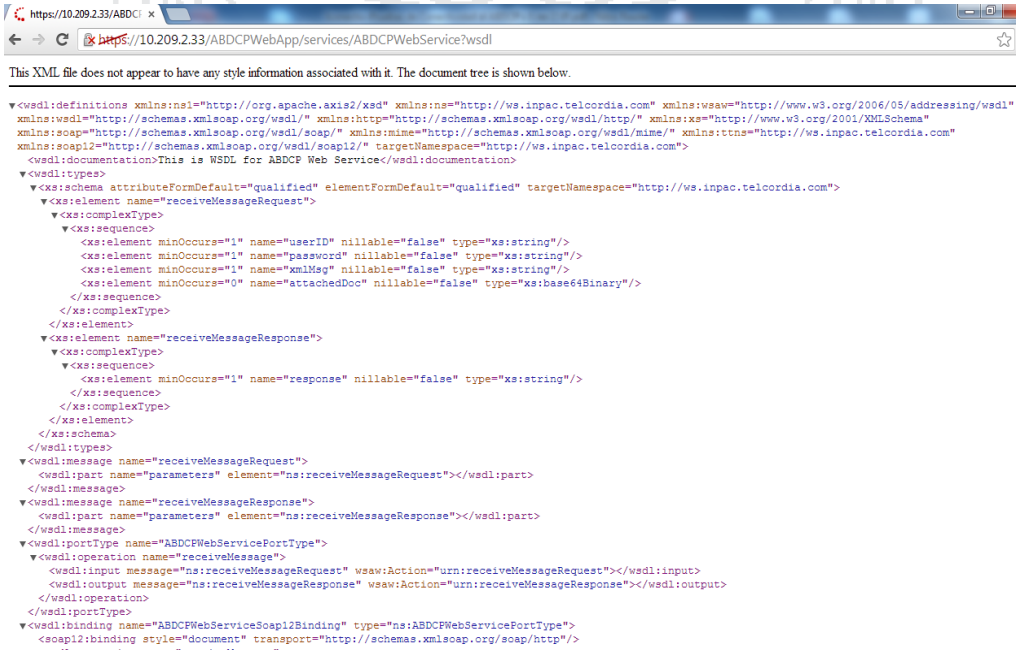


telnet 10.209.3.130 443



telnet 10.209.3.130 22





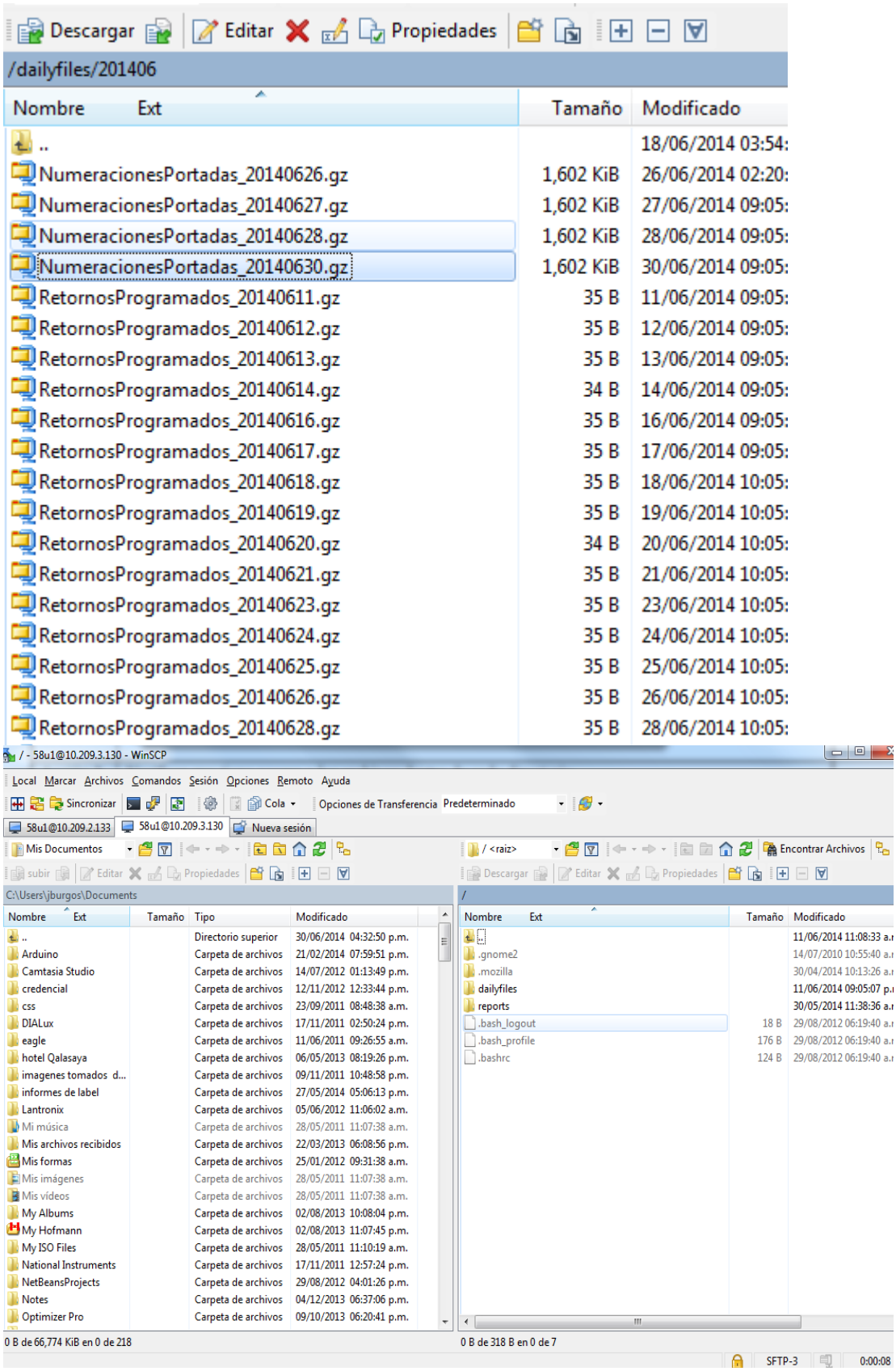
https://10.209.3.130/ABDC

https://10.209.3.130/ABDCPWebApp/services/ABDCPWebService?wsdl

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
▼<wsdl:definitions xmlns:ns1="http://org.apache.axis2/xsd" xmlns:ns="http://ws.inpac.telcordia.com" xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:ttns="http://ws.inpac.telcordia.com"
 xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" targetNamespace="http://ws.inpac.telcordia.com">
  <wsdl:documentation>This is WSDL for ABDCP Web Service</wsdl:documentation>
 ▼<wsdl:types>
  ▼<xs:schema attributeFormDefault="qualified" elementFormDefault="qualified" targetNamespace="http://ws.inpac.telcordia.com">
   ▼<xs:element name="receiveMessageRequest">
    ▼<xs:complexType>
     ▼<xs:sequence>
        <xs:element minOccurs="1" name="userID" nillable="false" type="xs:string"/>
        <xs:element minOccurs="1" name="password" nillable="false" type="xs:string"/>
        <xs:element minOccurs="1" name="xmlMsg" nillable="false" type="xs:string"/>
        <xs:element minOccurs="0" name="attachedDoc" nillable="false" type="xs:base64Binary"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   ▼<xs:element name="receiveMessageResponse">
    ▼<xs:complexType>
     ▼<xs:sequence>
        <xs:element minOccurs="1" name="response" nillable="false" type="xs:string"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:schema>
  </wsdl:types>
 ▼<wsdl:message name="receiveMessageRequest">
    <wsdl:part name="parameters" element="ns:receiveMessageRequest"></wsdl:part>
  </wsdl:message>
 ▼<wsdl:message name="receiveMessageResponse">
    <wsdl:part name="parameters" element="ns:receiveMessageResponse"></wsdl:part>
  </wsdl:message>
 ▼<wsdl:portType name="ABDCPWebServicePortType">
  ▼<wsdl:operation name="receiveMessage">
     <wsdl:input message="ns:receiveMessageRequest" wsaw:Action="urn:receiveMessageRequest"></wsdl:input>
     <wsdl:output message="ns:receiveMessageResponse" wsaw:Action="urn:receiveMessageResponse"></wsdl:output>
   </wsdl:operation>
  </wsdl:portType>
 ▼<wsdl:binding name="ABDCPWebServiceSoap12Binding" type="ns:ABDCPWebServicePortType">
    <soap12:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  ▼<wsdl:operation name="receiveMessage">
```


```
[root@localhost ~]# wget https://10.209.2.33/ABDCPWebApp/services/ABDCPWebService?wsdl --no-check-certificate
--2014-06-30 11:26:15--  https://10.209.2.33/ABDCPWebApp/services/ABDCPWebService?wsdl
Connecting to 10.209.2.33:443... conectado.
WARNING: cannot verify 10.209.2.33's certificate, issued by `/C=XX/L=Default City/O=Default Company Ltd/CN=10.209.2.33':
  Self-signed certificate encountered.
PeticiÃ³n HTTP enviada, esperando respuesta... 200 OK
Longitud: no especificado [text/xml]
Saving to: `ABDCPWebService?wsdl.1'

    [ <=>                                                                                                                                ] 3.052       --.-K/s   in 0s

2014-06-30 11:26:16 (6,09 MB/s) - `ABDCPWebService?wsdl.1' saved [3052]

[root@localhost ~]# wget https://10.209.3.130/ABDCPWebApp/services/ABDCPWebService?wsdl --no-check-certificate
--2014-06-30 11:28:50--  https://10.209.3.130/ABDCPWebApp/services/ABDCPWebService?wsdl
Connecting to 10.209.3.130:443... conectado.
WARNING: cannot verify 10.209.3.130's certificate, issued by `/C=CL/ST=SANTIAGO/L=SANTIAGO/O=ADEXUS/OU=TEC/CN=10.209.3.130':
  Self-signed certificate encountered.
PeticiÃ³n HTTP enviada, esperando respuesta... 200 OK
Longitud: no especificado [text/xml]
Saving to: `ABDCPWebService?wsdl.2'

    [ <=>                                                                                                                                ] 3.053       --.-K/s   in 0,001

2014-06-30 11:28:51 (4,69 MB/s) - `ABDCPWebService?wsdl.2' saved [3053]
```



ANEXO III

WSDL DEL SERVIDOR SOAP ABDCP

This XML file does not appear to have any style information associated with it. The document tree is shown below.

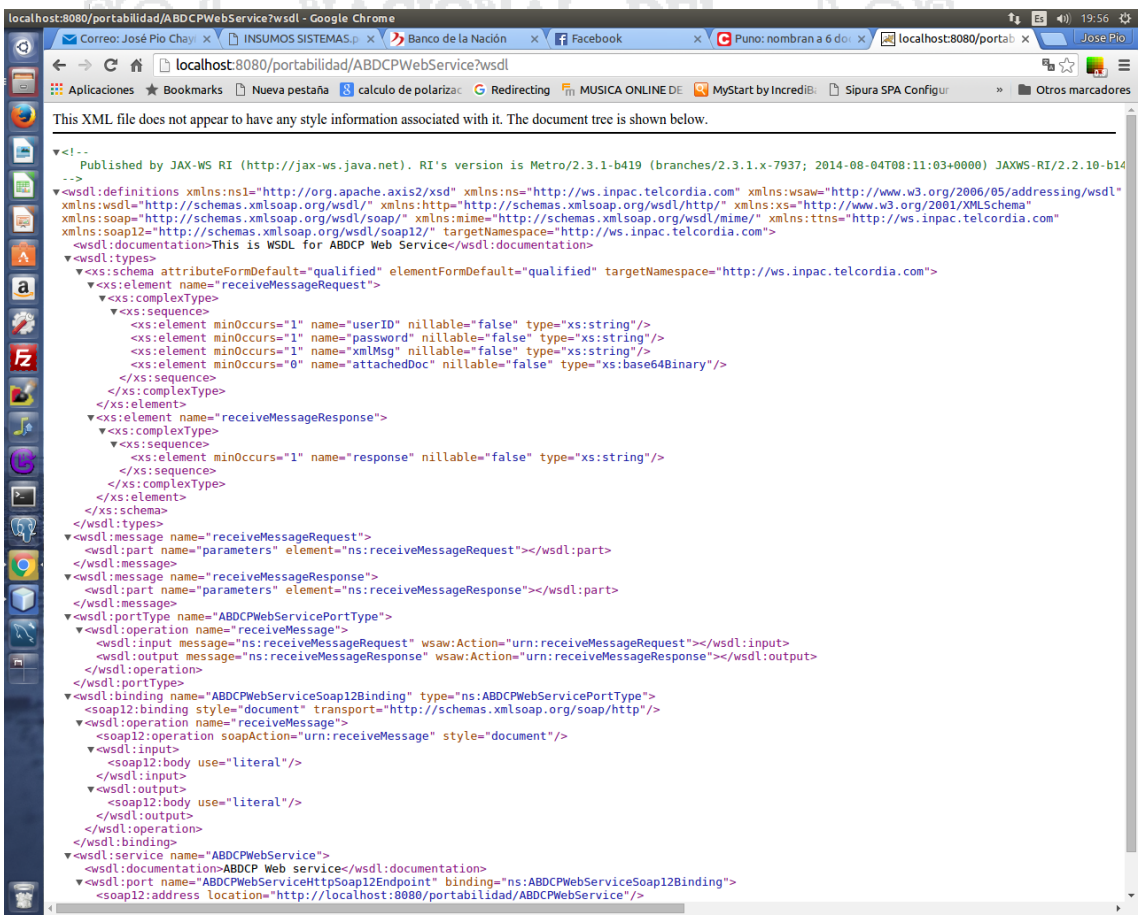
```
<wsdl:definitions xmlns:ns1="http://org.apache.axis2/xsd" xmlns:ns="
http://ws.inpac.telcordia.com" xmlns:wsaw="http://www.w3.org/2006/05
/addressing/wsdl"xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns
:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:xs="http://www.w
3.org/2001/XMLSchema"xmlns:soap="http://schemas.xmlsoap.org/wsdl/soa
p/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:ttns="h
ttp://ws.inpac.telcordia.com"xmlns:soap12="http://schemas.xmlsoap.or
g/wsdl/soap12/" targetNamespace="http://ws.inpac.telcordia.com">
  <wsdl:documentation>This is WSDL for ABDCP Web
  Service</wsdl:documentation>
  <wsdl:types>
    <xs:schema attributeFormDefault="qualified" elementFormDefault="
    qualified" targetNamespace="http://ws.inpac.telcordia.com">
      <xs:element name="receiveMessageRequest">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="1" name="userID" nillable="false"
             type="xs:string"/>
            <xs:element minOccurs="1" name="password" nillable="fals
            e" type="xs:string"/>
            <xs:element minOccurs="1" name="xmlMsg" nillable="false"
             type="xs:string"/>
            <xs:element minOccurs="0" name="attachedDoc" nillable="f
            alse" type="xs:base64Binary"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="receiveMessageResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="1" name="response" nillable="fals
            e" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="receiveMessageRequest">
    <wsdl:part name="parameters" element="ns:receiveMessageRequest">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="receiveMessageResponse">
    <wsdl:part name="parameters" element="ns:receiveMessageResponse"
    ></wsdl:part>
  </wsdl:message>
  <wsdl:portType name="ABDCPWebServicePortType">
    <wsdl:operation name="receiveMessage">
      <wsdl:input message="ns:receiveMessageRequest" wsaw:Action="ur
      n:receiveMessageRequest"></wsdl:input>
      <wsdl:output message="ns:receiveMessageResponse" wsaw:Action="
      urn:receiveMessageResponse"></wsdl:output>
    </wsdl:operation>
  </wsdl:portType>
```

```
  <wsdl:binding name="ABDCPWebServiceSoap12Binding" type="ns:ABDCPWe
  bServicePortType">
    <soap12:binding style="document" transport="http://schemas.xmlso
    ap.org/soap/http"/>
    <wsdl:operation name="receiveMessage">
      <soap12:operation soapAction="urn:receiveMessage" style="docum
      ent"/>
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="ABDCPWebService">
    <wsdl:documentation>ABDCP Web service</wsdl:documentation>
    <wsdl:port name="ABDCPWebServiceHttpSoap12Endpoint" binding="ns:
    ABDCPWebServiceSoap12Binding">
      <soap12:address location="http://10.209.3.130/ABDCPWebApp/serv
      ices/ABDCPWebService/"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

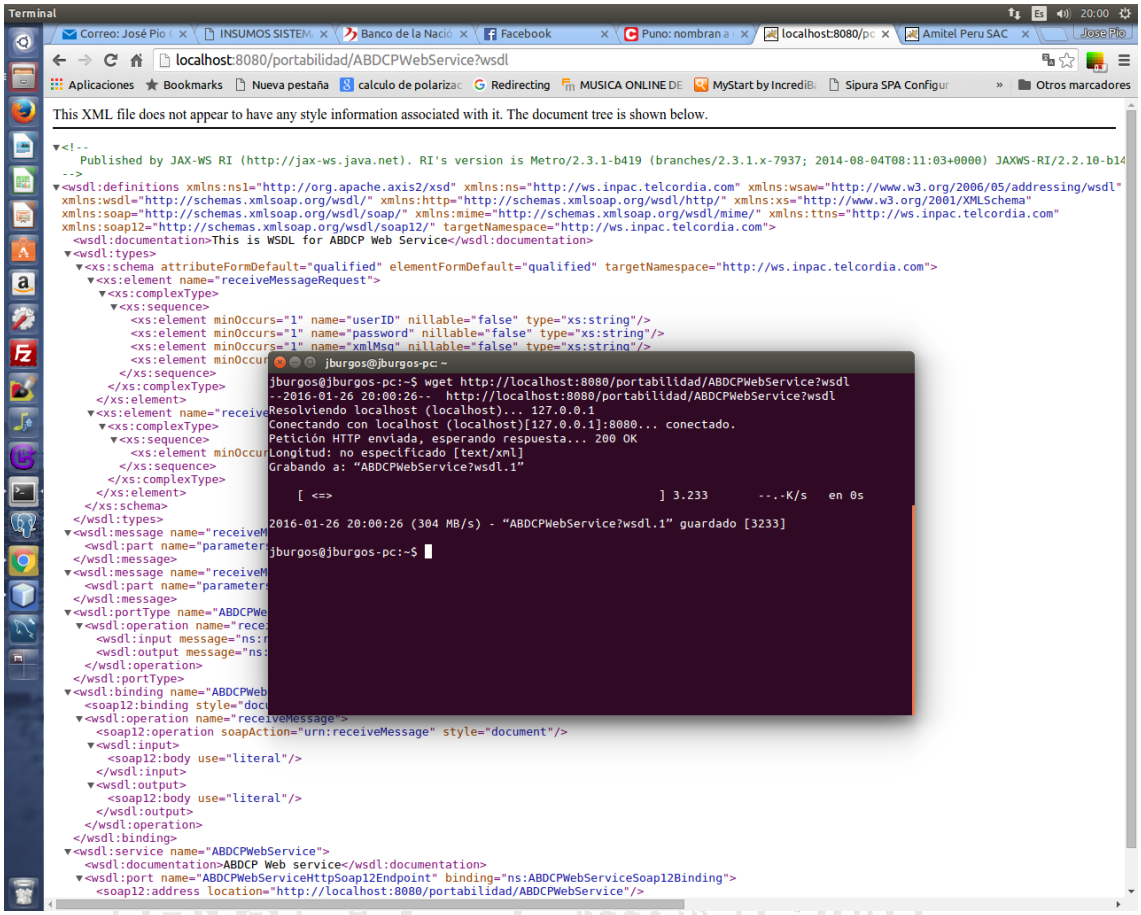
ANEXO IV PRUEBAS DE OPERATIVIDAD

Servidor SOAP Local (lista de servicios)

Servidor SOAP Local (WSDL)



Servidor SOAP local (Prueba de funcionamiento)



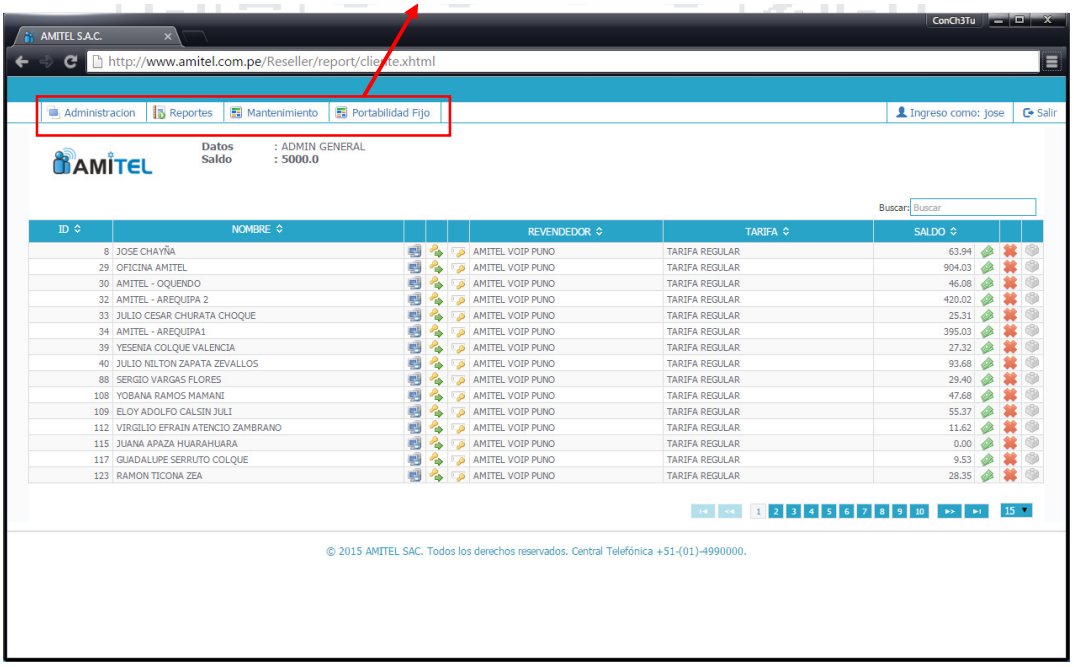
Cliente SOAP (Petición de consulta previa)

ANEXO VI MANUAL DE USUARIO

- ACCESO AL SISTEMA

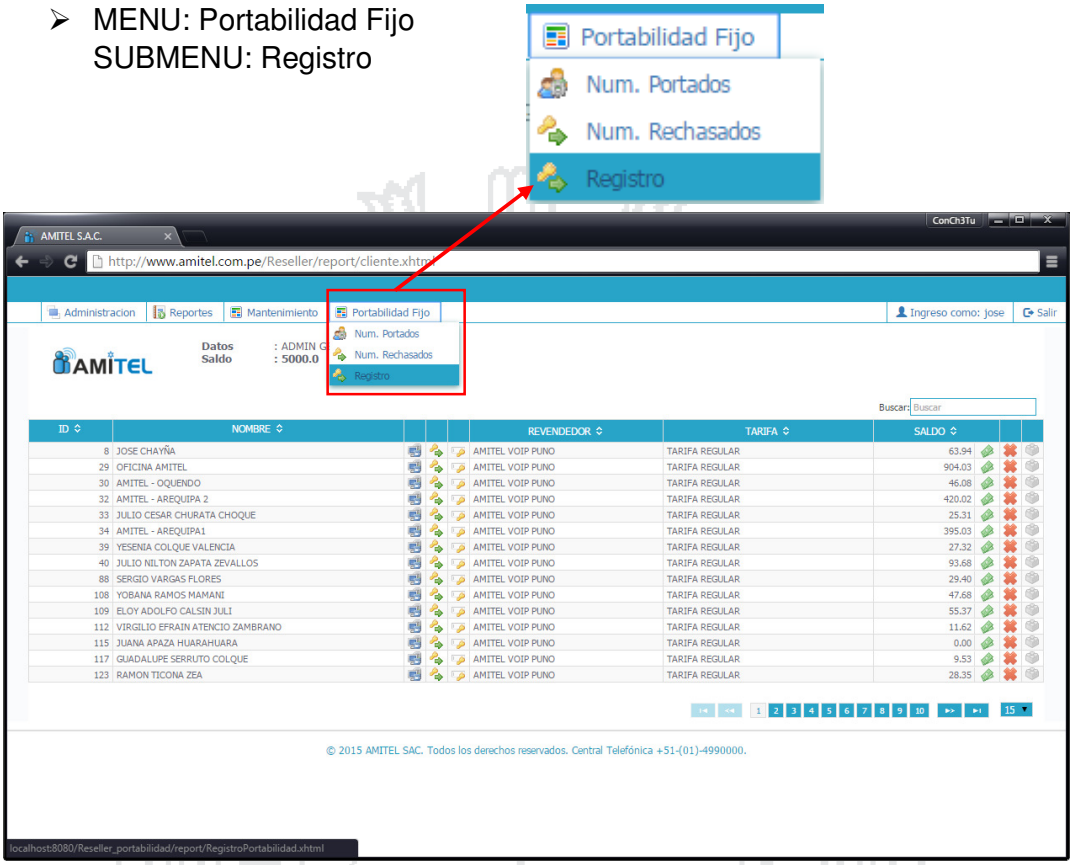
Para ingresar al sistema de Portabilidad Fija deberá especificar el nombre de usuario y contraseña que le fue asignado.

Existen 4 menús principales, los cuales tienen una serie de opciones que se deben de configurar



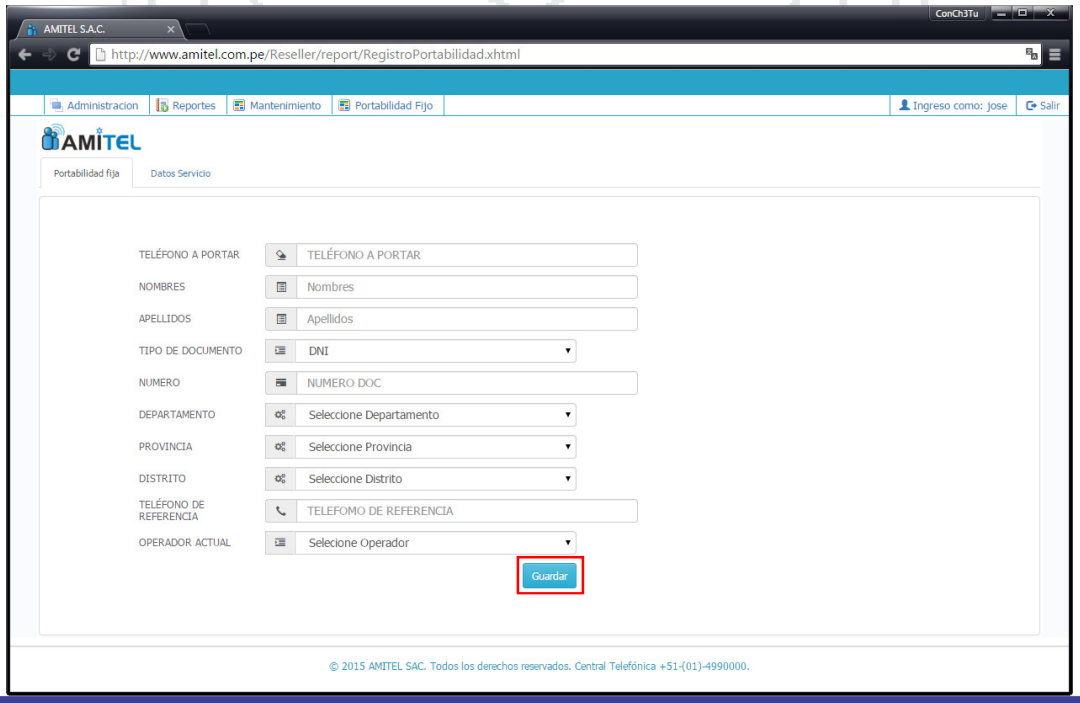
A continuación se explica el funcionamiento de del menú Portabilidad Fijo.

- MENU: Portabilidad Fijo
  SUBMENU: Registro

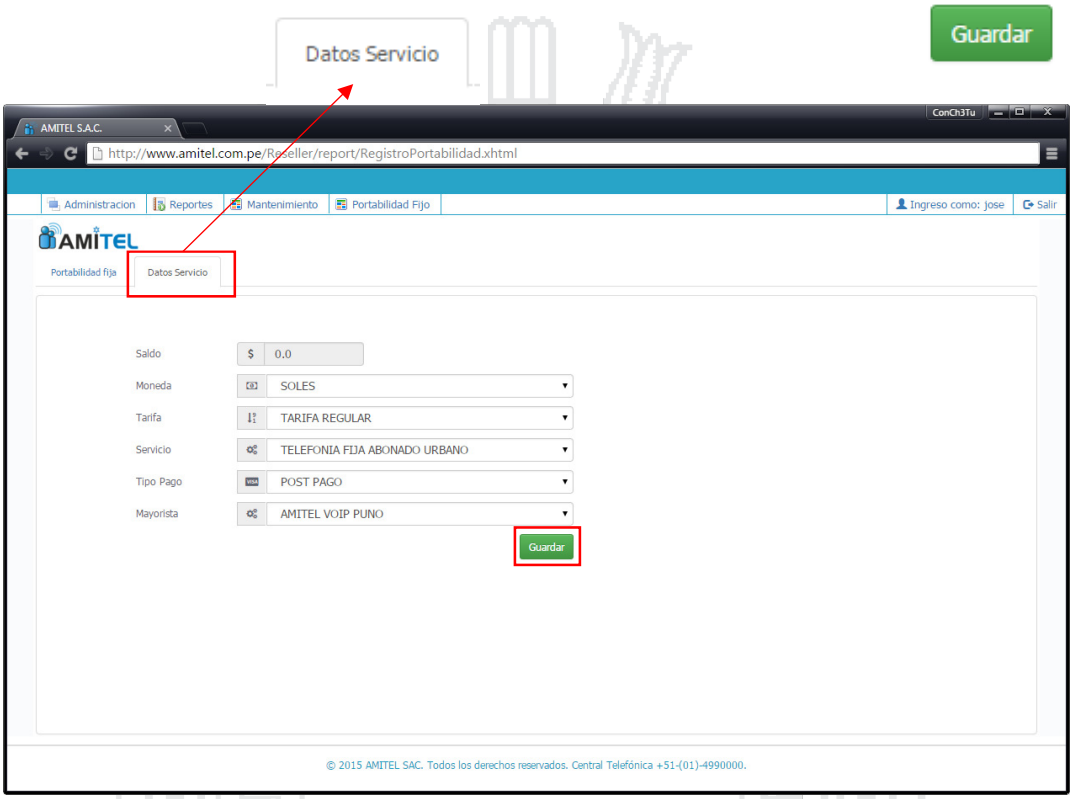


En esta pantalla se deben de capturar los datos del cliente como número Teléfono a Portar, Nombres y Apellidos, tipo de documento de identidad, número de documento de identidad, lugar de procedencia como departamento, provincia y distrito, teléfono de referencia y el operador actual.

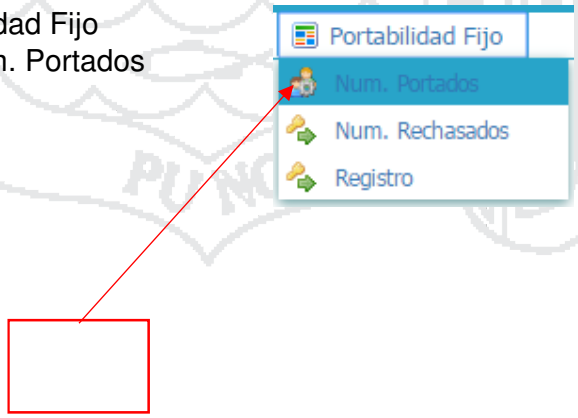
Una vez llenado los datos solicitados de manera correcta, presionar 

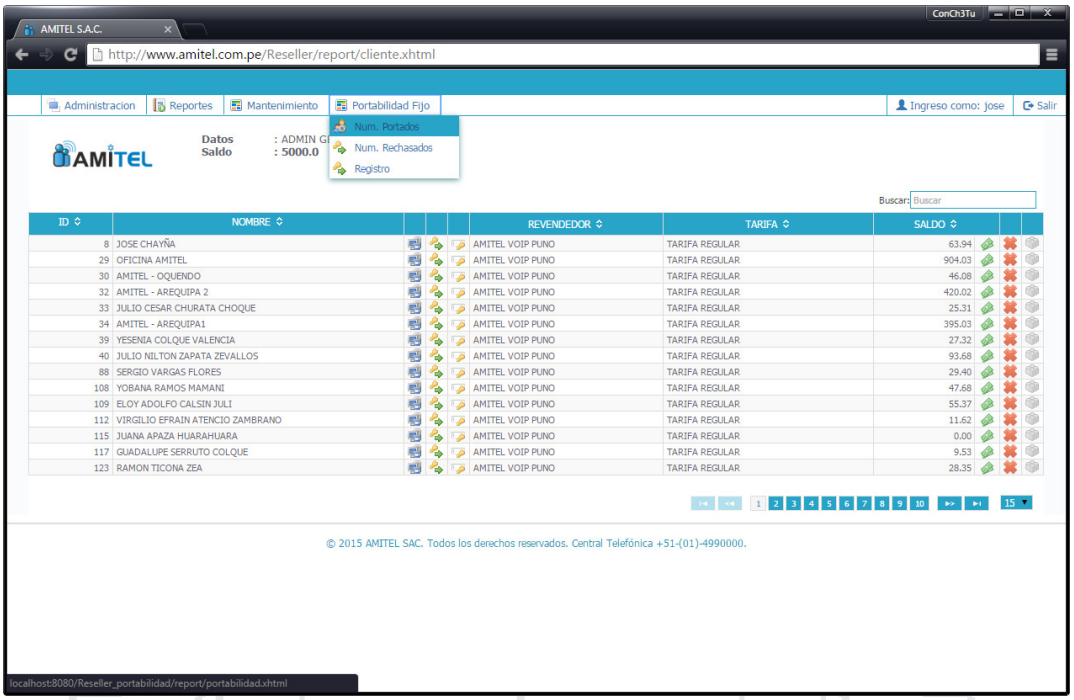


- PESTAÑA: Datos de Servicio
  Esta pestaña se genera los datos de Saldo, Moneda, Tarifa, Servicio, Tipo de pago y Mayorista.
  Una vez verificado los datos de manera correcta presionar el botón



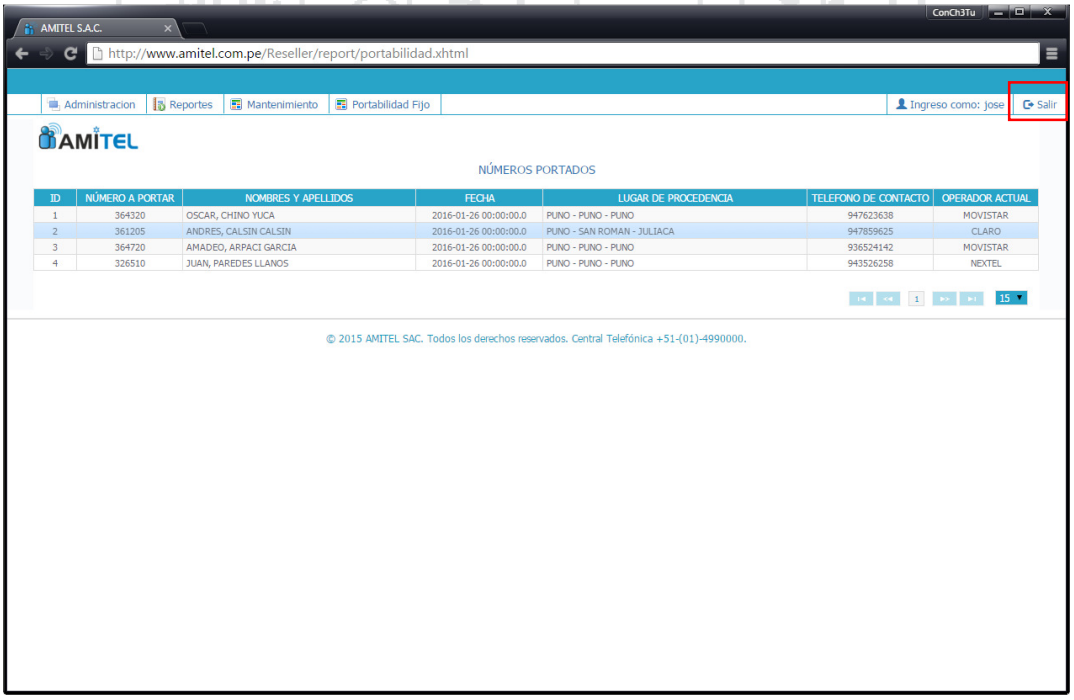
- MENU: Portabilidad Fijo
  SUBMENU: Num. Portados





En esta pantalla es donde se muestra los números portados que agregamos en el Submenú 

con los datos capturados como número Teléfono a Portar, Nombres y Apellidos, tipo de documento de identidad, número de documento de identidad, lugar de procedencia como departamento, provincia y distrito, teléfono de referencia y el operador actual.



Nota: Si se presiona el botón 

se cerrará la sesión y re direccionara al ventana de login.