Tesis Doctoral

Modelo para el diseño de Sistemas Gestores de *Workflows* con funcionalidades Colaborativas, *Cloud* y Móviles.

Autor

Edgar Castelán Maldonado

Tutor

Miguel A. Brigos Hermida

Co-Tutor

Joaquín Fernández Sánchez

Programa de Doctorado en Ingeniería Multimedia

Escuela Técnica Superior de Ingeniería Industrial

Universidad Politécnica de Cataluña

Barcelona, 2014

Tesis presentada para obtener el título de Doctor

por la Universidad Politécnica de Cataluña

Agradecimientos

Profesores:

Joaquín Fernández, Miguel A. Brigos, J. M. Monguet, Carmina Saldaña.

Compañeros:

Marco Ferruzca, Mónica Sampieri, Yliana Rivero, Eduardo Huerta, Yadira Alatriste Susana H. Badillo, Alfredo Gutiérrez, Hugo Dávila, Daniel Castillo, Bruno Alves, y a todo el equipo de trabajo del LAM.

Amigos:

Roald Dijkstra, Aida Salazar, Patricia Tamayo, James Bateman, Fabio Muniz, Nate Finch, Simisola Okoya, Raj Belbase, Rubén Hernández.

Familia:

A mis padres Ociel y Ligia, a mi hermano Ociel y mi hermana Isis, y a toda mi familia en México por apoyarme en esta etapa de mi vida.

Título

Modelo para el diseño de Sistemas Gestores de *Workflows* con funcionalidades Colaborativas, *Cloud* y Móviles.

Palabras Clave

Diseño, *workflow*, procesos de negocios, sistemas gestores, aplicaciones móviles, servicios móviles, colaboración, computación en la nube, arquitectura de software.

Abstract

This research was developed in the context of WFMS (*Workflow Management Systems*), mobile applications, cloud computing, and collaborative systems. Currently the design of WFMS is based on the reference model proposed by the WfMC (*Workflow Management Coalition*). The problem that exists today in the design and development of WfMS is that the reference model proposed by the WfMC was designed many years before the rise of mobile technologies, cloud computing and collaborative systems. It is important to create a new model for the design of WfMS taking in to account the new technological features and functionalities offered by mobile devices, collaborative and cloud computing services along with new paradigms of collaboration that can occur when using these technological solutions. This research has the general objective of obtain a model for the design of WfMS with Collaborative, Cloud and Mobile functionalities.

Resumen

Este trabajo de investigación se desarrolla en el contexto de WfMS (*Workflow Management Systems*), aplicaciones móviles, *cloud computing* y sistemas colaborativos. Actualmente el diseño de WfMS está basado en el modelo de referencia propuesto por la WfMC (*Workflow Management Coalition*). Actualmente el diseño y desarrollo de WfMS sigue el modelo de referencia propuesto por la WfMC, que fue diseñado con anterioridad a que surgierán las tecnologías móviles, *cloud computing* y los sistemas colaborativos. Es importante crear un nuevo modelo para el diseño de WfMS que tenga en cuenta las nuevas características y funcionalidades tecnológicas que ofrecen los dispositivos móviles, los servicios colaborativos y de *cloud computing,* junto con los nuevos paradigmas de colaboración que se pueden dar al utilizar estas soluciones tecnológicas. Esta investigación tiene como objetivo principal el proponer un modelo para el diseño de WfMS con funcionalidades Colaborativas, *Cloud* y Móviles.

Índice

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1. Introducción

1.1. Contexto de la Investigación

El presente trabajo de investigación se desarrolla en el contexto de *Workflow Management Systems* (WfMS), aplicaciones móviles, *cloud computing* y sistemas colaborativos.

En el mapa conceptual que se aprecia en la figura 1 se explica el universo de conocimiento de esta investigación. La parte central de la figura se puede apreciar el concepto principal de la investigación los WfMS. Al converger con las otras tecnologías crea el marco teórico-tecnológico que da rigor a la investigación.

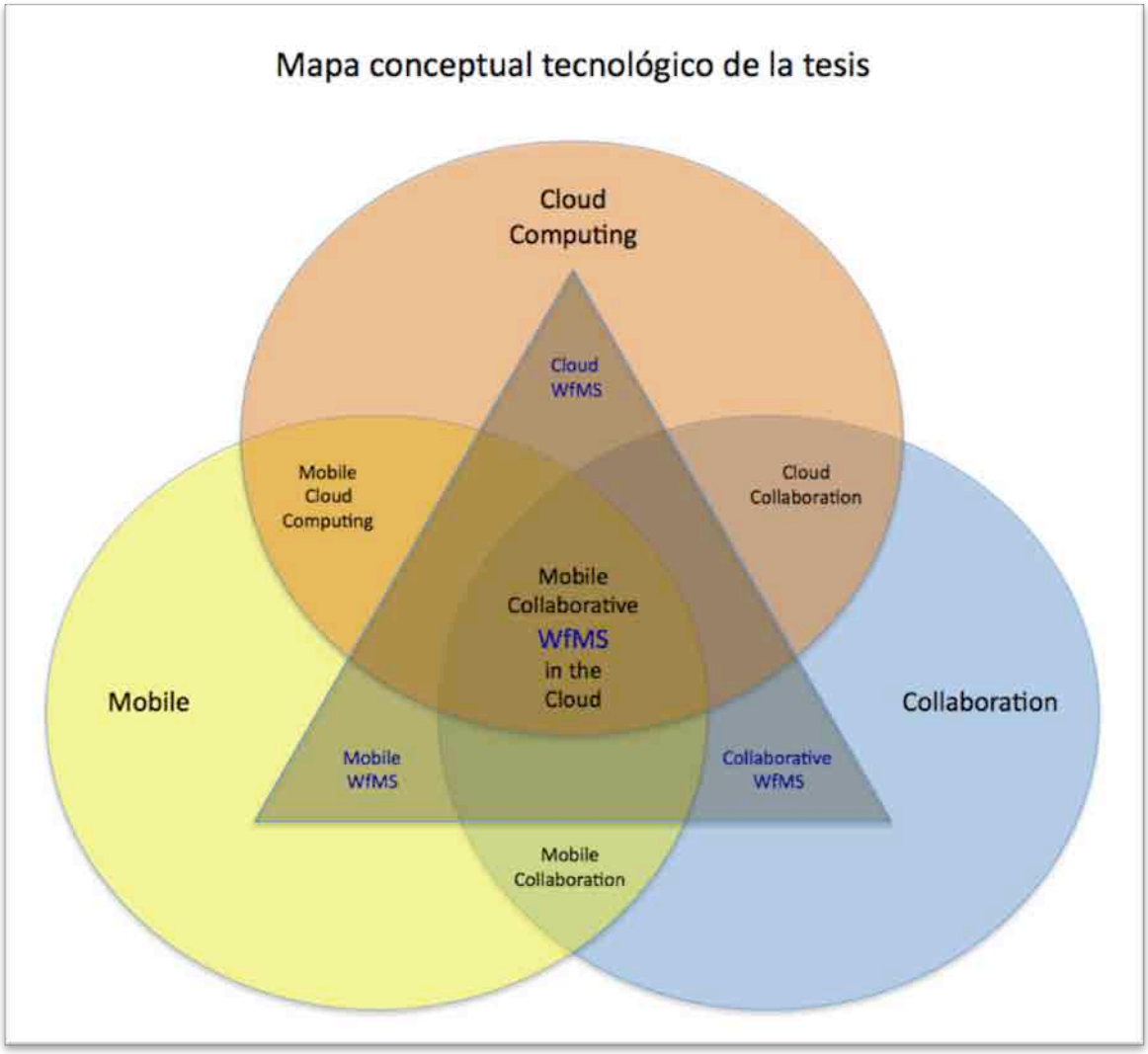

Figura 1 : Mapa conceptual tecnológico de la investigación.

El término *Workflow* es usado como sinónimo de proceso de negocios y se puede definir como "la automatización de un proceso de negocios, en parte o en su totalidad, donde documentos, información y tareas son pasados de un participante a otro para realizar una acción, de acuerdo con un conjunto de reglas y procedimientos para alcanzar o contribuir con todas las metas del negocio" (Hollingsworth 1995).

Para gestionar *Workflows* se hace uso de un WfMS, que se encarga de asegurar que la información correcta llegue a la persona correcta en el tiempo correcto, o que la información sea enviada a la aplicación correcta en el momento adecuado. Un WfMS cuenta con herramientas necesarias para diseñar, modelar y definir procesos y *Workflows*, incluyendo las actividades que lo componen. Provee un ambiente operacional en el cual los *Workflows* (procesos) son ejecutados, manejando las actividades de cada uno de los procesos en forma secuencial. Cuenta con las interfaces necesarias para que los usuarios y aplicaciones puedan interactuar con el sistema, con la finalidad de procesar cada una de las actividades (Hollingsworth 1995; Alonso et al. 1997).

Actualmente los dispositivos móviles pueden realizar tareas con similar nivel de complejidad que las de un ordenador de escritorio, lo que permite utilizar WfMS en este tipo de dispositivos. Sin embargo el uso de dispositivos móviles puede aportar nuevas funcionalidades que dan al usuario las facilidades tanto de movilidad, conexión permanente a *Internet* u otras comunicaciones. Permiten a los usuarios realizar tareas de una forma diferente a como se hace en un ordenador, teniendo la oportunidad de realizar tareas independientemente del lugar, del estado (en movimiento o estacionarios) y de la hora del día en que se encuentren.

La movilidad puede darse en espacio y tiempo; está relacionada con personas, dispositivos, aplicaciones y objetos. Los servicios móviles ayudan a los usuarios a superar restricciones de espacio y tiempo (Pura & Heinonen 2008) permitiendo el acceso a la información de forma universal, portable y flexible; gracias a la libertad de espacio y tiempo que tienen los servicios móviles (Balasubramanian et al. 2002). Esto quiere decir que si existe una red de comunicaciones, un servicio móvil siempre va a estar disponible independientemente del lugar dónde se encuentre el usuario y la hora del día en que quiera utilizar el servicio.

En la actualidad los dispositivos móviles cuentan con sensores que permiten recolectar información contextual de forma manual o automática (Bae et al. 2013). Los servicios móviles dependen en gran medida de la movilidad y por consecuencia del contexto.

Se puede entender por contexto "cualquier información que es utilizada para describir las características de la situación en la que se encuentra una entidad (persona, objeto o lugar) y que es considerada relevante en la interacción entre el usuario y la aplicación" (Dey et al. 2001). Por ejemplo muchos usuarios utilizan información contextual de geolocalización para hacer saber a sus contactos de redes sociales en qué lugar fue tomada la fotografía que han colgado en su muro.

Las aplicaciones móviles pueden hacer uso de una gran variedad de servicios móviles a través de *Mobile Cloud Computing* y de esta forma proveer a los usuarios móviles de servicios como por ejemplo de localización, facilidades de almacenamiento y procesamiento, fuera del dispositivo móvil (Dinh & Lee 2011; Fernando et al. 2013).

Mobile cloud computing ofrece la posibilidad de utilizar infraestructuras, plataformas y *software* a un bajo coste y bajo demanda a través de dispositivos móviles. Ayuda a mejorar la movilidad y portabilidad de la computación móvil extendiendo el tiempo de vida de la batería y mejorando la capacidad de almacenamiento y procesamiento.

Las herramientas colaborativas permiten a los usuarios publicar, compartir y gestionar información. La creación conjunta de información y el poder compartirla son características clave en este tipo de herramientas, los datos e información de los usuarios son almacenados y procesados en la nube (Yeh et al. 2013; Neyem et al. 2012). Los contenidos son creados y gestionados completamente por el usuario y la función principal de una herramienta colaborativa es permitir compartir estos contenidos (West et al. 2012).

En un ambiente colaborativo en la nube los usuarios pueden contribuir e interactuar con otros usuarios utilizando diferentes dispositivos; como por ejemplo: ordenadores portátiles, tabletas o móviles y utilizando diferentes plataformas: *Linux, Windows, iOS y Android*, ya sea en forma de aplicaciones nativas o en versión *Web* para colaborar virtualmente sobre *Internet* sin restricciones en cuanto a tiempo, localización geográfica y a veces colaborando en diferentes idiomas. Todo lo mencionado anteriormente tiene una gran influencia en el tipo de colaboración que el usuario puede tener con otros

usuarios, es por ello que el contexto y la situación del usuario son dos aspectos importantes en los ambientes colaborativos (Smari et al. 2013; Li et al. 2013).

1.2. Motivación

Actualmente el diseño de WfMS está basado en el modelo de referencia propuesto por la WfMC (*Workflow Management Coalition*)[1], organización que se encarga de crear y contribuir con estándares relacionados con procesos de negocios. La WfMC propuso en 1995 un modelo de referencia para estandarizar el diseño de WfMS, el modelo de referencia es una descripción general de la arquitectura que debe tener un WfMS. Describe las funcionalidades de los componentes de software que son parte esencial en un WfMS y la forma en que deben interactuar entre ellos. El modelo ha sido desarrollado de manera neutral en cuanto a tecnología se refiere, lo que le permite ser independiente de cualquier arquitectura en particular o implementación tecnológica (Hollingsworth 1995).

Sin embargo el modelo de referencia propuesto por la WfMC fue diseñado muchos años antes de que surgierán las tecnologías móviles, *cloud computing* y los sistemas colaborativos, lo que supone una diferencia para el diseño y desarrollo de WfMS, debido a las características propias de los dispositivos y aplicaciones móviles que permiten al usuario trabajar en diferentes contextos, utilizar información relacionada con el contexto, y utilizar servicios en la nube (como por ejemplo de procesamiento, comunicación, almacenamiento, colaboración, medios sociales, etc.). Se considera importante crear un nuevo modelo para el diseño de WfMS que tenga en cuenta las nuevas características y funcionalidades tecnológicas que ofrecen los dispositivos móviles, los servicios colaborativos y de *cloud computing*, procurando los nuevos paradigmas de colaboración que se puden dar al utilizar estas soluciones tecnológicas.

[1] http://www.wfmc.org/

1.3. Objetivos de la Investigación

1.3.1. Objetivo principal

Obtener un modelo tecnológico para el diseño y desarrollo WfMS con funcionalidades Colaborativas, *Cloud* y Móviles.

1.3.2. Objetivos específicos

a) Proponer un Marco Teórico-Tecnológico basado en el mapa conceptual de la investigación (ver fig. 1).

b) Diseñar un modelo tecnológico de WfMS con funcionalidades Colaborativas, *Cloud* y Móviles.

c) Hacer una implementación del modelo tecnológico para obtener una arquitectura de software que sirva para desarrollar aplicaciones móviles de WfMS.

d) Desarrollar una aplicación móvil basada en la implementación de la arquitectura de *software*.

e) Evaluar las funcionalidades del modelo tecnológico con expertos en el uso de WfMS mediante un estudio *Delphi*.

1.4. Contribuciones

Esta investigación tiene como objetivo general el obtener un modelo para el diseño de WfMS con funcionalidades Colaborativas, *Cloud* y Móviles, haciendo las siguientes contribuciones:

- Un modelo para el diseño de WfMS con funcionalidades colaborativas, *cloud* y móviles.
- Una Arquitectura Concreta resultado de la implementación del nuevo modelo.
- Una Arquitectura de Software para el desarrollo de WfMS resultado de la implementación de la Arquitectura Concreta.

- Una aplicación móvil de WfMS para la plataforma *iOS* y dispositivos *iPad* resultado de la implementación de la arquitectura de *software.*
- Una metodología para realizar un estudio *Delphi* utilizando una aplicación móvil de WfMS con herramientas colaborativas en la nube.

1.5. Comunicación de la Investigación

Los resultados de la investigación fueron presentados en forma de artículo de investigación en el siguiente congreso:

- Castelán, E., Brigos, M. A., & Fernández, J. (2014). *A SOFTWARE REFERENCE ARCHITECTURE FOR THE DESIGN AND DEVELOPMENT OF MOBILE WORKFLOW LEARNING APPLICATIONS. In 8th International Technology, Education and Development Conference Valencia - 10th - 12th March 2014.*

La estructura de este trabajo de investigación consta de los siguientes capítulos:

a) Metodología de Investigación.

Se explica la metodología de investigación que sirvió de guia para alcanzar los objetivos propuestos en la tesis.

b) Marco teórico tecnológico.

Se presenta el marco de referencia que sirve para la fundamentación teórica y tecnológica de la investigación.

c) Diseño, implementación y evaluación del Modelo.

Se explican los conceptos y procedimientos que se deben seguir para disñear el modelo de WfMS. Se explica el diseño de las funcionalidades del modelo y la forma en que se implementa y evalúa.

d) Conclusiones

Se hace un resúmen de los resultados obtenidos en la investigación, las limitaciones de la investigación y se mecionan las posibles investigaciones que pueden realizarse en el futuro.

Capítulo 2

Metodología de Investigación

2. Metodología de Investigación.

Debido a que el objetivo principal de esta investigación es el diseño, implementación y evaluación de un artefacto hemos elegido la metodología de investigación en el campo de los sistemas de información conocida como investigación diseño-ciencia.

De acuerdo con Hevner et al. (2004) un artefacto en el campo de las tecnologías de información puede definirse en forma de *constructos* (vocabulario y símbolos), *modelos* (abstracciones y representaciones), *métodos* (algoritmos y procedimientos) e *instancias* (implementaciones y prototipos de sistemas). Estos artefactos son representados de una forma estructurada que van desde software, lógica formal, y de descripciones matemáticas rigurosas a descripciones en lenguaje natural.

En el campo de investigación comportamiento-ciencia de los sistemas de información, el objeto de estudio es casi siempre un artefacto tecnológico implementado en un contexto organizacional. Las teorías buscan predecir o explicar el fenómeno que ocurre con respecto al uso del artefacto, la utilidad percibida, y el impacto en los individuos y la organización (Delone & Mclean 2003).

En cambio el paradigma de investigación diseño-ciencia en el ciclo de investigación de los sistemas de información está completamente orientado a la solución de problemas y tiene como meta principal la creación y evaluación de artefactos que sirvan para un propósito práctico, estos artefactos deben ser completamente relevantes y novedosos en su entorno de aplicación (Nicolaou & Geerts 2011).

Esta metodología debe ser utilizada para resolver problemas de forma única e innovadora o para tratar de mejorar problemas anteriormente resueltos de una forma más efectiva o eficiente (Hevner et al. 2004).

Hevner et al. (2004) explica que para poder entender el diseño-ciencia como un paradigma de investigación de los sistemas de información, se debe comprender que el diseño es tanto un proceso (actividades) como un producto (artefacto), que describe cómo actúa el mundo (procesos) y como es percibido (artefactos).

El proceso de diseño es una secuencia de actividades expertas que producen un artefacto innovador. La evaluación del artefacto provee una retroalimentación de información y un mejor entendimiento del problema con el fin de mejorar tanto la calidad del producto como la calidad del proceso de diseño. Durante este proceso creativo de construir y evaluar el investigador debe ser competente para incluir como parte de la investigación el poder ir mejorando y evolucionando tanto el proceso de diseño como el artefacto.

2.1. Ciclos de investigación en el paradigma Diseño-Ciencia

En la figura 2 se pueden ver los tres ciclos inherentes a la investigación diseño-ciencia en el marco de investigación de los sistemas de información propuesto por (Hevner 2007), y que sirve para entender, ejecutar y evaluar investigaciones.

El ciclo de relevancia sirve de puente entre el ámbito contextual del proyecto de investigación con las actividades de investigación del diseño-ciencia. El ciclo de rigor conecta las actividades del diseño-ciencia con los fundamentos científicos de la base de conocimiento. El ciclo de diseño es iterativo entre las actividades principales para construir y evaluar el diseño de artefactos y procesos de la investigación (Hevner & Chatterjee 2010).

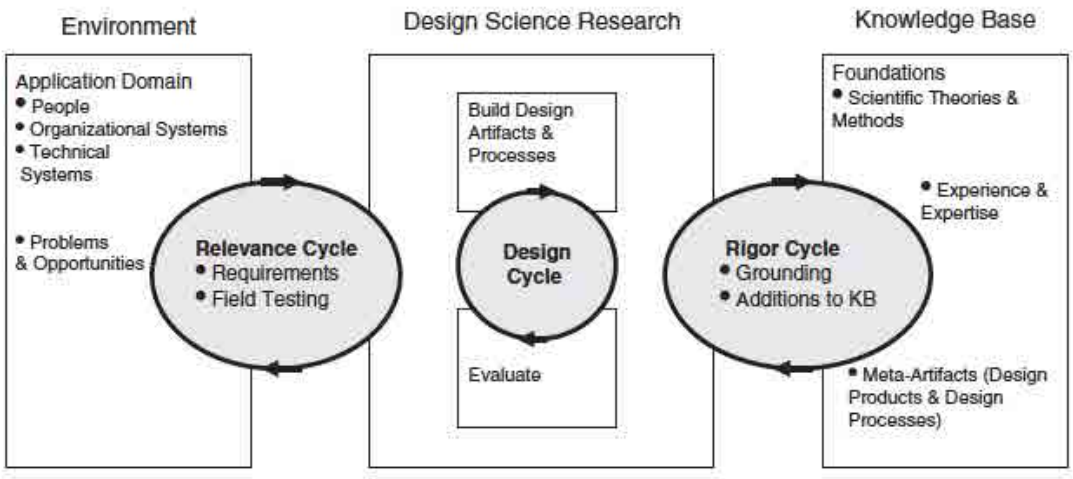

Figura 2: Ciclos de investigación en el paradigma Diseño-Ciencia, fuente (Hevner & Chatterjee 2010).

2.1.1. Ciclo de relevancia

En la figura 2 el ámbito define el espacio del problema donde reside el fenómeno de interés. Está compuesto por personas, organizaciones y las tecnologías existentes o que existirán en el futuro. Las necesidades de negocios que existen dentro de este ámbito son posicionadas relativamente con la infraestructura tecnológica, aplicaciones, arquitecturas de comunicación y posibilidades de desarrollo. De todo lo anterior es donde surge el problema percibido por el investigador (Hevner et al. 2004).

El ciclo de relevancia es el que inicia la investigación diseño-ciencia en un ámbito que además de proveer los requerimientos de la investigación, identificando el problema a solucionar, provee también los criterios para realizar la evaluación final de los resultados de la investigación. El resultado de la investigación diseño-ciencia debe ser devuelto al ámbito para ser estudiado y evaluado en el dominio de aplicación.

Los resultados de las pruebas de campo determinarán si es necesario realizar iteraciones adicionales del ciclo de relevancia en el proyecto de investigación. Se podría dar el caso que el nuevo artefacto tuviera deficiencias en su funcionalidad o en sus cualidades inherentes que podrían limitar su utilidad en la práctica.

Otro caso puede ser que las pruebas de campo den como resultado que los requerimientos de entrada para la investigación fueron incorrectos o incompletos, diseñando un artefacto que es inadecuado para resolver el problema propuesto.

Una nueva interacción del ciclo de relevancia deberá comenzar con una retroalimentación del ámbito del campo de pruebas y de un nuevo planteamiento de los requerimientos de la investigación, estos requerimientos son obtenidos a partir de la experiencia obtenida (Hevner & Chatterjee 2010).

2.1.2. Ciclo de rigor

La base de conocimiento (ver figura 2), hace uso de las ciencias del comportamiento para aportar las teorías y metodologías que explican o predicen el fenómeno relacionado con el problema. Investigaciones previas y resultados de otras disciplinas proveen teorías, marcos, instrumentos, modelos y métodos que van a ser de utilidad en la fase de diseño y construcción.

El diseño-ciencia toma de la base de conocimiento científico teorías y métodos que proveen los fundamentos para el rigor científico de la investigación diseño-ciencia. De esta base de conocimientos se obtienen las experiencias y conocimientos que definen el estado del arte en el campo de aplicación de la investigación. De la base de conocimientos es de donde podemos obtener los artefactos y procesos existentes en el campo de aplicación (Hevner & Chatterjee 2010).

El ciclo de rigor provee conocimiento adquirido en investigaciones anteriores para asegurar que el proyecto de investigación sea innovador. El ciclo de rigor supedita a investigar a fondo y hacer referencias a la base de conocimiento con el propósito de garantizar que los diseños producidos son contribuciones de investigación (Hevner & Chatterjee 2010).

La consideración de rigor en el diseño-ciencia está basado en las habilidades del investigador para seleccionar y aplicar las teorías y métodos apropiados para la construcción y evaluación del artefacto. Las aportaciones a la base de conocimiento como resultado de la investigación diseño-ciencia pueden ser: cualquier adición o extensión a las teorías y métodos originales hechos durante la investigación, nuevos artefactos, y todas las experiencias ganadas como resultado ejecutar el ciclo del diseño iterativo así como las pruebas del artefacto en el campo de aplicación (Hevner & Chatterjee 2010).

2.1.3. Ciclo de diseño

Este ciclo de actividades de investigación itera rápidamente entre la construcción del artefacto, la evaluación y la retroalimentación para refinar nuevamente el diseño. La naturaleza de este ciclo es generar diseños alternativos y evaluarlos con respecto a los requerimientos hasta que se alcance un diseño satisfactorio. Los requerimientos son proveídos por el ciclo de relevancia, las teorías y métodos de diseño y evaluación son tomados del ciclo de rigor. Es importante entender las dependencias que tiene el ciclo de diseño en los otros dos ciclos y al mismo tiempo apreciar su relativa independencia durante el proceso de investigación. Debe haber un esfuerzo balanceado entre construir

y evaluar el artefacto. Ambas actividades deben estar completamente basadas en relevancia y rigor (Hevner & Chatterjee 2010).

2.2. Pautas para el desarrollo de una investigación Diseño-Ciencia

En la tabla 1 se pueden ver resumidos las pautas propuestas por Hevner et al. (2004) para asistir a los investigadores y entender los requerimientos para desarrollar y llevar a cabo una investigación diseño-ciencia de forma efectiva. Los investigadores deben utilizar sus habilidades y juicio para determinar cuándo, dónde y cómo aplicar cada una de las pautas en un proyecto de investigación específico.

Pauta	Descripción
1: Diseño como un artefacto	La investigación diseño-ciencia debe producir un artefacto viable en forma de constructo, modelo, método o instancia.
2: Relevancia del problema	El objetivo de la investigación diseño- ciencia es el desarrollo de soluciones basadas en tecnología para resolver problemas relevantes e importantes.
3: Evaluación del diseño	La utilidad, calidad y eficacia del diseño del artefacto debe ser rigurosamente demostrada vía métodos de evaluación debidamente ejecutados.
4: Contribuciones de la investigación	Una investigación diseño-ciencia efectiva debe proveer contribuciones claras y que puedan ser verificadas en las áreas de diseño del artefacto, bases de diseño y/o metodologías de diseño.
5: Rigor en la investigación	La investigación diseño-ciencia se apoya en la aplicación de métodos rigurosos tanto en la construcción como en la evaluación del diseño del artefacto.
6: Diseño como un proceso de búsqueda	La búsqueda de un artefacto efectivo requiere utilizar medios disponibles para alcanzar un fin deseado mientras se satisfacen las leyes del ambiente del problema.
7: Comunicación de la investigación	Las investigaciones diseño-ciencia deben ser presentadas de forma efectiva a las audiencias orientadas a la tecnología y a las orientadas a la gestión.

Tabla 1: Pautas para la investigación Diseño-Ciencia, fuente (Hevner et al. 2004).

2.2.1. Aplicación de las pautas

De acuerdo con Hevner et al. (2004), las contribuciones surgen de la utilidad. Si los artefactos ya existentes son adecuados, entonces la investigación diseño-ciencia que crea el nuevo artefacto es innecesaria (irrelevante). Si el nuevo artefacto no se correlaciona adecuadamente con el mundo real (rigor), este artefacto no puede proveer utilidad. Si el artefacto no resuelve el problema (búsqueda, implementación) no tiene utilidad. Si la utilidad no es demostrada (evaluación), entonces no hay una base sobre la cual aceptar que se provee una contribución (contribución). Si el problema, el artefacto y su utilidad no son presentadas de manera tal que las implicaciones para la investigación y la práctica no sean claras, entonces la publicación en la literatura de sistemas de información no es apropiada (comunicación).

Es por esto que se debe tratar de satisfacer los lineamientos propuestos en la tabla 1, a continuación se describe la forma en que se siguen cada uno de ellos dentro de esta investigación.

2.2.1.1. Relevancia del problema

Hoy en día los dispositivos móviles pueden realizar tareas con el mismo nivel de complejidad que las de un ordenador de escritorio, lo que permite utilizar WfMS en este tipo de dispositivos. Las plataformas móviles junto con los servicios colaborativos y de *cloud computing*, permiten extender las funcionalidades de las aplicaciones móviles y superar la limitación de recursos que tienen estos dispositivos. Debido a las características propias de los dispositivos y aplicaciones móviles que permiten al usuario trabajar en diferentes contextos, utilizar información relacionada con el contexto, utilizar servicios de procesamiento, comunicación, almacenamiento, colaboración, medios sociales, etc. es necesario diseñar un nuevo modelo para el desarrollo de WfMS con funcionalidades colaborativas, *cloud* y móviles que tome en cuenta estas características.

2.2.1.2. Rigor en la investigación

Los fundamentos teóricos utilizados en la investigación tienen que ver con las metodologías para el diseño de modelos y arquitecturas en el campo de los sistemas de información (ver apartado 3.6), así como el uso de modelos y arquitecturas de referencia ya reconocidos en el campo de investigación de WfMS (ver apartado 3.8), *clud computing* (ver apartado 3.3) y colaboración (ver apartado 3.5).

Respecto a los fundamentos teóricos en la parte de evaluación, en el ciclo de diseño se utiliza un modelo multidimensional de arquitecturas de software (ver apartado 3.6.6) con la finalidad de evaluar los diseños iterativos del artefacto (modelo de WfMS). Para la evaluación del artefacto en su campo de aplicación se utiliza como metodología de evaluación un estudio *Delphi* (ver aparatado 4.7).

2.2.1.3. El diseño como un proceso de búsqueda

El diseño es esencialmente un proceso de búsqueda para descubrir una solución efectiva para un problema. Durante la fase de diseño se deben identificar los patrones de arquitectura y las soluciones tecnológicas que existen para poder diseñar un modelo de WfMS que cumpla con todos los requisitos y necesidades que se necesitan para solucionar el problema. Esto se debe hacer siguiendo los fundamentos teóricos que sirven de guía para el diseño del artefacto.

2.2.1.4. El diseño como artefacto

El artefacto principal de esta investigación es un modelo de WfMS con funcionalidades colaborativas, *cloud* y móviles. La arquitectura concreta y la arquitectura de software que resulta de la implementación del modelo y la aplicación móvil desarrollada son también artefactos que aporta esta investigación.

2.2.1.5. Evaluación del diseño

Para la evaluación en el ciclo de diseño (ver figura 2) se utiliza un modelo de multidimensional de arquitecturas de software (ver apartado 3.6.6) con la finalidad de evaluar los artefactos que se van diseñando para saber qué tan buena es la solución e ir mejorando el artefacto hasta que se obtenga la solución efectiva para el problema.

Para evaluar el artefacto en el campo de aplicación se realiza un estudio *Delphi* (ver aparatado 4.7).

2.2.1.6. Contribuciones de la investigación

Esta investigación hace las siguientes contribuciones:

- Un modelo para el diseño de WfMS con funcionalidades colaborativas, *cloud* y móviles.
- La implementación del modelo da como resultado una arquitectura concreta y una arquitectura de software para el diseño de WfMS.
- Una aplicación móvil de WfMS para la plataforma *iOS* en dispositivos *iPad*, resultado de la implementación de la arquitectura de *software.*
- La realización de un estudio *Delphi* utilizando una aplicación móvil de WfMS y herramientas colaborativas en la nube.

2.2.1.7. Comunicación de la investigación

Los resultados de la investigación fueron presentados en forma de artículo de investigación en el siguiente congreso:

- Castelán, E., Brigos, M. A., & Fernández, J. (2014). *A SOFTWARE REFERENCE ARCHITECTURE FOR THE DESIGN AND DEVELOPMENT OF MOBILE WORKFLOW LEARNING APPLICATIONS. In 8th International Technology, Education and Development Conference Valencia - 10th - 12th March 2014*.

Se están escribiendo los siguientes artículos para ser presentados en revistas indexadas:

- *A REFERENCE MODEL FOR THE DESIGN AND DEVELOPMENT OF MOBILE WFMS.*

- *THE PRACTICAL CASE OF A DELPHI STUDY WITH MOBILE WfMS AND CLOUD COLABORATION TOOLS*

Capítulo 3

Marco Teórico Tecnológico

3. Marco Teórico Tecnológico

3.1. Introducción

En este capítulo se presenta la base de conocimiento que sirve como fundamentación teórica y tecnológica de la investigación. Se presentan las tecnologías y sus respectivas arquitecturas que servirán como referencia para le definición del modelo de WfMS con funcionalidades colaborativas, cloud y móviles.

Se hace una descripción conceptual de los modelos y arquitecturas que sirven para diseñar un modelo de WfMS. También se describe el modelo de referecia de WfMS, el cual sirve para estandarizar el diseño de WfMS. Se hace una descripción general de la arquitectura que debe tener un WfMS, las funcionalidades de los componentes de software que son parte esencial en un WfMS y la forma en que deben interactuar entre ellos.

3.2. Tecnologías para el desarrollo de aplicaciones *Web*

Las tecnologías *Web* son las plataformas, arquitecturas y leguajes de programación utilizadas para desarrollar páginas y aplicaciones *Web*. En su más básico funcionamiento, utilizando la arquitectura Cliente-Servidor, un navegador *Web* del lado del cliente muestra ficheros escritos en HTML los cuales han sido obtenidos desde un servidor de páginas *Web*. Las páginas y aplicaciones *Web* requieren de una o varias tecnologías del lado del servidor para poder ser creadas casi siempre de una forma dinámica y muchas veces adaptando su contenido dependiendo de acciones que se toman del lado del cliente. Las aplicaciones *Web* pueden ir mejorando en interacción y complejidad cuando las páginas *Web* tienen insertado código que se ejecuta del lado del cliente como puede ser *JavaScript* [2] y programación escrita en *Java*[3].

La arquitectura más básica, que se le conoce como cliente servidor, se puede ver en la Figura 3, donde se muestra cada una de las partes implicadas en la generación e interacción de aplicaciones y páginas *Web*.

[2] https://developer.mozilla.org/en-US/docs/Web/JavaScript
[3] http://www.oracle.com/technetwork/java/index.html

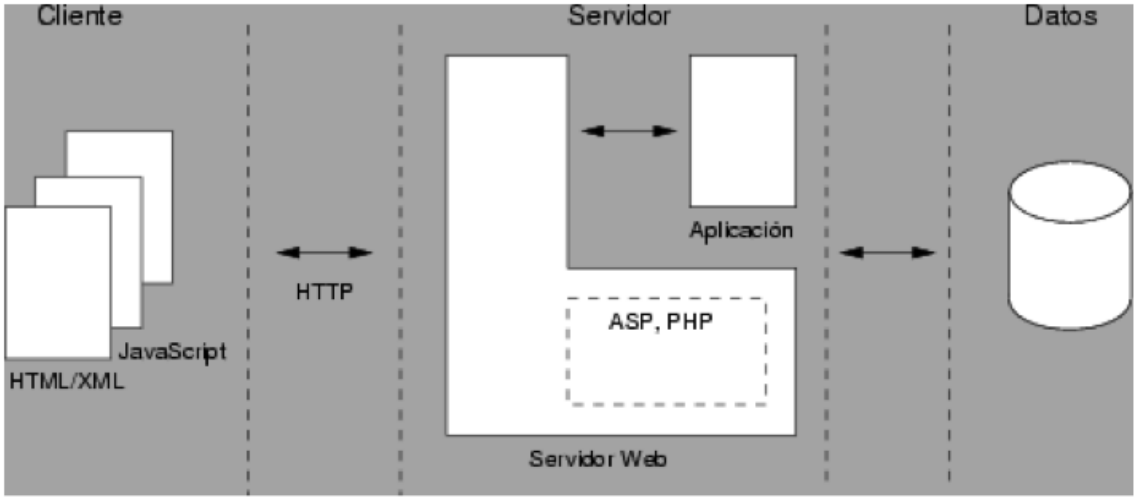

Figura 3: Arquitectura cliente servidor.

A continuación se describen las diferentes tecnologías *Web* que son consideradas esenciales en el contexto de las aplicaciones y páginas *Web*. Teniendo en cuenta que la arquitectura cliente servidor se puede implementar a su vez en una arquitectura de tres capas (3-*Tier*). A estas capas se las conoce como capa de presentación, capa de negocios y capa de datos, también se describen las tecnologías implicadas en cada una de ellas.

3.2.1. Tecnologías Web del lado del Cliente (Capa de Presentación).

La Capa de Presentación es la que se encarga de mostrar la información al usuario y se puede considerar como el nivel más alto de las tres capas. HTML[4], *JavaScript* y CSS[5] son las herramientas esenciales para desarrollar aplicaciones en la Capa de Presentación, a continuación se describe cada una de ellas.

[4] http://www.w3.org/html
[5] http://www.w3.org/Style/CSS/

3.2.1.1. HTML

HTML es la abreviación que se le da al Lenguaje de Marcado de Hipertexto (*Hipertex Mark-up Language*); es el lenguaje de base en que están escritas las páginas *Web*, y es donde se describe la estructura de la información de la página. El lenguaje HTML se describe completamente en forma de texto, usando elementos que son conocidos como etiquetas. Las etiquetas describen y dan estructura al contenido de la página *Web*. HTML es el lenguaje de marcación que los navegadores leen para renderizar las páginas *Web*. Un documento escrito en HTML solo contiene etiquetas HTML y texto plano.

3.2.1.2. Hojas de Estilo

CSS es la abreviación que se le da a *hojas de estilo en cascada* (Cascading Style Sheets). Una hoja de estilos sirve para definir como se deben mostrar los elementos HTML en cuanto a su apariencia se refiere, por ejemplo el color de la fuente y su tamaño. Fue desarrollado en 1997 por el WC3[6] (*World Wide Web Consortium*) para resolver el problema que se encontraron los desarrolladores a la hora de crear sitios *Web* de gran tamaño. Desarrollar sitios *Web* se convirtió en un proceso muy costoso y repetitivo, a la hora de formatear el contenido de las páginas *Web*, con fuentes y colores diferentes para cada una de las páginas del sitio. Desde entonces HTML se usa para estructurar el contenido y CSS para formatear el contenido que previamente se estructuro usando las etiquetas HTML.

Dar formato a un documento HTML mediante CSS puede hacerse creando una Hoja de Estilos embebida en el documento HTML, pero se tendrá una hoja de estilos para cada una de las páginas del sitio *Web*, modificar la hoja de estilos requiere hacerlo en todas las demás páginas *Web* donde se tengan que modificar los Estilos. Para solucionar este problema las hojas de estilos normalmente se guardan en un fichero aparte, con el formato de extensión .css, esto permite cambiar la apariencia de todas las páginas del sitio *Web* modificando un solo archivo.

[6] http://www.w3.org/

CSS ha ido evolucionando desde sus inicios y al igual que HTML, se ha hecho de forma iterativa, lo que quiere decir que la versión más actual es una modificación de la anterior y así sucesivamente con las versiones anteriores. La versión actual se le conoce como CS3, no está finalizada del todo y continúa con actualizaciones hasta estas fechas.

3.2.1.3. JavaScript

JavaScript está basado en el lenguaje de programación *ECMAScript*[7]. Es un lenguaje de programación interpretado y soporta el paradigma de programación orientada a objetos. Desde sus inicios fue implementado para ejecutar código del lado del cliente, con la finalidad de interactuar con el usuario, por ejemplo validar lo que el usuario escribe de forma para que la información que se envía al servidor sea la correcta. También se pueden realizar interacciones más complejas como por ejemplo controlar el navegador, abrir nuevas ventanas, comunicarse de forma asíncrona con el servidor o modificar el contenido que se muestra en la página *Web*. El código de *JavaScript* va embebido en la página *Web*, y se interpreta conforme la página *Web* se carga en el navegador o dependiendo de la interacción con el usuario. El código se puede ejecutar cuando el usuario arrastra el ratón sobre un elemento de la página *Web*, introduce texto con el teclado, envía un formulario o cierra la página. Todos los navegadores de páginas *Web* contienen un intérprete para *JavaScript*.

3.2.2. Tecnologías Web del lado del servidor

3.2.2.1. Tecnologías Web de la Capa de Aplicación

La *Capa de Aplicación* es la que se encarga del control de las aplicaciones y el proceso de datos y sirve como intermediario entre la capa de presentación y la capa de datos. En la capa de Aplicación es donde se encuentran todos los lenguajes y entornos de desarrollo de aplicaciones *Web* que sirven para procesar la información y pasarla a la siguiente capa, ya sea hacia una capa superior o inferior.

[7] http://www.ecma-international.org/ecma-262/5.1/

El principal propósito de las tecnologías *Web* del lado del servidor es actualizar y modificar contenido *Web* con el fin de crear páginas *Web* dinámicas. Los lenguajes más conocidos para desarrollar aplicaciones del lado del servidor son: PHP[8] y ASP.NET[9]. Cada uno de estos lenguajes ayuda en tareas específicas para el desarrollo de sitios *Web* del lado del servidor. Con la programación del lado servidor es fácil mantener páginas *Web* específicas para sitios *Web* que son de gran tamaño.

3.2.2.1.1. PHP

Es el lenguaje de programación más utilizado para crear páginas *Web* dinámicas, es un lenguaje de programación que se puede utilizar para todo tipo de tareas. PHP es un lenguaje de programación con el que se puede ejecutar aplicaciones en diferentes tipos de servidores *Web*. El código escrito en PHP es escrito junto con el código HTML y compilado en el servidor *Web*, devuelve una página HTML al navegador. PHP es el lenguaje utilizado con frecuencia para desarrollar el código de las funciones que se comunican con los SGBD (Sistemas Gestores de Bases de Datos) y así tener páginas *Web* con acceso a bases de datos.

3.2.2.1.2. ASP.NET

Las páginas de servidor activo ASP, se desarrollan con el entorno de programación de *Microsoft .Net*[10] y pueden ser programadas en cualquiera de los lenguajes disponibles en el entorno (*Visual Basic*, C#, etc.). Las páginas *Web* ASP desarrolladas en .Net solo pueden ser almacenadas y ejecutadas en un servidor *Web* que tenga instalado el software propietario de *Microsoft*. Las páginas ASP tienen la misma función que las páginas escritas en PHP, entre las cuales destaca el acceder a los SGBD.

[8] http://es.php.net/
[9] http://www.asp.net/
[10] http://www.microsoft.com/net

3.2.3. Tecnologías Web de la Capa de Datos

La *Capa de Datos* es donde se encuentran todas las tecnologías *Web* que tienen como función el almacenamiento y recuperación de la información, en esta capa la información se mantiene aislada de las aplicaciones de la Capa de Aplicación lo que permite que las demás capas mejoren en escalabilidad y eficacia, este tipo de tecnologías son conocidas como servidores de bases de datos.

Los servidores de bases de datos son los que se encargan del almacenamiento y recuperación de los datos que van a ser mostrados en las páginas *Web*, esto se hace mediante SGBD que guardan y recuperan la información en las bases de datos.

Una base de datos sirve para tener la información organizada y clasificada; está compuesta por campos, registros y tablas.

Un campo es una pieza de información individual como por ejemplo la edad de una persona. Un conjunto de campos crean un registro, los campos de un registro están relacionados y tienen información en común como por ejemplo el nombre, apellido, edad, sexo y estado civil de una persona, son campos que pertenecen a una entidad en este caso la entidad es una persona. Un conjunto de registros se les conoce como un archivo de Base de Datos.

Los SGBD, son los encargados de comunicarse con las páginas *Web*, con otros DBMS y con otros sistemas, con el fin de almacenar y recuperar datos. Los SGBD más populares en el entorno de páginas *Web* son: MyQSL[11], SQLite[12], *Microsoft SQL Server*[13] y *Oracle*[14].

3.3. Cloud Computing

La Computación en la Nube, mejor conocida en inglés como *Cloud Computing* es un nuevo paradigma donde tecnologías de la información y sus recursos son puestos a disposición en forma masiva y escalable a través de Internet a un grupo de usuarios

[11] http://www.mysql.com/
[12] http://www.sqlite.org/
[13] http://www.microsoft.com/en-us/sqlserver/default.aspx
[14] http://www.oracle.com/index.html

externos. Este nuevo modelo se basa en la arquitectura orientada a servicios donde todos los recursos computacionales de *software y hardware* como por ejemplo: redes, servidores, almacenamiento, aplicaciones y servicios, son suministrados con el mínimo esfuerzo de gestión y configuración por parte del proveedor. Además de ofrecerlos de una forma flexible, adaptable y bajo demanda a todos los usuarios; que dependiendo de sus necesidades pueden incrementar la capacidad y prestaciones de computo en el mismo instante que lo requieran, pagando solo los recursos que consumen sin necesidad de invertir en nueva infraestructura, licencias o costes de capacitación e implementación (Xu 2012; Wang 2011; Niu et al. 2013; Subashini & Kavitha 2011; Khan et al. 2013; Aceto et al. 2013; Mell & Grance 2011).

3.3.1. Principales Características del Cloud Computing

Las Principales Características del *Cloud Computing* (Mell & Grance 2011; Esayas 2012; Hua et al. 2013; Subashini & Kavitha 2011; Dinh & Lee 2011; Azeemi et al. 2013) se pueden resumir en las siguientes:

- Arquitectura basada en servicios: cuando lo que se ofrece en la nube se basa en el modelo: “pagar solo por lo que se usa”, tal y como se hace con los servicios de electricidad y agua. Los servicios en *Cloud Computing* son transparentes para el usuario.

- Autoservicio bajo demanda: el usuario puede administrar las capacidades y prestaciones de los recursos de cómputo conforme lo requieran sus necesidades sin tener que interactuar de forma humana con el proveedor. Por ejemplo, puede aumentar el tamaño de almacenamiento en la nube o disminuir el poder de procesamiento del servidor sin necesidad de que el proveedor tenga que intervenir.

- Acceso a cualquier hora y desde cualquier lugar: los recursos están disponibles a través de Internet mediante el uso de estándares que permiten el acceso a los clientes desde cualquier plataforma conocida, ya sea móvil o estacionaria y que tenga una conexión a *Internet*. En este caso tanto el proveedor del servicio como

el cliente tienen diferente ubicación geográfica, por lo que el cliente accede a los recursos de manera virtual a través de *Internet*.

- Recursos mancomunados: los recursos que ofrece el proveedor son usados de forma mancomunada por múltiples usuarios, los diferentes recursos ya sean físicos o virtuales son asignados y reasignados dinámicamente de acuerdo con la demanda y necesidades de los usuarios.

- Elasticidad: los recursos en la nube tienen un poder de respuesta rápido al realizar los cambios de las capacidades de cómputo que el usuario necesita y en muchos casos de forma automática.

- Medición del servicio: los recursos en la nube cuentan con un sistema de medición que permite controlarlos y optimizarlos automáticamente, de esta forma se puede monitorizar y controlar su uso. En este modelo de servicio el usuario paga solo por los servicios que ha consumido.

3.3.2. Modelos de Servicio

En el *Cloud Computing* todos los recursos son implementados como «servicios» y se clasifican de acuerdo con la forma en que se ofrecen. Actualmente se conocen tres modelos principales: Infraestructura como Servicio (IaaS), Plataforma como servicio (PaaS) y *Software* como servicio (SaaS) (Xu 2012; Harman et al. 2013; Park & Ryoo 2013; Mell & Grance 2011; Aceto et al. 2013; Subashini & Kavitha 2011; Fernando et al. 2013).

Existen otros modelos para proveer servicios que derivan de alguno de los anteriormente mencionados; estos son: Almacenamiento de Datos como Servicio (DaaS), Comunicaciones como Servicio (CaaS), Negocio Como Servicio (BaaS) y Seguridad como Servicio (SecaaS) (Khan et al. 2013). Realmente todos los servicios son ofrecidos en forma de aplicaciones de *software* en la nube y quienes los proveen han creado diferentes clasificaciones para poder describir mejor sus características (Park & Ryoo 2013). A todos los modelos de provisión de servicios en *Cloud Computing* se

les puede precisar en un solo modelo el cual se le conoce como: Todo como Servicio (XaaS)(Xu 2012). A continuación se explican los tres modelos más utilizados a fin de definir y diferenciar los servicios de *Cloud Computing.* En la figura 4 se muestran ejemplos de los servicios que los usuarios tienen disponibles para cada modelo.

3.3.2.1. *Infrastructure as a Service - IaaS*

Este modelo de servicios ofrece a los usuarios infraestructura de *hardware* como por ejemplo procesamiento, almacenamiento y redes. Los servicios son estandarizados para poder ser ofrecidos de una mejor manera, todo esto es posible mediante el uso de «máquinas virtuales» donde el usuario puede instalar sistemas operativos que está acostumbrado a utilizar para ejecutar aplicaciones. El perfil de usuario en este modelo de servicios es el de un administrador de sistemas.

3.3.2.2. Platform as a Service - PaaS

Este modelo de servicios se refiere a las posibilidades que se le dan al usuario para desarrollar, implementar, mantener y alojar aplicaciones en el ambiente de desarrollo que es soportado por la nube del proveedor, en este modelo el usuario no tiene acceso a la infraestructura como por ejemplo los servidores, sistemas operativos, etc. Solo tiene el control de las aplicaciones y la configuración del ambiente de alojamiento y desarrollo. El perfil del usuario en este modelo de servicios es el de un desarrollador de aplicaciones.

3.3.2.3. Software as a Service - SaaS

En este modelo de servicios las aplicaciones son almacenadas y ejecutadas remotamente en la nube del proveedor; las aplicaciones están disponibles bajo demanda para todos los clientes que tiene acceso a ellas a través de *Internet*. Las aplicaciones son accedidas a través de diferentes interfaces como pueden ser navegadores *Web*, dispositivos móviles o interfaces de programación. Las aplicaciones que se ofrecen al usuario final

se proveen como un servicio en lugar de ser un producto que se instala en el ordenador o dispositivo móvil del usuario.

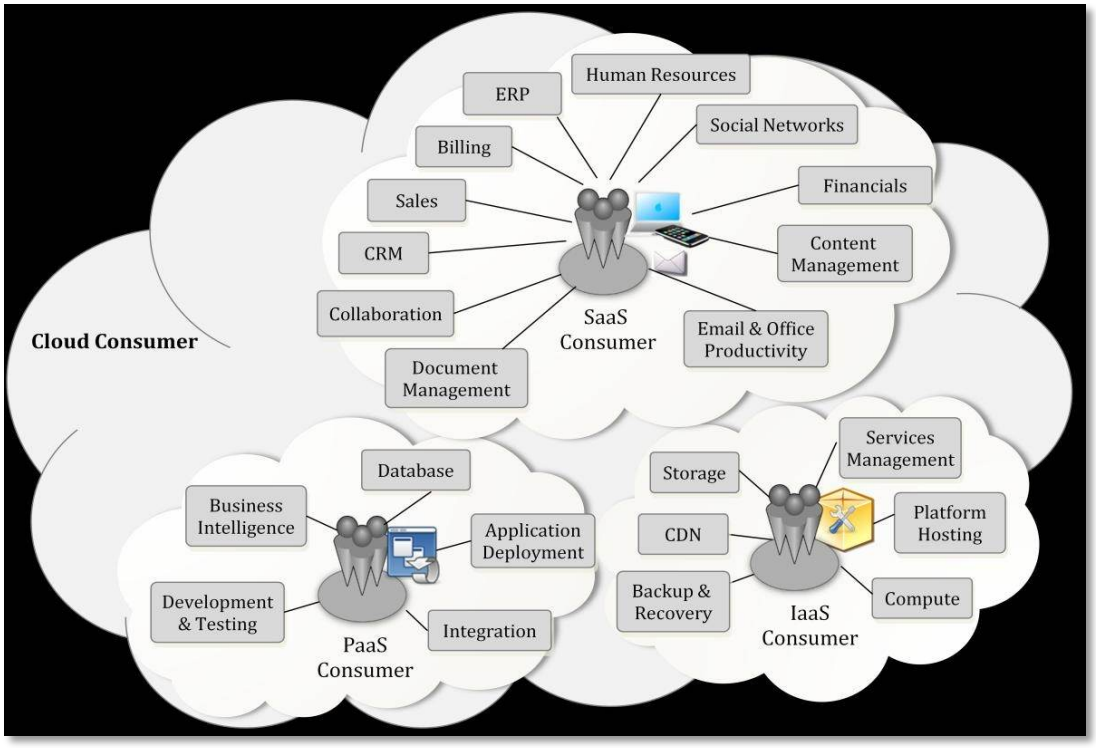

Figura 4: Modelos de Servicios en *Cloud Computing* y Servicios que ofrecen, fuente (Liu et al. 2011).

3.3.3. Modelos de Implementación del Cloud Computing

Los modelos en que es implementada la infraestructura de un sistema de *Cloud Computing*, pueden ser clasificados de acuerdo al nivel de exclusividad que el usuario desea tener en cuanto a los recursos que desea consumir. Estos modelos pueden ser: públicos, privados, comunitarios e híbridos.

En un servicio de *Cloud Computing* «público» todos los recursos están disponibles para cualquier usuario que tenga la disponibilidad de acceder a ellos a través de *Internet*. En un servicio de *Cloud Computing* «privado» toda la infraestructura y los recursos son operados exclusivamente para una organización y los usuarios que pertenecen a esta. En un servicio de *Cloud Computing* «comunitario», la infraestructura y los recursos son utilizados por un grupo específico de usuarios que comparten las mismas necesidades,

por ejemplo de seguridad, privacidad y almacenamiento. Un servicio de *Cloud Computing* «híbrido» es la combinación del uso de los tres modelos anteriores (Subashini & Kavitha 2011; Mell & Grance 2011; Esayas 2012; Aceto et al. 2013).

3.3.4. Arquitectura de Software de Cloud Computing

La Arquitectura de *Software* de *Cloud Computing* (ver figura 5), normalmente es representada como una Arquitectura de Capas (Harman et al. 2013).

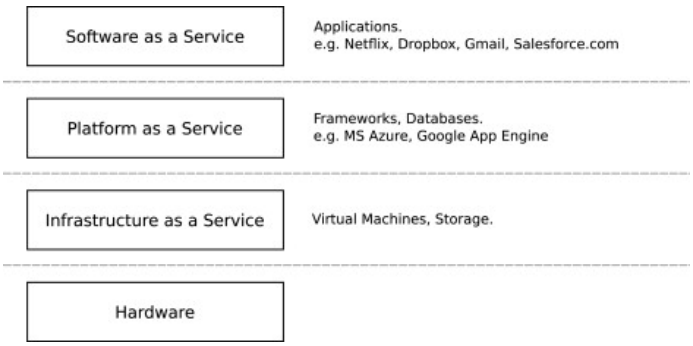

Figura 5: Arquitectura por Capas de *Cloud Coputing*, fuente (Harman et al. 2013).

La «Capa de *Hardware»* consiste en servidores físicos. El proveedor de servicios en la nube es responsable de la ejecución, mantenimiento, administración y actualización de los recursos de acuerdo con las necesidades de los usuarios. Los servidores están localizados en diferentes lugares geográficos, en centros de datos. Estos servidores ejecutan programas supervisores para la administración y provisión de recursos e incluyen sistemas administradores de máquinas virtuales que permiten a los usuarios crear máquinas virtuales remotamente en tiempo real (Harman et al. 2013; Khan et al. 2013).

La «Capa de Infraestructura» es la que permite al usuario hacer uso de máquinas virtuales, cada máquina virtual es una instancia de las diferentes configuraciones de máquinas virtuales que ofrece el proveedor en cuanto a memoria y poder de procesamiento disponible. Cada una de estas instancias monta una imagen del sistema operativo de la máquina virtual, el sistema operativo administra los recursos de

hardware y sirve de interfaz para la interacción de usuarios y aplicaciones con los recursos de *hardware*.

La Capa de Infraestructura también permite a los usuarios utilizar servidores de almacenamiento de información en forma masiva; la información puede ser almacenada, descargada y consultada desde cualquier lugar que tenga una conexión a internet, a este servicio se le conoce como DaaS. La Capa de Infraestructura también provee servicios de comunicación a los usuarios de forma rápida, segura y confiable, a este servicio se le conoce como CaaS (Harman et al. 2013; Khan et al. 2013).

La Capa de Aplicación provee plataformas de desarrollo de aplicaciones que incluyen interfaces de programación de aplicaciones (APIs) para que los desarrolladores puedan tener acceso total a los recursos y servicios de la nube. Los servicios de la plataforma de aplicación ejecutan las aplicaciones desarrolladas por los usuarios y automáticamente manejan la escalabilidad de recursos que la aplicación va consumiendo conforme van aumentado sus requerimientos (Harman et al. 2013; Khan et al. 2013).

La «Capa de Software» es la más utilizada y permite a los usuarios acceder y utilizar aplicaciones que están instaladas en los servidores de la nube. Los usuarios pueden acceder a estas aplicaciones desde cualquier dispositivo móvil o estacionario conectado a *Internet* sin importar sí sus capacidades de procesamiento y almacenamiento son limitados. Las aplicaciones ofrecen un acceso transparente al usuario sin necesidad de instalar actualizaciones o parches (Harman et al. 2013; Khan et al. 2013).

3.3.5. Mobile Cloud Computing

La Computación en la Nube Móvil (*Mobile Cloud Computing*) ofrece la posibilidad de utilizar infraestructuras, plataformas y software a un bajo coste y bajo demanda a través de dispositivos móviles. Las aplicaciones móviles pueden hacer uso de una gran variedad de servicios móviles que son provisionados a través de servicios de *Cloud Computing.* La integración de *Cloud Computing* en un ambiente móvil, para proveer a los usuarios móviles de servicios y facilidades de almacenamiento y procesamiento, fuera del dispositivo móvil, es lo que se conoce como *Mobile Cloud Computing* (Dinh & Lee 2011; Fernando et al. 2013).

Debido a las limitaciones de recursos que tienen los dispositivos móviles es complicado que las aplicaciones móviles puedan tener una intensiva actividad computacional o una capacidad de almacenamiento masivo en el mismo dispositivo (Khan et al. 2013). Por ejemplo, aplicaciones móviles basadas en localización y medios sociales hacen uso de varios sensores del dispositivo móvil, lo que conlleva a un alto nivel de uso de la batería, lo cual limita que el usuario pueda hacer uso de los sensores por un tiempo prolongado (Fernando et al. 2013).

Cloud Computing ayuda a mejorar la movilidad y portabilidad de la computación móvil extendiendo el tiempo de vida de la batería y mejorando la capacidad de almacenamiento y procesamiento. La computación móvil en la nube hereda las ventajas que ofrece el *Cloud Computing* como por ejemplo: escalabilidad, facilidad de integración, autoservicio bajo demanda y provisión de servicios mancomunados (Dinh & Lee 2011).

3.4. Tecnologías Móviles

Una Plataforma Móvil también se le conoce como «Sistema Operativo Móvil», un sistema operativo para móviles es el conjunto de funciones y programas que se encargan de administrar y optimizar los recursos de un dispositivo móvil como por ejemplo la conexión a *Internet*, los recursos multimedia, la cámara fotográfica y otros medios de entrada de datos. El sistema operativo es el que se encarga de ejecutar las aplicaciones que son desarrolladas por terceros para esta plataforma.

La mayoría de los sistemas operativos móviles que existen hoy en día son desarrollados para una arquitectura de *hardware* específica y con muy poca flexibilidad de ser adaptados a otros tipos y arquitecturas de *hardware*.

Un sistema operativo para móviles es muy similar a un sistema operativo estándar cómo por ejemplo *Linux*, *Windows* o *Mac*, con la diferencia de que los sistemas operativos para móviles son más pequeños y livianos en cuanto a programación se refiere ya que deben ejecutarse en dispositivos con recursos limitados como por ejemplo la memoria y capacidad de proceso.

Los sistemas operativos más conocidos actualmente son *iOS* de *Apple*[15], *Android*[16], *Windows Phone*[17] y *BlackBerry*[18] (Espada et al. 2013; Székely et al. 2013). Destacan dentro de estos *iOS* y *Android*, estos manejan casi el noventa por ciento de los dispositivos móviles inteligentes en el mundo (ver fig. 5).

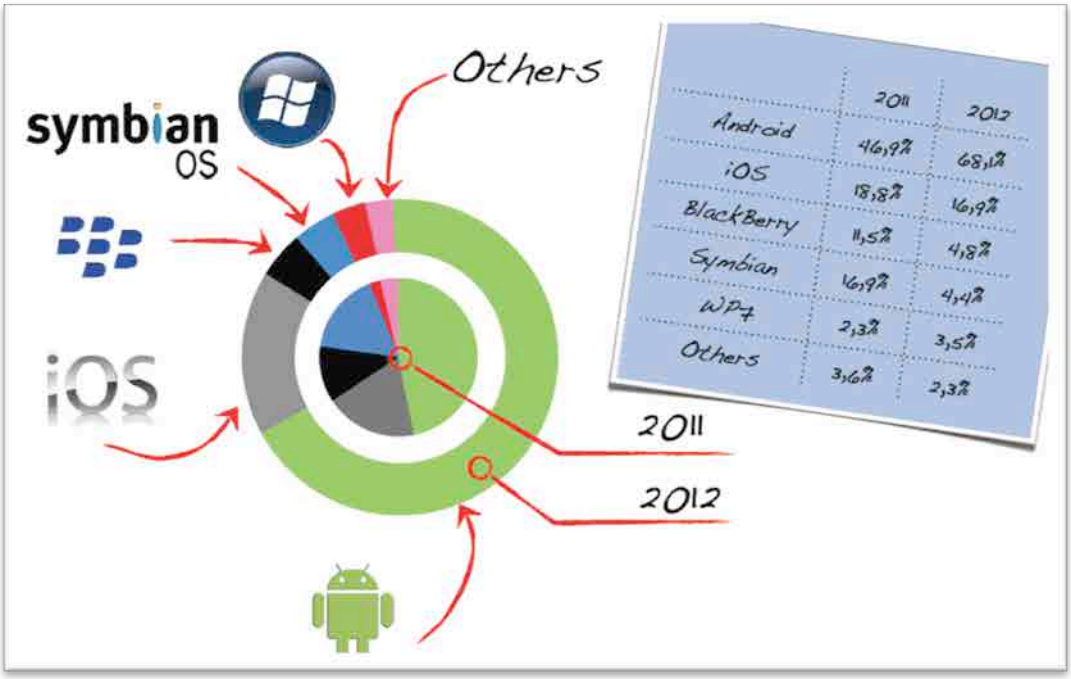

Figura 6: Plataformas Móviles, fuente (Friese 2012).

3.4.1. Plataforma iOS

iOS es el Sistema operativo desarrollado por la empresa *Apple*, *iOS* es esta basado en el Sistema operativo de escritorio *Mac OS X*[19] el cual a su vez está basado en una variante del Sistema operativo *Unix*[20] (Gavalas et al. 2011). *iOS* es un Sistema operativo con un ambiente cerrado (Székely et al. 2013). Apple no permite que terceros desarrollen extensiones para el Sistema operativo o se hagan modificaciones. La distribución de las

[15] https://developer.apple.com/
[16] http://developer.android.com/index.html
[17] http://dev.windowsphone.com/en-us
[18] http://developer.blackberry.com/
[19] https://developer.apple.com/technologies/mac/
[20] http://www.unix.org/

aplicaciones desarrolladas para *iOS* se hace mediante el *Apple Store* y los usuarios deben tener una cuenta de usuario registrado en la tienda para poder descargar las aplicaciones. El desarrollo de aplicaciones para *iOS* debe seguir y cumplir ciertos criterios impuestos por *Apple* para poder obtener el permiso para ser distribuidas en la tienda de *Apple*.

iOS al igual que todos los sistemas operativos modernos cuenta con una completa interfaz gráfica, completamente desarrollada para interactuar con el usuario mediante gestos manuales conocidos como *touch* (Fernández-López et al. 2013). En un Sistema operativo *touch* como es *iOS* las aplicaciones pueden ser abiertas con un simple toque con el dedo y el cambio de una pantalla a otra se hace simplemente deslizando el dedo para ir navegando por las diferentes pantallas.

El acceso a las opciones y configuración del sistema están restringidas y solo se puede acceder con la aplicación de configuración. Gestionar archivos y carpetas como en un sistema operativo de escritorio no es posible, todo esto se debe a que Apple desde sus inicios ha diseñado *iOS* como un sistema operativo sencillo y fácil de usar. *iOS* puede ser ejecutado en toda la familia de dispositivos móviles *iPhone*, *iPad* y *Apple tv*, dependiendo del modelo de dispositivo hay una versión del Sistema operativo que se puede ejecutar, la versión más actual de *iOS* es la versión 7 y se muestra en la figura 7.

El ambiente de desarrollo para aplicaciones en *iOS* que ofrece *Apple* se llama *Xcode*[21] y cuenta con herramientas tanto para desarrollo como para hacer pruebas de las aplicaciones programadas en un simulador. *Xcode* tiene integrado un compilador para los leguajes de programación C/C++ y *Objective-c*, este último es el lenguaje que ofrece la mayoría de las librerías con las se programan aplicaciones para *iOS*. El ambiente de desarrollo se descarga desde la tienda de *Apple* y solo puede ejecutarse en ordenadores con el Sistema operativo *Mac OS X*.

[21] https://developer.apple.com/xcode/

Figura 7: Sistema Operativo iOS versión 7.

3.4.2. Plataforma Android

El Sistema Operativo *Android* fue inicialmente desarrollado por la empresa *Android Inc*. y después comprado por *Google* en el 2005 y posteriormente liberado como software libre con licencia Apache. Es actualmente usado por una gran mayoría de fabricantes de dispositivos móviles.

Android se compone de una serie de funcionalidades de software para dispositivos móviles pero también destacan el sistema operativo, librerías y aplicaciones móviles, se pueden desarrollar aplicaciones por terceros. *Android* está basado en el *kernel* del sistema operativo *Unix*, *Android* es de código abierto, lo cual implica que el Sistema operativo puede ser modificado para cada dispositivo móvil en el que se quiera instalar. Ofrece una interfaz completamente gráfica basada en estilo *touch* y se muestra en la figura 7.

Android ofrece un entorno de desarrollo abierto llamado *Android SDK*[22] con el que los programadores de aplicaciones móviles pueden acceder directamente al hardware utilizando librerías diseñadas con este propósito (Chen et al. 2011). El lenguaje de

[22] http://developer.android.com/sdk/installing/studio.html

programación utilizado en este ambiente de desarrollo es *Java*[23]. *Android* utiliza una máquina virtual para ejecutar programas desarrollados en *Java* así como librerías estándar para SQLite[24], *Web*kit[25] y librerías multimedia (Lim et al. 2013). Las opciones más utilizadas para desarrollar en el ambiente de *Android* son *Eclipse*[26] y *Android Studio*, estos dos entornos de desarrollo se pueden instalar en diferentes plataformas y son completamente gratis.

La desventaja que tiene *Android* al ser de código abierto es la fragmentación, ya que hay muchos dispositivos de diversos fabricantes, ejecutando su propia versión de *Android*, lo que a la hora de desarrollar una aplicación, implica hacer pruebas en cada uno de los emuladores, que ofrecen los fabricantes de los dispositivos, asegurándose que el usuario, no tendrá problemas a la hora de ejecutar la aplicación y en caso de tenerlos hacer los cambios en la programación.

La distribución de las aplicaciones móviles programadas en *Android*, se hace en la tienda de aplicaciones *Google Play*[27], pero también se pueden distribuir utilizando tiendas alternativas de los propios fabricantes de dispositivos móviles.

[23] http://www.java.com/en/download/faq/develop.xml
[24] http://www.sqlite.org/
[25] http://www.webkit.org/
[26] http://www.eclipse.org/
[27] https://play.google.com/

Figura 8: Sistema Operativo *Android*.

3.4.3. Plataforma *Windows Phone*

La Plataforma *Windows Phone* está desarrollada por *Microsoft* y liberada en 2010 para sustituir la *Plataforma Windows Mobile*, cabe señalar que estas dos plataformas son incompatibles, por lo que las aplicaciones desarrolladas en *Windows Mobile* no pueden ejecutarse en *Windows Phone*. La interfaz llamada Metro, es la característica más novedosa ya que ofrece una pantalla de inicio con una serie de mosaicos, que muestran información ligada a aplicaciones y sensores del dispositivo móvil que se actualizan en tiempo real, esta interfaz se muestra en la figura 8.

La última versión del sistema operativo es *Windows Phone* 8 y está basado, en el sistema operativo de escritorio *Windows 8*[28]. *Microsoft* ofrece un ambiente de desarrollo llamado *Visual Studio Express*[29] completamente gratis y se pueden programar

[28] http://windows.microsoft.com/en-us/windows-8/meet
[29] http://www.microsoft.com/visualstudio/eng/products/visual-studio-express-products

aplicaciones en los lenguajes *C/C++, C# y VB.NET*. Este entorno de desarrollo solo se puede ejecutar en un ordenador con *Windows 8* versión profesional.

Las aplicaciones desarrolladas por terceros para *Windows Phone* son distribuidas en la tienda de *Microsoft* llamada *Marketplace*[30]. *Nokia* el principal fabricante de dispositivos móviles para *Windows Phone* tiene también una tienda donde se pueden distribuir las aplicaciones, pero en un futuro no muy lejano puede cambiar debido a la compra de la empresa *Nokia* por parte de *Microsoft*.

Figura 9: Sistema Operativo *Windows Phone* 8.

3.4.4. Aplicaciones Móviles

Una «Aplicación Móvil» es un tipo de software que está diseñado para ejecutarse en un dispositivo móvil (Lim et al. 2013). Hay diferentes tipos de aplicaciones móviles desde el punto de vista del desarrollo de software, pero en general todas son desarrolladas con el mismo propósito, ofrecer servicios al usuario final del dispositivo móvil, los servicios más comunes que ofrecen las aplicaciones móviles son de geolocalización, multimedia, Internet y servicios que las aplicaciones de escritorio han ofrecido durante muchos años.

[30] http://www.windowsphone.com/en-us/store

Actualmente la forma de desarrollar una aplicación móvil depende de muchos factores de hardware y de software a tomar en cuenta y como resultado existe una clasificación de aplicaciones móviles (ver tabla 2), a continuación se explican cada una de estas aplicaciones.

	Applications			
Language(s)	Native	Web	Client-Side Web	Hybrid
various **	✔			
Java / Ruby / HTML		✔		
JavaScript / HTML			✔	✔
Platforms				
Android	✔	✔	✔	✔
iOS	✔	✔	✔	✔
WP8	✔	✔	✔	✔
Platform access	✔	✘ *	✘ *	✔

** Java, Objective-C, C# *Limited acces

Tabla 2: Clasificación de Aplicaciones Móviles y las Plataformas en que pueden ser ejecutadas, fuente (Friese 2012).

3.4.4.1. Aplicaciones Nativas

Una Aplicación Móvil Nativa es aquella que es desarrollada con un lenguaje de programación para una plataforma y hardware móvil especifico. Una aplicación móvil nativa utiliza las librerías propias de la plataforma para acceder a las características y funciones del dispositivo móvil como pueden ser el GPS, la conexión a Internet, el acelerómetro, etc. Desarrollar una aplicación nativa asegura el acceso a todas las características del dispositivo móvil, ofreciendo de esta manera un mejor desempeño y confiabilidad (Mikkonen & Taivalsaari 2013). Las aplicaciones nativas pueden muchas veces trabajar sin conexión a Internet.

La principal desventaja de una aplicación nativa es que es específica para una plataforma, por lo que si se quiere ejecutar la misma aplicación en otra plataforma se tiene que programar completamente desde cero utilizando el lenguaje de programación específico para la plataforma que se quiera programar ya que son completamente incompatibles unas con otras (Mikkonen & Taivalsaari 2013). Una aplicación nativa debe ser instalada en el dispositivo móvil, lo que implica que si hay una nueva actualización de la aplicación esta debe ser reinstalada nuevamente o actualizada lo que supone una cierta probabilidad de que la aplicación pierda temporalmente ciertas funcionalidades.

3.4.4.1.1. Arquitectura de una Aplicación Móvil Nativa

Una aplicación móvil en la mayoría de los casos debe acceder a datos, servicios y recursos que solo están disponibles en la *Web*, que pueden ser tanto públicos como privados. Las aplicaciones móviles nativas usan las mismas técnicas y protocolos que un navegador para acceder a los servicios disponibles en la *Web* y se están convirtiendo en la manera más usual de consumir datos y servicios (Mikkonen & Taivalsaari 2013).

En la figura 10 se muestra la arquitectura de una aplicación móvil nativa y se puede observar la forma en que la aplicación tiene acceso y comunicación con un sistema *Backend* para poder acceder a datos y servicios.

La aplicación móvil se encarga de acceder a los recursos del dispositivo móvil, presentar la interfaz gráfica de la aplicación y ejecutar la lógica de programación para que el usuario interactúe con la interfaz. El sistema *Backend* se encarga de la lógica más compleja y el procesamiento de datos para poder ofrecer los servicios a la aplicación móvil. Muchas veces el sistema *Backend* puede hacer uso de servicios de terceros para poder cumplir con las peticiones de la aplicación móvil.

Como podemos observar los sistemas *Backend* son los mismos servidores *Web* que se suelen utilizar para ofrecer servicios a otros sistemas y aplicaciones, muchas veces el mismo sistema *Backend* está diseñado para atender a cualquier tipo de aplicación que requiera de sus servicios sin importar si es una aplicación móvil o de escritorio.

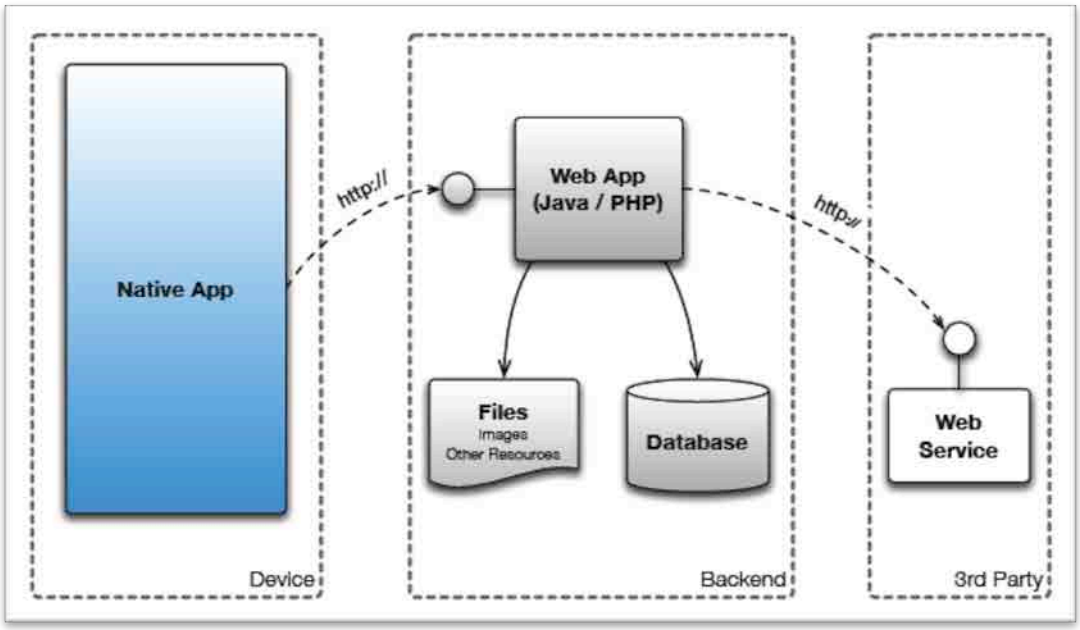

Figura 10: Arquitectura de una Aplicación Móvil Nativa, fuente (Friese 2012).

3.4.4.2. Aplicaciones Móviles Multiplataforma

Una aplicación móvil multiplataforma tiene como principal característica el poder ejecutarse en diferentes plataformas móviles, la aplicación móvil multiplataforma más universal es el navegador *Web*, pero existen otros tipos de aplicaciones multiplataforma. Las aplicaciones multiplataforma normalmente tienen ciertas desventajas con respecto a una aplicación nativa entre ellas el acceso restringido a los recursos de hardware del dispositivo móvil como por ejemplo el giroscopio, acelerómetro, etc.

Otra desventaja es que la interfaz no tiene el mismo aspecto que una aplicación nativa. Estas desventajas se deben a que las tecnologías con las que se desarrollan las aplicaciones multiplataforma no tienen el mismo grado de accesibilidad que las tecnologías de desarrollo nativas, las cuales tienen acceso en su totalidad a las funciones de hardware y de software de la plataforma móvil. A continuación se explican cada una de las aplicaciones multiplataforma que existen.

3.4.4.2.1. Aplicaciones Web Móviles

Una Aplicación *Web* Móvil está basada en tecnologías que son abiertas, accesibles y con un alto grado de interoperabilidad (Mikkonen & Taivalsaari 2013), principalmente la tecnologías utilizadas para su desarrollo son HTML5, CSS y *Java Script* (Espada et al. 2012). Todas estas tecnologías pueden ser ejecutadas en cualquier navegador móvil independientemente de la plataforma en la que se esté ejecutando.

Una aplicación *Web* móvil en la mayoría de los casos es la adaptación de una página *Web* de escritorio en una versión móvil. Las aplicaciones *Web* muchas veces tratan de imitar la interfaz de una aplicación nativa para que el usuario tenga una mejor experiencia de uso. Una de las principales ventajas de una aplicación *Web* móvil es que está disponible para cualquier usuario sin necesidad de tener que acceder a una tienda y comprar la aplicación, no requiere instalaciones manuales o actualizaciones.

Otra de las ventajas es que una aplicación *Web* móvil es fácil de programar ya que se utilizan tecnologías que llevan muchos años en el mercado. El costo de desarrollar este tipo de aplicaciones es muy bajo porque cualquier navegador *Web* móvil puede ejecutar una aplicación *Web* desarrollada en HTML.

3.4.4.2.2. Arquitectura de una Aplicación Web Móvil

3.4.4.2.2.1. Aplicación Web Móvil

Hay dos formas de desarrollar una aplicación *Web* móvil; la primera es ejecutando una aplicación HTML en el navegador del dispositivo móvil, el navegador *Web* del dispositivo móvil solo se encarga de presentar la interfaz del usuario y dejando toda la lógica, procesamiento de datos y creación de la interfaz del usuario al sistema *Backend*, el sistema *Backend* es el que se encarga de la administración de recursos como pueden ser imágenes, vídeos etc. y del almacenamiento de datos. El sistema *Backend* se encarga en su totalidad de proporcionar todos los servicios a la aplicación móvil que se ejecuta en el navegador y si es necesario accediendo a servicios de terceros para cumplir con su tarea. Se debe poner mucho esfuerzo para crear la interfaz con tecnologías como HTML y CSS ya que de no hacerlo la experiencia del usuario es muy pobre, el uso de recursos multimedia y gráficos es limitado en una aplicación *Web* móvil pero con el tiempo los navegadores *Web* para móviles van mejorando en cuanto a redimiendo para manejar

recursos multimedia, HTML5 cada vez tiene más librerías para acceder a este tipo de recursos. En la figura 10 se muestra la arquitectura de una aplicación *Web* móvil.

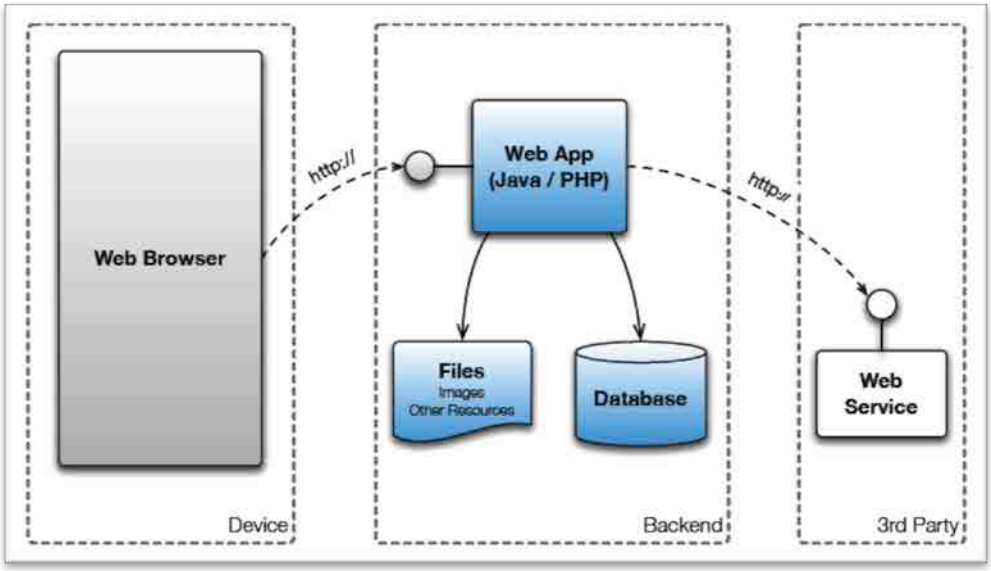

Figura 11: Arquitectura de una Aplicación *Web* Móvil, fuente (Friese 2012).

3.4.4.2.2.2. ***Aplicación Web Móvil del lado del Cliente***

Otra forma de desarrollar una aplicación *Web* Móvil es con la arquitectura que se muestra en la figura 11. Una aplicación *Web* del lado del cliente ejecuta una aplicación HTML en el navegador del dispositivo móvil, la aplicación *Web* utiliza librerías escritas en *JavaScript* que se ejecutan en el navegador del dispositivo móvil. Estas librerías sirven para crear la interfaz gráfica de la aplicación, acceder y almacenar datos en el dispositivo móvil de forma temporal, tener acceso a recursos de la plataforma móvil como pueden ser acceso a las fotos del usuario y utilizar el GPS.

El sistema *Backend* se encarga de dar los servicios que la aplicación móvil necesita y también se encarga del procesamiento de datos y el acceso a servicios de terceros cuando la aplicación móvil lo requiera.

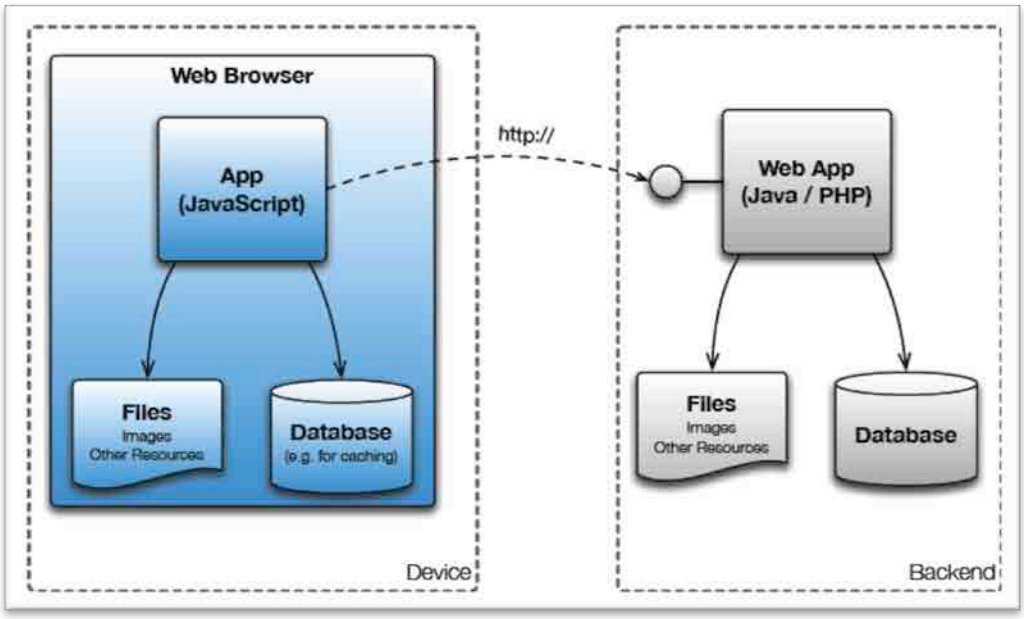

Figura 12: Arquitectura de una Aplicación *Web* Móvil del lado del Cliente, fuente (Friese 2012).

En la actualidad hay librerías desarrolladas por terceros en *Java Script* que están disponibles para desarrollar la aplicación móvil *Web* del lado del cliente. Las librerías que más se utilizan son *jQuery Mobile*[31] y *Sencha Touch*[32]. Estas librerías incluyen funciones que permiten crear la interfaz gráfica como si fuera la de una aplicación móvil nativa, esta funciones permiten al usuario resolver la mayoría de los problemas de compatibilidad, de los diferentes navegadores para móviles a la hora de crear interfaces gráficas, ya que las librerías están diseñadas para poder utilizarse en todos los navegadores móviles actuales.

La mayoría de estas librerías tratan de imitar el aspecto nativo de la plataforma móvil *iOS*. Estas librerías también ofrecen funciones para almacenar datos temporalmente y para acceder a recursos de la plataforma móvil en la que se está ejecutando la aplicación. La mayoría de los navegadores para móviles utilizan el motor de navegación llamado *WebKit*. Este motor de código abierto se encarga de renderizar y mostrar las páginas *Web* en el navegador móvil, las librerías permite que la interfaz tenga el mismo aspecto en casi todos los navegadores sin importar la plataforma móvil en la que se esté ejecutando la Aplicación *Web* Móvil. En la figura 12 se muestra la

[31] http://jquerymobile.com/
[32] http://www.sencha.com/products/touch

interfaz de una aplicación desarrollada en jQuery y ejecutada en el navegador *Web* de *iOS*.

Figura 13: Interfaz de una aplicación desarrollada en *jQuery*.

3.4.4.3. Aplicaciones Móviles Híbridas

Las «Aplicaciones Móviles Híbridas» son las que utilizan tecnologías *Web* abiertas y librerías de desarrollo que contienen funciones específicas para una plataforma móvil en particular (Espada et al. 2013). Estas librerías ofrecen todas las capacidades para acceder a funcionalidades de la plataforma móvil que solo se pueden utilizar en aplicaciones nativas. Estas librerías funcionan como una extensión de las tecnologías *Web* para que puedan ser desarrolladas como aplicaciones móviles, y de esta forma tener funcionalidades de una aplicación nativa.

Este tipo de aplicaciones son desarrolladas bajo del principio de que solo es necesario desarrollar la aplicación una sola vez y poder portarla a diferentes plataformas móviles (Corral et al. 2012), todo esto gracias a que las tecnologías *Web* y herramientas que se utilizan, son conocidas y utilizadas por todos las plataformas móviles, y solo se tienen que agregar las librerías para acceder a las funcionalidades, de cada una de las

plataformas en las que se quiera utilizar la aplicación. Las herramientas más utilizadas para este tipo de aplicaciones son: la herramienta de software libre llamada *PhoneGap*[33] y la herramienta propietaria de *Adobe* llamada *FlashBuilder*[34].

3.4.4.3.1. Arquitectura de una Aplicación Móvil Híbrida

En la figura 13 se muestra la Arquitectura de una Aplicación Móvil Híbrida donde se puede observar que en el dispositivo móvil la aplicación se ejecuta como si fuera una aplicación nativa, pero realmente la parte lógica de la aplicación, la que se encarga de interactuar con el usuario, está desarrollada con *HTML, CSS y Java Script*, la parte que se encarga de acceder a las funcionalidades de la plataforma está desarrollada en código nativo de la plataforma, una serie de librerías especiales hacen de puente para que las tecnologías *Web* puedan utilizar las funcionalidades de la plataforma. El sistema *Backend* se encarga de ofrecer todos los servicios que la aplicación móvil necesita ya sea por parte de las tecnologías *Web* o por la parte nativa de la aplicación móvil.

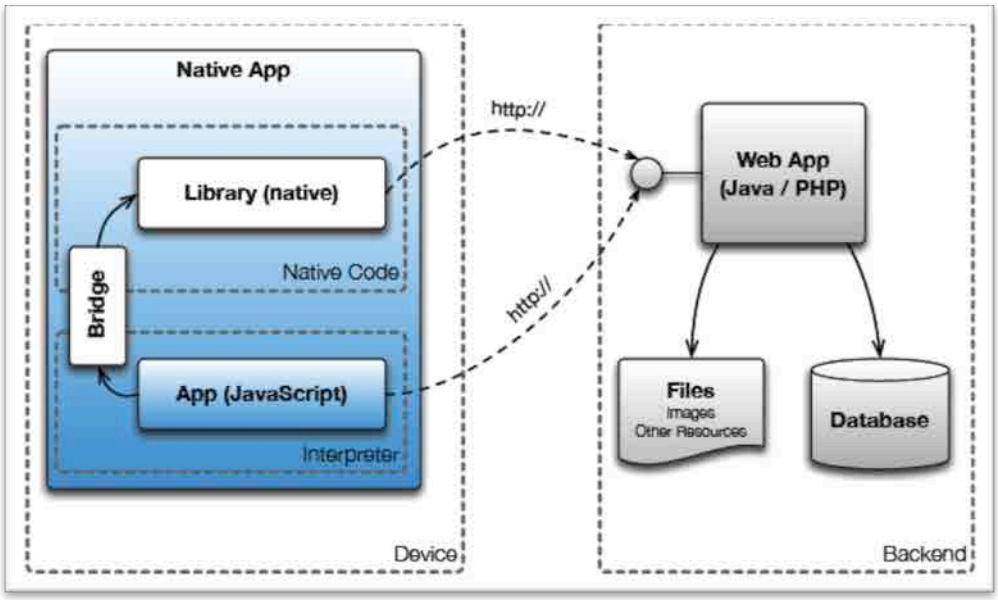

Figura 14: Arquitectura de una Aplicación Móvil Híbrida, fuente (Friese 2012).

[33] http://www.sencha.com/products/touch
[34] http://www.adobe.com/es/products/flash-builder.html

3.4.5. Ecosistemas de Software

Manikas & Hansen (2013) definen un Ecosistema de Software como: “la interacción de un conjunto de actores en la parte superior de una plataforma tecnológica común, que da como resultado en una serie de soluciones de software o servicios, cada actor es motivado por un conjunto de intereses o modelos de negocio y conectado con el resto de los actores y el ecosistema en su conjunto, la plataforma tecnológica es estructurada de una manera que permite la participación y la contribución de los diferentes actores”. Los actores involucrados en un ecosistema de software son propios de la plataforma y externos a ella (Bosch 2009, Molder et al. 2011). Todos estos actores involucrados operan intercambiando recursos, información, servicios y artefactos (Jansen et al. 2009).

3.4.5.1. Ecosistemas Móviles

Un «Ecosistema Móvil» es la descripción de una plataforma móvil (*Android*, *iOS*, etc.), sus productos y servicios que dependen de esta plataforma. En un Ecosistema Móvil también está incluido el Ecosistema de Software de la plataforma móvil (Hyrynsalmi 2012). En un Ecosistema Móvil (ver fig. 15) los actores involucrados van desde fabricantes de dispositivos móviles, operadores de telefonía móvil, proveedores de infraestructura, contenidos, software y servicios, desarrolladores de aplicaciones y actores que a al mismo tiempo son consumidores o proveedores de los servicios de otros actores. En un Ecosistema Móvil siempre hay actores emergentes y actores nuevos debido al constante cambio tecnológico y de software que tienen la plataformas móviles (Basole 2008). Los Ecosistemas Móviles pueden ser diferentes para cada plataforma, por ejemplo *iCloud* es un actor exclusivo de la plataforma *iOS* (ver fig. 16).

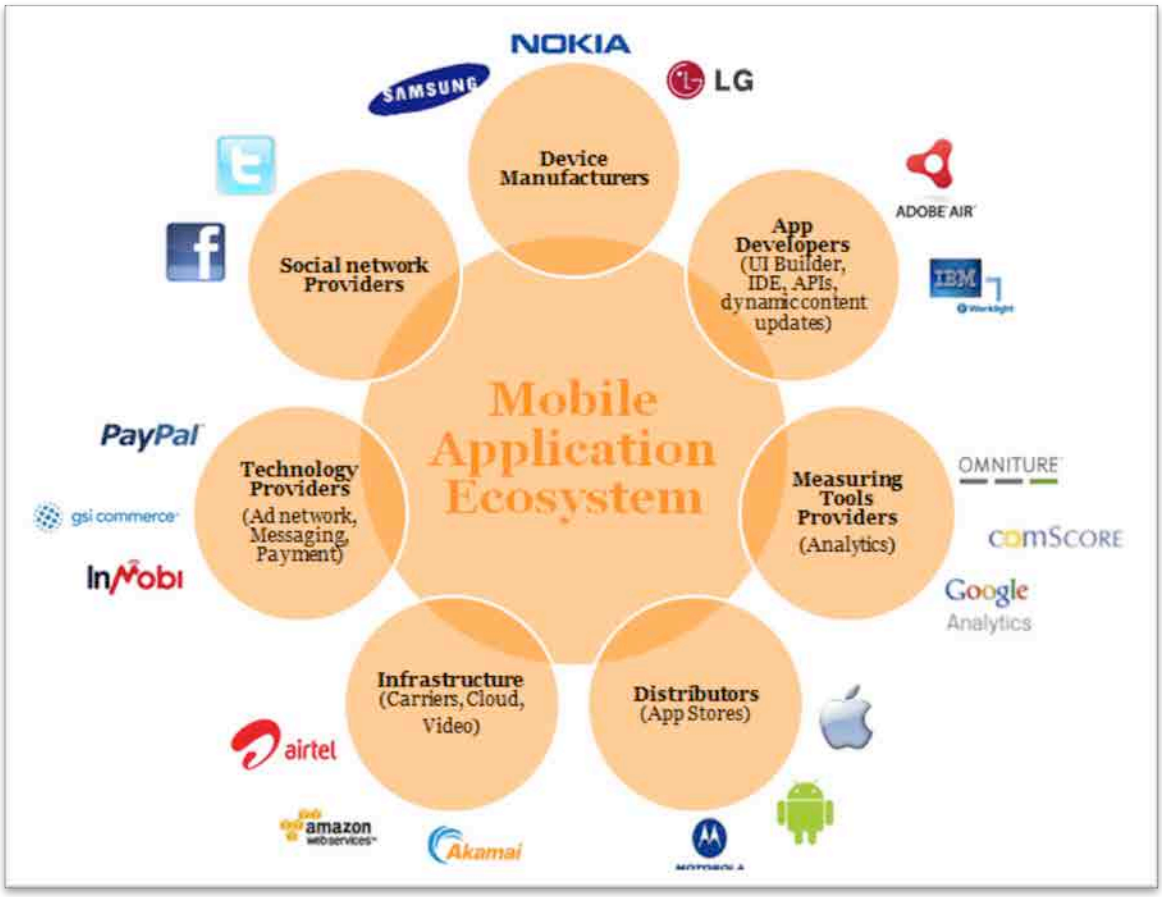

Figura 15: Actores de un Ecosistema Móvil, fuente (Kavyanidhi 2012).

3.4.6. Backend as a Service (BaaS)

Es una implementación de *Cloud Computing* que facilita a los desarrolladores de aplicaciones móviles la configuración, uso y operación de recursos y servicios *Backend* para aplicaciones móviles. BaaS provee a las aplicaciones móviles procesamiento y almacenamiento en la nube, además de las funcionalidades más comunes como son el manejo y autenticación de usuarios, integración con redes sociales, servicios de notificaciones, información contextual con respecto a la localización del usuario y acceso a fuentes de datos de terceros como por ejemplo *Oracle*. En la figura 15 se muestran los principales BaaS que existen actualmente en el mercado y los servicios que ofrecen.

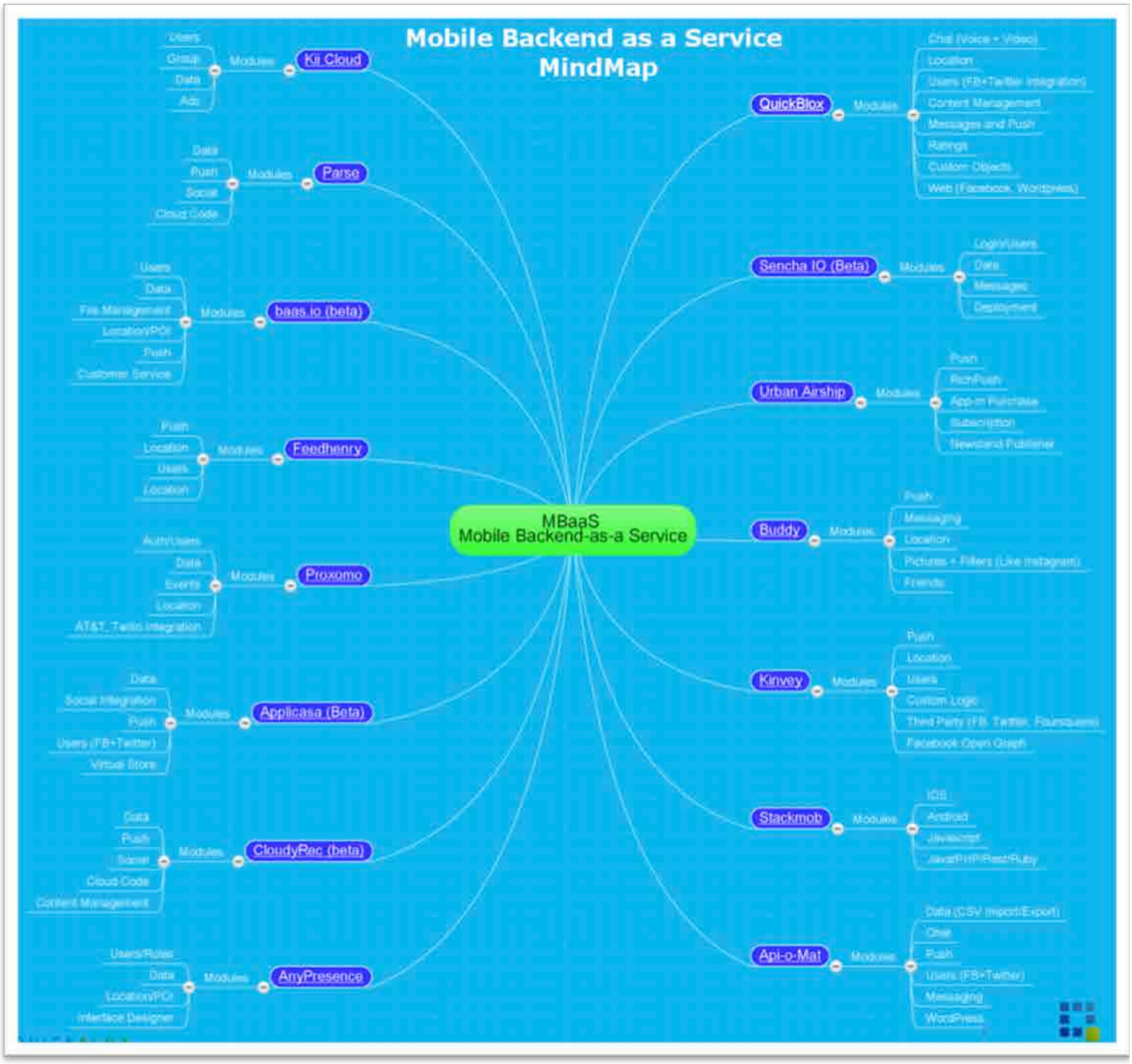

Figura 16: Proveedores *Mobile Backend as a Service* que existen actualmente en el mercado y las funcionalidades que ofrecen, fuente (Flautero 2012).

Los servicios ofrecidos por un BaaS son accesados a través de *kits* de desarrollo personalizados (SDKs) para una plataforma móvil en concreto y librerías de programación de aplicaciones (APIs). BaaS es una abstracción de las funcionalidades y

recursos de IaaS[35] y PaaS que una aplicación móvil necesita para ser desarrollada y elimina todas las dificultades de tener que administrar y ejecutar una infraestructura propia. BaaS tiene el mismo modelo de servicios en la nube que un PaaS, optimizándose y escalándose automáticamente cuando es necesario. En la figura 16 se muestra el Ecosistema Móvil Global incluyendo los BaaS para móviles.

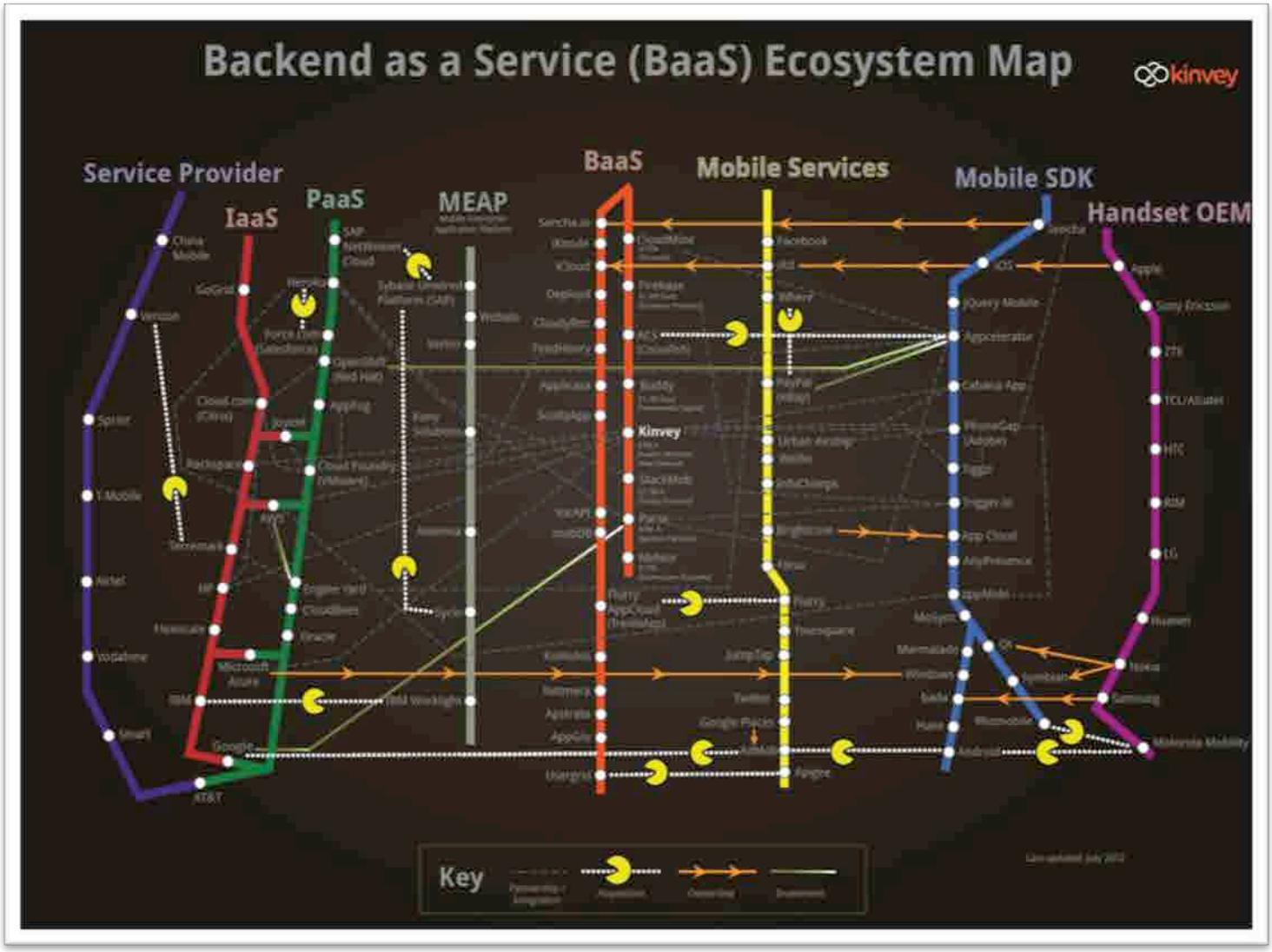

Figura 17: Ecosistema Móvil Global, fuente (Sridhar 2012).

3.4.7. Servicios Móviles

Los Servicios Móviles son recursos disponibles en la nube que son ofrecidos por terceros ya sea de forma privada o pública. Los servicios móviles pueden ser

[35] Modelos de implementación de *Cloud Computing* para ofrecer la Infraestructura como Servicio y la Plataforma como Servicio.

categorizados de muchas formas debido a que tienen una amplia gama de características dependiendo del tipo de servicio, algunos servicios tienen características de comunicación, otros tienen la característica principal de proveer información o entretenimiento, algunos son orientados a grupos de usuarios otros están orientados a usuarios individuales, algunos servicios son proveedores de información mientras otros necesitan obtener información por parte del usuario o por los censores del dispositivo móvil y algunos son una combinación de todas estas características (Bouwman et al. 2012).

3.4.7.1. Taxonomía de Servicios Móviles

Henten et al. (2009) propone una Taxonomía de Servicios Móviles tomando en cuenta dos dimensiones (ver figura 17).

La primera dimensión toma en cuenta dos tipos de servicios móviles: servicios adaptados para dispositivos móviles como por ejemplo los servicios de e-mail, banca móvil, y los servicios desarrollados específicamente para su uso en dispositivos móviles como pueden ser los servicios de localización.

La segunda dimensión toma en cuenta el tipo de contenido que provee el servicio: contenido que es el resultado del procesamiento de información como por ejemplo datos financieros y por otro lado contenido creativo que tiene un valor cultural o de entretenimiento como por ejemplo juegos para móviles.

Tomando en cuenta estas dos dimensiones se pueden agrupar los servicios móviles en cuatro categorías:

1. Servicios con contenido adaptado para móviles, en esta categoría entran los servicios de búsqueda, e-mail.

2. Servicios con contenido que es reutilizado para dispositivos móviles, como por ejemplo televisión móvil, música, juegos.

3. Servicios para la funcionalidad del dispositivo móvil, como por ejemplo localización, realidad aumentada.

4. Servicios móviles *Web* 2.0, como por ejemplo servicios de redes sociales.

Shao (2008) señala que los servicios móviles desde la perspectiva de contenidos generados por el usuario pueden clasificarse en servicios que le permiten al usuario consumir información o entretenimiento, servicios participativos que se enfocan en la interacción social y la formación de grupos sociales y por último servicios de producción que permiten crear contenidos al usuario mediante su capacidad de expresión (pasatiempo, artística, cultural, etc).

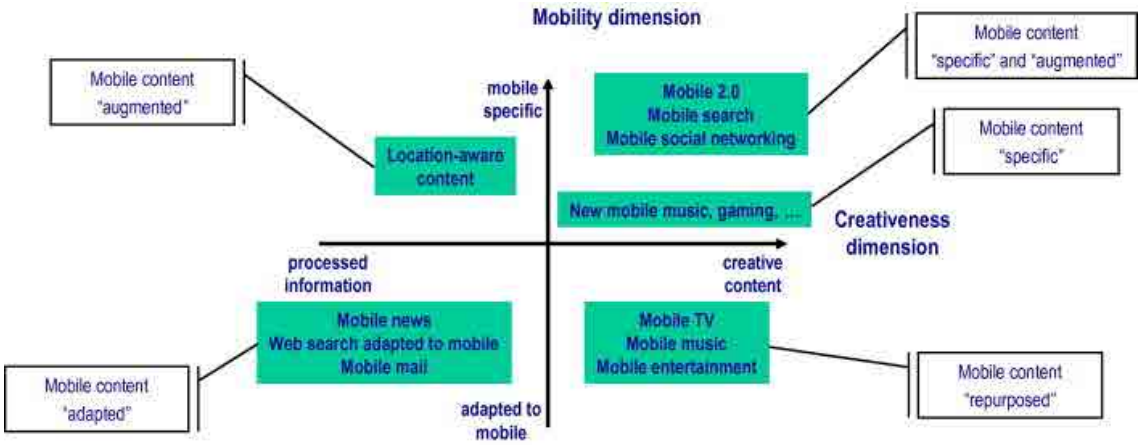

Figura 18: Dimensiones, Servicios y Contenidos Móviles (Henten et al. 2009).

3.4.7.2. Servicios Móviles, el Contexto y la Información Contextual

De acuerdo con Bouwman et al. (2012) la «Movilidad» es un concepto que debe ser entendido correctamente y en sobretodo para poder explicar lo que es el Contexto en Aplicaciones Móviles. La Movilidad puede darse en espacio y tiempo y está relacionada con personas, dispositivos, aplicaciones y objetos. Los Servicios Móviles ayudan a los usuarios a superar restricciones de espacio y tiempo (Pura & Heinonen 2008), permitiendo el acceso a la información de forma universal, portable y flexible, gracias a la libertad de espacio y tiempo que tienen los servicios móviles (Balasubramanian et al.

2002). Esto quiere decir que un servicio móvil siempre va a estar disponible independientemente del lugar dónde se encuentre el usuario y la hora del día en que quiera utilizar el servicio.

Dey et al. (2001) define Contexto como "cualquier información que es utilizada para describir las características de la situación en la que se encuentra una entidad (persona, objeto o lugar) y que es considerada relevante en la interacción entre el usuario y la aplicación". Por ejemplo muchos usuarios utilizan Información Contextual de geolocalización para hacer saber a sus contactos de redes sociales en qué lugar fue tomada la fotografía que han colgado en su muro.

Gummerus & Pihlström (2011) clasifican el Contexto en cuatro categorías: Contexto del Usuario, Computacional, Físico y de Tiempo. El Contexto del Usuario se refiere a su situación social, cultural y a su localización. El Contexto Computacional se refiere la infraestructura, tecnología y costos asociados de las tecnologías que hacen posibles los servicios móviles. El Contexto Físico se refiere a temperatura, humedad, ruido, luz, tráfico. El Contexto de Tiempo se refiere al día, mes y año.

Los Servicios Móviles que utilizan el Contexto como fuente principal de información se les conoce como "*Mobile Pervasive Services*" y se puede traducir como Servicios Móviles que son dispersados por todo el Contexto de la entidad de la que se quiere obtener Información Contextual. Este tipo de Servicios Móviles tiene la característica de ejecutarse a cualquier hora y lugar con la mínima atención del usuario (Yang et al. 2005), y deben tener la capacidad de adaptar dinámicamente su comportamiento de acuerdo con la Información Contextual que tenga a su disposición. Un ejemplo de este tipo de servicios es el de las aplicaciones móviles de recordatorios, se puede asociar un recordatorio con una geolocalización específica, como por ejemplo asociar el recordatorio de entregar el libro en la biblioteca con la geolocalización de la universidad, de esta forma la aplicación siempre va a estar pendiente de dónde se encuentra el usuario para que cuando su geolocalización coincida con la de la universidad, notificarle que tiene que entregar el libro en la biblioteca.

Una de las funciones principales de los servicios móviles que utilizan Información Contextual, es estar siempre pendiente del Contexto, esto quiere decir debe tener la

capacidad de detectar cambios en la Información Contextual y en concordancia con esto, adaptar el comportamiento de sus servicios (Achilleos et al. 2010).

3.4.7.3. Clasificación de la Información Contextual

En la actualidad los dispositivos móviles cuentan con sensores que permiten recolectar Información Contextual de forma manual o automática (Bae et al. 2013). Entender que información es reconocida como Contextual, como modelarla y utilizarla es importante ya que cuanta más Información Contextual pueda ser recolectada más exactos serán los resultados ofrecidos al usuario (Zhu et al. 2012). Debido a que los servicios móviles dependen en gran medida de la movilidad y por consecuencia del Contexto, en la mayoría de las veces el Contexto, va a definir el desempeño de los servicios móviles (Gummerus & Pihlström 2011).

3.4.7.3.1. Clasificación en base a como se percibe la Información Contextual

Bae et al. (2013) clasifica la Información Contextual en base a la forma en que la gente percibe como es generada la Información Contextual que recibe, y en base a como las personas perciben íntimamente el tipo de proveedor de información contextual.

Hay dos formas de percibir la Información Contextual:

- Subjetivamente donde la información se percibe de una manera interpretativa, el usuario interpreta que la información fue escrita manualmente por alguien, hay que tomar en cuenta que el contexto puede influenciar en como el usuario interpreta el ambiente que lo rodea.
- La forma Objetiva es percibida por el usuario como Información Contextual que fue detectada automáticamente por sensores, este tipo de Información Contextual tiene un bajo nivel de expresión ya que no es necesaria la intervención humana para obtenerla.

3.4.7.3.2. Clasificación Centrada en los Humanos

Zhu et al. (2012) hace una clasificación de la Información Contextual Centrada en los Humanos como entidad principal, esta clasificación toma en cuenta tres factores: humano, natural y cultural. Donde cada uno de los factores dependiendo de la situación tiene un peso específico. La clasificación está basada en tres categorías:

1. El contexto humano que trata del propio ser Humano y tiene dos subcategorías: El contexto fisiológico, por ejemplo temperatura corporal, presión arterial, pulso, son recolectados por simples instrumentos. El contexto mental, por ejemplo placer: alegría, enojo, tristeza, que refleja su estado de ánimo; los cuales son difíciles de capturar por sensores pero puede ser representado por expresiones faciales y gesticulaciones corporales o representadas por el usuario con iconos emocionales mejor conocidos como *emoticons*.

2. El contexto natural que se refiere al cuándo y al dónde tiene dos subcategorías: el contexto de tiempo, como por ejemplo: cuando un evento ha terminado o empezado a ejecutarse. El contexto de espacio, por ejemplo: humedad temperatura, localización o información de un lugar en particular.

3. El contexto cultural representa el grupo de atributos de la información de las relaciones humanas de la entidad y tiene dos subcategorías: El contexto de grupo, por ejemplo: el puesto que ocupa y la información de la organización a la que pertenece. El contexto social, por ejemplo: la nacionalidad, tradiciones y creencias.

Achilleos et al. (2010) y Bae et al.(2013) mencionan que Dey & Abowd (2000) clasifica la información contextual en cuatro categorías: Identidad, Tiempo, Localización y Actividad. De acuerdo con Nardi (1995) mediante la actividad del usuario se puede conocer información contextual referente a sus intenciones deseos y metas.

3.4.7.3.3. Clasificación de acuerdo al nivel de Validez y Calidad de la Información

Achilleos et al. (2010) señala que los servicios que utilizan información contextual la obtienen de diferentes fuentes como por ejemplo sensores, usuarios, repositorios y perfiles entre otros, por lo que se debe identificar y clasificar las fuentes de donde se obtiene la información contextual para así poder manejar y distinguir la información contextual de una mejor manera.

Achilleos et al. (2010) utiliza la clasificación propuesta por (Henricksen & Indulska 2006), donde las fuentes de información contextual están clasificadas en cuatro categorías: Estática, Perfiles, Detectada por sensores y Derivada.

Choi et al. (2012) utiliza la misma clasificación destacando la frecuencia con la que cambia el Contexto y la Información. Para saber que tan válido es el contexto que proveen estas fuentes, se puede determinar por la Persistencia de la Información contextual. La calidad del contexto también puede ser considerada en base a las fuentes de donde proviene la información contextual. A continuación explicamos cada una de estas fuentes de Información contextual, su nivel de *validez y calidad*:

- Estática: es información con un alto grado de persistencia y calidad ya que esta información nunca cambia y tomando en cuenta que la información que ha introducido el usuario es correcta, como ejemplo podemos citar la fecha de nacimiento del usuario y su lengua nativa.

- Perfiles: es información que es proporcionada por el usuario por lo que es información que cambia muy pocas veces y solo cuando el usuario la modifica, por lo tanto el contexto es confiable, su nivel persistencia y calidad es casi del mismo nivel que el de la información estática, la calidad puede verse afectada cuando el usuario olvida actualizar su perfil.

- Detectada por sensores: es información que es capturada por sensores, este tipo de información es altamente impredecible debido a que es información que cambia frecuentemente, es muy inestable debido a que se utilizan sensores que son inestables y pueden tener un funcionamiento erróneo o poca exactitud, por lo que la validez y calidad del contexto son muy inferiores al de las dos fuentes anteriormente descritas.

- Derivada: es información que esta derivada en base a otra fuente de información contextual como por ejemplo la temperatura ambiente de la localización donde se encuentra el usuario. Su persistencia es muy pobre ya que es información altamente impredecible y que cambia con frecuencia por lo cual el contexto tiene poca validez y su calidad depende de la calidad de otras fuentes que no se sabe si son confiables.

Achilleos et al. (2010) sugiere que otra forma de determinar la validez del contexto es utilizando restricciones temporales y es de mucha importancia porque de esta forma se puede designar cuándo la información contextual está caducada. Propone dos tipos de restricciones: la primera son restricciones comparativas en donde se establece un tiempo en el que expira la información contextual que se obtiene de la fuente, la segunda son restricciones temporales absolutas donde se designa un intervalo de tiempo entre el inicio y expiración de la información.

La privacidad del contexto también es importante y cada tipo de información contextual debe ser tratada de acuerdo con su nivel de sensibilidad, se deben dar diferentes permisos de acceso para cada fuente de Información contextual con el fin de proteger información que no debe ser compartida con otros usuarios.

3.4.8. REST APIs

Los Servicios Móviles son ofrecidos por arquitecturas *Backend*, la interfaz que conecta a las aplicaciones móviles con los sistemas *Backend* suelen ser servicios *Web* basados

en el protocolo http y sus principales funcionalidades son: Identificadores Uniformes de Recursos (URIs), tipos de medios (MIME), requerimientos (*request*) y respuestas (*response*). A este tipo de servicios *Web* se les conoce cómo REST (Arquitectura Orientada a Recursos).

Un recurso es cualquier información que puede ser referenciada por una URI como por ejemplo un documento, una imagen, datos de un usuario o un servicio (Arroqui et al. 2012). La URI que se muestra a continuación devuelve los datos de un usuario en la implementación REST de la API de *Facebook* "https://graph.facebook.com/nombre_usuario".

La interacción con los recursos y su manipulación se hace mediante el clásico patrón *request/response* haciendo uso de los métodos que el protocolo http proporciona (*GET, PUT, DELETE, y POST*) (Jamal & Deters 2011). Los recursos que son devueltos por el servicio en el *request* son representados en formato XML o JSON.

El mecanismo de autenticación y autorización que se utiliza para controlar el acceso a servicios específicos y recursos es el mismo que se utiliza en el protocolo http (Price et al. 2013). Para añadir seguridad se puede implementar el protocolo seguro https, es el único método de seguridad disponible para servicios REST (Bertram & Kleiner 2012). Cada *request* que se hace incluye toda la información necesaria para que el servidor pueda saber qué servicio debe proporcionar por lo que cada *request* es independiente y no tiene relación con otros *request* por lo que el servidor no necesita tener un registro del estado de los *request* y su relación entre ellos (Mohamed & Wijesekera 2012).

Los servicios *Web* basados en REST son la solución más utilizada por los servicios *Web* 2.0, *Twitter*, *Facebook* y *Flickr* (Price et al. 2013). Los servicios *Web* basados en REST son más eficientes y consumen menos recursos que los servicios tradicionales basados en SOAP lo que hace que su implementación y provisión se adapte a las necesidades de los dispositivos móviles que tienen recursos limitados (Mohamed & Wijesekera 2012). En los servicios *Web* tradicionales las interfaces usualmente no pueden ser portadas para proveer servicios a diferentes clientes, en cambio en los servicios REST cada uno de los clientes que tiene acceso a los recursos es tratado de la misma forma y el proveedor del servicio no toma en cuenta si el servicio está siendo utilizado por una aplicación móvil nativa, una aplicación *Web* móvil o una página *Web* convencional (Mikkonen & Taivalsaari 2013).

3.4.9. Identidad Federada de Usuarios en Servicios Móviles

Hoy en día las aplicaciones móviles y *Web*, hacen uso de una gran variedad de recursos en la nube, comparten y consumen información con otras aplicaciones que están protegidas del acceso a Clientes y Usuarios no autorizados, Identificarse de la forma tradicional con el modelo de autenticación Cliente-Servidor tiene muchas desventajas, el Cliente debe pedir permiso para acceder al recurso protegido autenticándose mediante el uso de las credenciales que le permitan acceder al recurso.

El propietario del recurso debe compartir sus credenciales con terceras aplicaciones si quiere dar permiso para que las terceras aplicaciones tengan acceso al recurso protegido. Las terceras aplicaciones almacenan las credenciales del propietario del recurso protegido para poder tener acceso en el futuro, si un tercero pone en peligro las credenciales, la clave de acceso y el recurso se ven en peligro de ser expuestos a un posible robo o acceso ilegal, si se revoca el acceso a la aplicación, al cliente que ha puesto en peligro las credenciales cambiando la clave de acceso, se está revocando el acceso a todos los demás clientes que tenían permiso de acceso al recurso.

El propietario no tiene la capacidad de limitar el acceso a ciertas partes o características del recurso y tampoco puede limitar la duración del acceso sin afectar a todos los clientes (Torroglosa-García et al. 2013).

La forma de solucionar estos problemas es implementando el inicio de sesión único (*Single Sing On* - SSO) mediante el uso de identidades federadas, de esta forma se elimina la necesidad de almacenar claves de acceso y correr el peligro de que sean robadas o que el Usuario tenga que recordar la clave de usuario y contraseña cada vez que tenga que a acceder a un recurso, los Usuarios no necesitan iniciar sesión para acender a recursos protegidos, su identidad y credenciales de acceso se proporcionan a terceros de una forma segura (Riesner et al. 2013).

Identidad Federada es el término que se le da a la manera de proveer una forma segura y amigable de compartir la identidad del usuario entre varias organizaciones y aplicaciones, además de una forma estandarizada. Utilizando identidades federadas estandarizadas los usuarios inician sesión solo una vez utilizando el método tradicional

de inicio de sesión utilizando su nombre de usuario y contraseña o mediante un servidor de servicios de autenticación.

Cuando el usuario quiere acceder la aplicación de un tercero la organización encargada de autenticar al usuario le proporciona a la aplicación de una forma segura y transparente la identidad del usuario, de esta manera las claves de acceso no son requeridas, la aplicación a la que el usuario quiere acceder siempre pude verificar la autenticidad de la identidad federada del usuario con la organización que autentica al usuario. La federación de identidades permite que los procesos de autenticación se den más allá de las fronteras organizacionales a las que pertenece el usuario (Riesner et al. 2013).

3.4.9.1. Estándar OAuth

El Estándar OAuth *(Open Authorization)* se ha convertido en la opción más utilizada para implementar el Inicio de Sesión Único mediante identidades federadas, hay otros estándares como por ejemplo *WS-TRUST* y *WS-FEDERATION* pero por su complejidad no han tenido tanto éxito. OAuth hace que los servicios basados en REST y http se hagan de una forma más segura.

OAuth es un protocolo de autorización seguro que sirve para integrar e interconectar REST APIs de servicios que son usados por las aplicaciones de redes sociales como *Facebook, Twitter, Dropbox, Google Drive* y muchas otras aplicaciones y servicios, el protocolo OAuth es usado para autorizar a Usuarios y terceros el acceso limitado a información y recursos protegidos sin compartir las claves de acceso del propietario, el acceso puede ser autorizado en nombre del propietario del recurso o permitiendo a terceros obtener el acceso por cuenta propia (Torroglosa-García et al. 2013; Noureddine & Bashroush 2013).

En lugar de usar las credenciales del propietario para acceder al recurso protegido, el cliente obtiene una estructura de datos llamada *token*, que contiene el alcance, tiempo que va a estar disponible y otros atributos de acceso al recurso. En el *token*, la información de autorización que da derecho de acceso al recurso se maneja por separado de la información de autenticación que contiene las credenciales de sesión (Torroglosa-García et al. 2013; Noureddine & Bashroush 2013).

EL protocolo *OAuth* define cuatro roles (Torroglosa-García et al. 2013):

- El propietario del recurso: es una entidad que tiene la capacidad de proveer acceso a un recurso protegido.

- El cliente: es una aplicación que hace peticiones para acceder a recursos Protegidos en nombre del propietario del recurso y obtener su autorización. La aplicación puede estar ejecutándose en un dispositivo móvil, desde un navegador *Web* o en un servidor.

- El servidor de autorización: es el que se encarga de autenticar al Propietario del Recurso y en caso de hacerlo satisfactoriamente, autorizar el acceso al Cliente expidiéndole un *token* de acceso, que a su vez es firmado digitalmente. El Servidor de Autorización es independiente del Servidor de Recursos por razones de seguridad y escalabilidad. Los Servidores de Autorización son respaldados por la organización *OAuth WRAP (OAuth Web Resource Authorization Profiles)* (Noureddine & Bashroush 2013).

- El servidor de recursos: Es el servidor que contiene el recurso protegido y tiene la funcionalidad de aceptar y responder peticiones de acceso a recursos protegidos mediante *tokens* de acceso. El servidor de recursos puede verificar si el *token* de acceso fue expedido por un servidor de autorización confiable usando la firma digital que trae el *token* (Noureddine & Bashroush 2013).

Torroglosa-García et al. (2013) y Noureddine & Bashroush (2013) explican la forma en la que se da la interacción desde que el cliente inicia la petición de acceso al recurso hasta que accede al recurso, se puede ver la figura 19 y se explica a continuación:

1. El cliente pide autorización de acceso al propietario del recurso protegido.

2. El propietario del recurso da su autorización y el cliente obtiene las credenciales que autorizan el uso del recurso protegido.

3. El cliente presenta al servidor de autorización sus credenciales para que pueda ser autenticado y pide que le sea expedido un *token*.

4. El servidor de autorizaciones valida las credenciales del cliente y el tipo de autorización que le ha sido otorgado y si el cliente es autenticado correctamente con sus credenciales al Servidor le expide un *token*.

5. El cliente hace la petición al servidor de recursos con la finalidad de acceder al recurso protegido autenticándose con el *token*.

6. El servidor de recursos valida el *token* de acceso y si este es válido el cliente puede tener acceso al recurso.

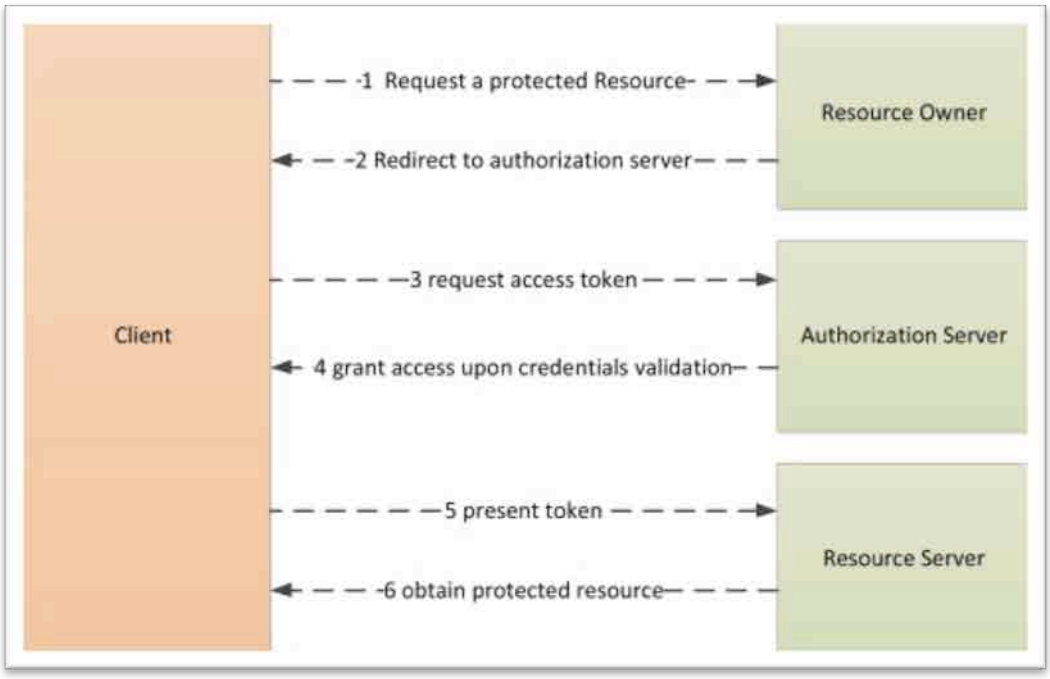

Figura 19: Interacción de los cuatro roles en el protocolo OAuth, fuente (Noureddine & Bashroush 2013).

Un ejemplo de autenticación de un usuario se puede ver en la figura 20, en este ejemplo el usuario, de una forma fácil y sencilla utiliza su identidad de usuario en *Facebook* para identificarse como usuario en aplicaciones de terceros. El usuario desea ejecutar el juego *High* 5 Casino utilizando su identidad de usuario en *Facebook* para no tener que

crear un nuevo usuario dentro del juego, la aplicación *High* 5 Casino le informa al usuario que va a pedir en su nombre a *Facebook* sus datos de identificación (paso 1 en la fig. 19), al presionar el botón *Play Gam*e el usuario da permiso a la aplicación y se ejecutan los pasos 2 al 6 de la figura 19. En este ejemplo el recurso protegido es la identidad del usuario.

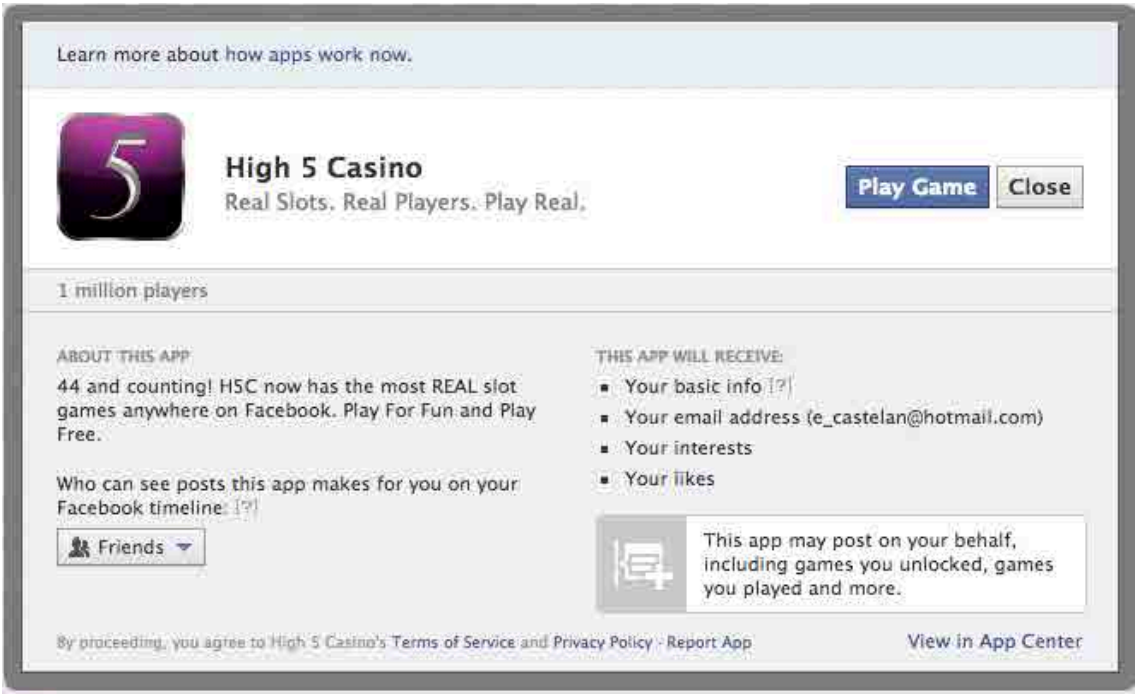

Figura 20: Aplicación *High* 5 Casino pidiendo permiso al usuario para utilizar sus datos de identificación de usuario de *Facebook*.

Un ejemplo de autenticación de una aplicación se puede ver en la figura 21, en este ejemplo la aplicación WOLF desea acceder a recursos protegidos de la cuenta de Google Drive del usuario, la aplicación WOLF es su rol de cliente pide permisos al usuario para poder acceder a estos recursos, si el usuario otorga el permiso al cliente, la aplicación WOLF se autentica con el servidor de autorización y presenta las credenciales con los permisos que le ha dado el usuario, si el servidor de autorización autentica que la aplicación WOLF es válida para acceder a los recursos protegidos le expide un *token* (ver fig. 22), la aplicación WOLF ahora puede hacer uso del *token* para

acceder a los recursos protegidos cuando lo requiera, siempre y cuando el *token* no esté caducado.

Figura 21: El Cliente (*WOLF*) pide permiso al Usuario de acceder a los recursos protegidos.

Figura 22: El Cliente (*WOLF*) recibe el *token* para acceder a los recursos protegidos.

3.4.10. Notificaciones en Dispositivos Móviles

Garzonis (2010) define notificación como parte de un sistema basado en eventos, donde el evento es algo que puede ser observado y notificado describiendo el evento. Notificar es el acto de enviar un mensaje a través de un mecanismo apropiado.

Los sistemas de notificación deben dar soporte para que los usuarios puedan acceder a información adicional de las actividades más actuales a través de fuentes de información secundarias. La información que se notifica al usuario debe ser relevante e importante y se debe entregar a los usuarios de una forma eficiente y efectiva sin distraer al usuario de la tarea que se encuentra realizando. Estos sistemas deben saber cómo manejar la atención del usuario para que este pueda estar realizar una tarea principal y al mismo tiempo poder recibir información adicional de una actividad o tarea secundaria.

McCrickard & Chewar (2003) menciona que se deben tener en cuenta tres parámetros críticos que deben ser manipulados y balanceados a la hora diseñar los requerimientos y metas particulares de un sistema de notificaciones, estos parámetros son:

- Interrupción: es causada por un evento que requiere por parte del usuario el cambio de atención parcial o total de la tarea que está realizando a la notificación. Esta interrupción puede variar en sutileza y persistencia con la finalidad de satisfacer con las necesidades de la tarea secundaria como por ejemplo su urgencia.

- Reacción: es la respuesta inmediata del usuario al evento que lo está interrumpiendo, esta reacción puede ser controlada manipulando la semántica del estímulo que presenta el sistema de notificaciones (sonidos, colores, etc.).

- Comprensión: se refiere a la habilidad de ayudar a los usuarios a entender las razones por las cuales la notificación ha sido invocada y de esta manera dar a los usuarios el soporte para que puedan utilizar esta información contextual para evaluar sus estrategias a largo plazo.

3.4.10.1. Métodos de entrega de información

La diseminación de información involucra la entrega de información a un grupo de consumidores desde una o más fuentes. En la tabla 3 se pueden ver los mecanismos de entrega que se pueden utilizar para la diseminación de información.

Pull		Solicita información cuando es requerida, cuando la solicitud es recibida en el servidor, el servidor localiza la información de interés y la envía al cliente.
Push		La entrega de datos tiene que ver con enviar información al cliente previo a cualquier petición.
Periódica		Es presentada de acuerdo a una agenda preestablecida.
Aperiódica		Es manejada por eventos.
Unicast		Los ítems de datos son enviados de una fuente de datos a otra máquina.
1-N	*Multicast*	La información se envía a un grupo específico de clientes conocidos.
	Broadcast	Se envía información sobre un medio para un grupo no identificado de clientes.

Tabla 3: Mecanismos de entrega de información, fuente (Sallam & Siba 2011)

3.4.10.2. Taxonomía de Notificaciones

Mühl et al. (2006) propone una Taxonomía de Notificaciones basándose en las seis preguntas (ver fig. 23) que se debe hacer para realizar una notificación. Esta taxonomía agrupa las seis preguntas en cuatro dimensiones que se explican a continuación:

- Contenido (¿ qué ?) Esta dimensión tiene que ver con el tipo de mensaje que va a ser enviado, el cual puede ser de tres tipos:

  1. Reporte: es un mensaje que provee la descripción de las características y circunstancias de la actividad o tarea.

  2. Recordatorio: es un mensaje que especifica cuando se debe realizar una tarea o actividad.

  3. Alarma: es un mensaje que tiene como propósito advertir al usuario acerca de un evento.

- Tiempo (¿ cuándo y por qué ?) Tiene que ver con el cuándo es apropiado entregar una notificación considerando el contexto del que envía y recibe la notificación. El tiempo en que se debe entregar la notificación también está relacionado con el propósito de la notificación: antes (prevenir algo), mientras (algo está ocurriendo) y después (algo ha ocurrido).

- Actor (¿ quién ?) Tiene que ver con el individuo o grupo de individuos que recibirán la notificación.

- Presentación (¿ cómo y dónde ?) tiene que ver con cómo la notificación será presentada y en cual dispositivo, y la forma en que se presenta, pasiva (sin intervención del actor) o activa (con un cierto nivel de intervención).

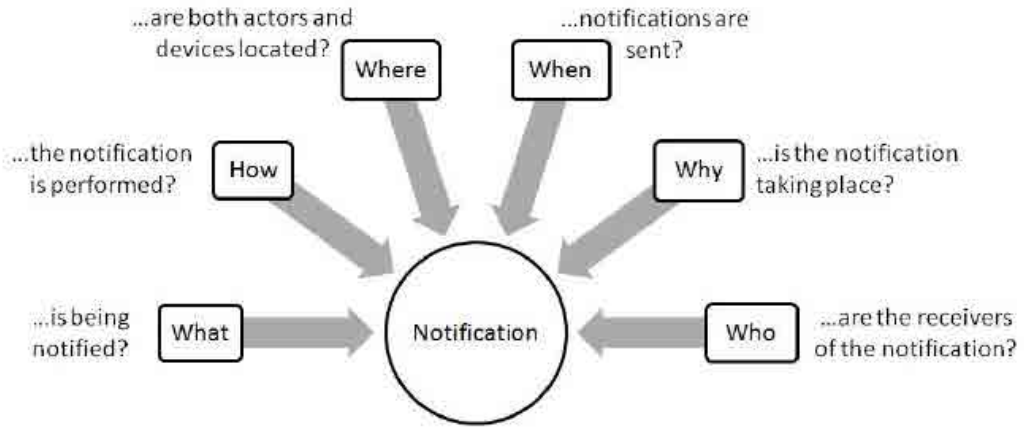

Figura 23: Seis preguntas para hacer una notificación, fuente (Sandra Nava-Muñoz 2009)

3.5. Sistemas y Herramientas Colaborativas

Se puede definir una herramienta colaborativa como una plataforma que sirve para la agrupación de dos o más usuarios colaborando y contribuyendo para alcanzar una meta en común. En una herramienta colaborativa en la nube los servicios de comunicación y coordinación se ejecutan en un espacio compartido de la nube, de forma centralizada o distribuida en diferentes nubes, además son utilizados desde diferentes plataformas móviles o de escritorio, ya sea en forma de aplicaciones nativas o en versión *Web* para colaborar virtualmente sobre *Internet*. Los servicios de colaboración en la nube tienen la ventaja de tener flexibilidad para poder gestionar y asignar los recursos colaborativos y expandirlos en caso de ser necesario (West et al. 2012; Lin & Tsai 2011; Nasirifard et al. 2011; Neyem et al. 2012). Los servicios colaborativos en la nube son accesibles globalmente, tienen un alto grado de tolerancia a fallos y tienen un alto grado de usabilidad (Yeh et al. 2013).

En un ambiente colaborativo en la nube los usuarios pueden colaborar con otros usuarios utilizando diferentes dispositivos, como por ejemplo ordenadores portátiles, tabletas o móviles y utilizando diferentes plataformas como por ejemplo *Linux, Windows, iOS y Android.* Los usuarios pueden colaborar sin restricciones en cuanto a tiempo, localización geográfica y a veces colaborando en diferentes idiomas. Todo lo mencionado anteriormente tiene una gran influencia en el tipo de colaboración que el usuario puede tener con otros usuarios, es por ello que el contexto y la situación del usuario son dos aspectos importantes a tener en cuenta para diseñar ambientes colaborativos (Smari et al. 2013; Li et al. 2013).

Las herramientas colaborativas permiten a los usuarios publicar, compartir y gestionar información. La creación conjunta de información y el poder compartirla son características clave en este tipo de herramientas, los datos e información de los usuarios son almacenados y procesados en la nube (Yeh et al. 2013; Neyem et al. 2012). Los contenidos son creados y gestionados completamente por el usuario y la función principal de una herramienta colaborativa es permitir compartir estos contenidos (West et al. 2012).

3.5.1. Escenarios en los que se puede dar la Colaboración Móvil

Se puede clasificar la forma en que se da la colaboración móvil teniendo en cuenta dos aspectos que tienen que ver con la movilidad, uno es el tiempo, por ejemplo que los usuarios coincidan «simultáneamente» en un tiempo dado para colaborar, esto sólo se puede ocurrir cuando los usuarios puedan trabajar de manera sincronizada. La otra característica es la localización del usuario, dependiendo de dónde se encuentra el usuario se va a ver afectada su «disponibilidad», se puede entender en este caso que los usuario estén al alcance uno del otro para poder colaborar. Dos usuarios están al alcance uno del otro siempre y cuando tengan la capacidad de poder comunicarse e intercambiar información y que el tiempo de respuesta sea aceptable para cada uno de los usuarios (Herskovic et al. 2009).

La simultaneidad se puede dar en dos dimensiones: una es que los usuarios tengan presencia simultánea en un tiempo dado, lo que quiere decir que trabajan de forma sincronizada (*Simultaneous*). La otra es cuando los usuarios no tienen presencia al mismo tiempo pero de alguna forma tienen la capacidad de trabajar de manera asíncrona (*Non - Simultaneous*) (Herskovic et al. 2009).

La disponibilidad se puede dar también en dos dimensiones: una es estar disponibles y al alcance de otros usuarios (*Reachable*) y la otra lo opuesto, no estar disponibles y por lo tanto no poder estar al alcance de otros usuarios (*Unreachable*) (Herskovic et al. 2009).

Combinando estas dimensiones dan como resultado cuatro posibles escenarios de colaboración en los que el usuario podrá estar con otros usuarios, en la figara 24 se puede ver los cuatro escenarios y a continuación se describen cada uno de ellos.

3.5.1.1. Simultaneous and Reachable

En este escenario de colaboración los dos usuarios trabajan al mismo tiempo y la interacción es de una forma directa. Un ejemplo de este tipo de escenarios puede ser dos

usuarios utilizando la herramienta colaborativa *Google Drive* editando al mismo tiempo un documento de *Word* en la nube.

3.5.1.2. Simultaneous and Unreachable

En este escenario de colaboración los usuarios están trabajando de manera síncrona pero no pueden comunicarse entre ellos por lo tanto son inalcanzables el uno para otro. Un ejemplo de este escenario pueden ser dos usuarios que han descargado documentos de *Dropbox*[36] y han activado la opción para acceder a los documentos cuando no tienen conexión y de esta manera poder trabajar a pesar de estar desconectados.

3.5.1.3. Non-Simultaneous and Reachable

En este escenario de colaboración los usuarios están trabajando en diferentes periodos de tiempo pero un servicio de sincronización o de comunicación les permite comunicarse a pesar de estar trabajando de manera asíncrona. Un ejemplo de este escenario puede ser la herramienta colaborativa y de almacenamiento de archivos en la nube llamada *BOX*[37], con esta herramienta los usuarios pueden sincronizar archivos entre ellos y dejar comentarios, gracias a esto, los usuarios pueden trabajar de forma asíncrona y tener comunicación utilizando esta herramienta.

3.5.1.4. Non-Simultaneous and Unreachable

Este escenario de colaboración los usuarios difícilmente pueden trabajar de forma colaborativa, los dos usuarios no tienen forma de tener comunicación entre ellos y trabajan en horarios diferentes por lo que la sincronización es escasa. En un momento dado uno o ambos usuarios tendrán que cambiar su estado en alguna de las dimensiones para poder tener un mejor grado de colaboración. Un ejemplo puede ser el tratar de tener una conexión de datos cada cierto intervalo de tiempo y usar la herramienta *Dropbox* para sincronizar los archivos y de esta manera estar actualizado.

[36] http://www.dropbox.com/
[37] https://www.box.com/

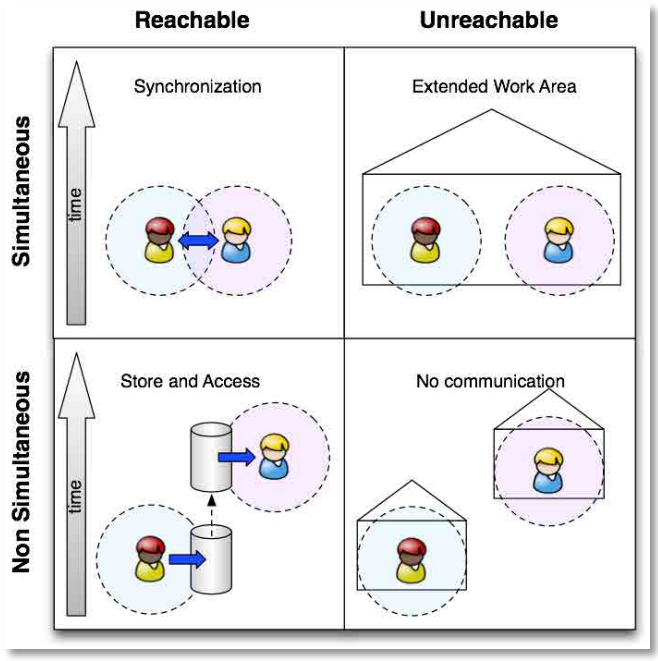

Figura 24: Diferentes escenarios de colaboración, fuente (Herskovic et al. 2009).

3.5.2. Estilos de Interacción Colaborativa

Debido a que las aplicaciones móviles colaborativas en la nube se dan en ambientes distribuidos, dónde los servicios de comunicación, colaboración y coordinación siempre están disponibles sin importar el contexto o la situación de movilidad del usuario, la colaboración entre usuarios se puede dar de acuerdo a tres estilos de interacción: colaboración atendida, colaboración parcialmente desatendida y colaboración desatendida (Rodríguez-Covili et al. 2011). Cada uno de estos estilos depende del contexto del usuario, como por ejemplo si está con otras personas, si está ocupado ejecutando otras tareas o si se encuentra en movimiento o está inmóvil, etc. A continuación se explica cada uno de estos estilos.

3.5.2.1. Colaboración Atendida

Las interacciones se dan entre usuarios que necesitan estar completamente involucrados en el proceso colaborativo. Los usuarios tienen un estado de movilidad pasivo ya que sus interacciones requieren que su atención sea constante para que pueda participar de una forma eficiente. Un ejemplo de este tipo de interacción puede ser un usuario colaborando con un grupo de usuarios editando una hoja de cálculo con la herramienta

colaborativa *Google Drive*[38] en un *iPad* sentado en una cafetería o en su casa (Rodríguez-Covili et al. 2011).

3.5.2.2. Colaboración Parcialmente Desatendida

Es cuando un usuario interactúa con uno o varios dispositivos móviles pero no se comunica directamente con otros usuarios. En este tipo de interacciones normalmente la localización geográfica de los usuarios cambia constantemente. Un ejemplo de este tipo de interacciones puede ser cuando un usuario accede, actualiza o sincroniza un documento en *Dropbox*, los usuarios con los que se ha compartido este archivo tienen una copia del mismo, pero no se enteran de las transacciones que se están realizando en el documento. Solo el usuario que realiza los cambios en el documento se encuentra en un estado estacionario, los demás usuarios posiblemente se encuentran en un estado móvil (Rodríguez-Covili et al. 2011).

3.5.2.3. Colaboración Desatendida

Este tipo de interacciones se da cuando el usuario se encuentra en movimiento pero su dispositivo móvil está ocupado interactuando con un servicio colaborativo. Los usuarios normalmente no son conscientes de que en su dispositivo móvil se está dando este tipo de interacciones colaborativas con otros usuarios o servicios colaborativos. La colaboración desatendida normalmente se da para proveer servicios colaborativos de tipo contextual como por ejemplo compartir la localización del usuario con otros usuarios para saber si está disponible para realizar una tarea dependiendo de su localización, o saber si el usuario se encuentra en movimiento y no puede colaborar en ese momento (Rodríguez-Covili et al. 2011).

3.5.3. Clasificación de las Herramientas Colaborativas

Las herramientas colaborativas ofrecen funcionalidades diferentes dependiendo del tipo de colaboración y capacidad de procesamiento de información que requieren los

[38] https://drive.google.com/

usuarios para realizar las tareas colaborativas. Algunas herramientas son diseñadas con el propósito de realizar tareas sencillas y el procesamiento de la información es limitada. Las tareas con un bajo nivel de incertidumbre y fáciles de resolver se deben realizar con herramientas colaborativas sencillas y de esta manera ahorrar tiempo y hacer que la colaboración sea la más óptima para los usuarios, un ejemplo de una herramienta colaborativa sencilla y eficiente es *Wunderlist*[39] que permite crear listados de tareas por hacer y asignarlas a un grupo de usuarios, la herramienta se encarga de gestionar quien debe hacer las tareas y notificar al usuario cuando la fecha límite para realizar la tarea que le toca ha expirado, la herramienta tiene una funcionalidad limitada pero resuelve eficientemente la tarea para la que fue diseñada (Keith et al. 2010).

Las herramientas más complejas ofrecen un avanzado nivel de colaboración y procesamiento de información como por ejemplo la herramienta *Google Drive,* esta permite editar documentos en tiempo real entre varios usuarios y gestionar los documentos para dar permisos de lectura, escritura y permisos de quien puede acceder al documento. Los ambientes colaborativos con un alto nivel de incertidumbre en las tareas que se necesitan resolver requieren de herramientas que puedan realizar tareas complicadas para que los usuarios obtengan un óptimo nivel de colaboración (Keith et al. 2010).

Datta (2010) hace una clasificación de las herramientas colaborativas, tomando como punto de referencia al usuario y sus necesidades colaborativas, las clasifica en cuatro categorías: *de comunicación, redes sociales, organizativas y para crear/editar contenidos.*

3.5.3.1. Herramientas de Comunicación Colaborativas

Se pueden clasificar en los siguientes tipos:

- Foros de discusión: sirven para realizar discusiones entre usuarios con la posibilidad de que los usuarios puedan crear temas a discutir o hacer comentarios a una respuesta anterior de un tema en específico.

[39] https://www.wunderlist.com/en/

- Audio conferencia (VoIP): sirve para realizar reuniones entre dos o más usuarios en diferentes localizaciones geográficas.

- Vídeo conferencia: sirve para realizar presentaciones o reuniones entre dos o más usuarios en diferentes localizaciones geográficas.

- *Whiteboarding*: sirve para hacer anotaciones y dibujos en tiempo real sobre cualquier medio que se presenta en la pantalla como por ejemplo: documentos, imágenes, pantalla, diapositivas, etc.

- Conferencia vía *Web*: sirve para hacer reuniones y presentaciones entre dos o más usuarios en diferentes localidades, este tipo de herramientas permite presentar e interactuar con contenidos como en las herramientas *Whiteboarding.*

- Mensajería instantánea: sirve para tener comunicación en tiempo real entre dos o más usuarios utilizando mensajes de texto, la comunicación puede también darse de forma asíncrona ya que el usuario no necesariamente tiene que responder en tiempo real.

3.5.3.2. Herramientas Colaborativas Sociales

Se pueden clasificar en los siguientes tipos:

- Plataformas de redes sociales: son herramientas que sirven para crear conexiones con otros usuarios que comparten un interés común o experiencia en un tema específico.

- *Blogs*: Espacios que sirven para presentar y discutir opiniones, pensamientos y reflexiones de temas específicos que el usuario quiere presentar a otros usuarios que visitan su *blog*, también a veces sirve para compartir información y contenidos como vídeos e imágenes de un tema en específico.

- *Microblogs*: Estas herramientas sirven para distribuir mensajes de texto cortos de forma masiva a grupos de usuarios.

3.5.3.3. Herramientas Colaborativas Organizativas

Se pueden clasificar en los siguientes tipos:

- Compartición de Archivos: su principal propósito de este tipo de herramientas es que un archivo esté disponible para varios usuarios en su versión más actual.

- Planeación de proyectos: sirven para planificar proyectos, tareas y recursos dentro del proyecto.

- Gestión de proyectos: estas herramientas sirven para gestionar todo el ciclo de vida de un proyecto, se pueden gestionar tareas, recursos, fechas límite de las metas, etc.

- Calendarios y eventos en grupo: estas herramientas sirven para crear eventos y reuniones de grupos de usuarios, facilita que los usuarios se pongan de acuerdo en la fecha del evento o reunión, estas herramientas casi siempre están interconectadas con otras herramientas colaborativas.

3.5.3.4. Herramientas Colaborativas para Crear y Editar Contenidos

Se pueden clasificar en los siguientes tipos:

- *Wikis*: sirven para crear y editar documentos que son mostrados como páginas *Web*, en una *Wiki* normalmente cualquier usuario tiene la libertad de colaborar en la creación o edición del documento.

- De propósito general: son herramientas que sirven para colaborar en la creación de contenidos y que ofrecen opciones para crear documentos de texto, presentaciones, hojas de cálculo y contenidos gráficos.

3.5.4. Clasificación de las Herramientas Colaborativas Móviles de acuerdo con su grado de uso en el dispositivo

De acuerdo con el grado de uso, las herramientas colaborativas en un dispositivo móvil se pueden clasificar en tres categorías: extensión móvil, híbridas y sistemas móviles completos (Saucedo-Tejada et al. 2013).

3.5.4.1. Extensión móvil

Este tipo de sistemas no necesitan de dispositivos móviles y la mayoría del trabajo es solo un añadido al sistema colaborativo (Saucedo-Tejada et al. 2013).

3.5.4.2. Híbrido

La operación de estos sistemas depende del trabajo que se realiza en el dispositivo móvil; los ordenadores estacionarios aún son requeridos para la mayoría de las funciones del sistema, algunas partes del trabajo deben ser exclusivamente realizadas por ordenadores estacionarios (Saucedo-Tejada et al. 2013).

3.5.4.3. Sistemas Móviles Completos

Los ordenadores estacionarios son solo una parte opcional de este tipo de sistemas, en este tipo de sistemas es dónde se pueden dar todos los tipos de colaboración incluyendo la colaboración espontánea debido a que son ejecutados en dispositivos móviles y pueden ser llevados a cualquier lugar y pueden ser iniciados en cualquier momento ya que no se necesita una conexión a un ordenador estacionario (Saucedo-Tejada et al. 2013).

3.5.5. Software y Webs Sociales como Herramientas Colaborativas

En la actualidad han surgido una gran cantidad de sistemas y herramientas que permiten a los usuarios crear comunidades en línea, estos sistemas cuentan con herramientas que permiten a los usuarios crear, publicar y compartir contenidos, los usuarios son creadores y al mismo tiempo consumidores de contenidos, tienen un rol más activo y la colaboración se da de forma diferente que en las *Webs* tradicionales.

Los usuarios pueden recolectar, compartir recursos y publicar materiales que otros usuarios pueden ver, comentar, votar, guardar y retransmitir. Los usuarios también pueden publicar contenidos propios como por ejemplo fotos, vídeos, perfiles de usuarios y actualizaciones de actividades, esto hace que el trabajo se convierta en un proceso abierto, colaborativo y permite tener comunicación entre muchos usuarios.

El proceso de trabajo y sus resultados son públicos y accesibles a otros usuarios que están interesados en el tema. Este tipo de sistemas y herramientas sociales son el resultado de la unión de las redes sociales y medios sociales. Las redes sociales permiten a la gente estar conectada con otras personas en comunidades en línea. Los medios sociales tienen como función principal compartir contenidos creados por el usuario. La distinción entre estos dos tipos de sistemas está desapareciendo rápidamente ya que las redes sociales actualmente ofrecen como una de sus principales funcionalidades el compartir contenidos creados por los usuarios y los sistemas de medios sociales ofrecen funcionalidades para crear perfiles personales y comunidades (Kim et al. 2010; Valtonen et al. 2010; Hansen et al. 2011).

3.5.5.1. Medios Sociales

Los Medios Sociales son herramientas que cuentan con unas funcionalidades que permiten la interacción social entre usuarios. Estas funcionalidades son el resultado de juntar técnicas sociales con tecnología para permitir a los usuarios (Kim et al. 2010; Valtonen et al. 2010; Hansen et al. 2011):

1. Crear información colaborativa.

2. Darle un nuevo sentido a la gestión de la información, ya que se puede buscar, compartir y evaluar toda la información disponible entre los usuarios.

3. Y así permitir también conectar, informar, inspirar y seguir a otros usuarios con nuevas formas de colaboración.

3.5.5.2. Principales Características de los Servicios de Medios Sociales

Los medios sociales varían en cuanto a sus funcionalidades y la manera de interactuar entre los usuarios. La manera en cómo se conectan lo usuarios entre ellos varía en las decisiones técnicas que se tomen para diseñar estas herramientas, estas decisiones de diseño tienen una influencia en la interacción social, como por ejemplo: qué es lo que el usuario puede ver, quién puede replicar a quién, por cuanto tiempo el contenido va a estar visible, quién puede estar ligado a qué, quién puede tener interacción con quién.

Las prácticas sociales de los usuarios, su personalidad y sus antecedentes también tienen influencia en como son usados los sistemas de medios sociales. Los sistemas de medios sociales como por ejemplo los grupos de discusión y los sistemas de correo electrónico han sido adaptados de una forma satisfactoria a las necesidades de muchos grupos sociales. Se deben tomar en cuenta las diferencias y similitudes de los sistemas de medios sociales para poder hacer una clasificación de las principales características clave que los servicios de medios sociales pueden tener (Hansen et al. 2011), a continuación se describen las características más importantes.

3.5.5.2.1. Tamaño de la Población de Productores y Consumidores

Los servicios de medios sociales pueden variar en términos del número de «productores» y «consumidores» que tienen. Normalmente los consumidores son al mismo tiempo productores o pueden cambiar de papel de un momento a otro. Los sistemas de medios sociales pueden soportar diferentes escalas de producción y consumo de contenidos digitales. Por ejemplo en el sistema de medios *Twitter*[40] un

[40] https://twitter.com/

único productor puede crear un *Tweet*[41] y millones de consumidores pueden leerlo, o se puede dar el caso de que un productor crea un *Tweet* que lo consume un solo usuario.

Otro ejemplo puede ser el sistema de medio sociales *YouTube*[42], un grupo de Productores muy grande, crea contenidos que pueden ser compartidos a un grupo de consumidores muy grande, este grupo de consumidores puede cambiar de un momento a otro en tamaño, al igual que el número de productores. En la red social de *Facebook* un grupo de consumidores mediano, puede estar inscrito a un grupo de *Facebook*, donde un grupo de productores mediano cuelga sus contenidos.

Los medios sociales permiten a individuos alcanzar grandes grupos a través de *blogs*, *podcasts* y vídeos colgados en *YouTube*. Sitios de redes sociales permiten a los usuarios alcanzar grupos medianos de consumidores permitiendo que los contactos del usuario puedan crear listados de actualizaciones de su actividad, *posts* que hace en *Facebook* o en su página personal de *Twitter* con todos los *Tweets* que ha hecho el usuario (Hansen et al. 2011).

3.5.5.2.2. Ritmo de Interacción

Tradicionalmente el «ritmo de interacción» en los medios sociales se ha hecho distinguiéndolos entre medios sociales de comunicación síncronos y asíncronos. Los sistemas asíncronos como por ejemplo el correo electrónico y foros de discusión, tienen un intervalo de interacción de horas, días o tal vez semanas. Estos sistemas permiten que el usuario pueda tomarse el tiempo de preparar su participación sin tener que coordinarse con las otras personas con las que está en comunicación, por lo que posiblemente pueda hacer contribuciones más cuidadosas. Los usuarios con los que colabora pueden estar en diferentes zonas horarias.

Los sistemas síncronos como los *chats*, mensajería instantánea y las vídeo conferencias requieren que los usuarios tengan una coordinación temporal y requiere que los participantes tengan una interacción al mismo tiempo y cara a cara. En la interacción síncrona se crea un ambiente ideal ya que los usuarios tienen que comunicar y ajustar sus reacciones uno frente a otro en tiempo real.

[41] Un *Tweet* es un mensaje corto de 140 caracteres.
[42] https://www.youtube.com/

Recientemente la distinción de los medios sociales con estas dos formas de comunicación ha ido desapareciendo, pero depende en gran medida de las expectativas que el usuario tiene en su ritmo de interacción, por ejemplo es normal que un usuario responda un *post* o a una actualización del estado de un usuario un una red social en unos minutos o sí lo desea lo puede hacer un día después sin ningún problema, en *Twitter* responder a un *Tweet* en unos minutos o un día después también es aceptable. Hoy en día *Google Hang Outs* y *Google Chat* están integrados en el sistema de correo electrónico *Gmail*[43] y la red social *Google Plus*[44], haciendo que la distinción entre los modos de comunicación síncrona y asíncrona en los medios sociales desaparezca (Hansen et al. 2011).

3.5.5.2.3. Género de Elementos Básicos

Los elementos básicos son los elementos esenciales de un sistema de medios social, estos elementos varían tanto en tamaño como en tipo. La diferencia en el tamaño de los Elementos Básicos produce diferentes patrones de interacción y los diferentes tipos de elementos ayudan a entender las similitudes y diferencias entre estas herramientas.

La forma en que están diseñadas las herramientas de medios sociales es un factor decisivo en el diseño del tamaño de los elementos básicos, por ejemplo en un sistema de mensajería el tamaño reducido de la caja de texto donde se escriben los mensajes, promueve brevedad y de esta forma insta a escribir mensajes cortos, mientras que la plataforma *Wikipedia* está diseñada para poder crear largos documentos de texto, ofreciendo varios niveles de cabeceras y generando tablas de contenidos de forma automática. En *Twitter*, los *Tweets* pueden alcanzar hasta 140 caracteres mientras que un *e-mail* puede tener unas cuantas líneas o tener varios párrafos.

Normalmente cada una de las herramientas de medios sociales tiene uno o varios elementos básicos que son su razón de ser y con los que han ido evolucionando a través del tiempo, por ejemplo: fotos en *Flickr*[45], vídeos en *YouTube*, marcadores en

[43] https://mail.google.com
[44] https://plus.google.com
[45] http://www.flickr.com/

Delicious[46], gente en *Facebook*, *Tweets* en *Twitter*. Cada uno de estos Elementos Básicos provee diferentes mecanismos y niveles para atraer e interactuar con los usuarios, identificar los elementos básicos de un sistema de medios sociales es importante porque son la razón principal de la interacción y de la creación de redes sociales, cuando estos elementos son interconectados o intercambiados entre los usuarios (Hansen et al. 2011).

3.5.5.2.4. Control de los Elementos Básicos

Los sistemas de medios sociales proveen diferentes niveles de Control sobre sus elementos básicos. Algunos sistemas implementan el control de sus elementos básicos clasificando a los usuarios en anónimo, registrados y usuarios con privilegios especiales como los administradores, restringiendo en cada uno de estos perfiles de usuario quién puede crear, editar, eliminar, leer, invitar, responder y compartir contenidos de varios tipos. Por ejemplo en *Twitter* los usuarios deben estar registrados para crear un *Tweet*, pero cualquier persona puede leer los *Tweets* de otros usuarios sin necesidad de estar registrados.

Entre más abierta es una comunidad hay más posibilidades de que los usuarios den un uso mal intencionado a sus funcionalidades y una comunidad demasiado cerrada reduce el número de personas que pueden contribuir y colaborar en la comunidad. Una comunidad abierta puede atraer contribuciones y colaboraciones de muy buena calidad incluyendo usuarios que están dispuestos a corregir y prevenir el mal uso que otros usuarios dan a la comunidad.

En *facebook* los usuarios pueden reportar cuando un contenido es ofensivo y pedir que este sea eliminado. En *Wikipedia,* que es una comunidad abierta donde usuarios registrados y no registrados pueden contribuir libremente, las contribuciones de baja calidad hechas por usuarios no registrados son eliminadas por otros usuarios. Los tipos de barreras de entrada que una comunidad debe tener son decisivas en la forma en cómo se construye una comunidad.

[46] http://delicious.com/

Las comunidades que han tenido éxito han puesto ciertos límites para evitar que usuarios no registrados abusen o tomen ventaja de los recursos que se crean y mantienen dentro de la comunidad. Poner límites puede impactar en el tipo de interacciones que la gente está dispuesta a aceptar debido a la forma en que esto afecta a la audiencia en la comunidad (Hansen et al. 2011).

3.5.5.2.5. Tipos de Conexiones

La forma en que los elementos básicos de los sistemas de medios sociales son conectados es muy variada, por lo que se debe entender cada una de estas formas de «lazos» o conexiones para poder diseñar y entender las redes de cada uno de los sistemas de medios sociales.

Los elementos básicos de los sistemas de medios sociales se pueden Interconectar entre ellos de forma explícita o implícita. Las conexiones explícitas son creadas conscientemente e intencionalmente por parte del usuario mientras que las conexiones implícitas son resultado de los comportamientos que tienen los usuarios en las redes e *Internet.*

Los ejemplos más comunes de conexión explícita en los medios sociales pueden ser: crear conexiones con amigos en las redes sociales, en este tipo de conexión cada uno de los usuarios debe aceptar la conexión para que esta pueda realizarse, seguir un usuario en *Twitter,* etiquetar varias fotos con la misma etiqueta, y agregar a un contacto a un sistema de mensajería instantánea o vídeo conferencia como por ejemplo *Skype*. Las conexiones implícitas son por ejemplo cuando el usuario responde a un *post,* o cuando el usuario presiona el botón *plus* (+) de *Google Plus* en un *post*, a pesar de que estas acciones han sido intencionales, no fueron realizadas con la intención explícita de crear una conexión con la persona.

Las conexiones directas e Indirectas son otra distinción importante que se puede hacer con la forma en cómo se conectan los elementos básicos. Si dos personas se convierten en amigos en *Facebook* la conexión es mutua por lo tanto es indirecta, lo contrario de *Tweeter,* donde un usuario puede seguir a otro usuario sin necesidad de que este último lo apruebe. Las conexiones que son reciprocas son más fuertes que las que no lo son.

Las conexiones pueden tener diferentes valores y pesos, por ejemplo en las redes sociales los usuarios pueden ser amigos o no lo que se traduce en una conexión binaria con valor de 0 (no eres amigo) o 1 (eres amigo); el grado y la intensidad de la conexión de dos amigos en una red social puede medirse por el número de mensajes que se envían o por el número de etiquetas que hace un usuario del otro.

La localización ha servido para expandir los servicios de medios sociales, permitiendo que las conexiones se realicen entre lugares, objetos y personas. Con los dispositivos móviles los medios sociales pueden integrar información de la localización y actividad del usuario y de esta forma hacer conexiones de formas novedosas, se pueden crear vínculos y conexiones con solo estar en el mismo lugar que otros usuarios a pesar de no coincidir al mismo tiempo (Hansen et al. 2011).

3.5.5.2.6. Retención del Contenido

Lo sistemas de medios sociales puede variar respecto a cuánto tiempo retienen el contenido, hay sistemas de medios que registran permanentemente cada acción que se realiza en el sistema y es presentada como un historial a los usuarios, por otro lado hay sistemas como los de mensajería instantánea o video conferencia que no registran ninguna de las interacciones a menos que se configure para que lo haga. La mayoría de los sistemas de medios sociales se encuentran en un punto medio, la mayoría de estos sistemas varia en la política de retención de los medios, dependiendo de las políticas específicas para cada tipo de medio o de la configuración del usuario (Hansen et al. 2011).

3.5.5.3. Redes y Medios Sociales Abiertos

Hoy en día, los desarrolladores de aplicaciones, pueden crear aplicaciones que hacen uso de las grandes cantidades de información y contenidos que están almacenados en los sitios *Web* sociales y ponerlos a disponibilidad de los miembros de estos sitios sociales. Es posible exportar contenidos a otros sitios y mantener un seguimiento de las conexiones en línea de los usuarios que hacen en un sitio para utilizarlas en otro sitio

diferente y todo esto mediante APIs de código abierto que los sitios *Web* sociales ofrecen a los desarrolladores.

Por ejemplo *Facebook* permite a los desarrolladores hacerlo a través de su interface abierta para la programación de aplicaciones (*Facebook* API)[47], al igual que *Google* con su API llamada *OpenSocial*[48]. Estas dos APIs ofrecen a los desarrolladores ahorrarse el tener que programar procedimientos para crear conexiones entre usuarios puesto que en *Facebook* y *Google* los usuarios ya las tienen. Las aplicaciones que hacen uso de la *API* de *Facebook* permiten a los desarrolladores alcanzar un segmento de usuarios en *Facebook*, los usuarios de *Facebook* pueden encontrar estas aplicaciones interesantes y útiles ya que pueden tener acceso a la base de datos de los usuarios, amigos, eventos y grupos que hay en *Facebook* como si estuviesen dentro de esta aplicación y al mismo tiempo las aplicaciones expanden los servicios que *Facebook* puede ofrecer a sus usuarios (Kim et al. 2010).

Para eliminar el problema de que los usuarios tengan que establecer nuevamente conexiones con amigos en cada sitio *Web* social y tener que identificarse varias veces para conectar con sus amigos, *Facebook, Google* y otros sitios *Web* sociales extendieron su plataforma para solucionar este problema (Kim et al. 2010), a esta solución se le conoce como *Facebook Connect*[49] y *Google Friend Connect*[50]. Así es posible que un usuario de una *Web* social por ejemplo *Facebook* pueda iniciar sesión en otra *Web* social por ejemplo *Games.com* utilizando las credenciales de inicio de sesión única (SSO), de esta forma el usuario de *Facebook* puede conectarse e interactuar con nuevos amigos que pertenecen al sitio social *Games.com.*

3.5.6. Requerimientos de un Sistema Colaborativo Móvil

Cuando se diseña un Sistema Colaborativo Móvil se deben tener en cuenta ciertos requisitos con los que debe cumplir el sistema, de acuerdo con Herskovic et al. (2011) estos requisitos se pueden clasificar en siete categorías:

[47] https://developers.facebook.com/
[48] http://opensocial.org/
[49] https://developers.facebook.com/docs/facebook-login/
[50] http://www.google.com/friendconnect/home/overview

- Flexibilidad de la Interacción del Usuario
- Protección de la Interacción del Usuario
- Comunicación
- Heterogeneidad e Interoperabilidad
- Servicios para el soporte de la Interacción Autónoma
- Presencia del Usuario
- Consistencia y Disponibilidad

Es importante tener en cuenta estos requisitos debido a que la utilidad de un sistema es determinado por las características que se incluyen en estos requerimientos (Neyem et al. 2012).

3.5.6.1. Flexibilidad en la Interacción del Usuario

En las actividades colaborativas con el uso de dispositivos móviles el contexto de trabajo es dinámico por lo cual las «aplicaciones colaborativas móviles» deben reaccionar y responder a los cambios que se dan en el ambiente. Por ejemplo, la disponibilidad de los recursos compartidos puede verse afectada debido a la movilidad de los usuarios; otro ejemplo es el cambio de tamaño y estructura del grupo colaborativo (Neyem et al. 2008). Una forma de resolverlo es proporcionando flexibilidad mediante un componente de detección automática del usuario que provea información contextual para implementar mecanismos de presencia, otra forma es a través de un componente para detectar la conexión y desconexión del usuario para permitir a las aplicaciones trabajar en modo off-line y cambiar a modo on-line bajo demanda (Neyem et al. 2012).

3.5.6.2. Protección de la Interacción del Usuario

Los sistemas colaborativos móviles deben tener la funcionalidad de proteger el trabajo y los recursos del usuario de un posible ataque o mal uso por parte de otros usuarios. Se deben proveer mecanismos de privacidad y de verificación de la identidad del usuario; por ejemplo, las sesiones de trabajo deben ser ofrecidas en un espacio de interacción donde el usuario pueda elegir entre sesiones públicas, privadas o por invitación.

También se le debe permitir al usuario elegir la información que quieren compartir con los otros usuarios y ofrecer un espacio público para poder compartir información. Además se debe identificar al usuario cada vez que se accede a un recurso compartido, el nivel de verificación de la identidad del usuario se puede poner como una opción bajo demanda (Neyem et al. 2012).

3.5.6.3. Comunicación

La Comunicación es la base principal que permite que la coordinación y colaboración pueda realizarse (Ellis et al. 1991). La interacción entre los usuarios de una aplicación móvil colaborativa incluye el intercambio de mensajes, alarmas y notificaciones. Debido a la alta incertidumbre de conexión y comunicación de un sistema móvil, es necesario ofrecer varios canales de comunicación para que el usuario pueda interactuar y colaborar con otros usuarios, estos canales de comunicación pueden ser sistemas de mensajería síncronos, asíncronos o mediante sistemas de notificaciones «push» (Neyem et al. 2012).

3.5.6.4. Heterogeneidad e Interoperabilidad

El tipo de dispositivo que un usuario utiliza, no debe ser un obstáculo para interactuar y trabajar con otros usuarios en un ambiente colaborativo móvil, por esta razón las aplicaciones colaborativas deben operar en función de datos y servicios y no del tipo de dispositivo (Neyem et al. 2008). El uso de tecnologías que están consolidadas como un estándar y el uso de información contextual que permita obtener información y características de los dispositivos pueden ayudar a mejorar la interoperabilidad y permitir que la colaboración e interacción sea transparente para el usuario, a pesar de que los dispositivos sean de diferentes características (Neyem et al. 2012).

3.5.6.5. Servicios para el Soporte de la Interacción Autónoma

La movilidad de los usuarios genera desconexiones y reconexiones de manera frecuente; las aplicaciones colaborativas deben ofrecer servicios que permitan al usuario tener la opción de acceder una conexión alternativa en caso de no contar con una conexión inalámbrica. Además de ofrecer al usuario la posibilidad de crear una conexión punto a punto, con otro dispositivo y el descubrimiento de otros dispositivos que puedan facilitar una conexión de datos (Neyem et al. 2008; Herskovic et al. 2011).

3.5.6.6. Presencia del Usuario

La interacción entre los usuarios en los sistemas colaborativos móviles se da bajo demanda e identificar la disponibilidad de colaboradores potenciales es esencial en este tipo de sistemas. Una forma de resolver esta necesidad es mediante mecanismos que distinguen la presencia y el estado en que se encuentra el usuario; como por ejemplo, saber si está conectado o desconectado, saber su disponibilidad para colaborar o por el contrario si se encuentra ocupado, notificar de su presencia y disponibilidad a los demás usuarios (Papadopoulos 2006; Herskovic et al. 2011).

3.5.6.7. Consistencia y Disponibilidad de los Datos

En un sistema colaborativo móvil los usuarios trabajan y colaboran al mismo tiempo con documentos que deben estar actualizados para ofrecer la última versión del documento a cada uno de los usuarios. Las frecuentes desconexiones de los usuarios generan inconsistencias y la indisponibilidad de la información compartida. Por lo que se debe contar con mecanismos que permitan la replicación, sincronización y resolución de conflictos para poder ofrecer información consistente y siempre disponible (Neyem et al. 2008; Herskovic et al. 2011).

3.5.7. Arquitectura de una Aplicación Móvil Colaborativa

La colaboración solo se puede dar con la ayuda de servicios de coordinación y comunicación (Ellis et al. 1991), es por esto que los sistemas colaborativos móviles requieren de funcionalidades independientes en tres puntos clave: Comunicación, Coordinación y Colaboración (Neyem et al. 2012).

En la figura 25 se muestra una arquitectura de tres capas que estructura la funcionalidad básica que debe tener una aplicación móvil colaborativa.

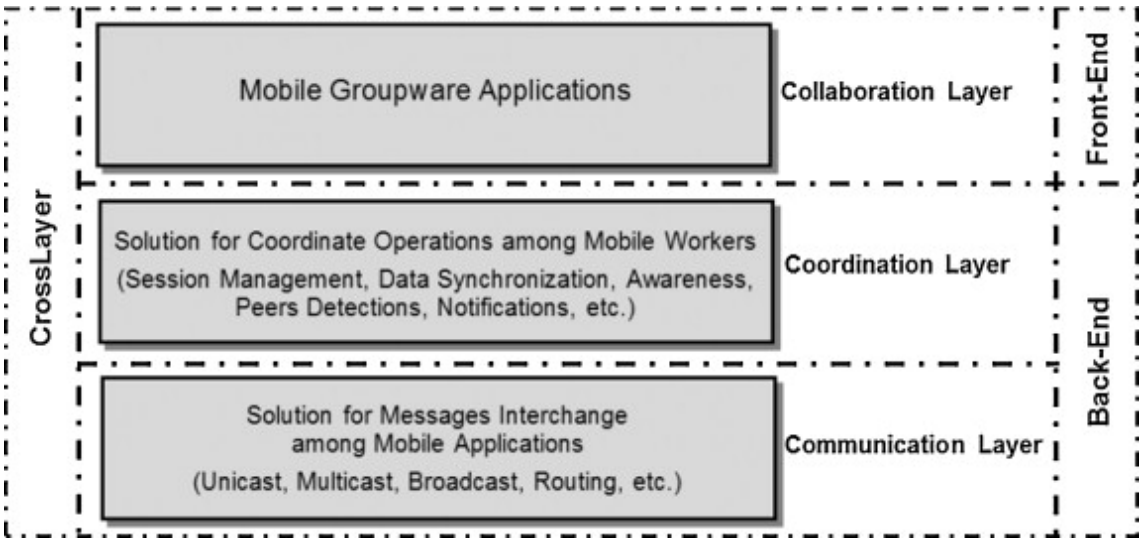

Figura 25: Arquitectura por Capas de una Aplicación Móvil Colaborativa, fuente (Neyem et al. 2012).

3.5.7.1. *Collaboration Layer*

La capa de colaboración da soporte para el manejo de las interacciones entre los usuarios móviles que colaboran para alcanzar las metas del grupo, provee soluciones que resuelven las funcionalidades que deben ser incluidas en una aplicación móvil colaborativa y que son presentadas en forma de interfaz gráfica al usuario, estas soluciones utilizan los servicios que son subministrados por la capa de coordinación.

3.5.7.2. Coordination Layer

La capa de coordinación se centra en coordinar las operaciones de los usuarios móviles de manera que estos usuarios tengan una vista consistente de la tarea que se está realizando por todo el grupo; proporciona soluciones que tienen que ver con aspectos de diseño típicos de un grupo colaborativo como por ejemplo: gestión de sesiones, compartir información, gestión de usuarios y roles, todo esto tomando en cuenta los contextos en que se lleva a cabo el trabajo colaborativo cuando se utilizan servicios de comunicación inestables.

3.5.7.3. Communication layer

La capa de comunicación se encarga de suministrar servicios que resuelvan las necesidades de comunicación entre los usuarios de las aplicaciones móviles colaborativas (Neyem et al. 2012).

3.6. Definición de Modelos y Arquitecturas

3.6.1. Modelo de Referencia

Un modelo de referencia, es la descomposición funcional de un problema en forma de componentes y su conexión entre ellos (Albin 2003). De acuerdo con Bass et al. (2012) “Es la división de funcionalidades, junto con la forma en que fluye la información entre las diferentes piezas del modelo”.

Es una forma simplificada de ver un problema y la manera en que puede ser resuelto. Un modelo de referencia (ver fig. 26) es usado en conjunto con Patrones de Arquitectura para dar forma a una arquitectura de referencia (Angelov et al. 2012).

3.6.2. Patrones de Arquitectura

Un patrón de arquitectura es una plantilla que sirve de modelo para una arquitectura de software, que tiene la finalidad de expresar un esquema estructurado y organizativo de un sistema de «software» (Gamma et al. 1999). Es una serie de reglas que proveen un

«*framework*» abstracto, con soluciones que sirven para resolver problemas recurrentes y ampliamente conocidos (*Microsoft* Patterns & Practices 2009).

Varios patrones de arquitectura pueden ser combinados en una arquitectura de software cuando se construye una arquitectura; diferentes patrones pueden ser aplicados a diferentes elementos, y así mismo es posible aplicar un solo patrón a todos los elementos de una arquitectura de software (Buschmann et al. 1996). En la Tabla 4 se muestran los patrones de arquitectura más comunes y las áreas donde se aplican.

Category	Architecture Patterns
Communication	Service-Oriented Architecture (SOA)
Deployment	Client/Server, N-Tier, 3-Tier
Structure	Component-Based, Layered Architecture

Tabla 4: Arquitecturas y los campos donde más se utilizan, fuente (*Microsoft Patterns & Practices* 2009)

3.6.3. Arquitectura de Referencia

Una arquitectura de referencia provee una solución en forma de plantilla para la arquitectura de software de una categoría de sistemas específica (Bass et al. 2012). Una arquitectura de referencia es el resultado de transponer un modelo de referencia y patrones de arquitectura, se compone de elementos de software que juntos resuelven el problema definido por un modelo de referencia (Bass et al. 2012).

En la mayoría de los casos estas arquitecturas están basadas en las mejores prácticas que se han ido acumulando a través del tiempo, dentro de un campo de aplicación específico (Greefhorst & Proper 2011).

Debido a que una arquitectura de referencia puede definirse en niveles de abstracción, debe ser lo suficientemente genérica para ser usada en una amplia categoría de sistemas y no solo para un sistema en específico (Clements et al. 2010). Las arquitecturas de

referencia se crean para facilitar el diseño y desarrollo de sistemas en múltiples proyectos (Angelov et al. 2012).

Los patrones de arquitectura son soluciones ampliamente reconocidas para implementar arquitecturas de software, pero se pueden dar casos donde las arquitecturas de referencia sirven también como patrones de arquitectura.

Hay que considerar que un patrón de arquitectura es independiente del dominio donde se aplica y tiene como objetivo resolver ciertas cualidades arquitectónicas; en cambio una arquitectura de referencia tiene como propósito definir las funcionalidades requeridas en el dominio (previamente definidas en el modelo de referencia) y su interacción con el ambiente del dominio.

Los patrones de arquitectura son usados en el diseño de arquitecturas de referencia para obtener las cualidades arquitectónicas deseadas, en la figura 26 se puede ver la relación entre los modelos, patrones y arquitecturas (Angelov et al. 2012).

Se puede dar el caso donde las arquitecturas de referencia son diseñadas como resultado de una investigación en un nuevo campo de conocimiento, a este tipo de arquitecturas de referencia se les conoce como «futuristas».

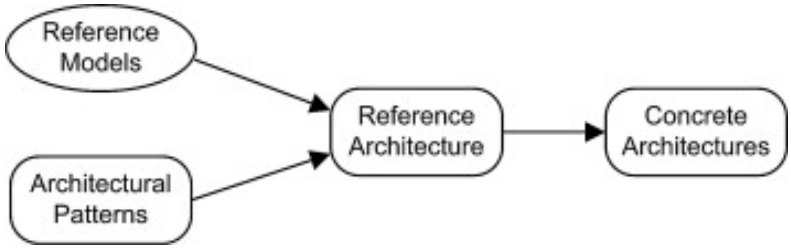

Figura 26: Relación entre Modelos, Patrones y Arquitecturas (las flechas indican entrada de datos de), fuente (Angelov et al. 2012).

3.6.4. Arquitecturas Concretas

Este tipo de arquitecturas son diseñadas implementando una arquitectura de referencia (ver fig. 26) y se utilizan para el desarrollo de una aplicación de software específica. Las arquitecturas de software concretas son diseñadas en un contexto específico y reflejan las metas del negocio en concreto.

A una arquitectura concreta también se le conoce como arquitectura de solución (Greefhorst & Proper 2011), Kruchten (1995) le da el nombre de «arquitectura de software», éste es tal vez el término que más se ha utilizado en los últimos años cuando se habla de este tipo de arquitecturas.

Bass et al. (2012) define una arquitectura de software de la siguiente manera:

> "La arquitectura de software de un programa o sistema computacional es la estructura o estructuras del sistema, abarca elementos de software, las propiedades visiblemente externas de estos elementos, y las relaciones entre estos elementos. Las propiedades externamente visibles, es lo que un elemento puede ofrecer o esperar de otros elementos, como por ejemplo los servicios que provee, características de desempeño, manejo de errores, uso compartido de recursos, etc."

La arquitectura de software define los elementos de software utilizados y la relación entre ellos. En una arquitectura de software, las interfaces públicas de los elementos son conocidas, pero no incluyen detalles o elementos que se requieren específicamente para su implementación.

3.6.5. Arquitecturas Estándar

Se puede dar el caso donde una arquitectura concreta puede ser utilizada para diseñar varias aplicaciones de software del mismo tipo, pero esto no quiere decir que se convierte en una arquitectura de referencia; ya que una arquitectura de referencia tiene como característica principal el ser genérica por naturaleza, lo que implica que puede ser utilizada en múltiples y en diferentes contextos. Satisfaciendo las necesidades de los elementos involucrados en estos contextos, la naturaleza genérica se alcanza diseñando la arquitectura en altos niveles de abstracción, lo que implica no tener en cuenta las diferencias introducidas por los contextos.

Cuando un tipo de aplicación se repite varias veces, es recomendable crear diseños abstractos de alto nivel (Arquitectura de Referencia), de manera que puedan ser reutilizados en múltiples ocasiones cuando se trate de este tipo de aplicación (Angelov et al. 2012).

Una arquitectura estándar, es la especialización de una arquitectura de referencia dentro de una organización en particular (Stoitsev 2012), esto quiere decir que servirá solo para desarrollar sistemas de información que cumplen con las necesidades de esta organización. En la figura 27 se pueden ver los niveles de abstracción de cada una de las arquitecturas. Las arquitecturas de referencia son abstractas en un nivel de estandarización organizacional, esto significa que podrá ser utilizada en múltiples organizaciones; las arquitecturas estándar y concretas son abstractas en un nivel de usuarios para un tipo específico de sistemas de información dentro de una organización.

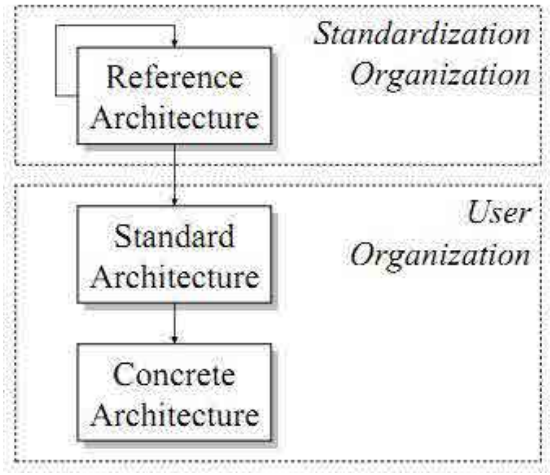

Figura 27: Arquitectura de Referencia y Estándar, fuente (Stoitsev 2012).

3.6.6. Dimensiones Organizacionales y Modelo Tridimensional para el Diseño de Arquitecturas

Los diferentes tipos de arquitecturas pueden ser organizadas de acuerdo con tres dimensiones conocidas: *abstraction, aggregation y realization*, cada una de estas dimensiones puede tener cierto nivel y variar dependiendo del tipo de arquitectura (Grefen 2012).

3.6.6.1. Abstraction

"Define que tan concreto es el modelo de arquitectura con respecto a los bloques de construcción del software. En su más alto nivel de abstracción un modelo de arquitectura, no provee información alguna de los tipos de sistemas y de los proveedores de software. En su más bajo nivel de abstracción, hasta la versión del producto debe ser mencionada explícitamente" (Stoitsev 2012).

3.6.6.2. Aggregation

"Define que tan detallado es el modelo de arquitectura con respecto al número de componentes, en su más alto nivel de agregación el sistema es solo una caja negra mientras que en su nivel más bajo todos los subsistemas son identificados a detalle" (Stoitsev 2012).

3.6.6.3. Realization

"Define que tan cerca está el modelo de la implementación tecnológica del sistema. Las descripciones de arquitectura a lo largo de esta dimensión pueden ser categorizadas en el rango de modelos muy orientados al negocio y muy orientados a la tecnología" (Stoitsev 2012).

En (Stoitsev 2012) , (Grefen 2012) y (Stoitsev & Grefen 2012) se propone un modelo multidimensional para diseñar y organizar los modelos de arquitecturas de acuerdo con estas dimensiones (ver fig. 28).

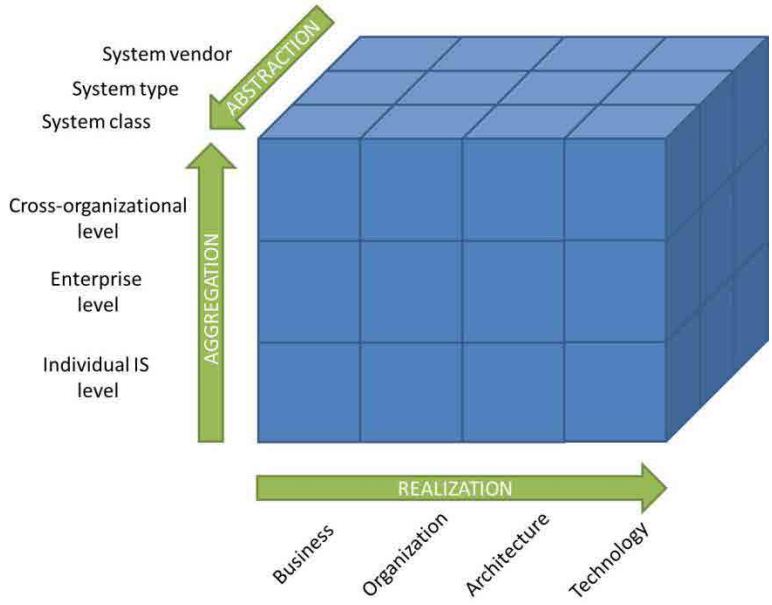

Figura 28: Espacio de Diseño Multidimensional, fuente (Stoitsev 2012).

Cada proyecto que tiene como propósito diseñar sistemas de información, de alguna manera tiene que pasar por algunas de las celdas que están entre la celda de inicio y final (ver fig. 29). Durante este recorrido, diferentes arquitecturas se crean en la forma de modelos, en el inicio del proceso se diseña un modelo altamente orientado al negocio, el cual es abstracto y con un alto nivel de agregación, mientras que al final se obtiene un modelo bien definido, orientado a la tecnología y con todas las especificaciones acerca de la versión del software y sus subsistemas (Stoitsev & Grefen 2012).

De acuerdo con Grefen (2012), para poder llegar al final, se debe tomar la decisión de personalizar la arquitectura con respecto a la tecnología, se debe tomar la decisión de escoger una tecnología específica para el sistema de información que se está diseñando. Esto se hace normalmente en la segunda mitad de la dimensión de realización, cuando los requerimientos del negocio aun no son dependientes de la tecnología (Stoitsev & Grefen 2012).

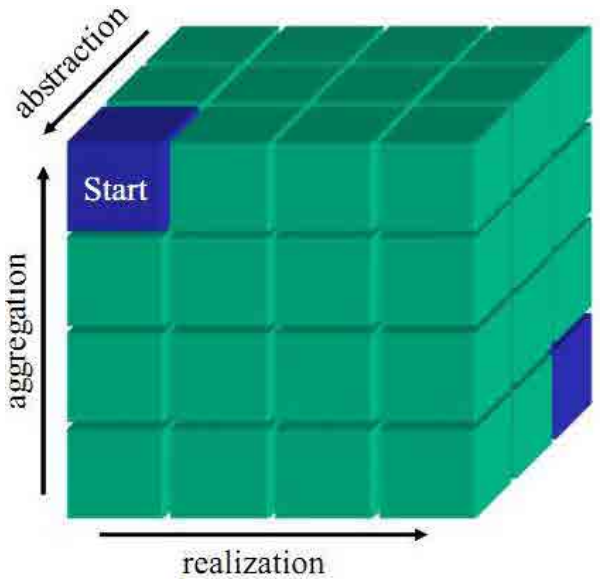

Figura 29: Recorrido del proceso de diseño de la arquitectura de un sistema de información, fuente (Stoitsev & Grefen 2012).

3.6.7. Principales Patrones de Arquitecturas

3.6.7.1. Service Oriented Architecture (SOA)

Esta arquitectura permite que las funcionalidades de un sistema, sean ofrecidas en forma de servicios, los servicios utilizan interfaces basadas en estándares, que permiten que los servicios puedan ser descubiertos, publicados e invocados. SOA permite empaquetar procesos de negocios en servicios que son operativos entre ellos, utilizando protocolos y formatos de datos para poder comunicarse e intercambiar información. Clientes y otros servicios pueden acceder a los servicios locales o acceder a servicios remotos a través de una conexión de red (*Microsoft* Patterns & Practices 2009).

Los principales beneficios de una arquitectura SOA son:

- Reusar los servicios más comunes con interfaces estándar, permite incrementar las oportunidades tecnológicas y de negocios.

- Permite un alto nivel de abstracción, debido a que los servicios son autónomos y accedidos a través de un contrato formal.

- Alta disponibilidad para que los servicios puedan ser descubiertos, los servicios pueden ofrecer descripciones que permiten a otros servicios y aplicaciones localizarlos y automáticamente determinar el tipo de interfaz.

- Interoperabilidad, los protocolos y formatos de datos están basados en estándares y permite que los proveedores y consumidores de servicios estén implementados en diferentes plataformas.

- Permite racionalizar los recursos, los servicios pueden ser granulares y de esta forma eliminar la duplicidad de funcionalidades, cada servicio puede proveer una funcionalidad específica.

3.6.7.2. Client/Server Architecture

Esta arquitectura es utilizada en sistemas distribuidos y se divide en dos aplicaciones: sistemas que hacen de cliente y sistemas que hacen de servidor. Los clientes hacen peticiones y consultas al servidor, las aplicaciones se comunican a través de una red de comunicaciones. La representación más simple esta arquitectura es una sola aplicación de servidor que es consultada por múltiples clientes, se le conoce como arquitectura 2-*Tier*.

La información de la aplicación y la lógica de negocios residen en el servidor lo que hace que la escalabilidad del servidor sea complicada, la dependencia de un servidor centralizado afecta la confiabilidad del sistema en caso de haber fallos en el servidor (*Microsoft* Patterns & Practices 2009), las ventajas de esta arquitectura son:

- Un alto nivel seguridad, debido a que los datos son almacenados en el servidor, el cual siempre está protegido contra el acceso indebido o ataques maliciosos. Versus los sistema clientes que cuentan con un nivel bajo de seguridad y control.

- El acceso a datos es centralizado, la administración y actualización de la información es más sencilla que en otras arquitecturas, debido a que toda la información reside en el servidor.

3.6.7.3. Component-Based Architecture

Esta arquitectura está centrada en la descomposición de un sistema en componentes individuales, ya sean lógicos o funcionales, e incluyendo interfaces de comunicación que contienen métodos, eventos y propiedades. Los componentes son diseñados para ser reutilizados en diferentes escenarios y aplicaciones, y se pueden sustituir por otros componentes que tiene funciones similares, son diseñados para operar en diferentes contextos y ambientes, pueden ser combinados con otros componentes para ofrecer nuevos comportamientos, y son diseñados para depender lo menos posible de otros componentes (*Microsoft* Patterns & Practices 2009).

Los principales beneficios de esta arquitectura son:

- Es fácil de implementar, se pueden reemplazar componentes sin afectar otros componentes o a todo el sistema.

- El uso de componentes hace que la complejidad sea mínima.

- El uso de componentes con interfaces que tienen una funcionalidad definida permite que se desarrollen otros componentes sin afectar otras partes del sistema.

3.6.7.4. Layered Architecture Architecture

Esta arquitectura se centra en agrupar las funcionalidades de un sistema en distintas «capas» que son apiladas verticalmente una encima de la otra. La funcionalidad de cada una de las capas está asociada a un rol o responsabilidad. En este tipo de arquitectura cada una de las capas se comunica con la capa que está justo debajo para asignarle responsabilidades; cada capa debe saber cómo resolver los requerimientos hechos por la capa que se encuentra arriba de ella y si es necesario pasarlos a la capa por debajo de ésta.

Las capas de un sistema pueden estar en el mismo ordenador o distribuidas en diferentes ordenadores y en lugares físicos diferentes, utilizando interfaces de comunicación para poder estar en contacto entre ellas. Un ejemplo de esta arquitectura puede ser un sistema *Web* que consta de tres capas: de presentación la cual está relacionada con la interfaz, de negocios la cual contiene toda la funcionalidad de los procesos lógicos de negocios, y la de datos que se encarga de las funcionalidades del acceso y recuperación de datos (*Microsoft* Patterns & Practices 2009).

Los principales beneficios de esta arquitectura son:

- Las capas están separadas tecnológicamente lo que permite hacer cambios en una capa sin afectar todo el sistema.

Se puede separar las funcionalidades más importantes en diferentes capas lo que ayuda a identificar dependencias y organizar las capas de forma que sean más accesibles. Distribuir las capas en diferentes ordenadores mejora la escalabilidad y la tolerancia a fallos.

- Asignar un rol funcional a cada una de las capas permite que puedan ser reutilizadas.

3.6.7.5. N-Tier y 3-Tier Architecture

En este tipo de arquitecturas las funcionalidades están separadas por segmentos de forma parecida a como se hace en la arquitectura por capas, cada segmento puede ser alojado en sistemas que se encuentran en lugares físicos diferentes. Una arquitectura N-*Tier* tiene al menos tres niveles, separados lógicamente y en diferentes ordenadores físicos.

Estas arquitecturas proveen una mejor escalabilidad, disponibilidad, gestión y utilización de recursos; debido a que hace una implementación distribuida y una descomposición funcional tanto de sus aplicaciones como componentes de servicio. Cada nivel (*Tier*) es completamente independiente de los niveles que no están inmediatamente por arriba o por debajo de ella; cada capa debe saber cómo resolver los

requerimientos hechos por la capa que está arriba de ella y si es necesario pasarlos a la capa que está debajo de ella, la comunicación entre los niveles casi siempre es de manera asíncrona para tener una mejor escalabilidad.

Un ejemplo es una aplicación *Web* 2.0, la capa de presentación es implementada en la máquina del cliente y la capa de negocios y de datos son implementadas en uno o más servidores (*Microsoft* Patterns & Practices 2009).

Los principales beneficios de esta arquitectura son:

- Su mantenimiento y escalabilidad es relativamente fácil debido a que cada nivel es independiente, hacer cambios o actualizaciones sin afectar los otros niveles es posible.

- Flexibilidad, cada nivel es manejado o escalado independientemente.

- La disponibilidad aumenta debido a que permite utilizar fácilmente componente que se pueden escalar sin problemas y sin afectar otros componentes.

3.7. Gestión de Procesos de Negocios

La gestión de procesos de negocios mediante el uso de sistemas de información tiene sus orígenes en la década de los setenta; desde entonces mucho esfuerzo se ha hecho por realizar electrónicamente, de una forma completa la gestión y manejo de procesos de negocios, pero ha resultado ser una tarea difícil debido a que hasta el día de hoy hay tareas que solo los humanos pueden realizar cuando se trata de procesos de negocios (Aalst & Hee 2004).

3.7.1. Definición de Proceso de Negocios

Un «proceso de negocios» se puede definir como:

> "Un proceso que se centra en la producción de productos determinados, estos productos pueden ser físicos como por ejemplo un avión o un puente, o productos menos tangibles como por ejemplo un diseño, la consulta de un documento o una evaluación. En otras palabras el producto pude ser también un servicio" (Aalst & Hee 2004). En forma más resumida se puede definir como "un conjunto de tareas que están relacionadas lógicamente y que son ejecutadas para alcanzar un resultado de un negocio específico" (Davenport & Short 1990).

El término proceso puede ser aplicado a todo tipo escenarios que se pueden dar en un trabajo, hay diferentes tipos de trabajos como por ejemplo comprar artículos en el supermercado, construir una casa, hornear una pizza, en cada uno de estos trabajos hay un resultado tangible que es producido o modificado y al que se le puede dar el nombre de «caso». Un caso no requiere específicamente dar como resultado un objeto tangible ya que también puede ser algo más abstracto como por ejemplo, hacer una transacción bancaria. Cada uno de estos casos tiene un inicio y un final, es diferente y se puede distinguir de otros casos. Por naturaleza propia es un proceso que tiene que ser ejecutado, el proceso consiste en un número de tareas que deben ser llevadas a cabo y una serie de condiciones que determinan el orden en que son realizadas las tareas; a un proceso se le conoce también como procedimiento. Una tarea es la unidad lógica de trabajo que es llevada a cabo en su totalidad por un recurso. Un recurso es el nombre genérico que se le da a una persona, máquina, grupo de personas o un grupo de máquinas que pueden realizar una tarea (Aalst & Hee 2004).

3.7.2. Definición de Workflow

El término «*workflow*» es usado como sinónimo de proceso de negocios y se puede definir como "la automatización de un proceso de negocios, en parte o en su totalidad, en donde, documentos, información y tareas son pasados de un participante a otro para realizar una acción, de acuerdo con un conjunto de reglas y procedimientos para alcanzar o contribuir con todas las metas del negocio" (Hollingsworth 1995). Un

workflow representa las características funcionales y operacionales de cada uno de los procesos de una organización, indicando el orden en que la actividades deben ser ejecutadas y quien debe ejecutarlas, también representa el flujo de la información utilizado para dar soporte a todas las actividades definidas en el proceso, y los mecanismos para la monitorización que miden y controlan el proceso (Yang 2000).

3.7.3. Clasificación de *Workflows*

Los *workflows* pueden ser clasificados por la complejidad y estructura de la tarea que deben efectuar, otra forma de clasificarlos puede ser agrupándolos, tomando en cuenta las similitudes de los procesos de negocios y el valor que tiene en las empresas (ver fig 30). Los *workflows* pueden agruparse en cuatro categorías: Administrativos, *ad-hoc*, Colaborativos y de Producción (Alonso et al. 1997).

Otra forma de clasificación puede ser en base a la tecnología a la que están ligados, las tres categorías más importantes son, *workflows* centrados en e-mail, centrados en documentos y centrados en procesos (Alonso et al. 1997).

3.7.3.1. *Workflows* Administrativos

Los *workflows* administrativos tienen la característica de tener pasos que ya han sido establecidos de antemano y las reglas son conocidas por todos los que participan en ellos, este tipo de *workflows* son procesos burocráticos, como por ejemplo pagar una multa de tráfico o realizar la matrícula en la universidad.

3.7.3.2. Workflows ad-hoc

Los *workflows* ad-hoc son similares a los administrativos, con la excepción de que son hechos a la medida de procesos que son únicos o cuando se tiene la necesidad de hacer una excepción, los *workflows* hechos a la medida dependen mucho de los usuarios que hacen uso de ellos.

3.7.3.3. Workflows Colaborativos

Un *workflow* Colaborativo se caracteriza por su número de participantes y la interacción entre ellos, casi siempre, en cada uno de los pasos del *workflow* se dan una serie de iteraciones sobre el mismo paso hasta que se alcanza un nivel de satisfacción o consenso, o puede que se genere la necesidad de regresar a un paso anterior para volver a empezar nuevamente; un ejemplo claro es el trabajo de varios autores en la escritura de un mismo documento.

3.7.3.4. Workflows de Producción

Los *workflows* de producción son procesos de negocio relacionados con funciones críticas de las organizaciones en las que son implementados, su característica principal es que son ejecutados en gran escala por lo que se necesitan herramientas de monitorización que permitan análisis estadísticos de la ejecución y estado de los procesos (Alonso et al. 1997).

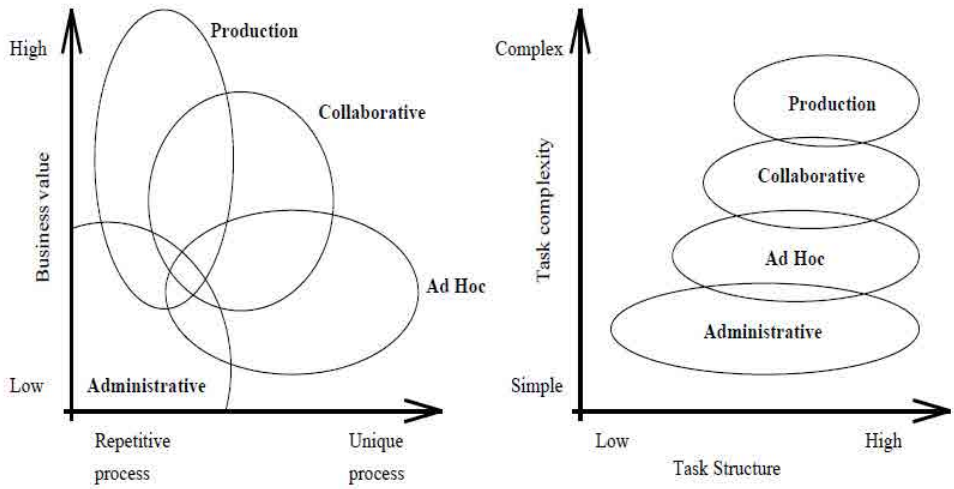

Figura 30: Clasificación de *Workflows*, fuente: (Alonso et al. 1997).

3.7.4. Sistemas de Gestión de Workflows

La función principal de un sistema de gestión de *workflows* es asegurarse que las actividades adecuadas son ejecutadas por los servicios correctos en el tiempo correcto (Bala & Chana 2012), de la forma más efectiva y eficiente como sea posible (van der Aalst & Kumar 2001). Un sistema de gestión de *workflows* se encarga de asegurar que la información correcta llegue a la persona correcta en el tiempo correcto, o que la información sea enviada a la aplicación correcta en el momento adecuado. El sistema de gestión de *workflows* no realiza ninguna de las tareas que son parte del proceso (Aalst & Hee 2004). Un sistema de gestión de *workflows* cuenta con herramientas necesarias para diseñar, modelar y definir, procesos y *workflows,* incluyendo las actividades que lo componen. Provee un ambiente operacional en el cual los *workflows* (procesos) son ejecutados, manejando las actividades de cada uno de los procesos en forma secuencial. Cuenta con las interfaces necesarias para que los usuarios y aplicaciones puedan interactuar con el sistema, con la finalidad de procesar cada una de las actividades (Hollingsworth 1995; Alonso et al. 1997).

3.7.4.1. Funcionalidades de un Sistema Gestor de Workflows

En la figura 29 se pueden ver las principales características funcionales desde un punto de vista global, el Sistema de Gestión da soporte a tres áreas funcionales (Hollingsworth 1995).

- Funciones en tiempo de construcción, estas tienen que ver con la definición y el modelado de los procesos de *workflows* y las actividades de las que están compuestos.

- Funciones de control en tiempo de ejecución, estas tienen que ver con la gestión de los procesos de *workflows* en un ambiente operacional y la secuenciación de las diferentes actividades que son manejadas como parte de cada proceso.

- Interacciones en tiempo de ejecución con usuarios y aplicaciones de software para el procesamiento de actividades y pasos.

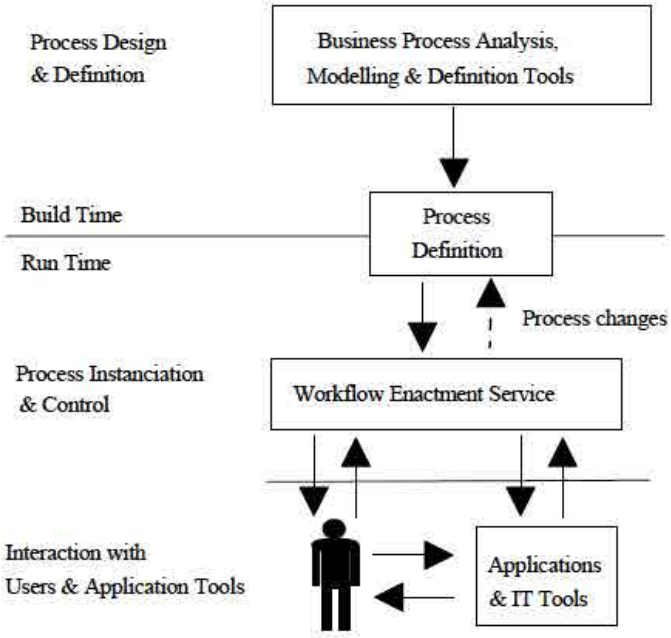

Figura 31: Principales características de un sistema gestor de *Workflows*, fuente (Hollingsworth 1995).

En la fase de tiempo de construcción (*Build time*):

1. Los procesos de negocios son transformados en un formato computacional que sea legible para sistema de gestión de *workflows* y de esta manera poder procesarlo.
2. La forma de hacer esta conversión es mediante el uso de herramientas de modelado, análisis y definición de procesos.
3. El producto que resulta de esta fase se le da el nombre de “definición del proceso”.
4. La definición del proceso contiene el número de pasos y actividades que se tienen que llevar a cabo para completar el proceso, así como la relación de recursos humanos y computacionales que van a realizar cada una de las actividades y las reglas que van a gobernar.
5. También contiene la forma en que se llevan a cabo las tareas, la forma en que es representado el proceso de definición puede ser en forma textual, gráfica o en un lenguaje de notación.

En la fase de tiempo de ejecución (*Run Time*):

1. Varias funcionalidades del sistema se encargan de controlar que la definición del proceso sea interpretada.
2. En base a ésta interpretación crear una instancia operacional del proceso y despachar las actividades que lo componen en el tiempo correcto.
3. Asegurándose que los recursos adecuados realicen las tareas que les han sido asignadas a cada una de las actividades, invocando los recursos humanos o informáticos para que realicen la tarea según sea el caso.

3.7.4.2. Terminología básica en un Sistema Gestor de Workflows

En la figura 32 se muestra la relación términos básicos, la definición de un proceso (*process definition*) es la representación un proceso de negocios (*business process*) en un formato en el que puede ser manipulado automáticamente y de esta forma poder ser modelado o interpretado por un sistema gestor de *workflows*. La definición de un proceso está compuesta de actividades (*activities*), el orden temporal en que estas actividades deben ser ejecutadas con la información de cada una de las actividades, como por ejemplo los participantes, las aplicaciones y datos asociados a la actividad.

Una actividad es cada uno de los pasos que conforman el proceso, las actividades son asignadas a usuarios que tienen la capacidad de ejecutarlas (*manual activities*) o son asignadas a aplicaciones de *software* para que sean ejecutadas de forma automatizada (*automated activities*).

La instancia de un proceso (*process instance*) es la representación informática de un proceso previamente definido, el sistema gestor se encarga de interpretar la definición del proceso para poder crear y ejecutar la instancia, una vez que la instancia del proceso está en estado de ejecución, el sistema gestor va creando y ejecutando cada una de las instancias de las actividades (*activity instances*), siguiendo la lógica de ejecución de las tareas, conforme fue indicado en el proceso de definición.

Cada una de las instancias de las actividades es asignada cómo un ítem de trabajo (*work item*) al usuario que ha sido elegido para realizar la tarea o en su defecto a la aplicación

que se encarga de realizar la tarea automáticamente (Ferreira 2009; Hollingsworth 1995; Alonso et al. 1997).

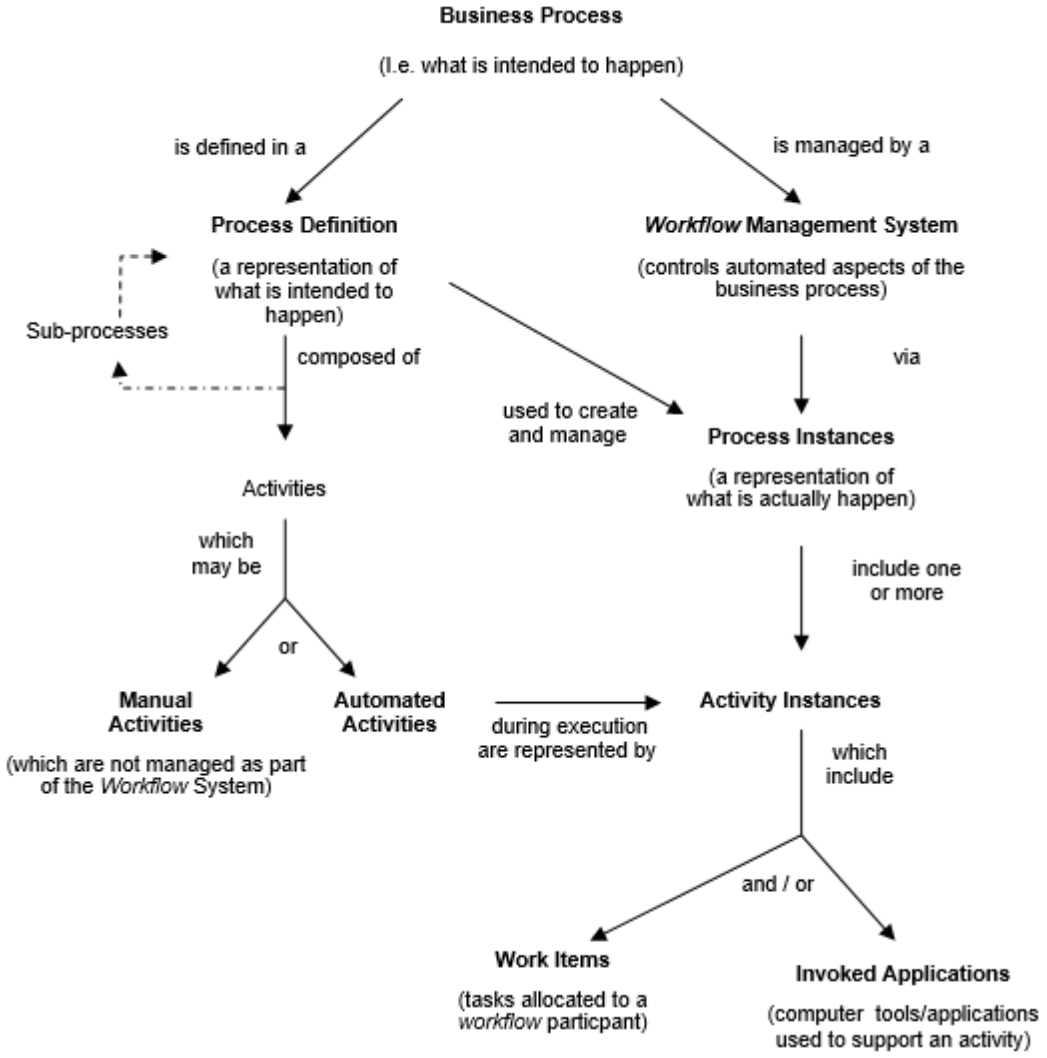

Figura 32: Relación entre las terminologías básicas de *Workflows*, fuente (Ferreira 2009).

3.7.5. Definición de Procesos

La definición de un proceso contiene toda la información necesaria acerca del proceso, esto va a permitir que pueda ser interpretado y ejecutado por el sistema de interpretación de *workflows* (Hollingsworth 1995), la definición del proceso incluye:

- Información de las condiciones que se deben dar para que el proceso pueda ser iniciado y finalizado.

- Las actividades que son parte del proceso y las reglas para ir avanzando entre ellas.

- Las tareas que los usuarios deben ejecutar y la información del tipo de usuario que la debe ejecutar.

- Referencias de las aplicaciones que deben ser invocadas.

- Información relevante que pueda ser necesaria para el *workflow.*

En la figura 31 se muestra el meta-modelo de la definición de un proceso donde se pueden identificar los elementos básicos que debe tener un proceso de definición.

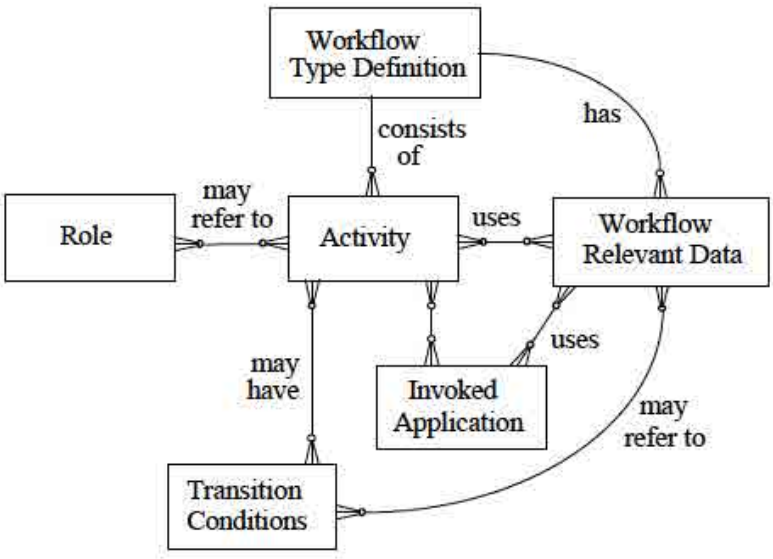

Figura 33: Meta-modelo de la Definición de un Proceso propuesto por WfMC (1995).

3.8. Modelo de Referencia de Sistemas Gestores de *Workflows*

La WfMC (*workflow management coalition*)[51] es una organización que se encarga de crear y contribuir con estándares relacionados con procesos de negocios. La organización está formada por desarrolladores, consultores, analistas, universidades y grupos de investigación que trabajan con *workflows* y la gestión de procesos de negocios. La WfMC propuso en 1995 un modelo de referencia para estandarizar la implementación de sistemas de gestión de *workflows* (Hollingsworth 1995), es una descripción general de la arquitectura que debe tener un sistema de gestión de *workflows* (ver fig. 34), describe las funcionalidades de los componentes de software que son parte esencial en un sistema gestor de *workflows* y la forma en que deben interactuar entre ellos. El modelo ha sido desarrollado de manera neutral en cuanto a tecnología se refiere, lo que permite al modelo ser independiente de cualquier arquitectura en particular o implementación tecnológica.

[51] http://www.wfmc.org/

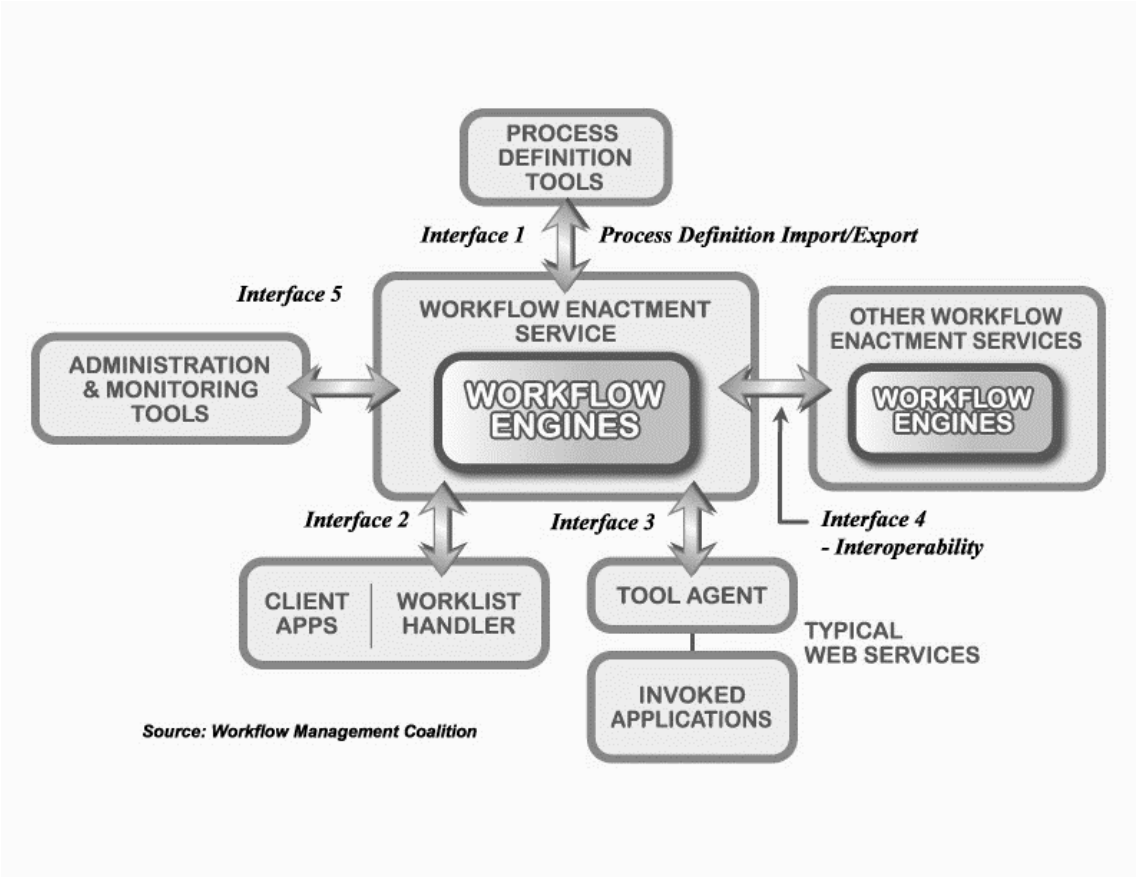

Figura 34: Modelo de Referencia de un Sistema Gestor de *Workflow*s, fuente (WFMC, 1995).

3.8.1. Workflow Enactment Service

El servicio de interpretación de *workflows* es la parte central del sistema, este módulo de software se encarga de la creación, manejo y ejecución de las instancias de *workflows* que han sido definidas con antelación. Para realizar su tarea el servicio de interpretación utiliza motores (*Engines*), un servicio de interpretación puede hacer uso de uno o varios motores para mejorar la escalabilidad del sistema y distribuir el trabajo entre ellos; dependiendo del grado de complejidad y características de los procesos y tareas un *workflow* puede ser distribuido en varios motores. Las aplicaciones funcionales del modelo se comunican con el servicio de interpretación a través de una interfaz de programación de aplicaciones para *workflows* (WAPI). Un Servicio de Interpretación de *workflows* pude estar físicamente centralizado o distribuido funcionalmente (Aalst & Hee 2004; Hollingsworth 1995; Meilin et al. 1998).

3.8.1.1. Workflow Engine

Un «motor» de *workflows* es el núcleo de un sistema de gestión, provee un entorno de ejecución y control para que los procesos previamente definidos puedan ser creados, interpretados y activados. Logísticamente provee las facilidades para que los procesos puedan ser completados. Un motor provee las siguientes funcionalidades (Aalst & Hee 2004) (Hollingsworth 1995):

- Interpretación de la definición de procesos.
- Control de los procesos (creación, activación, suspensión, finalización, etc).
- Controlar el flujo y orden en que se deben dar las actividades de los procesos.
- Acciones de supervisión (control, administración y auditoria).
- Autenticación de participantes.
- Entregar los ítems de trabajo a los recursos correctos.
- Iniciar una aplicación de software durante la ejecución de una actividad.
- Registro de datos históricos.
- Proveer un resumen del *workflow* y monitorear su consistencia.
- Identificar los ítems de trabajo para cada usuario y proveer una interfaz para permitir que los usuarios interactúen con los ítems.

3.8.1.2. Estados de Transición de los Procesos

Se puede decir que el servicio de interpretación de *workflows* es una máquina que se encarga de manejar logísticamente los estados y la transición de las instancias de procesos y actividades, los estados de los procesos y sus transiciones pueden cambiar

debido a factores externos y que no tienen nada que ver con el servicio de interpretación como por ejemplo el que una actividad fue completada, o a factores internos como por ejemplo decisiones de control ejecutadas por el motor de *workflows* (Hollingsworth 1995).

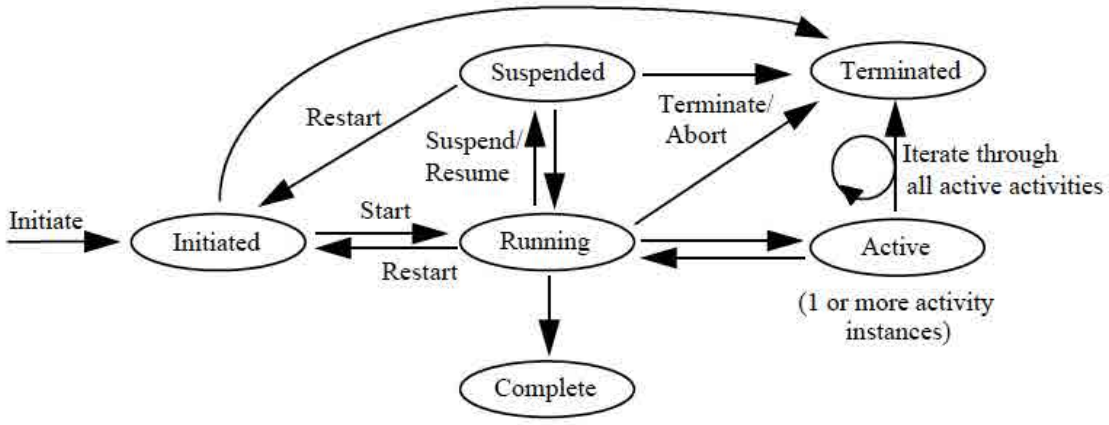

Figura 35: Estados de Transición para la Instancia de un Proceso, fuente (Hollingsworth 1995).

En la figura 35 se muestran los estados y las transiciones que puede tener la instancia de un proceso, las transiciones entre los estados están representadas por las flechas, los seis estados que puede tener la instancia de un proceso son (Hollingsworth 1995):

- *Initiated*: la instancia del proceso ha sido creada pero aún no cumple con las condiciones para poder ser ejecutada.

- *Running*: la instancia del proceso empieza a ejecutarse, una de las actividades del proceso deberá iniciar.

- *Active*: una o más actividades dentro del proceso han iniciado.

- *Suspended*: la instancia del proceso se pone en estado de suspensión y las actividades del proceso no se inician hasta que el proceso no sea puesto en estado de ejecución nuevamente.

- *Completed*: todas las actividades de la instancia del proceso han sido completadas y el proceso cumple con las condiciones para ser finalizado, la instancia del proceso deja de ejecutarse en el motor de *workflows* y es destruida.

- *Terminated*: la ejecución de la instancia del proceso ha sido detenida antes de que finalizara de forma correcta, la instancia del proceso es destruida.

3.8.1.3. Estado y Transición de las Actividades

En la figura 36 se muestran los Estados y las Transiciones que puede tener la instancia de una Actividad. Una vez que una actividad ha sido iniciada no puede ser suspendida o terminada, a continuación se explican los cuatro estados que puede tener una actividad (Hollingsworth 1995).

- *Inactive*: la actividad ha sido creada pero aún no ha sido activada, hasta no ser activada no podrá tener asignado un ítem de trabajo.

- *Active*: la actividad pasa a estar activa, se crea un ítem de trabajo y es asignado a la instancia de la actividad para que esta pueda ser realizada.

- *Suspended*: la actividad es suspendida y no se le puede asignar un ítem de trabajo hasta que no sea puesta en ejecución nuevamente.

- *Completed*: la ejecución de la instancia de la actividad se ha completado, el motor de *workflows* evalúa cual es la siguiente actividad a ejecutar y toma decisiones de control con respecto a la instancia del proceso.

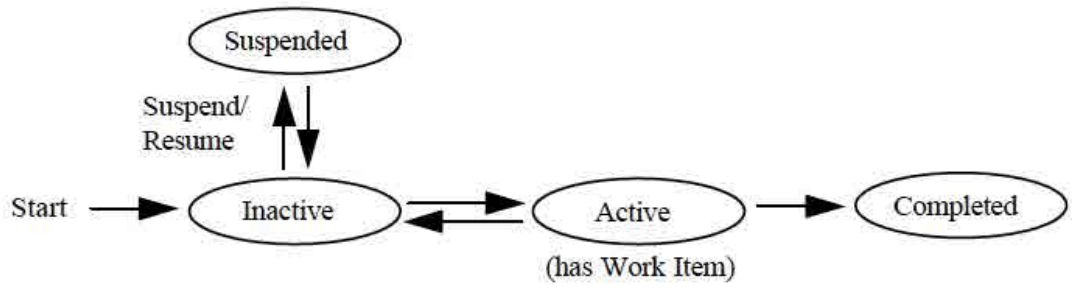

Figura 36: Estados de Transición para la Instancia de una Actividad, fuente (Hollingsworth 1995).

3.8.2. Interfaz de Programación de Aplicaciones para Workflows (WAPI)

Los módulos del sistema gestor, aplicaciones y recursos, pueden interactuar con el servicio de interpretación de *workflows* mediante una interfaz de Programación de Aplicaciones (WAPI), un sistema gestor está compuesto de varios módulos, estos deben intercambiar información para poder trabajar juntos, el modelo de referencia propuesto por la WfMC propone cinco interfaces (ver figura 34), que pueden tener comunicación con el sistema de interpretación de *workflows* mediante el uso de una WAPI. La WAPI se compone de funciones con parámetros y formatos de información preestablecidos y estandarizados, con la finalidad de permitir el intercambio de información relacionada con los *workflows* entre los diferentes módulos del sistema (Aalst & Hee 2004; Hollingsworth 1995).

3.8.2.1. Información de Control de un Workflow

Es información interna que sirve para el control de las instancias de los *workflows* que se están ejecutando, como por ejemplo identificar el estado en que se encuentra un proceso o una actividad dentro del sistema de interpretación. Esta información solo es utilizada por el sistema de interpretación o por los motores y no es compartida con los otros módulos del sistema (Hollingsworth 1995).

3.8.2.2. Información Relevante de un Workflow

Es información que está relacionada con todo lo que tiene que ver con el estado o transición en que se encuentra la instancia de un *workflow*, esta información es usada por el sistema gestor para saber cuál es la siguiente actividad que debe ser ejecutada (Hollingsworth 1995).

3.8.2.3. Información de las Aplicaciones del Workflow

Esta información es creada o utilizada por las aplicaciones con las que interactúa el *workflow* dentro de sus actividades o tareas, la información puede ser manipulada directamente por la aplicación o con algún tipo de interacción entre el usuario y la aplicación. Esta información no es relevante o de utilidad para el sistema gestor (Hollingsworth 1995).

3.8.3. Process Definition Tools – Interface 1

Las herramientas de definición de procesos sirven para tres tareas diferentes, la definición de procesos, la clasificación de recursos y el análisis de procesos. En la mayoría de los sistemas gestores estas tres herramientas están integradas en una sola y se le conoce como herramienta de análisis y definición de *workflows.* Este tipo de herramientas pueden ser desde herramientas informales utilizando una hoja de papel y un bolígrafo hasta herramientas altamente sofisticadas. El producto final de este proceso de modelado y diseño se le conoce como "proceso de definición" el cual va a ser interpretado en tiempo de ejecución por el motor de *workflows* dentro del servicio de interpretación.

Estas herramientas se comunican con el sistema de interpretación utilizando la "*Interface* 1" (ver fig. 34), utilizando funciones de la WAPI que sirven para establecer la conexión y desconexión, obtener un resumen de las definiciones de *workflows*, abrir, crear y guardar definiciones de procesos (Hollingsworth 1995; Meilin et al. 1998; Aalst & Hee 2004).

3.8.3.1. Herramienta de Definición de Procesos

La funcionalidad básica de un herramienta de definición de procesos son las siguientes (Aalst & Hee 2004):

- Establecer definiciones de procesos (nombre, descripción, fecha, componentes, versión, etc.).

- Modelar lógicamente el orden en que se van a ejecutar las actividades.

- Dar soporte para la gestión de diferentes versiones de la definición del proceso.

- Realizar la especificación de las tareas y atributos que debe tener el proceso.

- Verificar que la sintaxis de la definición del proceso sea la adecuada y poder detectar inconsistencias.

En cuanto a la «definición de tareas», la herramienta ofrece las siguientes funcionalidades para determinar las condiciones que debe llevar a cabo la tarea y las operaciones se necesitan ejecutar (Aalst & Hee 2004):

- Dar nombre y descripción a la tarea.

- Dar información relacionada con la tarea e instrucciones para que el recurso (usuario o aplicación) pueda realizar la tarea.

- Establecer los requisitos que debe tener el recurso que va a realizar la tarea

- Establecer las característica de la ruta y el orden que van a seguir las tareas.

- Especificar las aplicaciones que va a ser ejecutadas, las condiciones y el orden en que se deben ejecutar.

- Especificar reglas de decisión que van a determinar las tareas que van a ser ejecutadas posteriormente.

3.8.3.2. Herramientas de Clasificación de Recursos

Los recursos que van a realizar las tareas de las actividades de los *workflows* necesitan ser «clasificados», de esta forma las tareas pueden ser asignadas a usuarios, que cuentan con las características y las necesidades que requiere la tarea para ser llevada a cabo, la mayoría de los sistemas gestores cuentan con una herramienta de clasificación de recursos, que muestra la relación de las diferentes clases de recursos en forma gráfica. Los ítems que son más comunes para ser mostrados en una herramienta gráfica son: una lista de las «clases de recursos», que a su vez son subdivididos en «roles» (basados en funciones, habilidades y perfiles) y unidades orgánicas (basados en equipos, departamentos y ramos). Cualquier característica especifica de la clase del recurso y la relación entre las diferentes clases de recursos (por ejemplo jerarquía de roles o unidades orgánicas) (Aalst & Hee 2004).

3.8.3.3. Herramientas de Análisis de Workflows

Estas herramientas permiten que se pueda analizar semánticamente el proceso de definición y de esta forma evitar incongruencias que conlleven al mal funcionamiento del *workflow,* también permiten hacer una simulación para avaluar el tiempo que va a llevar el completar todo el proceso (Aalst & Hee 2004).

3.8.4. Workflow Client Applications – Interface 2

Los usuarios que están involucrados en la ejecución de un proceso, específicamente en las actividades que requieren recursos humanos, entran en contacto con el sistema gestor de *workflows* a través de las «aplicaciones para clientes»; éstas herramientas pueden estar incluidas en el sistema gestor o pueden ser desarrolladas por terceros, la forma de entrar en contacto es a través de una interfaz conocida como manejador de listados de trabajo (*Worklist Handler*), la interfaz está basada en el concepto de listado de trabajos, este listado contiene los ítems de trabajo asignados por el motor de *workflows* a un usuario en particular. Las aplicaciones para el cliente utilizan una serie

de funciones de la WAPI para comunicarse con el servicio de interpretación a través de la *Interface* 2 (ver apartado 3.8), estas funciones contienen comandos para conectarse y desconectarse, comandos de control para las actividades y procesos, comandos para saber el estado de los procesos y actividades, y comandos para la manipulación de los listados de trabajos (Hollingsworth 1995; Meilin et al. 1998). Las funcionalidades básicas ofrecidas por un *worklist handler* son las siguientes (Aalst & Hee 2004):

- La presentación de ítems de trabajo que van a ser realizados por el usuario.
- Proveer propiedades relevantes del ítem de trabajo.
- La habilidad de ordenar y seleccionar ítems de trabajo basándose en sus propiedades.
- Informar al usuario cuando una tarea ha iniciado o finalizado.
- Permitir al usuario reportar que ha terminado una actividad.
- Permitir al usuario bloquear o no aceptar un ítem de trabajo.

3.8.5. Invoked Applications – Interface 3

Las aplicaciones invocadas (*Invoked Applications*), son utilizadas por el servicio de interpretación para llevar a cabo una actividad dentro de un *workflow*, toda la información que se necesita para invocar estas aplicaciones, fue subministrada en el proceso de definición, incluyendo la información que la aplicación necesita para realizar la actividad; éstas aplicaciones no pertenecen al sistema gestor de *workflows* porque son aplicaciones que están asociadas con la realización de la tarea, pueden estar ejecutándose en la misma plataforma que el sistema gestor o pueden estar en una plataforma remota accesible a través de *Internet.*

Las aplicaciones se pueden distinguir entre aplicaciones interactivas y automatizadas, las primeras son iniciadas por el usuario al seleccionar un ítem de trabajo del listado de

tareas, estas herramientas son aplicaciones estándar como por ejemplo un procesador de textos, una hoja de cálculo o una aplicación desarrollada específicamente para el proceso de negocios. Las aplicaciones automatizadas no necesitan intervención del usuario para iniciar su ejecución ya que son iniciadas directamente por el servicio de interpretación a través de la "*Interface* 3" (ver apartado 3.8), tampoco necesitan interacción con el usuario para realizar sus tareas (Hollingsworth 1995; Meilin et al. 1998; Aalst & Hee 2004).

3.8.6. Other Workflow Enactment Services – Interface 4

El modelo de referencia ofrece la posibilidad de ligar varios «sistemas gestores», para que puedan colaborar entre ellos y de esta forma, poder transferir procesos o parte de ellos para ser ejecutados en otro sistema gestor, un intercambio de información estandarizado es necesario entre lo sistemas, esa es la función de la "*Interface* 4" (ver apartado 3.8) y se le conoce con el nombre de interfaz de interoperabilidad (Meilin et al. 1998; Aalst & Hee 2004).

3.8.7. Administration and Monitoring Tools – Interface 5

Las herramientas de administración y monitorización ayudan a ejecutar tareas administrativas y operacionales; como por ejemplo, la gestión y auditoria de roles y usuarios, supervisión de instancias de procesos y actividades, implementación de nuevos procesos de definición y la reconfiguración del sistema, éstas herramientas permiten solucionar problemas que son resultado del fallo del sistema y solucionar cuellos de botella en los procesos. Las herramientas de monitorización sirven para recolectar datos del desempeño que tienen los *workflows* en tiempo de ejecución, toda esta información histórica es importante para la gestión de *workflows*, la información es proveída por el servicio de interpretación en forma bruta, y son las herramientas de administración y monitorización, las que se encargan de interpretarlas y darles un formato, para ofrecer la información en forma de reportes a los usuarios. Estas herramientas se conectan con el servicio de interpretación a través de la "*Interface* 5" (ver apartado 3.8), haciendo uso de funciones implementadas en la WAPI para obtener información de monitorización y gestión (Meilin et al. 1998; Aalst & Hee 2004).

3.8.8. Sistemas de Gestión de Workflows en la Nube

La combinación de un sistema de gestión y servicios en la «nube» da como resultado a lo que se le conoce como un sistema gestor de *workflows* en la nube, en este tipo de sistemas, todos los componentes funcionales son implementados en diferentes infraestructuras y servicios en la nube, los componentes pueden estar separados de los motores, al ser alojados en diferentes infraestructuras (Liu et al. 2012). Un *workflow* en la nube consiste en una serie de tareas ejecutadas por un usuario o por un servicio en la nube, un flujo de control que especifica el orden de ejecución de las tareas y el flujo de datos que especifica los datos de entrada y salida para cada tarea (Minor et al. 2011).

3.8.8.1. Ventajas de un Sistema de Gestión Workflows en la Nube

Utilizando recursos «virtualizados» para la ejecución de *workflows*, en lugar de utilizar recursos tradicionales, no se tiene la necesidad de asegurar los recursos físicos con la finalidad de no ser atacados por códigos maliciosos, esto se logra mediante técnica de *sanboxing*[52]. La elasticidad de la nube permite cambiar la cantidad de los recursos y sus características en tiempo de ejecución, escalándose dinámicamente cuando hay una gran necesidad de utilizar recursos adicionales y disminuyendo el uso de recursos cuando no hay mucha demanda de uso del sistema gestor, esto permite que los sistemas gestores de *workflows* puedan ofrecer la calidad de servicio que sea requerida, al contrario de los sistemas gestores tradicionales que tienen la necesidad de reservar recursos de antemano para no verse limitados y poder ofrecer una buena calidad del servicio (Pandey et al. 2011; Mao et al. 2013).

Las herramientas, APIs y servicios ofrecidos por los proveedores de servicios en la nube son importantes para implementar sistemas de gestión, estos servicios facilitan la implementación, ejecución, monitorización y la escalabilidad de los Sistemas Gestores. Los proveedores de servicios en la nube como por ejemplo *Google App Engine*[53] y

[52] Es un ambiente de computación aislado y con restricciones por razones de seguridad, las máquinas virtuales son ejecutadas normalmente en este tipo de ambientes.
[53] https://appengine.google.com/start

Windows Azure[54] proveen plataformas donde se pueden desarrollar aplicaciones escalables e interactivas, esto hace que sea relativamente fácil portar los componentes funcionales de un sistema de gestión de *workflows* (Pandey et al. 2011).

3.8.8.2. Arquitectura de un Sistema de Gestión Workflows en la Nube

Por razones obvias es una adaptación de la arquitectura de referencia de *cloud computing* representada en capas (ver fig. 37), es prácticamente idéntica a la Arquitectura de cloud computing por lo que se puede traslapar una a la otra (Liu et al. 2012; Mao et al. 2013).

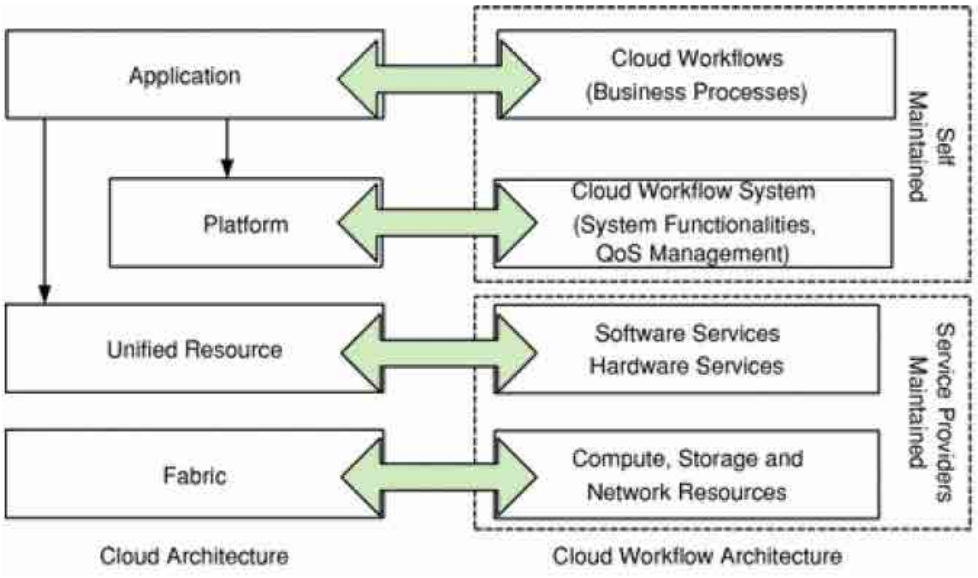

Figura 37: Arquitectura de un Sistema de *Workflows* en la Nube, fuente (Liu et al. 2012).

3.8.8.2.1. Capa de Aplicación

Consiste en aplicaciones de *workflows* en la nube, los usuarios que acceden a los servicios de esta capa lo hacen a través de aplicaciones *Web* con un estilo de servicio SaaS, el usuario no se tiene que preocupar por el mantenimiento del *software* ni de recursos de *hardware* que la aplicación debe utilizar (Liu et al. 2012; Mao et al. 2013).

[54] http://www.windowsazure.com/en-us/

3.8.8.2.2. Capa de Plataforma de Desarrollo

En esta capa es donde se encuentran los Sistemas Gestores de *workflows* en la Nube, todas las funcionalidades del sistema están incluidas en esta capa, los usuarios de esta capa son desarrolladores de aplicaciones de *workflows* que tienen acceso al ambiente de desarrollo que provee la plataforma, los servicios ofrecidos son de tipo PaaS (Liu et al. 2012; Mao et al. 2013).

3.8.8.3. Capa de Infraestructura

Es una réplica de la Capa de Infraestructura de *cloud computing*, consiste en recursos y servicios de software y de hardware que son requeridos para construir nuevos ambientes de desarrollo de *workflows* en la nube, y la ejecución de aplicaciones de *workflows*. En esta capa los recursos y servicios son ofrecidos en forma de máquinas virtuales (Liu et al. 2012; Mao et al. 2013).

3.8.8.4. Modelos de Implementación Sistemas Gestores de Workflows en la Nube

Existen tres modelos principales de implementación que son idénticos a los modelos de *cloud computing* (Pública, Privada, Híbrida) (Mao et al. 2013).

- Un sistema en la nube pública es la implementación más común, el sistema de gestión se ejecuta un recurso en la nube que es compartido con varios usuarios.

- Los sistemas de gestión en la nube privada permiten tener los beneficios de la nube sin tener que lidiar con las desventajas de una nube pública, en esta implementación se tiene un control total sobre cómo se maneja la información y sobre la gestión de la seguridad.

- En un sistema de gestión en la nube híbrida se pueden tener las ventajas de las dos implementaciones anteriores, se puede tener la información crítica del

sistema de gestión en la nube privada y utilizar la nube pública para situaciones en las que se hace un uso masivo de recursos.

3.8.8.5. Experiencias de Implementación del Modelo de Referencia de WfMC en la Nube

Las funcionalidades de un sistema de gestión de *workflows* en la nube pueden ser organizadas y diseñadas de la misma manera que en el modelo de referencia propuesto por la WfMC. La mayoría de los componentes funcionales del modelo de referencia pueden ser separados del motor de *workflows* para ser distribuidos y ejecutados en diferentes servicios en la nube, dependiendo del tamaño de carga de trabajo estos componentes pueden ser migrados o replicados a otros servicios en la nube o reforzados con recursos adicionales (Liu et al. 2012).

Weiping (2009) implementa todas las funcionalidades del Modelo de Referencia como un SaaS, escalando el número de motores de *workflows* conforme va aumentando el número de usuarios y procesos.

En (Zhao et al. 2012) y (Fei et al. 2011) se hace una implementación en la nube del Modelo de Referencia dividiéndolo en cuatro capas lógicas en base a sus funcionalidades.

- Capa de presentación, contiene las funcionalidades de diseño, presentación y visualización de *workflows*.

- Capa de gestión, en esta capa se encuentra el motor y la funcionalidad de monitorización.

- Capa de gestión de tareas, contiene las funcionalidades para la gestión de tareas de cada uno de los *workflows*.

- Capa operacional, contiene las herramientas de software que son invocadas para ejecutar las tareas y las fuentes de datos.

Zhao et al. (2012) propone varias formas de implementar en la nube, las cuatro capas lógicas en las que ha divido el modelo de referencia.

- Una solución es implementar solo la capa operacional en la nube, en este tipo de implementación el sistema gestor de *workflows* puede utilizar las aplicaciones en la nube como una categoría más de tareas que pueden ser ejecutadas dentro de un *workflow*.

- Otra forma de implementar el modelo en la nube, es poner la capa operacional conjuntamente con una o más de las otras capas, permitiendo la escalabilidad y elasticidad de cada una de las funcionalidades que están incluidas en cada una de éstas.

- Otra forma es implementar todas las capas del modelo en la nube, lo cual permite que el sistema gestor de *workflows* adquiera las ventajas y características de la computación en la nube para todas sus funcionalidades.

Capítulo 4

Diseño, Implementación y Evaluación del Modelo de WfMS

4. Diseño, Implementación y Evaluación del Modelo de WfMS

4.1. Introducción

En este capítulo Se explica la forma en que fueron diseñadas y moduladas las funcionalidades Móviles, *Cloud* y Colaborativas del modelo de WfMS y se hace una descripción de cada una de las funcionalidades del modelo. Se describe la forma en que se aplican los patrones de arquitectura junto con el modelo de referencia de WfMS para obtener el nuevo modelo de WfMS. Posteriormente se describe la forma en que se hace la implementación del modelo en un contexto específico y se describen las diferentes arquitecturas que resultan de está implementación. El resultado final de la implementación del nuevo modelo es una aplicación móvil de WfMS, se hace una descripción de las funcionalidades de la aplicación y la forma en que están relacionada con el modelo. En la parte final del capítulo se describe a detalle la experiencia y los resultados de la evaluación de las funcionalidades del modelo mediante un estudio *Delphi*.

4.2. Antecedentes

En la comunidad de investigación del LAM (Laboratorio de Aplicaciones Multimedia)[55], se planteó la implementación del «modelo de referencia» de WfMS (ver apartado 3.8) propuesto por Ferreira (2009); la cual dio como resultado un WfMS que fue incluido como parte de un Sistema de Gestión de Conocimiento y Comunidades de Aprendizaje llamado COLS[56]. De esta manera cualquier tarea dentro del sistema COLS podía ser llevada a cabo como un *Workflow*.

El sistema COLS se utilizó principalmente para la gestión de proyectos e investigaciones. Por lo que la labor gestora de *Workflows* dentro de esta comunidad fue en general para compartir tareas de investigación, de desarrollo de prototipos, de artefactos y la creación de contenidos de forma colaborativa. También se utilizaban *Workflows* como sistemas de validación de calidad en los productos e investigaciones realizados en la comunidad.

[55] El LAM es el lugar dónde se llevan a cabo las investigaciones del Doctorado en Ingeniería Multimedia de la Universidad Politécnica de Cataluña.
[56] COLS fue desarrollado por el grupo de investigación y desarrollo de software perteneciente al LAM.

Posteriormente se desarrolló una versión móvil del WfMS de la versión COLS completamente en HTML, siendo una adaptación de la versión *Web* de escritorio para ser utilizada en dispositivos móviles con un navegador *Web*. Posteriormente se agregó la funcionalidad para recibir notificaciones a través del sistema de mensajería SMS, permitiendo que los usuarios pudieran estar informados y actualizados en sus tareas y *Workflows*.

La versión *Web* para móviles del WFMS tenía muchas limitaciones debido a las características con las que contaban los dispositivos móviles *touch* de primera generación. El desarrollo de aplicaciones *Web* móviles con HTML no tenía la capacidad de poder ofrecer una interfaz y funcionalidades adecuadas para un dispositivo móvil.

Con la aparición de nuevos dispositivos móviles y sistemas operativos especialmente de estilo *touch*, como por ejemplo *iOS*, *Android* y con *frameworks* de desarrollo nativos para estas plataformas (referencia cruzada N), fue posible desarrollar aplicaciones móviles que realmente ofrecían funcionalidades adecuadas para realizar tareas de forma aceptable con un dispositivo móvil.

Después de varios desarrollos de aplicaciones móviles en las plataformas *iOS*, *Android* y de la necesidad de realizar una nueva versión móvil del WfMS basado en la implementación del modelo de referencia propuesto por Ferreira, el Equipo de Investigación y Desarrollo de Software del LAM concluyó que era necesario realizar una nueva implementación del Modelo de Referencia de WfMS. El nuevo modelo debía tener características y funcionalidades que permitiesen desarrollar WfMS para dispositivos y plataformas móviles tomando en cuenta las características de este tipo de dispositivos (ver aparatado 3.4).

Las aplicaciones móviles hoy en día no sólo utilizan las funcionalidades del dispositivo móvil para poder realizar las tareas para las que fueron diseñadas, también hacen uso de otras tecnologías que van desde servicios móviles (ver apartado 3.4.7) y de *cloud computing* (ver apartado 3.3) a tecnologías colaborativas con características *cloud* y sociales (ver apartado 3.5), por lo que también se tomaron en cuenta estas tecnologías al momento de realizar la adaptación del modelo de referencia de WfMS (ver apartado 3.8).

4.3. Génesis del modelo de WFMS con características Cloud, Móviles y Colaborativas

El nuevo modelo de WfMS con características *Cloud,* Móvil y Colaborativas es el resultado de transponer las funcionalidades del Modelo de Referencia de WfMS (ver apartado 3.8), junto con patrones de arquitectura *cloud*, móviles y colaborativos (ver apartados 3.3, 3.4 y 3.5). En la figura 38 se muestra el proceso que se debe seguir para obtener un modelo que permita el diseño de una arquitectura de software para el desarrollo de WfMS con funcionalidades móviles, *cloud* y colaborativas.

A este modelo genérico resultante (ver fig. 38) se le conoce como «arquitectura de referencia» (ver apartado 3.6.1), su estructura debe ser suficientemente genérica para que pueda ser usada como una plantilla que permita diseñar arquitecturas de software de diferentes categorías de WfMS (ver apartado 3.6.3). Esta arquitectura servirá para definir las funcionalidades requeridas por un WfMS y la interacción entre ellas.

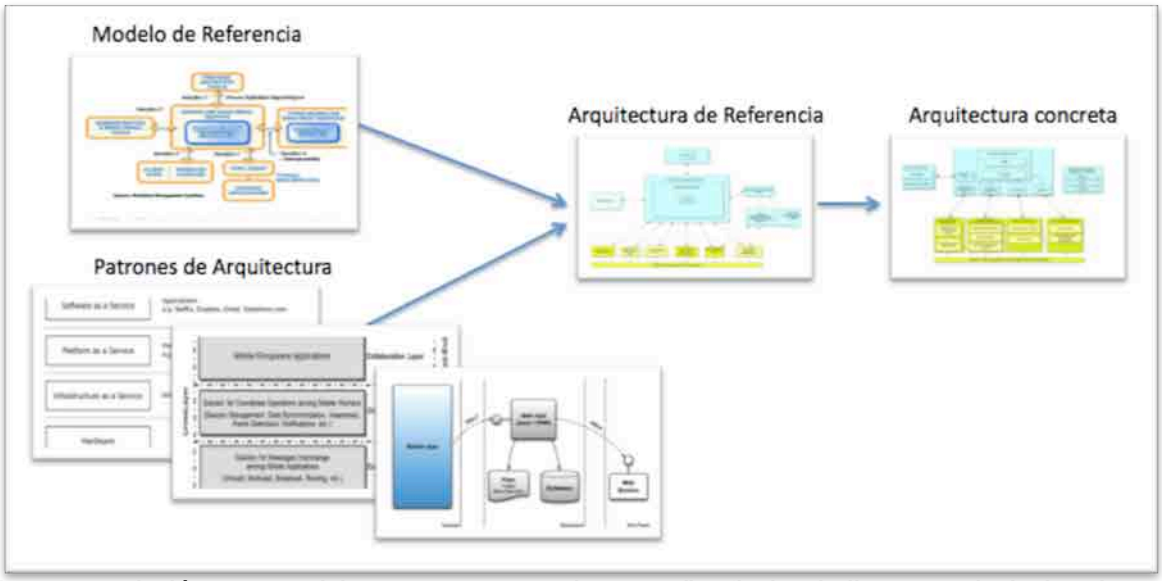

Figura 38: Relación entre Modelos, Patrones y Arquitecturas (las flechas indican entrada de datos), adaptación de (Angelov et al. 2012).

Los patrones de arquitectura móvil, *cloud* y colaborativos provienen de soluciones ampliamente conocidas que resuelven problemas recurrentes en el campo de las

arquitecturas de software (ver apartado 3.6.2), son completamente independientes del campo dónde se aplican, en este caso el de los WfMS.

4.3.1. Patrones de Arquitectura Móvil

Los «patrones de arquitectura móvil» (ver apartado 3.4) que fueron aplicados dan como resultado que la mayoría de las funcionalidades de la arquitectura de referencia sean implementadas en una arquitectura de *software* móvil, estas funcionalidades son mostradas en color amarillo en la arquitectura de referencia (ver fig. 39); más adelante cuando se haga la implementación de una «arquitectura concreta» (ver apartado 3.6.4), se tendrá que decidir el tipo de aplicación móvil (nativa, *Web*, híbrida, etc.) y por consiguiente la arquitectura *software* móvil que deberá tener la aplicación.

Debido a las características propias de los dispositivos móviles y de sus limitaciones (ver apartado 3.4), tres funcionalidades nuevas han sido añadidas en la arquitectura de referencia (ver fig. 39). La funcionalidad que permite utilizar información contextual dentro de los *Workflows* (ver apartado 3.4.7.2), la funcionalidad que permite utilizar servicios móviles para extender las funcionalidades del WfMS en la aplicación móvil (ver apartado 3.4.7) y la funcionalidad de gestión y almacenamiento de contenidos en la nube que ayuda a superar las limitaciones de recursos que tienen los dispositivos móviles (ver apartado 3.4.9).

4.3.2. Patrones de Arquitectura Cloud

Los «patrones de arquitectura cloud» (ver apartado 3.3.4) que fueron aplicados en la arquitectura de referencia permite que varias funcionalidades del WfMS tengan características *cloud* (ver apartado 3.3.1), estas funcionalidades son mostradas en color azul en la arquitectura de referencia (ver fig. 39). Cuando se haga la implementación de una arquitectura concreta, estas funcionalidades podrán ser desarrolladas con alguno de los modelos de implementación *Cloud* (ver apartado 3.3.3) y con alguno de los modelos de servicio *Cloud* (ver apartado 3.3.2).

4.3.3. Patrones de Arquitectura Colaborativos

Los «patrones de arquitectura colaborativos» (ver apartado 3.5) que fueron aplicados en la arquitectura de referencia permite que varias funcionalidades del WfMS tengan características colaborativas (ver apartado 3.5.6). De acuerdo con los patrones de arquitectura colaborativos se deben tener funcionalidades de comunicación, coordinación y de colaboración para que una arquitectura de software tenga características colaborativas.

La comunicación en la arquitectura de referencia (ver fig. 39) se da mediante las funcionalidades de los módulos de herramientas de notificación y servicios móviles. Estas funcionalidades deben resolver las necesidades de comunicación entre los usuarios del WfMS.

La coordinación se da mediante los módulos de herramientas de notificación, servicios móviles y las aplicaciones invocadas para realizar las tareas y actividades dentro de los *Workflows*. Estos módulos deben proveer funcionalidades con soluciones que tienen que ver con aspectos típicos de un grupo colaborativo como por ejemplo: gestión de sesiones, compartir información, gestión de usuarios y roles.

La colaboración se da mediante el módulo de aplicaciones invocadas y el gestor de almacenamiento y contenido en la nube, estos módulos ofrecen servicios y aplicaciones móviles con funcionalidades colaborativas, *cloud* y sociales. Estas funcionalidades deben dar soporte para el manejo de las interacciones entre los usuarios móviles que colaboran para alcanzar las metas del grupo.

4.4. Modulación de las funcionalidades Móviles, *Cloud* y Colaborativas en el Modelo de WfMS

En la figura 39 se muestra la arquitectura de referencia con los módulos (componentes de *software*) que debe tener un WfMS con características móviles, *cloud*, colaborativas y la forma en que deben interactuar entre ellos. Al igual que el Modelo de Referencia propuesto por la coalición de gestión de *Workflows*, el modelo ha sido desarrollado de manera neutral en cuanto a tecnología se refiere, lo que permite al modelo ser independiente de cualquier arquitectura móvil en particular o implementación tecnológica.

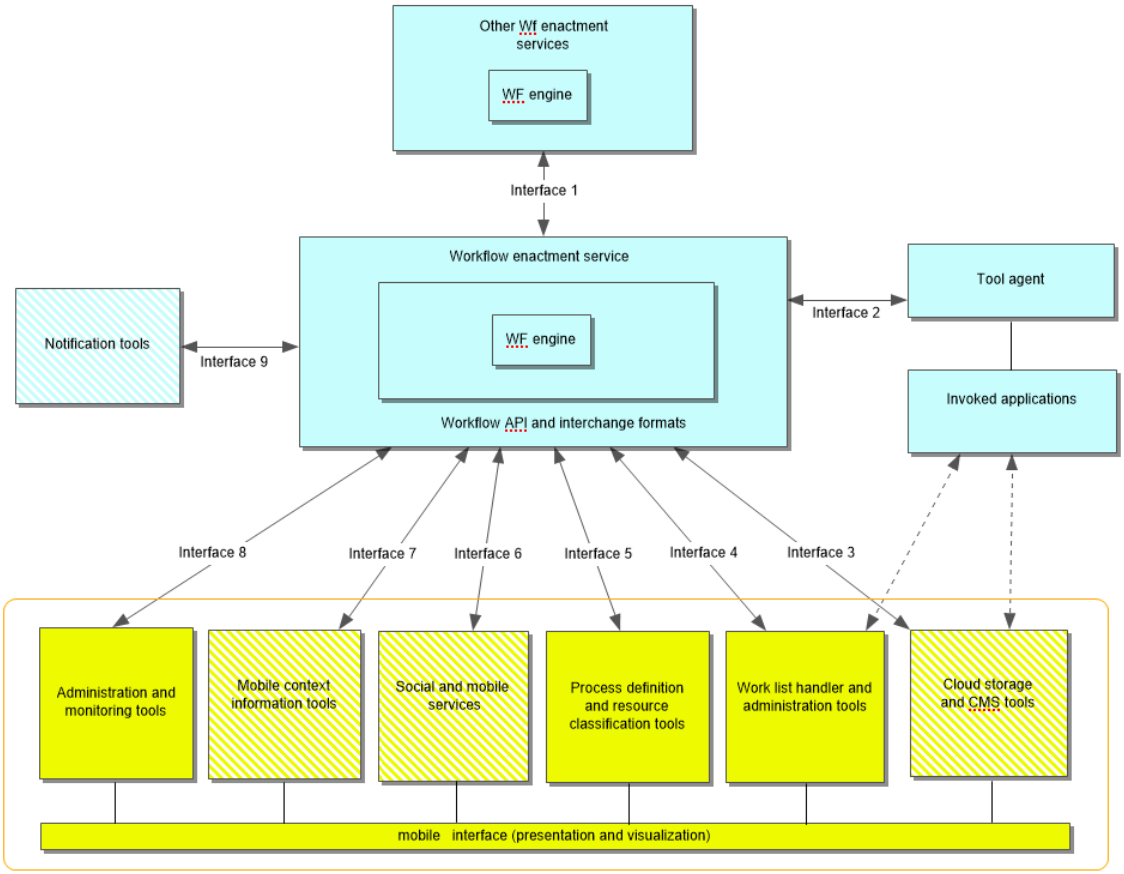

Figura 39: Arquitectura de Referencia de un WfMS con funcionalidades Móviles, *Cloud* y Colaborativas.

4.4.1. Workflow Enactment Service

El Servicio de Interpretación de *Workflows* es la parte central del sistema, este módulo de *software* se encarga de la creación, manejo y ejecución de las instancias de *Workflows* que han sido definidas con antelación (en el apartado 3.8.1 se explica con detalle las características y funcionalidades de un servicio de interpretación).

4.4.2. Other Workflow Enactment Services – Interface 1

Al igual que en el modelo de referencia, este modelo ofrece la posibilidad de conectar varios sistemas gestores, para que puedan colaborar entre ellos y de esta forma poder transferir procesos o parte de ellos para ser ejecutados en otro sistema gestor (ver aparatado 3.8.6).

4.4.3. Invoked Applications – Interface 2

Las aplicaciones invocadas (ver apartado 3.8.5) son utilizadas para llevar a cabo una actividad dentro de un *Workflow*, las aplicaciones invocadas pueden tener características *cloud,* sociales y colaborativas. Las aplicaciones invocadas pueden ser iniciadas por el usuario desde el modulo del sistema gestor de contenidos (*Interface* 3) y desde el módulo de listados y administración de tareas (*Interface* 4). También pueden ser invocadas directamente por el servicio de interpretación de *Workflows*.

4.4.4. Cloud Storage and Content Management Tool – Interface 3

Este módulo se encarga de proveer las herramientas necesarias para la gestión y almacenamiento de contenidos en la nube, de esta manera todos los contenidos que son utilizados dentro de las actividades de los *Workflows* son almacenados en la nube. Mediante el uso de servicios *cloud* con funcionalidades colaborativas y sociales los usuarios pueden asociar los contenidos con las actividades de los *Workflows* (ver apartado 0).

4.4.5. Work List Handler Tool – Interface 4

Este módulo ofrece una herramienta que se la conoce como manejador de listados de trabajo (*Work List Handler*), los listados muestran los ítems de trabajo asignados por el motor de *Workflows* a un usuario en particular (ver apartado 3.8.4). También ofrece una herramienta que permite visualizar en detalle toda la información de la instancia de un *Workflow*.

4.4.6. Process Definition and Resource Classification Tool - Interface 5

Este módulo ofrece una herramienta que sirve para la definición de procesos y para modelar lógicamente el orden en que se van a ejecutar las actividades del proceso (ver apartado 3.8.3). También ofrece una herramienta de clasificación de recursos que

permite acceder a listados de usuarios, de esta forma las tareas y actividades de los procesos pueden ser asignadas a usuarios que cuentan con las características y las necesidades que requiere la tarea para poder ser llevada a cabo.

4.4.7. Mobile Services Tool - Interface **6**

Este módulo ofrece las herramientas necesarias para acceder a servicios móviles disponibles en la nube que sirven para extender las funcionalidades del WfMS (ver apartado 3.4.7). De esta forma se superan las limitaciones de recursos computacionales que tienen los dispositivos móviles.

4.4.8. Context Information Tool - Interface **7**

Este módulo ofrece las herramientas para gestionar la información contextual (ver apartado 3.4.7.2) que puede ser capturada por los censores del dispositivo móvil y que pueden ser utilizadas como información relevante y operacional de un *Workflow* (ver apartado 3.8.2).

4.4.9. Administration and Monitoring Tools - Interface **8**

Este módulo ofrece las herramientas para la ejecución de tareas administrativas y operacionales como por ejemplo la supervisión de instancias de procesos y actividades. Las herramientas de monitorización sirven para recolectar datos del desempeño que tienen los *Workflows* en tiempo de ejecución y ofrecer la información a los usuarios (ver apartado 3.8.7).

4.4.10. Notification Tools - Interface **9**

Este módulo ofrece las herramientas de notificación y comunicación (ver apartado 3.4.10) para mantener a los usuarios actualizados sobre el estado de las instancias de los *Workflows* y las actividades dentro de estos. También permite que los usuarios puedan tener comunicación entre ellos para realizar las actividades de forma colaborativa (ver apartado 3.5.7.3).

4.4.11. Posicionamiento de la Arquitectura de Referencia en el Espacio 3D

En la sección 3.6.6 hemos presentado un espacio tridimensional propuesto por (Stoitsev & Grefen 2012), sirve para entender las relaciones entre los diferentes modelos que se necesitan para diseñar y desarrollar un sistema de software. En la figura 40 se puede ver resaltado en naranja en que parte del espacio tridimensional inicia el proceso de diseño de un WFMS, podemos ver la situación en la que se encuentra el modelo conocido como Arquitectura de Referencia con respecto a las tres dimensiones.

Podemos ver que la arquitectura de referencia tiene un nivel de abstracción alto, debido a esto la arquitectura va a servir para diseñar y desarrollar diferentes tipos de WfMS. La arquitectura de referencia es genérica y no provee información alguna del tipo de WfMS o de los elementos de software que se van a utilizar.

El nivel de agregación de la arquitectura de referencia es alto ya que solo describe sus componentes como una caja que menciona la funcionalidad que tiene cada una de estas dentro de la arquitectura, debido a esto la arquitectura puede ser utilizada para diseñar y desarrollar WfMS en múltiples organizaciones.

El nivel de realización de la arquitectura de referencia es de un nivel medio, debido es un modelo que describe los componentes de un WfMS, sus funcionalidades y la forma en que se comunican, contiene todas las funcionalidades que debe tener un WfMS por lo que es una arquitectura pero aún no se describe la tecnología que se debe utilizar para cada uno de los componentes del sistema lo que permite al modelo diseñar WfMS con diferentes soluciones tecnológicas.

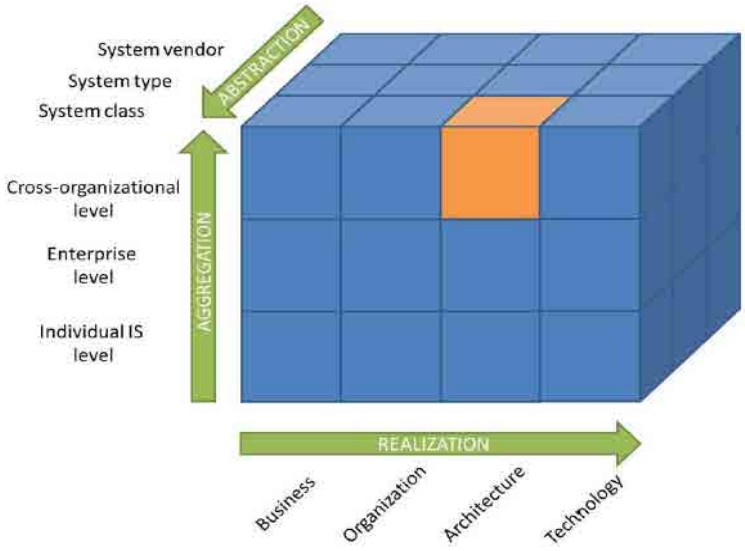

Figura 40: Posicionamiento de la Arquitectura de Referencia en el espacio 3D, fuente (Stoitsev 2012).

4.5. Implementación del modelo

4.5.1. Arquitectura Concreta de un WfMS con características Clud, Móvil y Colaborativas

Una «arquitectura concreta» es diseñada implementando una Arquitectura de Referencia (ver apartado 3.6.3), es utilizada para el desarrollo de una aplicación de *software* especifica. Las arquitecturas de software concretas (ver apartado 3.6.4) son diseñadas en un contexto específico y reflejan las metas del negocio en concreto.

En la figura 41 podemos ver la arquitectura concreta para el desarrollo de WfMS con funcionalidades *cloud*, móviles y colaborativas en el contexto de esta investigación. Se muestran las funcionalidades de cada uno de los componentes que tiene la arquitectura y su relación entre ellos. Para definir las funcionalidades con las que cuenta cada uno de los componentes de la arquitectura concreta es necesario tomar en cuenta las soluciones tecnológicas que pueden ser implementadas de acuerdo con los patrones de arquitectura *cloud*, móviles y colaborativos con los que ha sido diseñada la arquitectura de referencia de WfMS (ver apartado 4.3).

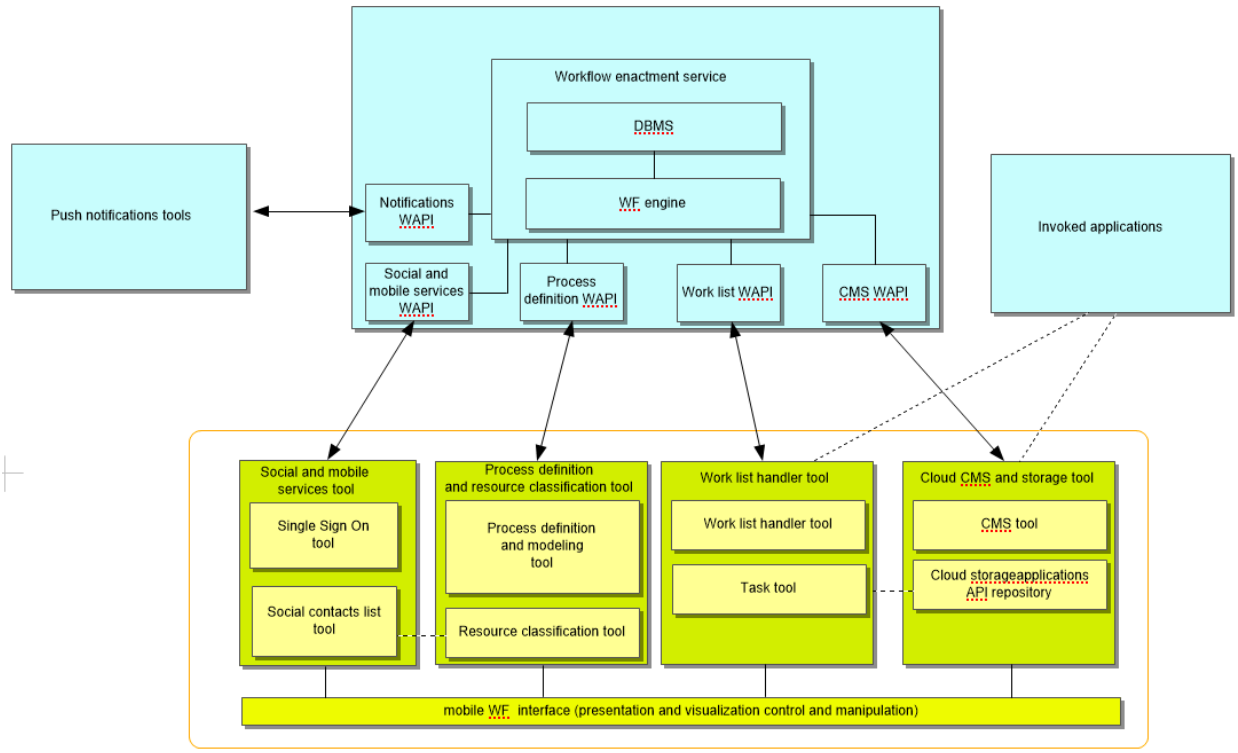

Figura 41: Arquitectura Estándar de un WfMS con funcionalidades Móviles, *Cloud* y Colaborativas.

4.5.1.1. Workflow Enactment Service

4.5.1.1.1. Workflow Engine

El Motor de *Workflows* es la parte central del WfMS, esta funcionalidad se encarga de la creación, manejo y ejecución de las instancias de *Workflows* que han sido definidas con antelación. Logísticamente provee las facilidades para que los procesos puedan ser completados (ver apartado 3.8.1.1).

4.5.1.1.2. DBMS (Data Base Management System)

El sistema manejador de bases de datos se encarga de almacenar información de control interna que sirve para el control de las instancias de los *Workflows* que se están ejecutando, como por ejemplo identificar el estado en que se encuentra un proceso o una actividad dentro del sistema de interpretación de *Workflows* (ver apartado 3.8.1.2).

4.5.1.2. WAPI (Workflow Application Programming Interface)

Los módulos del sistema gestor, aplicaciones y recursos, pueden interactuar con el servicio de interpretación de *Workflows* mediante una WAPI, un WfMS está compuesto de varios módulos, estos deben intercambiar información para poder trabajar juntos, en la figura 42 se muestran las interfaces que pueden tener comunicación con el sistema de interpretación de *Workflows* mediante el uso de una WAPI. La WAPI se compone de funciones con parámetros y formatos de información preestablecidos y estandarizados, con la finalidad de permitir el intercambio de información relacionada con los *Workflows* entre los diferentes módulos del sistema (ver apartado 3.8.2).

4.5.1.2.1. Patrones de Arquitectura y Soluciones Tecnológicas

Las soluciones tecnológicas que pueden ser implementadas en este módulo corresponden con los patrones de arquitectura del modelo de implementación y el modelo de servicio de *cloud computing*, este módulo por ejemplo puede ser implementado en una nube privada (ver apartado 3.3.3) y con el modelo de servicio *Software as a Service* (ver apartado 3.3.2).

Para que se pueda efectuar la comunicación y el intercambio de información entre el Sistema de Interpretación de *Workflows* y los otros módulos del sistema, la WAPI es implementada con patrones de arquitectura orientada a recursos con la solución tecnológica conocida como *REST services* (ver apartado 3.4.8).

4.5.1.3. Invoked Applications

Este módulo se encarga de ofrecer las funcionalidades para ejecutar las aplicaciones móviles colaborativas que son utilizadas para llevar a cabo una actividad dentro de un *Workflow*. El modulo del sistema gestor de contenidos junto con el módulo de listados y administración de tareas tienen comunicación con esta funcionalidad para ejecutar las aplicaciones colaborativas a petición del usuario.

4.5.1.3.1. Patrones de Arquitectura y Soluciones Tecnológicas

Las soluciones tecnológicas que pueden ser implementadas en este módulo son en forma de aplicaciones colaborativas con características *cloud* y sociales (ver aparatado 3.5) implementando el modelo de servicios de los patrones de arquitectura *cloud computing* (ver apartado 3.3.2). De acuerdo con los patrones de arquitectura colaborativos implementados (ver apartado 3.5.7) las aplicaciones colaborativas son el soporte principal para que se dé la interacción colaborativa entre los usuarios para que de esta forma las metas del grupo colaborativo puedan ser alcanzadas.

4.5.1.4. Cloud Storage and CMS Tool

4.5.1.4.1. CMS Tool

Esta herramienta se encarga de ofrecer las funcionalidades para gestionar contenidos con la finalidad de asociarlos con las actividades y tareas dentro de los *Workflows*, ofrece acceso a los contenidos almacenados en la nube mediante los servicios de almacenamiento con los que cuenta el CMS. Desde esta herramienta el usuario puede invocar a las aplicaciones colaborativas para visualizar los contenidos y trabajar en ellos.

4.5.1.5. Cloud Storage API Repository

Esta herramienta es un repositorio de las APIs que se necesitan para acceder a los servicios de almacenamiento en la nube con los que cuenta el CMS para que los usuarios puedan gestionar los contenidos almacenados en la nube y que son asociados con los *Workflows.*

.

4.5.1.5.1. Patrones de Arquitectura y Soluciones Tecnológicas

Las soluciones tecnológicas que pueden ser implementadas en este módulo son en forma de servicios de almacenamiento con características *cloud* y sociales (ver aparatado N) implementando el modelo de servicios *Software as a Service* y *Storage as a Service* de los patrones de arquitectura *cloud computing* (ver apartado N). De acuerdo

con los patrones de arquitectura colaborativos implementados (ver apartado N) la coordinación entre los usuarios es una característica principal de los sistemas colaborativos, los servicios de almacenamiento *cloud* permiten la gestión de sesiones, sincronización de contenidos, permisos para acceder y compartir de contenidos, estas características extienden las funcionalidades de coordinación del CMS.

4.5.1.6. Work List Handler Tool

4.5.1.6.1. Work List Handler Tool

Esta herramienta se encarga de ofrecer las funcionalidades para que el usuario tenga acceso a los listados que muestran los ítems de trabajo asignados por el motor de *Workflows* a un usuario en particular. Mediante esta funcionalidad el usuario puede ordenar y seleccionar los ítems de trabajo basándose en sus propiedades, ver propiedades relevantes de los ítems de trabajo y permitir al usuario bloquear o no aceptar un ítem de trabajo.

4.5.1.6.2. Task Tool

Esta herramienta se ejecuta cuando el usuario selecciona un ítem de trabajo en el listado de tareas. Las funcionalidades de esta herramienta se encargan de mostrar al usuario toda la información en detalle relacionada con la instancia de un *Workflow* y que es relevante para el usuario. Se muestran todos los pasos y actividades que conforman el *Workflow*, los documentos asociados y el paso que está activo dentro del *Workflow*. Desde esta herramienta el usuario puede invocar la funcionalidad de CMS para adjuntar documentos y también ejecutar las aplicaciones colaborativas para abrir los documentos asociados a las actividades y tareas del *Workflow*.

4.5.1.7. Process Definition and Resource Classification Tool

4.5.1.7.1. Process Definition and Modeling Tool

Esta herramienta ofrece las funcionalidades para crear la definición de los procesos (ver apartado 3.8.3.1) mediante una herramienta de modelado completamente gráfica, permite crear los pasos y actividades que van a formar parte del proceso, modelar lógicamente el orden en que se van a ejecutar las actividades, indicar las condiciones que se deben dar para que el proceso pueda ser iniciado y finalizado. También permite asociar información necesaria que pueda ser relevante para el proceso. La información del tipo de usuario que debe ejecutar cada una de las tareas se obtiene a través del módulo de clasificación de recursos.

4.5.1.7.2. Resource Classification Tool

Los recursos que van a realizar las tareas de las actividades de los *Workflows* necesitan ser clasificados (ver aparatado 3.8.3.2), de esta forma las tareas pueden ser asignadas a usuarios, que cuentan con las características y las necesidades que requiere la tarea para ser llevada a cabo. Esta herramienta cuenta con una funcionalidad que muestra las diferentes clases de recursos en forma gráfica. Los ítems que son más comunes para ser mostrados son: una lista de las clases de recursos, que a su vez son subdivididos en roles (basados en funciones, habilidades y perfiles) y unidades orgánicas (basados en equipos y departamentos).

La funcionalidad de clasificación de recursos hace uso del módulo de herramientas de servicios móviles para acceder a los contactos con los que colabora el usuario en sus redes sociales y de esta forma utilizarlos como recursos a los que se les pueden asignar tareas dentro de los procesos.

4.5.1.8. Mobile Services Tool

4.5.1.8.1. Single Sign On Tool

Esta herramienta (ver aparatado 3.4.9) ofrece las funcionalidades para que el WfMS pueda utilizar los servicios colaborativos que ofrecen los medios sociales (ver apartado 3.5.5) y de esta forma acceder en nombre del usuario a información y recursos protegidos en sus cuentas de redes sociales para utilizarlos dentro de las funcionalidades del WfMS.

4.5.1.8.2. User Contacts List Tool

Esta herramienta ofrece las funcionalidades para que el usuario pueda utilizar las listas de contactos y conexiones que tiene con otros usuarios en sus redes sociales para utilizarlos dentro del WfMS de forma colaborativa.

4.5.1.8.3. Patrones de Arquitectura y Soluciones Tecnológicas

Las soluciones tecnológicas que pueden ser implementadas en este módulo son los servicios colaborativos de redes sociales (ver aparatado 3.5.5.1) y la identidad federada de usuarios en servicios móviles (ver aparatado 3.4.9.1) implementando el modelo de servicios *Software as a Service* de los patrones de arquitectura *cloud computing* (ver apartado 3.3.2). Estas soluciones cumplen con los patrones colaborativos de la arquitectura (ver apartado 3.5.7).

4.5.1.9. Notification Tools

Esta herramienta se encarga de ofrecer servicios de notificación que están disponibles en la nube y que son utilizados dentro del WfMS para mantener al usuario informado del estado de las instancias de los procesos, actividades y tareas. También permite que los usuarios puedan tener comunicación entre ellos para realizar las actividades de forma colaborativa.

4.5.1.9.1. Patrones de Arquitectura y Soluciones Tecnológicas

Las soluciones tecnológicas que pueden ser implementadas en este módulo son los servicios de comunicación y notificación (ver apartado 3.4.10) que ofrecen los medios sociales (ver apartado 3.5.5.1), y los servicios de comunicaciones y notificaciones en la

nube implementando el modelo de servicios de los patrones de arquitectura *cloud computing* (ver apartado 3.4.6).

De acuerdo con los patrones de arquitectura colaborativos (ver apartado 3.5.7) las soluciones tecnológicas de notificación son necesarias para que se dé la colaboración e interacción entre los usuarios, es una funcionalidad esencial para la comunicación y coordinación en una arquitectura con características colaborativas.

4.5.1.10. Posicionamiento de la Arquitectura Concreta en el Espacio 3D

En la figura 42 se puede ver resaltado en naranja en que parte del espacio tridimensional se encuentra el modelo. Podemos ver que la arquitectura concreta tiene un nivel de abstracción medio, describe los sistemas de *software* que la arquitectura va a tener en cada una de sus funcionalidades pero aún no se definen los proveedores de software que van a utilizar cada uno de estos sistemas.

El nivel de agregación de la «arquitectura estándar» es medio, describe sus componentes especificando cada una de las funcionalidades que tienen los componentes, los subsistemas dentro de los componentes han sido identificados pero aún no se explican a detalle.

El nivel de realización de la arquitectura estándar es de un nivel medio, describe los componentes de un WfMS, cada una de sus funcionalidades y la forma en que se comunican, aun no se describe específicamente la tecnología que se debe utilizar para cada uno de los componentes del sistema lo que permite al modelo diseñar WfMS con diferentes tecnologías.

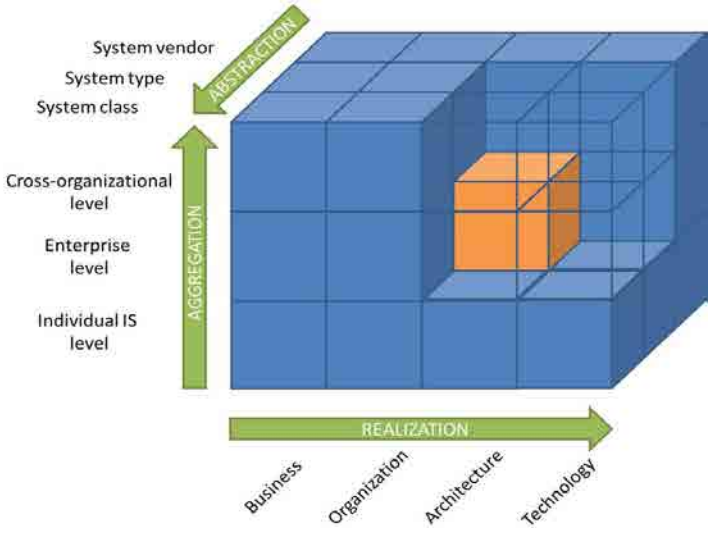

Figura 42: Posicionamiento de la Arquitectura de Estándar en el espacio 3D, basado en (Stoitsev 2012).

4.5.2. Implementación de la Arquitectura Concreta

4.5.2.1. Modelo Tecnológico de un WfMS

En la figura 43 se puede ver la arquitectura de *software* para un WfMS en forma de aplicación móvil nativa, que ha resultado de implementar la arquitectura concreta con soluciones tecnológicas específicas. En el apartado 3.4.4 se explican los patrones de arquitectura móviles que pueden ser implementados, de todos estos, el patrón de arquitectura móvil que se ha implementado en este caso es el de una aplicación móvil nativa. Esta arquitectura de *software* tiene como objetivo especificar la tecnología que se ha implementado en cada una de los módulos de una aplicación nativa de WfMS y la forma en que están relacionados y organizados.

4.5.2.1.1. Native Application

En la arquitectura de software la «aplicación móvil nativa» se encarga de ofrecer las funcionalidades del WfMS a los usuarios mediante una interfaz gráfica, ejecutándose en una plataforma móvil (ver apartado 3.4).

La aplicación móvil hace uso de las librerías propias de la plataforma para acceder a las características y funciones del dispositivo móvil. De esta forma las funcionalidades de

la aplicación móvil pueden tener comunicación con el servicio de interpretación de *Workflows* implementado en el *Backend* de la arquitectura de *software* y con los servicios móviles *cloud* que necesita la aplicación para extender las funcionalidades del WfMS.

4.5.2.1.2. Backend

La parte *Backend* de la arquitectura de software (ver fig. 44) se encuentra implementada en una nube privada (ver modelos de implementación de *cloud* computing en el apartado 3.3.3). En el *Backend* se encuentra alojado el servidor *Web* que se encarga de ejecutar las funcionalidades del servicio de interpretación de *Workflows.* Las funcionalidades del servicio de interpretación de *Workflows* son ofrecidas mediante el modelo de implementación de servicio *cloud* conocido como SaaS, permitiendo que el nivel de calidad de servicio de las funcionalidades sea siempre el mismo (ver apartado 3.8.8).

Las aplicaciones móviles acceden a las funcionalidades del servicio de interpretación de *Workflows* a través de APIs implementadas en forma de servicios REST (ver apartado 3.4.8) esto permite que se pueda ofrecer los servicios a diferentes clientes sin importar si es una aplicación nativa, *Web*, etc.

4.5.2.1.3. 3rd Party Services

Los servicios móviles, *cloud* y colaborativos ofrecidos por terceros son parte esencial de la arquitectura de software de un WfMS, extienden las funcionalidades de la aplicación móvil y dan soporte a las funcionalidades del *Backend*. Estos servicios son ofrecidos en los diferentes modelos de implementación de servicios de *cloud computing*.

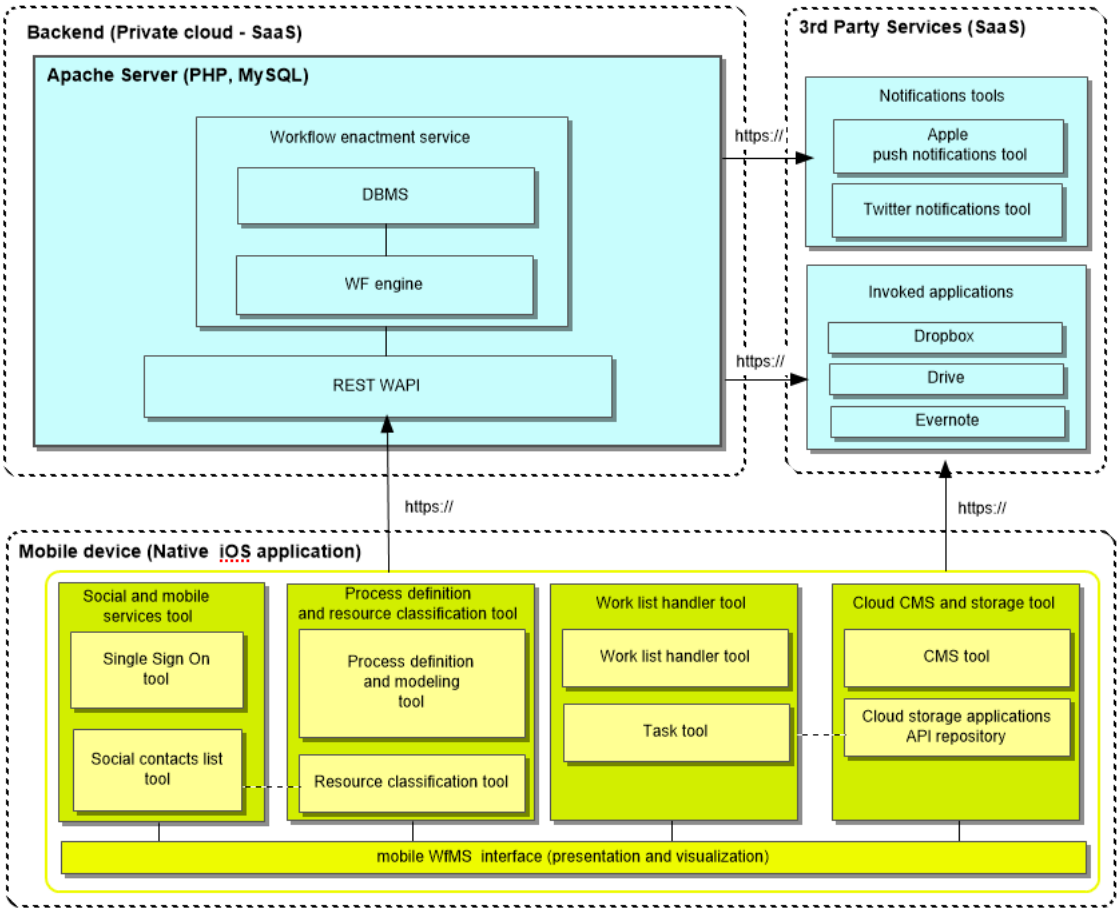

Figura 43: Arquitectura de *Software* de un WfMS para una aplicación móvil nativa.

4.5.2.2. Posicionamiento en el Espacio 3D

En la figura 45 se puede ver resaltado en naranja en que parte del espacio tridimensional se encuentra el modelo. Podemos ver que la arquitectura de *software* tiene un nivel de abstracción medio, describe los tipos de sistemas y la tecnología que una aplicación móvil nativa de WfMS debe tener, se definen los proveedores de *software* que van a ser implementados en cada uno de estos sistemas.

El nivel de agregación de la arquitectura de software es medio, describe sus componentes especificando cada una de las funcionalidades que tienen los sistemas que componen la arquitectura de software, todos los subsistemas dentro de los

componentes y el tipo de tecnología que se debe implementar han sido identificados pero aún no se explican a detalle.

El nivel de realización de la arquitectura de software es de un nivel alto, describe los componentes de un WfMS, cada una de sus funcionalidades y la forma en que se comunican, se describe la tecnología que se debe utilizar para cada uno de los componentes del sistema lo que permite al modelo diseñar WfMS con las soluciones tecnológicas que se describen en la arquitectura de la aplicación móvil nativa para WfMS.

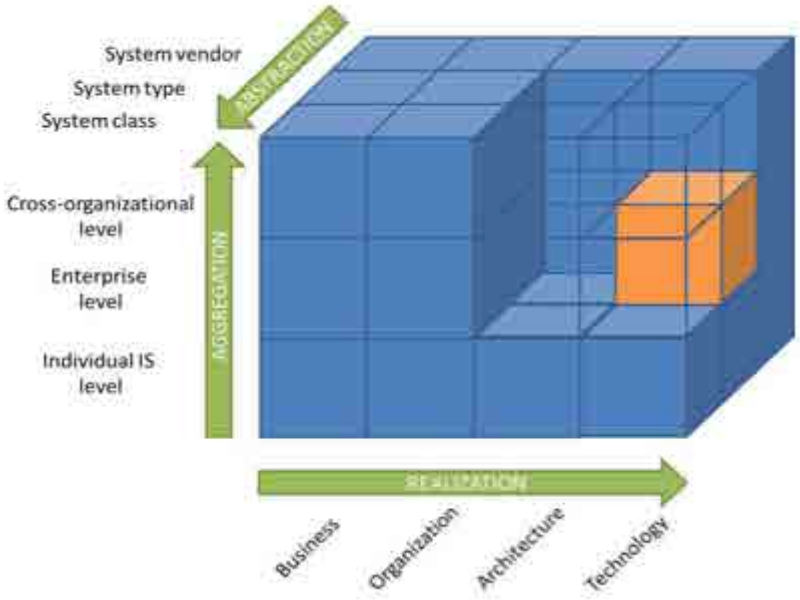

Figura 44: Posicionamiento de la Arquitectura de Software en el espacio 3D, basado en (Stoitsev 2012).

4.6. Aplicación Móvil de WfMS – *WOLF*

La implementación de la arquitectura de software dio como resultado un WfMS para la plataforma *iOS* (ver apartado 3.4.1), específicamente para el dispositivo móvil *iPad*. A esta herramienta se le dio el nombre de *WOLF* (*Work Linera Flow*) y está disponible en la tienda de aplicaciones conocida como *Apple Store*[57]. A continuación se explican las funcionalidades de la herramienta.

4.6.1. Registro de Usuarios y Login

Los usuarios tienen la posibilidad de Registrarse como usuarios de la aplicación *WOLF* de la forma en como se muestra en la figura 45. La primera opción es creando una cuenta de usuario con datos personales. La segunda opción es utilizar una cuenta de *Facebook* y de esta forma el usuario podrá identificarse dentro de la aplicación *WOLF* con su perfil (nombre, apellidos, foto, etc) que tiene en la red social.

[57] https://itunes.apple.com/es/app/wolf/id668607305

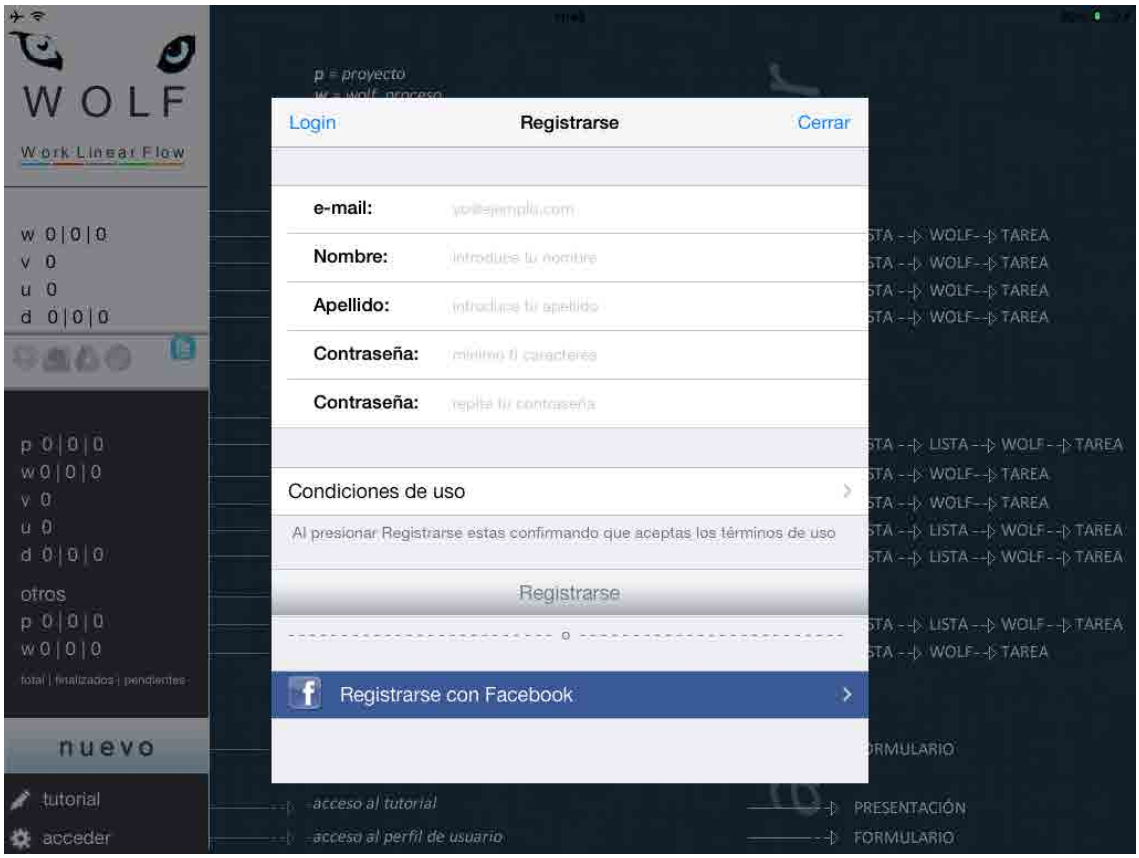

Figura 45: Funcionalidad para registrarse en la aplicación WOLF.

Facebook junto con otros servicios de medios sociales (ver fig. 46) están integrados en el sistema operativo *iOS*, esto permite a los usuarios utilizar su cuenta de *Facebook* en aplicaciones instaladas por terceros. En este caso la aplicación *WOLF* utiliza la funcionalidad *Single Sign On* (ver apartado 4.4.7 de la arquitectura) para acceder al perfil del usuario en *Facebook* y registrarlo en la aplicación *WOLF*.

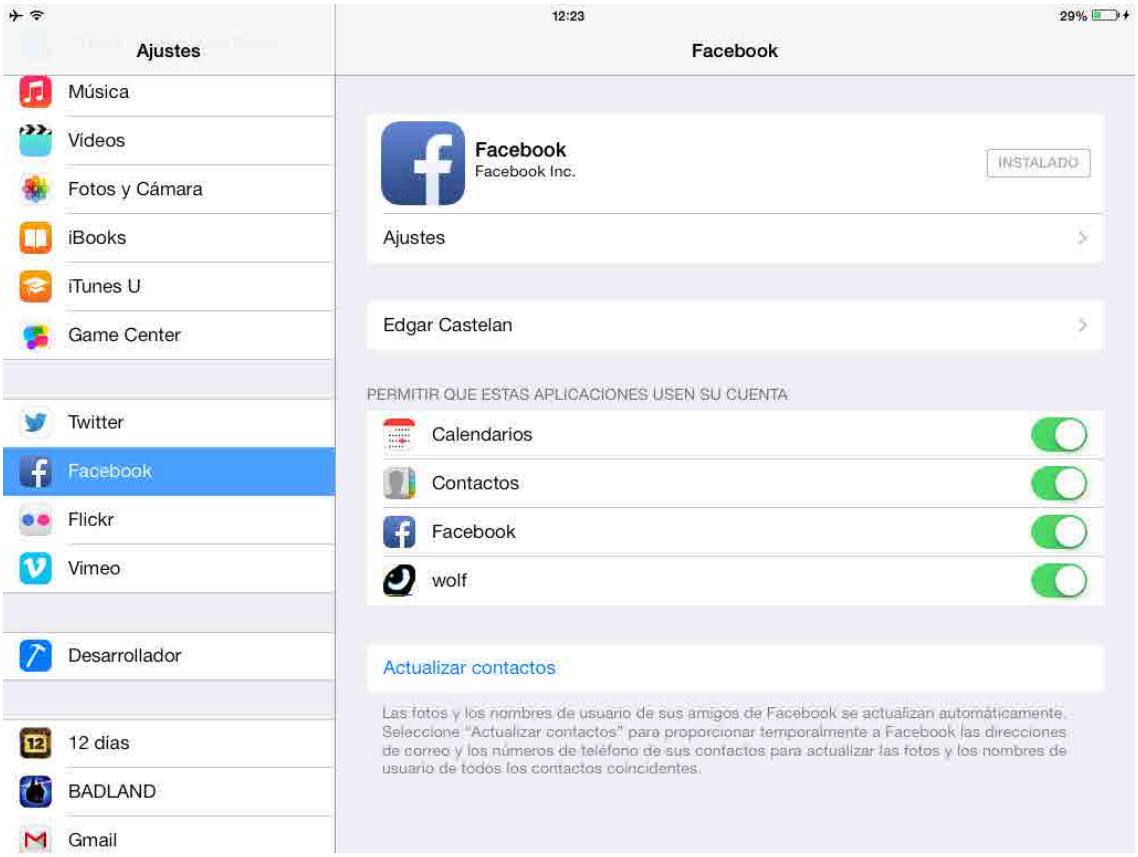

Figura 46: Cuentas de redes y medios sociales que están integradas en *iOS*.

La funcionalidad de *Log-in* (ver figura 47) permite al usuario acceder a la aplicación *WOLF* con su nombre de usuario y contraseña con los que se ha registrado previamente. Otra opción es utilizar su cuenta de *Facebook* con la que se ha registrado en la aplicación *WOLF*.

WOLF utiliza la solución tecnológica *Single Sign On* (ver apartado 3.4.9.1) para acceder al perfil del usuario en *Facebook* y de esta forma autenticar e identificar al usuario dentro de la aplicación.

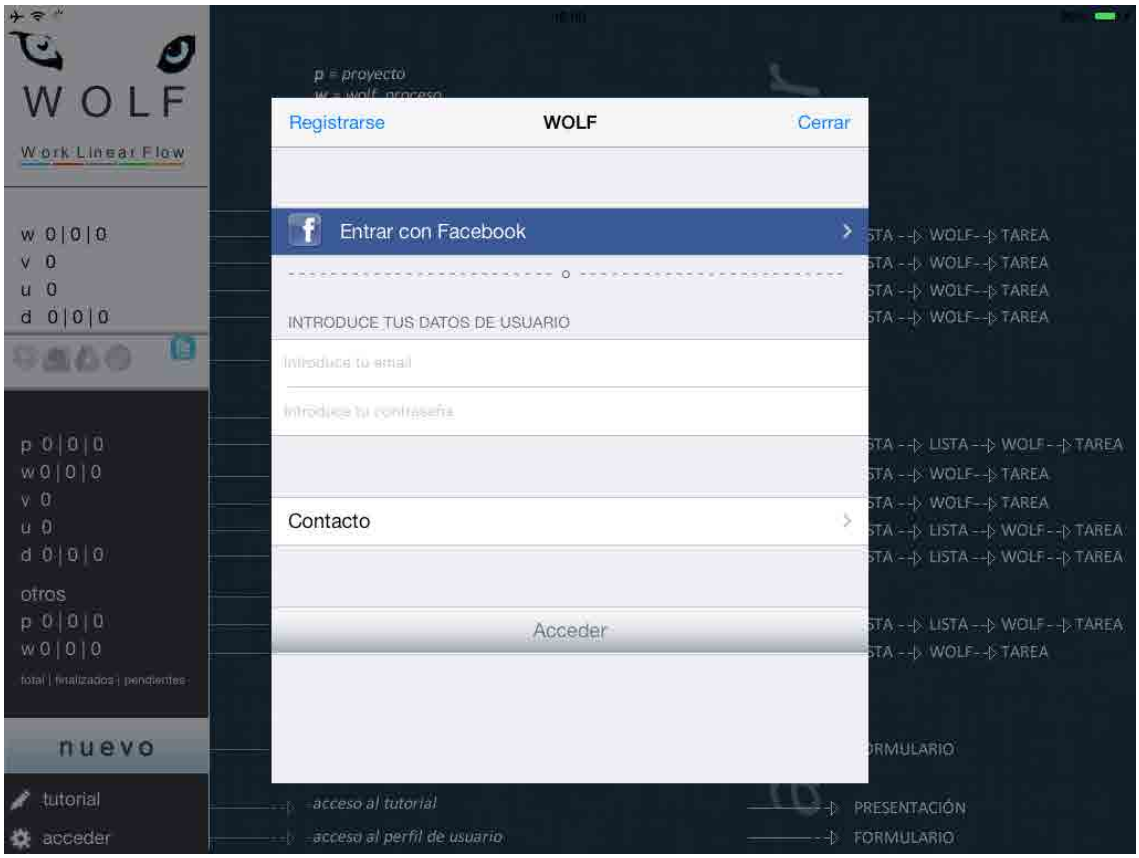

Figura 47: Funcionalidad de Log-in en la aplicación *WOLF*.

4.6.2. Menú principal

El menú principal de la aplicación (*Work List Handler Tool*, ver aparatado 9) es donde se muestra la información más relevante para el usuario y desde aquí se puede acceder a las funcionalidades principales de la aplicación *WOLF*.

Este menú ofrece las diferentes opciones que dan acceso a los listados que muestran los ítems de trabajo, en la figura 48 se muestra el menú y la descripción de las funcionalidades que tiene cada una de las opciones.

Figura 48: Menú principal (*Work List Handler Tool*) de la herramienta *WOLF*.

4.6.2.1. Listado de Tareas

Muestra la información de cada uno de los ítems de trabajo que el usuario tiene asignados. En la figura 49 se muestran el listado de ítems de trabajo que han sido asignados por el motor de *workflows* para el usuario. Desde este listado el usuario puede seleccionar un ítem para acceder a la funcionalidad que muestra en detalle toda la información del *workflow* y poder realizar la tarea que le fue asignada.

Figura 49: Listado de ítems de trabajo.

4.6.2.2. Listado de Workflows

Muestra en forma de ítems (ver fig. 50), información relevante de cada uno de los *workflows*, dependiendo de la opción que se seleccione en el menú se puede mostrar el Listado de *workflows* que han sido creados por el usuario o los que han sido creados por terceros pero que involucran al usuario en alguna tarea dentro del *workflow*.

La herramienta de listado ofrece la funcionalidad de administración de ítems permitiendo que el usuario pueda eliminar y cancelar los *workflows* de los que es propietario el usuario. Desde este listado el usuario puede seleccionar un ítem y de esta forma acceder a la herramienta de gestión de tareas donde se muestra en detalle toda la información del *workflow*. El listado está dividido en dos apartados, el apartado de finalizados muestra los *workflows* que ya fueron terminados, y el apartado de pendientes muestra los que aún tienen pasos y actividades por realizar y por lo tanto están pendientes de finalizar.

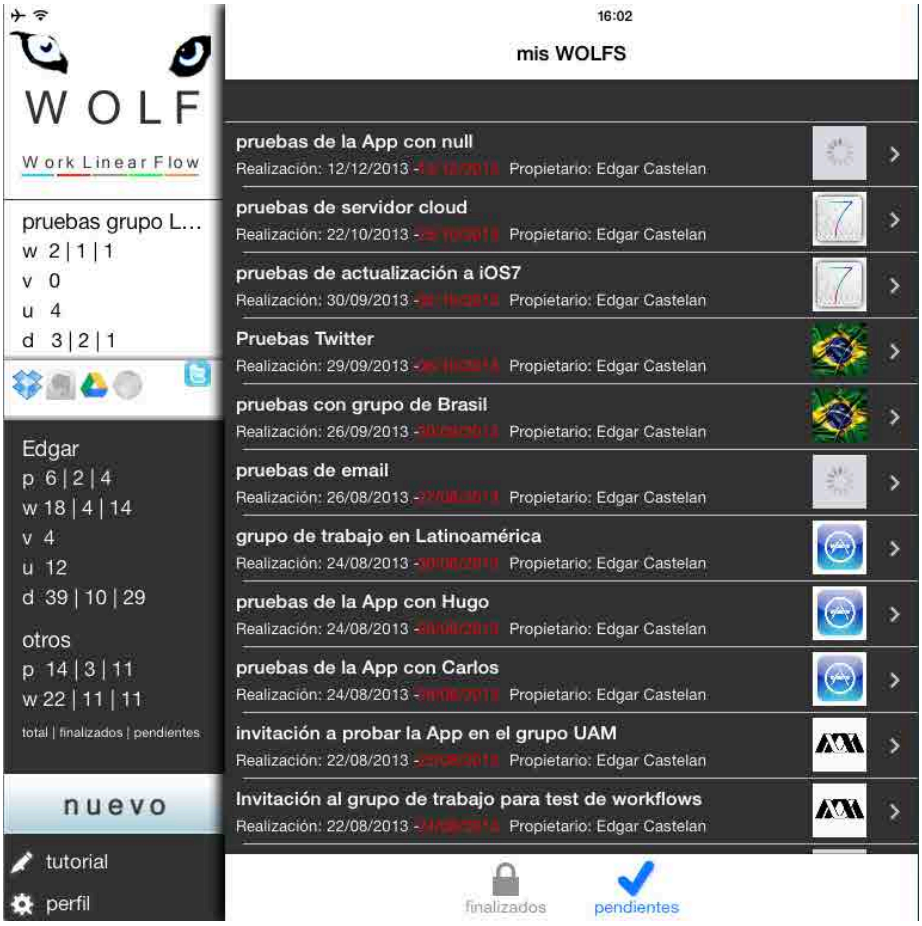

Figura 50: Listado de *Workflows*.

4.6.2.3. Listado de Usuarios

Muestra información en forma de ítems de cada uno de los usuarios con los que está colaborando el usuario de la aplicación (ver fig. 51). Cuando se selecciona un ítem de la lista de usuarios se muestra un listado con todos los *workflows* los que están colaborando los dos usuarios.

Figura 51: Listado de Usuarios.

4.6.2.4. Listado de Documentos

Muestra en forma de ítems, información relevante de cada uno de los Documentos que están asociados con los *workflows* en los que el usuario se encuentra colaborando (ver fig. 52).

La herramienta de listado ofrece la funcionalidad de administración de ítems permitiendo que el usuario pueda eliminar los documentos de los que es propietario. Desde este listado el usuario puede seleccionar un ítem y de esta forma acceder a la funcionalidad que muestra en detalle toda la información del *workflow* con el que está asociado el Documento.

Figura 52: Listado de Documentos.

4.6.2.5. Listado de Proyectos

Muestra en forma de ítems, información relevante de cada uno de los proyectos (ver fig. 53), dependiendo de la opción que se seleccione en el menú se puede mostrar el Listado de los Proyectos que han sido creados por el usuario o los que han sido creados por terceros que involucran al usuario en una o varias tareas dentro de los *workflows* que pertenecen al proyecto.

El listado ofrece la funcionalidad de administración de ítems permitiendo que el usuario pueda finalizar, cancelar y eliminar los proyectos que han sido creados por el mismo. Desde este listado el usuario puede seleccionar un ítem y de esta forma acceder a un listado con todos los *workflows* que pertenecen a el proyecto, también tiene la opción de administrar los ítems de este segundo listado. Cuando un proyecto es finalizado por el usuario se mostrará en el apartado finalizados y no se podrán crear nuevos *workflows* dentro del Proyecto.

Figura 53: Listado de Proyectos.

4.6.3. Herramienta de Gestión de Tareas

En la Herramienta de Gestión de Tareas (*Task Tool*, ver apartado 4.4.5) siempre es invocada desde la herramienta de listados al seleccionar uno de los ítems. Aquí es se muestran toda la información detallada del estado en que se encuentra la instancia del *workflow* y el paso que está activo dentro del *workflow* (ver fig. 54).

Ofrece las funcionalidades para que el usuario pueda rechazar o realizar las actividades que tiene asignadas dentro de los pasos del *workflow* y hacer comentarios al respecto. También puede adjuntar contenidos almacenados en la nube y abrirlos con las aplicaciones colaborativas con las que está asociado el contenido. Los contenidos se adjuntan con la herramienta de gestión de contenidos en la nube y las aplicaciones colaborativas en la nube son ejecutadas en un navegador *Web* dentro de la aplicación WOLF.

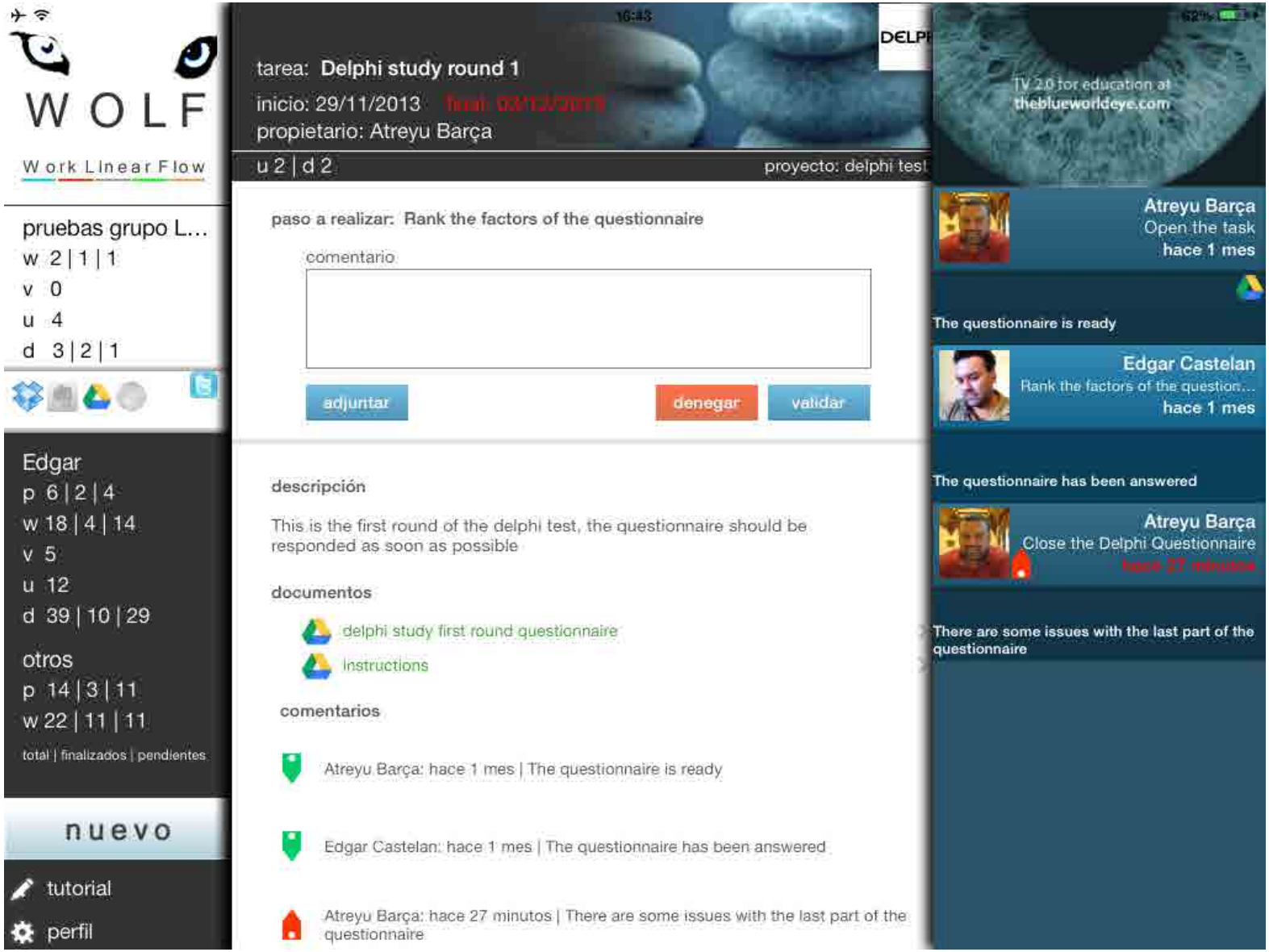

Figura 54: Herramienta de Gestión de Tareas (*Task Tool*).

4.6.4. Herramienta de Gestión de Contenidos en la Nube

En la figura 55 se muestra la herramienta (*Cloud Storage and Content Management Tool*, ver apartado 4.4.4) que ofrece las funcionalidades para que se puedan gestionar contenidos que están almacenados en la nube y asociarlos con las actividades dentro de un *workflow*.

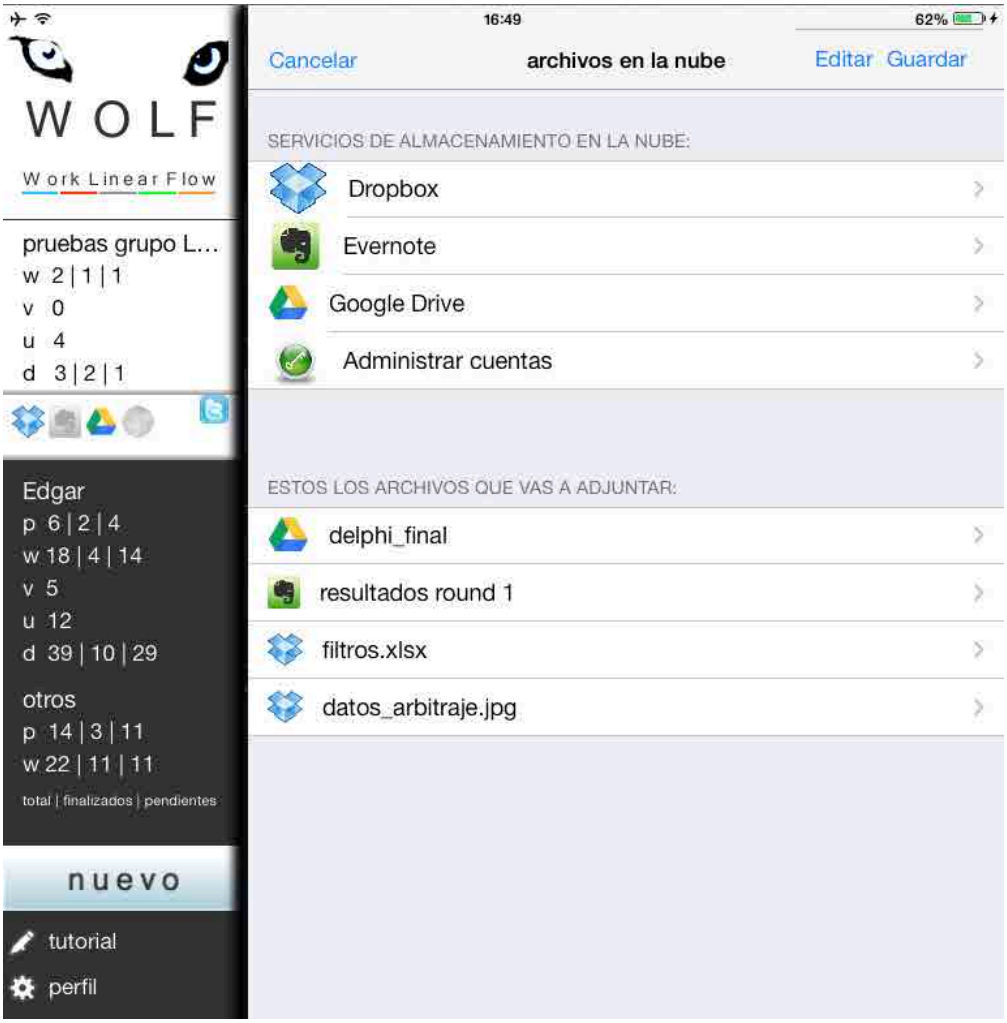

Figura 55: Herramienta de Gestión de Contenidos en la Nube.

En la figura 56 se muestra la herramienta de Gestión de Contenidos haciendo uso de la funcionalidad *Single Sign On* para obtener el permiso de acceder a la cuenta de *Dropbox.*

Figura 56: Funcionalidad *Single Sign On* para acceder a *Dropbox* desde la aplicación *WOLF*.

En la figura 57 se muestra la Herramienta de Gestión de Contenidos accediendo a los archivos y carpetas de *Dropbox*, el usuario puede navegar por las diferentes carpetas y seleccionar todos los archivos que sean necesarios sin la necesidad de salir de la aplicación *WOLF, Dropbox* se está utilizando como un servicio móvil dentro de la app WOLF.

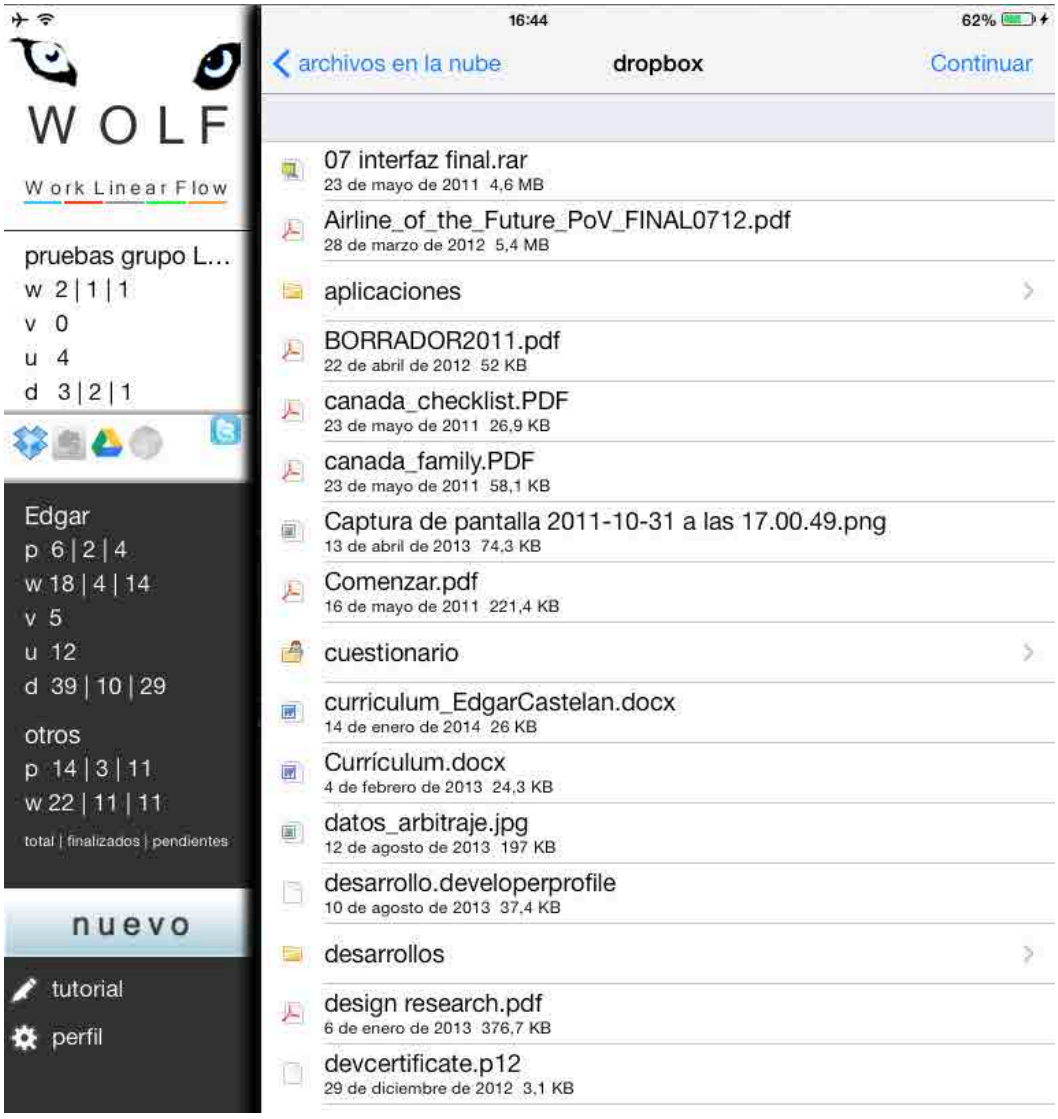

Figura 57: Funcionalidad de gestión de contenidos para *Dropbox*.

4.6.5. Herramienta de Definición de Procesos

En la figura 58 se puede ver la herramienta que ofrece las funcionalidades para la Definición de Procesos (*Process Definition Tool*, ver apartado 4.4.6), permite crear los pasos y actividades que van a formar parte del proceso y modelar lógicamente el orden en que se van a ejecutar. También permite asociar información necesaria que es relevante para el proceso.

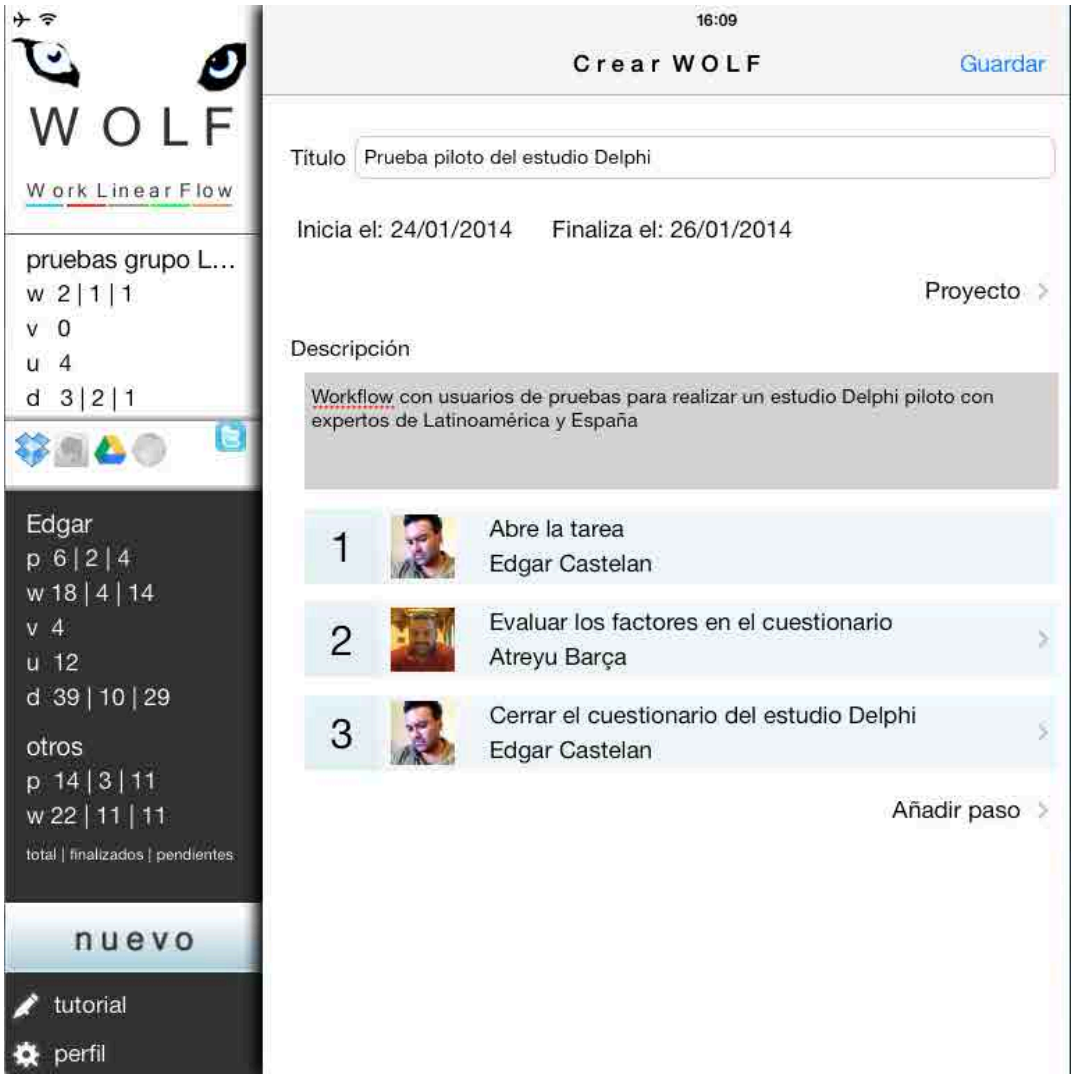

Figura 58: Herramienta de Definición de Procesos.

4.6.5.1. Asociar un Proyecto

En la figura 59 se puede ver la funcionalidad que permite asociar un Proyecto a la definición del proceso.

Figura 59: Funcionalidad para asociar un proyecto a la Definición del Proceso.

En caso de ser un Proyecto nuevo, se puede asociar un *hashtag* de *Twitter* para que los usuarios puedan hacer un seguimiento de los *Tweets* que los usuarios van publicando en relación con el *hashtag* del Proyecto. También se hace uso de un servicio móvil en la nube para hacer una búsqueda de imágenes por palabras clave, una de estas imágenes puede ser seleccionada como imagen distintiva del Proyecto (ver fig. 60).

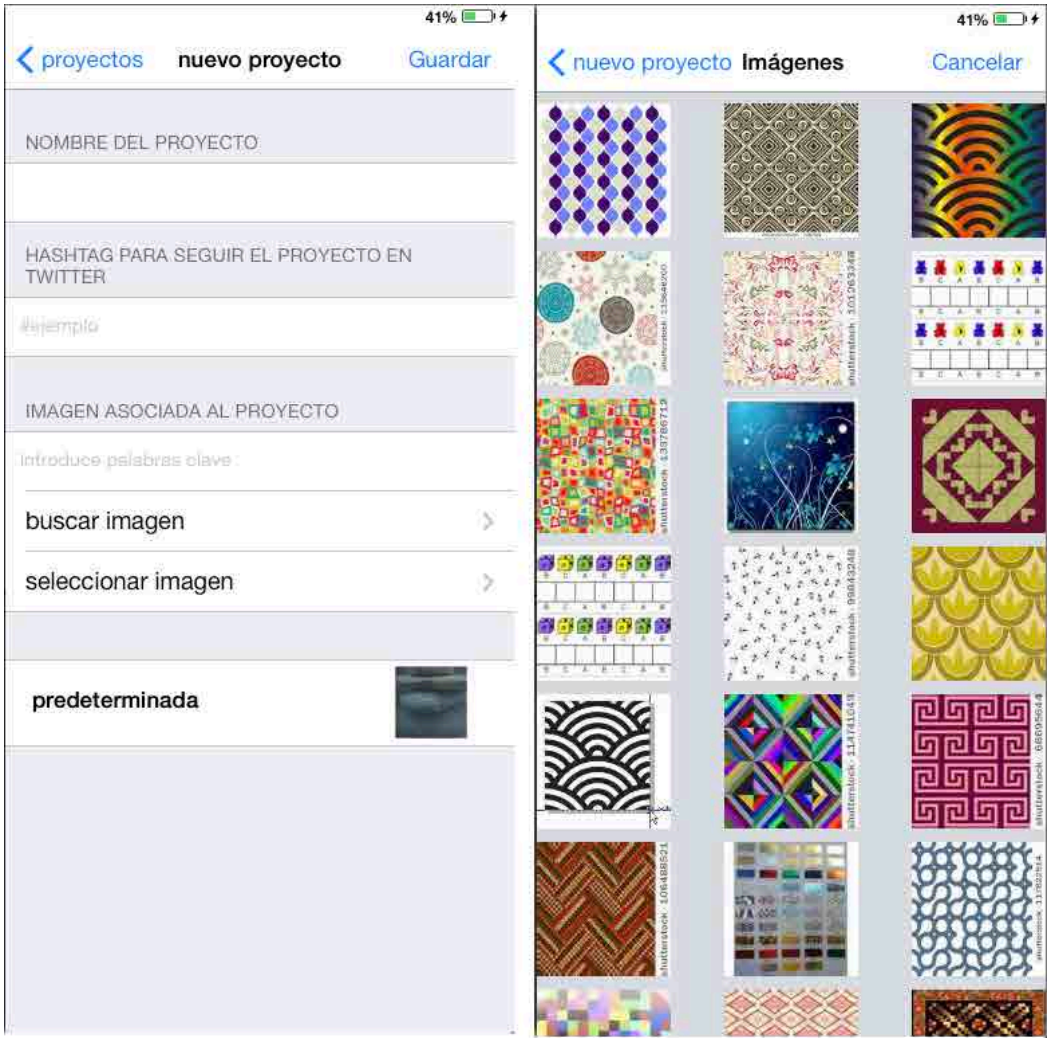

Figura 60: Funcionalidad para asociar un nuevo proyecto a la Definición del Proceso.

4.6.5.2. Herramienta de Clasificación de Recursos

En la figura 61 se muestra la funcionalidad que sirve para añadir un nuevo paso. En el campo "nombre del paso" se debe indicar la actividad que se va a realizar dentro del paso. Mediante el uso de la funcionalidad de Clasificación de Recursos (*Resource Classification Tool*, ver apartado 4.4.6) se debe seleccionar el usuario que debe realizar la actividad.

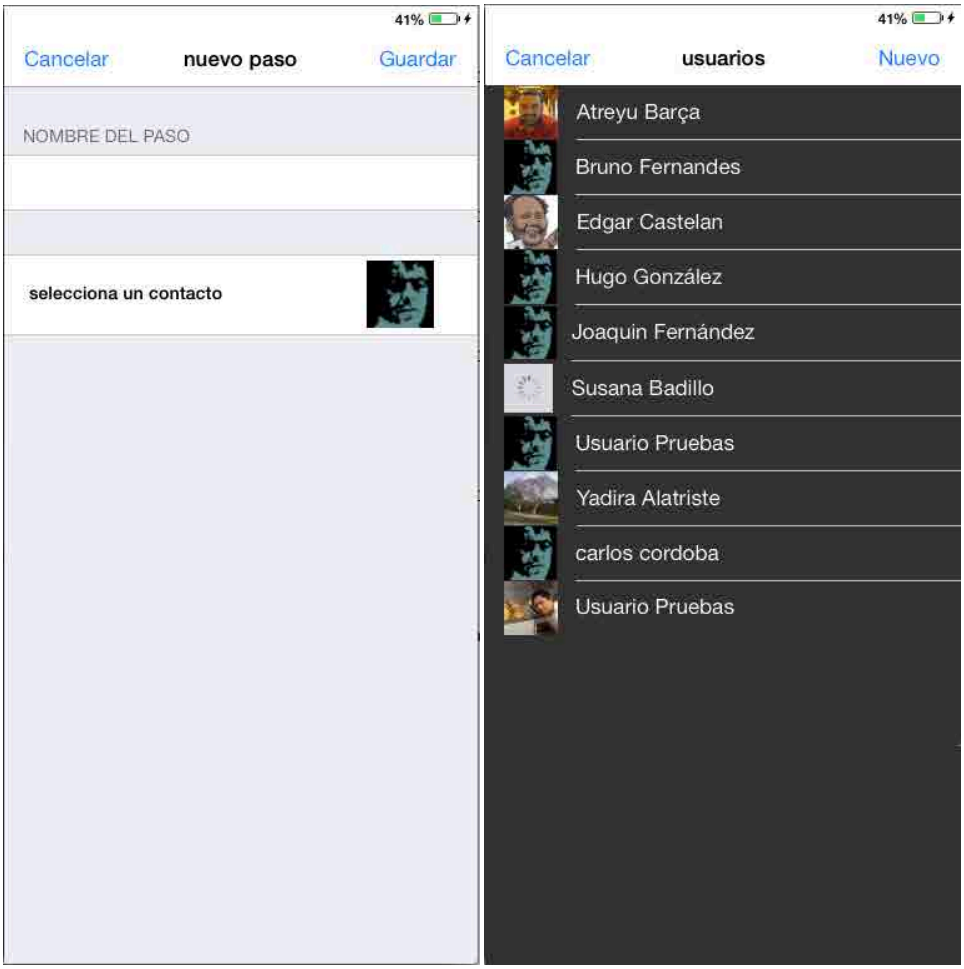

Figura 61: Funcionalidades para añadir un nuevo paso y seleccionar el usuario que debe realizar la actividad.

En caso que sea un usuario no registrado en la aplicación *WOLF* quien deba realizar la actividad, la funcionalidad de Clasificación de Recursos accede al servicio de contactos

que ofrece la plataforma *iOS*, de esta manera el usuario puede seleccionar el *e-mail* del contacto que quiere delegarle la actividad. *Facebook* junto con otras redes sociales (ver fig. 46) están integradas en el sistema operativo *iOS*, esto permite a los usuarios utilizar sus contactos de *Facebook* en aplicaciones de terceros. La aplicación *WOLF* puede tener acceso a los contacto de *Facebook* desde el servicio de contactos de *iOS* (ver fig. 62).

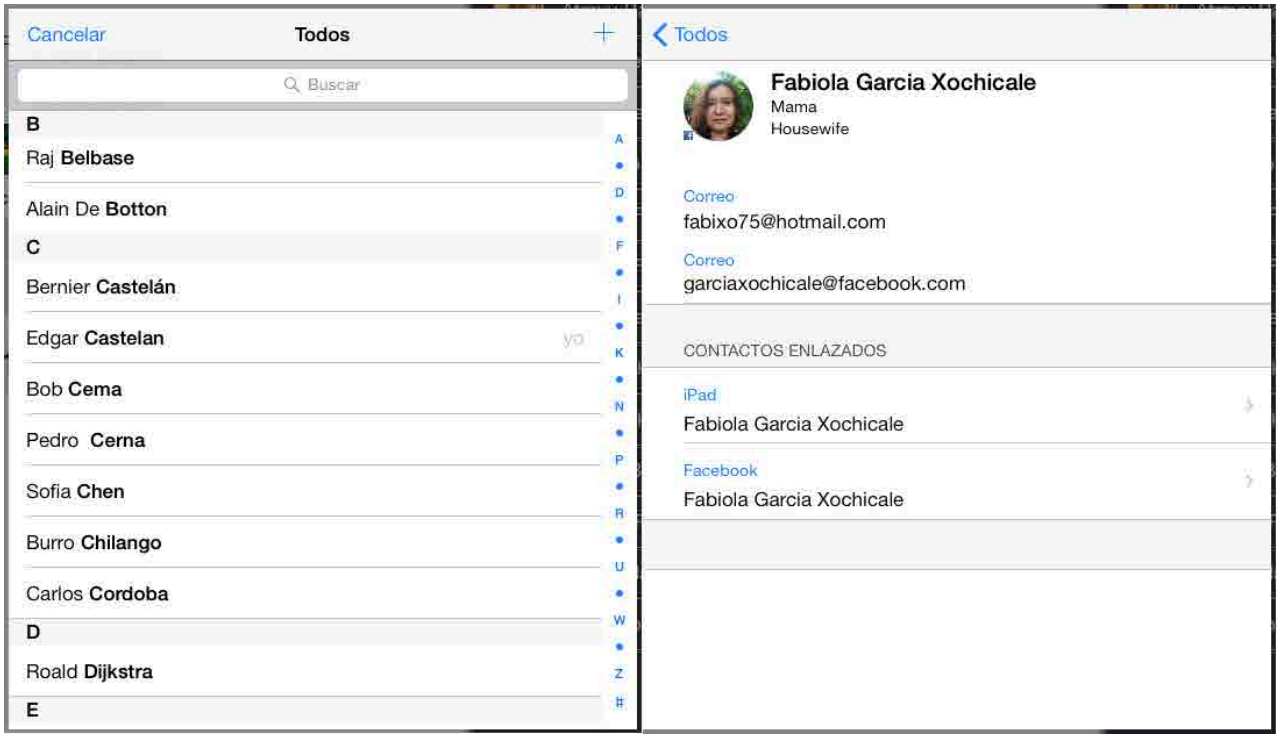

Figura 62: Servicio que accede a los contactos del usuario en su *iPad*.

4.6.6. Herramientas de Notificaciones

Estas herramientas (*Notification Tools* – ver apartado 4.4.10) sirven para mantener al usuario actualizado sobre el estado de los *workflows* y las actividades que realiza dentro de ellos.

4.6.6.1. Servicio de Notificaciones en la Plataforma *iOS*

En la figura 63 se muestra el centro de notificaciones de la plataforma *iOS*, aquí es donde el usuario puede ver las notificaciones que el motor de *workflows* le ha enviado para mantenerlo actualizado.

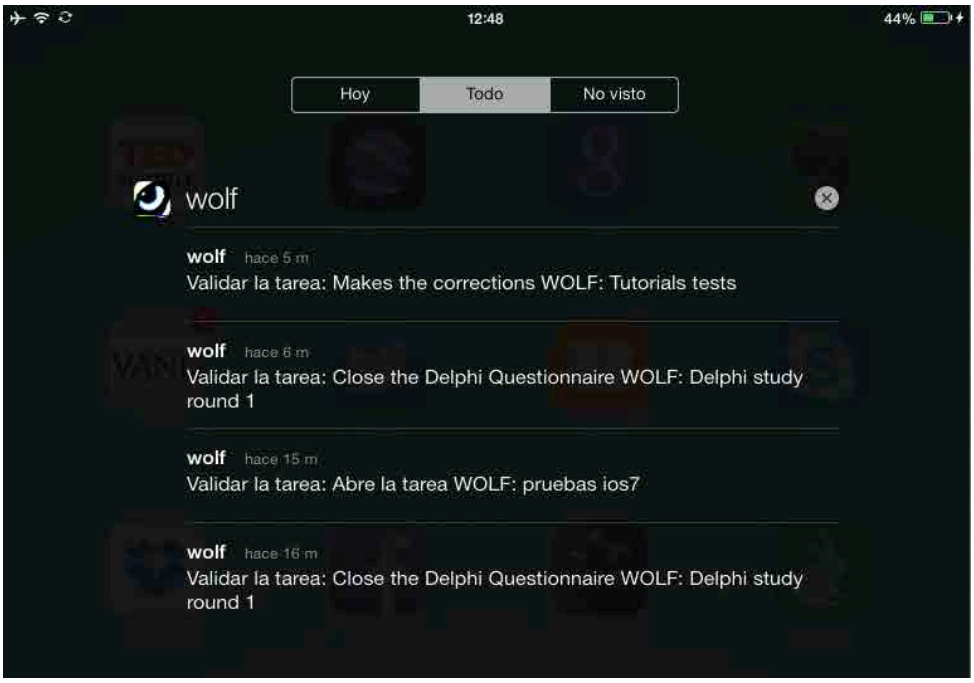

Figura 63: Centro de notificaciones en la plataforma *iOS*.

El usuario puede recibir notificaciones sin importar la actividad que se encuentra realizando, por ejemplo puede estar utilizando la aplicación Mapas en su *iPad* y recibir una notificación de la aplicación *WOLF* (ver fig. 64).

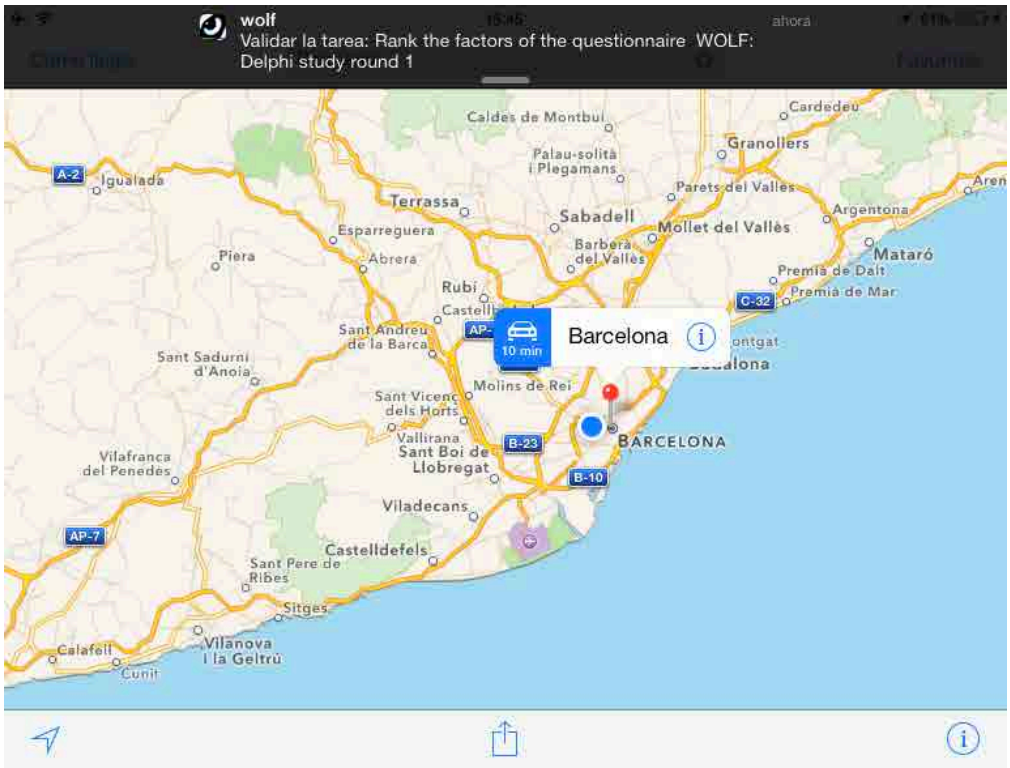

Figura 64: El usuario recibe una notificación de la aplicación *WOLF* dentro de la aplicación Mapas.

Cuando el *iPad* se encuentra apagado y recibe una notificación de la aplicación *WOLF*, la pantalla se enciende y muestra la notificación en la funcionalidad protector de pantalla (ver fig. 65), seguido de una alerta auditiva para atraer la atención del usuario.

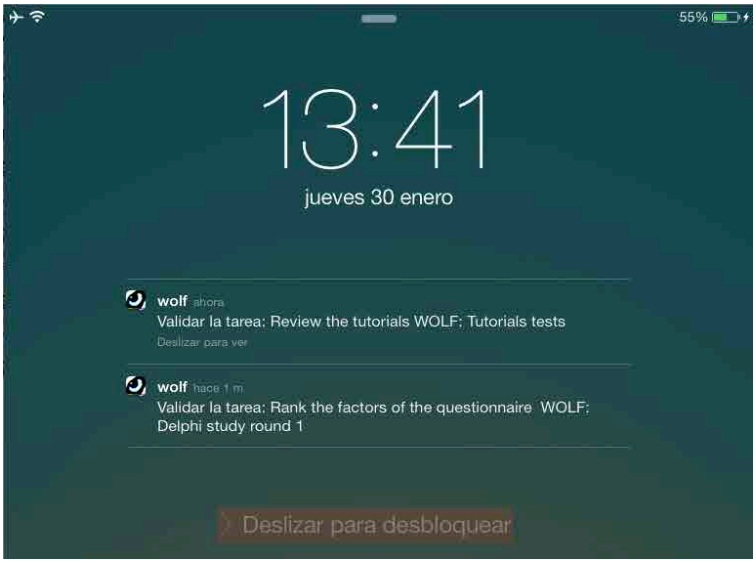

Figura 65: La funcionalidad protector de pantalla muestra una notificación de la aplicación *WOLF*.

El usuario también puede recibir una notificación de un evento importante relacionado con sus *Workflows* cuando está utilizando la aplicación *WOLF* (ver fig. 66).

Figura 66: El usuario recibe una notificación dentro de la aplicación WOLF.

4.6.6.2. Servicio de Notificaciones con Medios Sociales

En la figura 67 se muestra la funcionalidad que muestra los *tweets* asociados con el *hashtag* que ha sido asociado al proyecto que el usuario está consultando en la aplicación *WOLF*, el usuario puede leer todas las notificaciones que los usuarios han dejado en *Twitter* en relación con los *workflows* del proyecto. El usuario puede acceder a esta funcionalidad desde el menú principal de la aplicación *WOLF* presionando el ícono de *Twitter*.

Figura 67: Funcionalidad para ver las notificaciones en *Twitter* para el *hastag* #pruebasWolf.

4.7. Evaluación de las funcionalidades del modelo mediante un estudio *Delphi*

4.7.1. Objetivo del estudio

Evaluar de acuerdo con la opinión (consenso) de expertos, sí las soluciones tecnológicas basadas en los patrones de arquitectura colaborativas, *cloud* y móviles implementados en la arquitectura de software, mejoran las funcionalidades de un *WfMS*.

4.7.2. Participantes

Los Participantes del estudio son expertos en el uso de *WfMS*, herramientas colaborativas, servicios de *cloud computing* y usuarios expertos en la plataforma móvil *iOS*. Los Participantes utilizaron la aplicación *WOLF* por un periodo de tres meses que les permitió adquirir el dominio de las funcionalidades de la aplicación.

4.7.3. Consideraciones Éticas

Los participantes recibieron información de los objetivos del estudio y las implicaciones que tenía participar en el estudio, todos los participantes realizaron el estudio de forma voluntaria y anónima.

4.7.4. Diseño del Estudio

El diseño del estudio *Delphi* está basado en los lineamientos propuestos por Worrell et al. (2013) y por la experiencia empírica de McGinn et al. (2012).

Se identificaron catorce factores tecnológicos en la literatura existente con respecto a los patrones *cloud*, colaborativos y móviles, implementados en la arquitectura de referencia. Se diseñó un cuestionario con preguntas sobre la forma en que estos factores mejoran las funcionalidades de un WfMS (ver anexo A).

Utilizando una escala de *Likert* de cinco puntos (ver fig. 68) los participantes evaluaron cada uno de los factores. Esta escala es la más utilizada en cuestionarios de investigación (Li 2013) y sirve para preguntar a los encuestados su nivel de acuerdo con respecto a la pregunta que se les hace. También se les dio la oportunidad de dejar sus comentarios, relacionados con cada una de las preguntas del cuestionario.

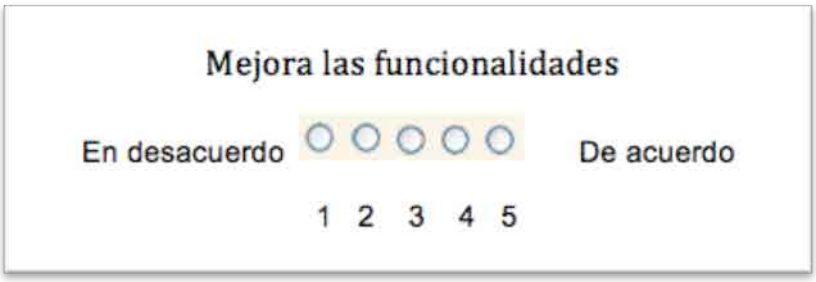

Figura 68: Escala de *Likert* para la evaluación de los factores.

4.7.5. Metodología de un estudio Delphi mediante el uso de Workflows

El estudio *Delphi* se realizó durante un periodo de quince días con tres rondas de cuestionarios, una ronda cada cinco días (ver fig. 69). Los cuestionarios fueron desarrollados utilizando la herramienta *Goolge Drive,* esta herramienta permite crear formularios y compartir el *link* para que otros usuarios puedan acceder y rellenar el formulario. El formulario en este caso contenía las preguntas del estudio *Delphi*, las respuestas de los usuarios se guardaron automáticamente en una hoja de cálculo.

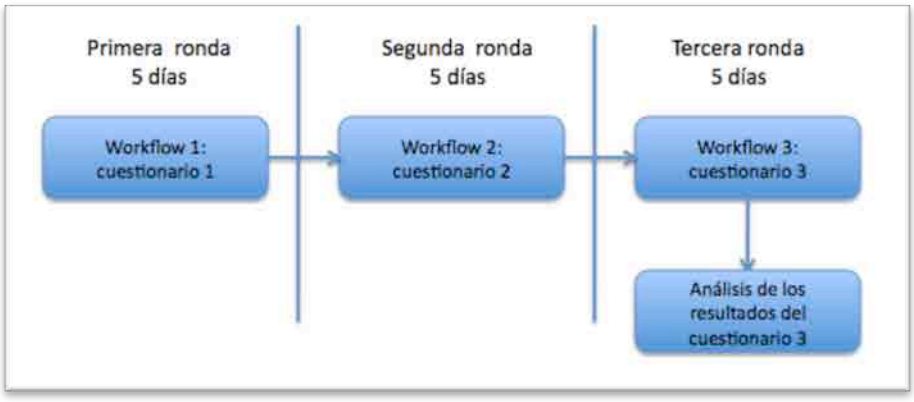

Figura 69: Rondas del estudio Delphi.

El estudio *Delphi* se realizó completamente con la aplicación *WOLF*. En cada una de las rondas se creó un *workflow* con tres pasos (actividades) y con dos usuarios, el Moderador y un Participante (ver figura 70).

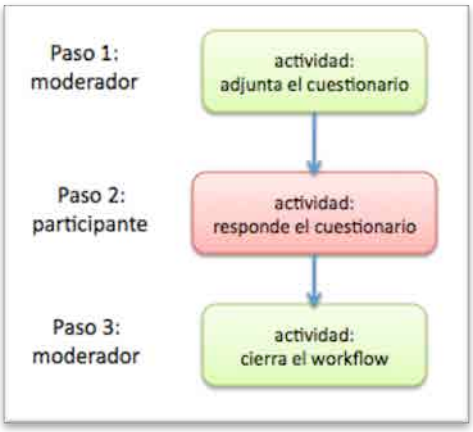

Figura 70: Pasos del *Workflow* para cada una de las rondas del estudio Delphi.

- En el primer paso del *workflow* (ver fig. 71), con la funcionalidad CMS de la aplicación, el Moderador adjuntaba el cuestionario con las instrucciones para realizar el estudio *Delphi*. Presionando el botón validar, el *workflow* avanzaba al paso número dos y el Participante recibía una notificación *push* en su *iPad* indicando que el cuestionario estaba abierto y podía ser rellenado.

- En el segundo paso (ver fig. 71), el Participante respondía el cuestionario y avanzaba el *workflow* al paso siguiente presionando el botón validar. El Moderador encargado de realizar el tercer paso recibía una notificación *push* en su *iPad* indicando que el Participante había respondido el cuestionario.

- En el tercer paso (ver fig. 71), el Moderador confirmaba que todo estaba correcto y cerraba el *workflow* validando el último paso. En caso de que existiera algún problema con el cuestionario, el Moderador podía retroceder el *workflow* un paso atrás presionando el botón denegar, si este fuera el caso el Participante recibía una notificación indicando que la actividad del segundo paso había sido denegada, lo que implicaba hacer la actividad nuevamente (responder el cuestionario).

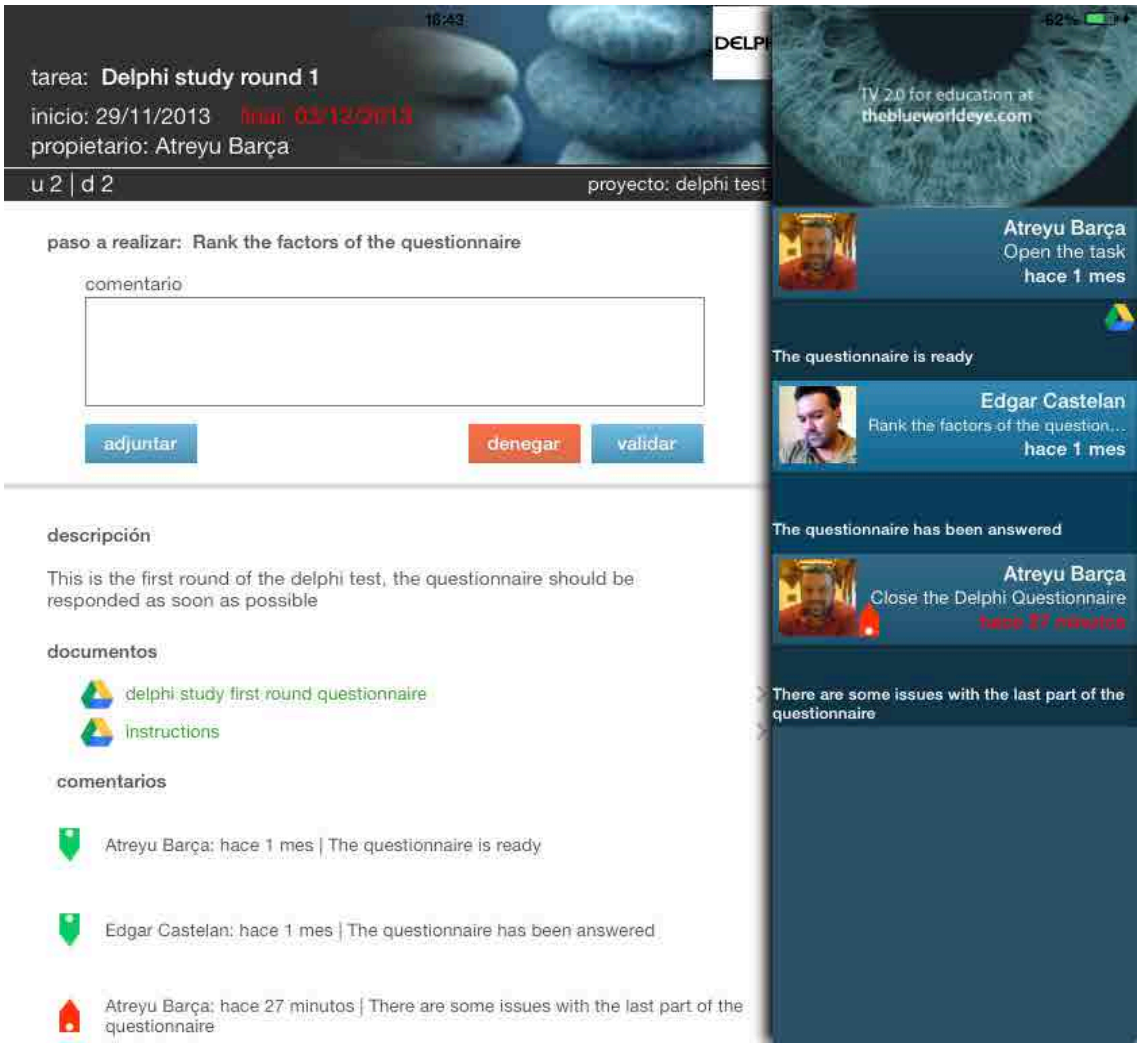

Figura 71: Ejemplo de un *Workflow* para cada una de las rondas del estudio *Delphi*.

En la primera ronda los participantes solo tenían que evaluar los factores que se mostraban en el cuestionario. En la segunda y tercera ronda, la distribución en porcentajes de las respuestas que los Participantes dieron para cada uno de los factores fueron mostradas en forma gráfica, junto con los comentarios hechos por los participantes (ver fig. 72). Esto se hizo con la finalidad de que los participantes tomaran en cuenta las respuestas de los otros participantes y en base a esto reevaluar sus respuestas y decidir si cambiaban, o no, su opinión.

Figura 72: Cuestionario para el estudio *Delphi* de la segunda ronda y resultados de la primera ronda.

4.7.6. Análisis

Los datos resultantes del cuestionario de la tercera ronda fueron utilizados para el análisis de cada uno de los factores y así determinar el tipo de consenso alcanzado en cada uno de ellos. En la tabla 5 se muestran los tipos de consenso propuestos por (McGinn et al. 2012) y que son utilizados en este estudio.

Tipo de consenso	Acuerdo en porcentaje
Alto	75% de los participantes o más, están de acuerdo
Moderado	más del 60% de los participantes están de acuerdo
Parcial	60% de los participantes están de acuerdo
Ausencia	menos del 60% de los participantes están de acuerdo

Tabla 5: Tipos de consenso, fuente (McGinn et al. 2012).

Para el análisis de resultados se utilizó la medida de estadística descriptiva conocida como Percentil y la medida de dispersión estadística conocida como Rango-Intercuartil.

El Nivel de consenso se puede medir obteniendo cual es el número de la escala de *Likert* donde los expertos alcanzaron consenso, teniendo en cuenta que 1 es estar en desacuerdo y 5 de acuerdo, cuanto más se acerca a 5 mejor nivel de consenso se ha obtenido.

Para obtener el Nivel de consenso se debe calcular los percentiles: P_{10} (décimo percentil) y P_{25} (veinticincoavo percentil). El P_{10} indica cual fue el número más bajo en la escala de *Likert* que el 90% de los participantes eligió para evaluar un factor. El P_{25} indica cual fue el número en la escala de *Likert* que el 75% de los participantes eligió.

Para saber la Fuerza que tiene el consenso se debe calcular el rango inter-cuartil, este sirve para medir estadísticamente que tan dispersas están las respuestas de los participantes. Un consenso se puede decir que es fuerte cuando el resultado del rango inter-cuartil da un valor de 0, y pierde fuerza alejándose de este valor de tal forma que para el valor 2 ya se considera que la fuerza es débil.

Por ejemplo, para el factor Servicios Móviles (ver tabla 6) de acuerdo con el percentil P_{10} el 90% de los participantes respondieron 4 o 5 en la escala de *Likert* de 5 puntos, mientras que el 75% de los participantes respondieron 5. El resultado del rango inter-cuartil es 0 lo que indica que el consenso es fuerte.

Caso contrario es el del factor Redes Sociales Abiertas, de acuerdo con el percentil P_{10} el 90% de los participantes respondieron 2, 3, 4 o 5 en la escala de *Likert* de 5 puntos, mientras que el 75% de los participantes respondieron 3, 4 o 5. El rango inter-cuartil es de 1 debido a que las respuestas de los participantes son dispersas.

4.7.7. Resultados del Estudio Delphi

En la tabla 6 se muestran los resultados de la evaluación de los catorce factores. Doce factores alcanzaron consenso de tipo Alto (> 75% de los participantes estuvieron de acuerdo). El factor Elasticidad alcanzo un consenso de tipo Moderado (> 60%). Solo el factor Redes Sociales Abiertas no obtuvo consenso (< 60%).

El nivel de consenso de diez factores fue de un nivel alto ($P_{25} = 5$, en escala de 1 a 5), el más alto nivel que se podía alcanzar. Tres factores obtuvieron un nivel aceptable ($P_{25} = 4$, en escala de 1 a 5).

La fuerza del consenso de los trece factores que alcanzaron algún tipo de consenso ha sido fuerte (rango inter-cuartil = 0). El factor Redes Sociales Abiertas obtuvo un resultado débil en lo que se refiere a la fuerza del consenso (rango inter-cuartil = 1), esto se debe a que las respuestas de los participantes fueron dispersas y por lo tanto no se logró alcanzar algún tipo de consenso.

Factores	Consenso - (acuerdo en %)	5-point Likert scort	Percentil P_{10}	P_{25}	Rango Inter-cuartil
Colaborativos					
Colaboración	Alto (> 75 %)	5	5	5	0
Comunicación	Alto (> 75 %)	5	5	5	0
Coordinación	Alto (> 75 %)	5	5	5	0
Redes Sociales Abiertas	Ausencia (< 60%)	-	2	3	1
Single Sign On	Alto (> 75 %)	5	5	5	0
Cloud					
Disponibilidad	Alto (> 75 %)	5	4	5	0
Elasticidad	Moderado (> 60 %)	4	3	4	0
Cloud Services	Alto (> 75 %)	4	4	4	0
Consistencia	Alto (> 75 %)	5	4	5	0
Sincronización	Alto (> 75 %)	5	5	5	0
Móviles					
Mobilidad	Alto (> 75 %)	5	5	5	0
Ubicuidad	Alto (> 75 %)	5	5	5	0
Comunicación	Alto (> 75 %)	4	3	4	0
Servicios Móviles	Alto (> 75 %)	5	4	5	0

Tabla 6: Resultados del estudio *Delphi*.

4.7.8. Interpretación de los Resultados

Utilizando el método *Delphi* se ha logrado obtener la opinión y el consenso de los participantes para evaluar los Factores Tecnológicos propuestos en el cuestionario. De acuerdo con los resultados de la tabla 5, los participantes han evaluado satisfactoriamente más del 90% de los factores tecnológicos, además de alcanzar consenso, este obtuvo un alto nivel de en cada uno de los factores, lo quiere decir que además de estar de acuerdo, los participantes le dieron la puntuación más alta a la mayoría de los Factores Tecnológicos.

El único Factor donde los participantes no lograron alcanzar consenso ha sido en el factor Redes Sociales Abiertas, de acuerdo con la opinión de los participantes, esto se puede deber a que hoy en día la mayoría de las Redes Sociales son más para uso personal que para uso profesional, por lo que las soluciones tecnológicas que ofrecen las Redes Sociales no son de mucha ayuda para las funcionalidades de un WfMS. Las Redes Sociales de tipo profesional de momento no están abiertas y no ofrecen las soluciones tecnológicas para proveer el tipo de servicios de las Redes Sociales de uso personal como *Facebook.*

En base a los resultados anteriores se puede llegar a la conclusión que las soluciones tecnológicas para los factores:

a) Colaborativos (Hansen et al. 2011; Kim et al. 2010; Riesner et al. 2013; Saucedo-Tejada et al. 2013; Neyem et al. 2012; Smari et al. 2013; Li et al. 2013; Herskovic et al. 2009; Rodríguez-Covili et al. 2011).

b) *Cloud* (Subashini & Kavitha 2011; Mell & Grance 2011; Esayas 2012; Aceto et al. 2013; Xu 2012; Park & Ryoo 2013; Fernando et al. 2013; Azeemi et al. 2013; Harman et al. 2013).

c) Móviles (Mühl et al. 2006; Sallam & Siba 2011; Arroqui et al. 2012; Bouwman et al. 2012; Pura & Heinonen 2008).

Basados en los patrones de arquitectura colaborativos (Hansen et al. 2011; Neyem et al. 2012; Herskovic et al. 2009), *cloud* (Liu et al. 2011; Mikkonen & Taivalsaari 2013; Dinh & Lee 2011) y móviles (Friese 2012; Flautero 2012), implementados en la arquitectura de *software,* mejoran las funcionalidades de un WfMS.

4.7.9. Mejoras del estudio

El estudio *Delphi* que aquí se presenta es una primera aproximación para la evaluación de los factores tecnológicos, el estudio tiene ciertas limitaciones debido a que los factores tecnológicos provienen de tres campos de aplicación diferentes, debido a ello cabe la posibilidad de que algún participante no posea los mismos niveles de

conocimientos y experiencia en todos los factores. Se podría mejorar este estudio *Delphi* con grupos de expertos en cada uno de los campos de aplicación tecnológicos a los que pertenecen los factores y de esta forma reforzar los resultados obtenidos.

Capítulo 5

Conclusiones

5. Conclusiones

Este trabajo de investigación se realizó en el contexto de WfMS (*Workflow Management Systems*), aplicaciones móviles, *cloud computing* y sistemas colaborativos.

La investigación plantea una resolución al problema de que el diseño de WfMS estaba basado en el modelo de referencia propuesto por la WfMC (*Workflow Management Coalition*)[58], este modelo es una descripción general de la arquitectura que debe tener un WfMS, describe las funcionalidades de los componentes de software, que son parte esencial en un WfMS y la forma en que deben interactuar entre ellos. Sin embargo el modelo de referencia propuesto por la WfMC fue diseñado muchos años antes de que surgierán las tecnologías móviles, *cloud computing* y los sistemas colaborativos que hacen uso de estas tecnologías.

Es importante tener un nuevo modelo para el diseño de WfMS que tenga en cuenta las nuevas características y funcionalidades tecnológicas que ofrecen los dispositivos móviles, los servicios colaborativos y de cloud computing, teniendo en cuenta las nuevas formas de colaboración que se dan al utilizar estas soluciones tecnológicas.

El diseño y desarrollo de nuevos WfMS debe tener en cuenta las características propias de los dispositivos y aplicaciones móviles que permitan al usuario trabajar en diferentes contextos, utilizar información relacionada con el contexto, utilizar servicios en la nube: de procesamiento, comunicación, almacenamiento, colaboración, medios sociales, etc.

Debido al impacto social de estas nuevas tecnologías se considera este este trabajo de investigación reelevante en el campo de WfMS. Este trabajo de investigación aporta una solución para resolver el problema antes mecionado.

El objetivo principal de esta investigación ha sido, por tanto, obtener un modelo tecnológico para el diseño y desarrollo WfMS con funcionalidades Colaborativas, *Cloud* y Móviles.

[58] http://www.wfmc.org/

Los objetivos específicos alcanzados en esta investigación fueron los siguientes:

- Se propuso un Marco Teórico-Tecnológico basado en el mapa conceptual de la investigación.

- Se diseñó un modelo tecnológico de WfMS con funcionalidades Colaborativas, *Cloud* y Móviles.

- Se realizó la implementación del modelo tecnológico para obtener una arquitectura de software que sirvió para desarrollar aplicaciones móviles de WfMS.

- Se desarrolló una aplicación móvil basada en la implementación de la arquitectura de software.

- Se evaluaron las funcionalidades del modelo tecnológico con expertos en el uso de WfMS mediante un estudio *Delphi*.

Para el desarrollo de esta investigación utilizamos el paradigma de investigación Diseño-Ciencia utilizado en el campo de investigación de los sistemas de información. Está completamente orientado a la solución de problemas y tiene como meta principal la creación y evaluación de artefactos que sirvan para un propósito práctico, estos artefactos deben ser completamente relevantes y novedosos en su entorno de aplicación (Nicolaou & Geerts 2011).

En la figura 74 se pueden ver los tres ciclos inherentes a la investigación diseño-ciencia en el marco de investigación de los sistemas de información propuesto por (Hevner 2007), y que sirven para entender, ejecutar y evaluar investigaciones.

El “ciclo de relevancia” sirve de puente entre el ámbito contextual del proyecto de investigación con las actividades de investigación del diseño-ciencia. El “ciclo de rigor” conecta las actividades del diseño-ciencia con los fundamentos científicos de la base de conocimiento. El “ciclo de diseño” es iterativo entre las actividades principales para construir y evaluar, el diseño de artefactos y procesos de la investigación (Hevner & Chatterjee 2010). Ver el apartado 2.1 para más detalle de la implementación de cada uno de los ciclos.

A continuación hacemos un resumen de los pasos que se llevaron a cabo para realizar la investigación con la metodología diseño-ciencia, en la figura 73 se puede ver en cuál de los ciclos de la investigación diseño-ciencia se realizaron cada uno de los siguientes pasos.

1) Se identifica un problema en el ámbito de los WfMS.

2) Se propusieron las características que debía tener el artefacto que se necesitaba para solucionar el problema y se seleccionó la forma en que iba a ser representado el artefacto, en este caso un modelo.

3) Se identificaron y propusieron los procesos de diseño que deberían ser utilizados para construir el artefacto.

4) Se justificó teóricamente, con metodologías y modelos ampliamente aceptados en el campo de sistemas de información, el artefacto y el proceso de diseño.

5) Durante el ciclo de diseño hemos utilizado un modelo de evaluación de arquitecturas de software.

6) Con la finalidad de introducir el artefacto dentro del campo de aplicación hemos realizado una implementación del modelo dando como resultado final una aplicación móvil de WfMS con funcionalidades colaborativas, *cloud* y móviles. Hemos llevado a cabo un estudio *Delphi* para evaluar las funcionalidades del nuevo artefacto y demostrar su utilidad en su campo de aplicación.

7) El resultado de esta investigación añade a la base de conocimiento un nuevo modelo que sirve para el diseño y desarrollo de WfMS con funcionalidades colaborativas, *cloud* y móviles. Se ha escrito un artículo para la comunidad científica donde se publican los resultados de esta investigación (ver apartado 5.4).

8) El objetivo principal y todos los objetivos propuestos en esta investigación se han cumplido satisfactoriamente, hemos logrado demostrar que el artefacto diseñado resuelve el problema planteado en esta investigación y que este artefacto provee utilidad en el campo de los WfMS.

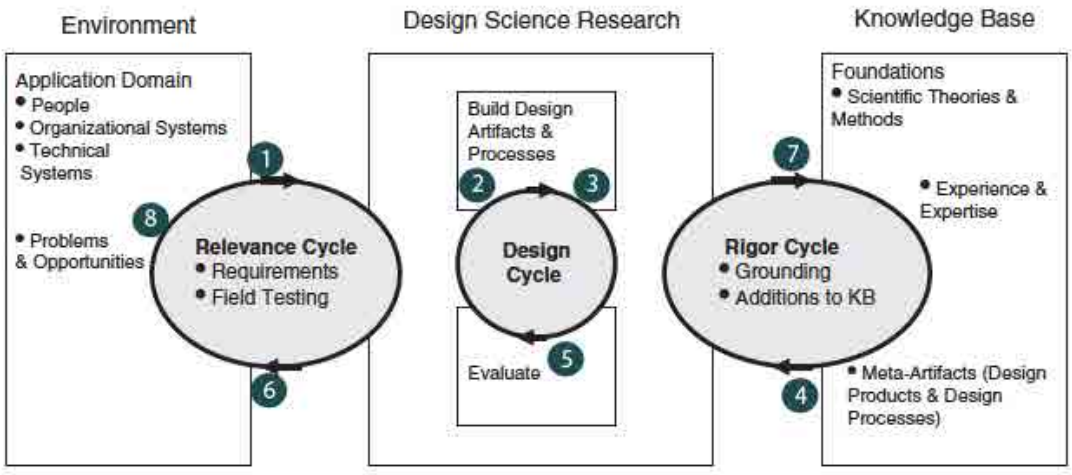

Figura 73: Relación entre lineamientos y los ciclos del Diseño-Ciencia, fuente (Hevner & Chatterjee 2010).

5.1. Conclusiones sobre el Modelo aportado

Se ha logrado obtener un Modelo para el diseño y desarrollo de WfMS con funcionalidades colaborativas, *cloud* y móviles. Esta es la principal aportación de este trabajo de investigación.

En la figura 75 se puede ver el proceso que se siguió en esta investigación para crear el modelo y la implementación de este modelo en sucesivas arquitecturas hasta llegar a la obtención de una aplicación móvil de WfMS con funcionalidades colaborativas, *cloud* y móviles.

El nuevo modelo (arquitectura de referencia en la figura 74) fue el resultado de transponer las funcionalidades del modelo de referencia de WfMS (ver apartado 4.3), junto con patrones de arquitectura móviles, *cloud* y colaborativos (ver apartados 4.3.1, 4.3.2 y 4.3.3).

Este nuevo modelo servirá para crear en diferentes contextos y campos de aplicación, arquitecturas de software para desarrollar aplicaciones de WfMS con funcionalidades colaborativas, *cloud* y móviles (ver fig. 75).

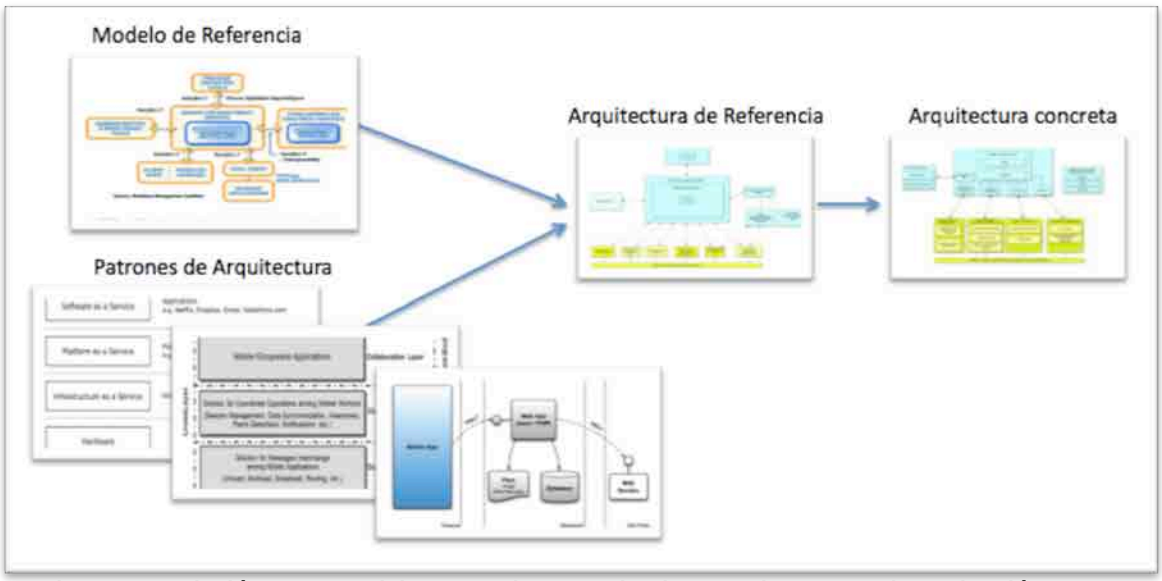

Figura 74: Relación entre modelos y arquitecturas implementadas en esta investigación.,

La implementación del nuevo modelo dio como resultado una Arquitectura Concreta, una Arquitectura Concreta está diseñada para un contexto específico y refleja los objetivos del negocio en concreto. En este caso el contexto fue el laboratorio de aplicaciones multimedia, dónde existía la necesidad de desarrollar aplicaciones de WfMS con funcionalidades colaborativas, *cloud* y móviles para el grupo de investigación del doctorado en multimedia.

Esta Arquitectura Concreta va a servir como plantilla para diseñar diferentes Arquitecturas de Software para el desarrollo de WfMS con las funcionalidades que se especifican en la arquitectura concreta.

En la figura 75 se muestran las arquitecturas resultado de la implementación del modelo de WfMS.

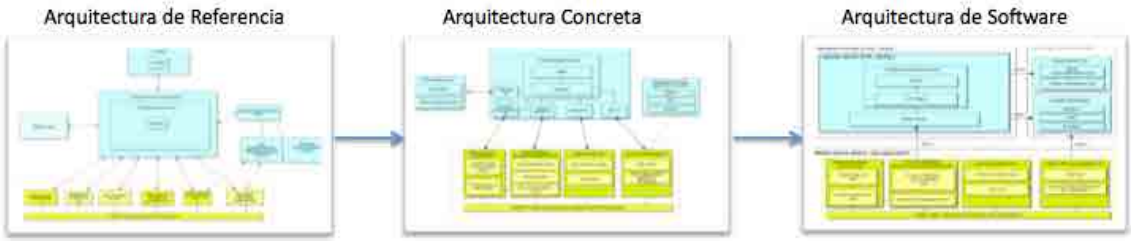

Figura 75: Relación entre las arquitecturas implementadas en esta investigación.

La implementación de la Arquitectura Concreta dio como resultado una Arquitectura de Software para desarrollar aplicaciones móviles nativas de WfMS. La Arquitectura de Software contiene el detalle todas las funcionalidades colaborativas, *cloud* y móviles de un WfMS.

La Aplicación Móvil fue el resultado de la implementación de la Arquitectura de Software, esta aplicación en particular fue desarrollada para la plataforma *iOS* y se ejecuta en los dispositivos *iPad*. La Aplicación Móvil ha sido publicada en la tienda de *Apple* con el nombre *WOLF* (*Work Linear Flow*) y puede ser utilizada gratuitamente por cualquier usuario u organización con la necesidad gestionar *Workflows*.

5.2. Conclusiones sobre el estudio *Delphi*

El objetivo del estudio *Delphi* fue evaluar, en base a la opinión de los participantes, si las soluciones tecnológicas basadas en los patrones de arquitectura colaborativas, *cloud* y móviles implementados en la arquitectura de software, mejoran las funcionalidades de un WfMS.

Se identificaron catorce factores tecnológicos en la literatura existente con respecto a los patrones *cloud*, colaborativos y móviles, implementados en el modelo. Los participantes evaluaron cada uno de los factores.

Los participantes tenían una amplia experiencia en el uso de WfMS y sus funcionalidades. Evaluaron satisfactoriamente más del 90% de los factores tecnológicos, además de alcanzar consenso se obtuvo un alto nivel de consenso en cada uno de los factores, esto quiere decir que además de estar de acuerdo, los participantes le dieron la puntuación más alta a la mayoría de los factores tecnológicos.

En base a los resultados obtenidos en el estudio *Delphi* se pudo llegar a la conclusión que las soluciones tecnológicas para los factores:

a) colaborativos (Hansen et al. 2011; Kim et al. 2010; Riesner et al. 2013; Saucedo-Tejada et al. 2013; Neyem et al. 2012; Smari et al. 2013; Li et al. 2013; Herskovic et al. 2009; Rodríguez-Covili et al. 2011).

b) *cloud* (Subashini & Kavitha 2011; Mell & Grance 2011; Esayas 2012; Aceto et al. 2013; Xu 2012; Park & Ryoo 2013; Fernando et al. 2013; Azeemi et al. 2013; Harman et al. 2013).

c) móviles (Mühl et al. 2006; Sallam & Siba 2011; Arroqui et al. 2012; Bouwman et al. 2012; Pura & Heinonen 2008).

basados en los patrones de arquitectura colaborativos (Hansen et al. 2011; Neyem et al. 2012; Herskovic et al. 2009), *cloud* (Liu et al. 2011; Mikkonen & Taivalsaari 2013; Dinh & Lee 2011) y móviles (Friese 2012; Flautero 2012), implementados en la arquitectura de *software,* mejoran las funcionalidades de un WfMS.

En esta investigación se ha presentado una nueva metodología para realizar un estudio *Delphi* mediante el uso de la aplicación móvil de WfMS que fue desarrollada en esta investigación.

En esta metodología cada uno de los cuestionarios del estudio *Delphi* se hicieron a través de un Wokflow y se hizo uso de herramientas colaborativas en la nube para almacenar tanto los cuestionarios así como también los resultados de la evaluación de los cuestionarios. Realizar el estudio *Delphi* desde el dispositivo móvil *iPad*, permite realizar la evaluación de los cuestinoario sin importar el lugar y la hora. El proceso de comunicación entre el moderador y el experto se da de una forma más estructurada y directa.

5.3. Aportaciones de la investigación

Todos los objetivos propuestos en esta investigación se han cumplido satisfactoriamente, siguiendo los requerimientos de la metodología de investigación diseño-ciencia. Se ha diseñado un artefacto, se ha comprobado que este artefacto resuelve el problema planteado en esta investigación y se ha demostrado que el artefacto provee utilidad en el campo de los WfMS. Al satisfacer estos puntos claves de la investigación diseño-ciencia, podemos llegar a la conclusión que hemos logrado hacer una aportación a la base de conocimiento de los WfMS que aporta una solución a la necesidad que había de contar con un modelo para el diseño de WfMS con funcionalidades colaborativas, cloud y móviles en el campo de los WfMS. La principal aportación de esta investigación es:

- Un modelo para el diseño de WfMS con funcionalidades colaborativas, cloud y móviles.

Adicionalmente esta investigación también hace las siguientes aportaciones:

- Una arquitectura concreta que servirá para el diseño de arquitecturas de software de WfMS basada en un contexto de aplicación específico.
- Una arquitectura de software para el desarrollo de aplicaciones móviles nativas de WfMS.
- Una aplicación móvil de WfMS para la plataforma *iOS* y dispositivos *iPad* resultado de la implementación de la arquitectura de software.
- Una metodología para ejecutar un estudio *Delphi* utilizando una aplicación móvil de WfMS con herramientas colaborativas en la nube.

5.4. Comunicación de la investigación

Los resultados de la investigación fueron presentados en forma de artículo de investigación en el siguiente congreso:

- Castelán, E., Brigos, M. A., & Fernández, J. (2014). *A SOFTWARE REFERENCE ARCHITECTURE FOR THE DESIGN AND DEVELOPMENT OF MOBILE WORKFLOW LEARNING APPLICATIONS. In 8th International Technology, Education and Development Conference Valencia - 10th - 12th March 2014.*

Se están escribiendo los siguientes artículos para ser presentados en revistas indexadas:

- *A REFERENCE MODEL FOR THE DESIGN AND DEVELOPMENT OF MOBILE WFMS.*

- *THE PRACTICAL CASE OF A DELPHI STUDY WITH MOBILE WfMS AND CLOUD COLABORATION TOOLS*

5.5. Limitaciones de la investigación

Este trabajo de investigación fue desarrollado en el contexto de WfMS, aplicaciones móviles, *cloud computing* y sistemas colaborativos, realizar esta investigación representó un gran reto debido a que cada una de estas tecnologías tiene su propio campo de estudio y aplicación.

Solo se realizó una implementación del modelo en un contexto y campo de aplicación, por lo que pudiera darse el caso que no hubiera un escenario adecuado que permitiera evaluar por completo las funcionalidades del modelo.

Seleccionar la forma adecuada de evaluar el artefacto en su campo de aplicación fue complicado debido a las diferentes metodologías que existen para realizar la evaluación del artefacto, se deben tomar en cuenta otras metodologías de evaluación en futuras investigaciones que refuercen la evaluación del artefacto.

El estudio *Delphi* realizado en esta investigación tiene ciertas limitaciones debido a que los factores tecnológicos provienen de tres campos de aplicación diferentes, debido a esto es posible que los expertos no tuvieran la experiencia necesaria para poder evaluar alguno de los factores.

5.6. Futuras investigaciones

Es posible desarrollar futuras investigaciones que mejoren y hagan nuevas aportaciones a la base de conocimiento de los WfMS tomado como base este trabajo de investigación, a continuación proponemos una serie de investigaciones se pueden llevar a cabo.

- Cada una de las nuevas funcionalidades del modelo son tecnologías nuevas e innovadoras en su campo de aplicación, se deben realizar futuras investigaciones por separado enfocadas a cada una de las funcionalidades del modelo con la finalidad de mejorar el modelo y sus funcionalidades.
- Se deben realizar futuras implementaciones del modelo en diferentes contextos y campos de aplicación y de esta forma poder hacer nuevas evaluaciones del modelo.
- Se deben realizar estudios *Delphi* con grupos de expertos en cada uno de los campos de aplicación tecnológicos que han sido aplicados en esta investigación y de esta forma reforzar los resultados obtenidos.

La aplicación móvil de WfMS tiene su propia línea de investigación, ya que es posible realizar investigaciones centradas en el uso de la aplicación móvil, a continuación se proponen las posibles investigaciones.

- Proponer una serie de indicadores para realizar investigaciones relacionadas con productividad y eficiencia.
- Implementar la aplicación móvil como un sistema para la evaluación de la calidad de las tareas y actividades que se hacen dentro de los *Workflows*.
- Implementar la aplicación móvil en el campo de WfLMS (*Workflow Learning Management Systems*) y ejecutar *Workflows* en ambientes educativos y de aprendizaje.

La metodología que se propone en esta investigación sobre cómo realizar un estudio *Delphi* utilizando la aplicación móvil de WfMS con herramientas colaborativas en la nube también es una futura línea de investigación, se deben realizar investigaciones

relacionadas con la aplicación de WfMS como herramientas para la realización estudios *Delphi*.

Referencias Bibliográficas

6. Referencias bibliográficas

Aalst, W. van der & Hee, K. van, 2004. *Workflow Management: Models, Methods, and Systems (Cooperative Information Systems series)*, The MIT Press. Available at: http://www.amazon.com/*Workflow*-Management-Methods-Cooperative-Information/dp/0262720469 [Accessed August 27, 2013].

Aalst, W. van der. & Kumar, A., 2001. A Reference Model for Team-Enabled *Workflow* Management Systems. *Data & Knowledge Engineering*, 38(3), pp.335–363.

Aceto, G. et al., 2013. Cloud monitoring: A survey. *Computer Networks*, 19(9), pp.2093–2115.

Achilleos, A., Yang, K. & Georgalas, N., 2010. Context modelling and a context-aware framework for pervasive service creation: A model-driven approach. *Pervasive and Mobile Computing*, 6(2), pp.281–296.

Albin, S.T., 2003. *The Art of Software Architecture: Design Methods and Techniques*, Wiley. Available at: http://www.amazon.com/The-Art-Software-Architecture-Techniques/dp/0471228869 [Accessed September 19, 2013].

Alonso, G. et al., 1997. Functionality and Limitations of Current *Workflow* Management Systems. *IEEE Expert*, 12.

Angelov, S., Grefen, P. & Greefhorst, D., 2012. A framework for analysis and design of software reference architectures. *Information and Software Technology*, 54(4), pp.417–431.

Arroqui, M. et al., 2012. RESTful *Web* Services improve the efficiency of data transfer of a whole-farm simulator accessed by *Android* smartphones. *Computers and Electronics in Agriculture*, 87(0), pp.14–18.

Azeemi, I.K., Lewis, M. & Tryfonas, T., 2013. Migrating To The Cloud: Lessons And Limitations Of “Traditional” IS Success Models. *Procedia Computer Science*, 16(0), pp.737–746.

Bae, S., Jang, J. & Kim, J., 2013. Good Samaritans on social network services: Effects of shared context information on social supports for strangers. *International Journal of Human-Computer Studies*, 71(9), pp.900–918.

Bala, A. & Chana, I., 2012. Design and Deployment of *Workflows* in Cloud Environment. *International Journal of Computer Applications*, 51(11), pp.0975–887.

Balasubramanian, S., Peterson, R.A. & Jarvenpaa, S.L., 2002. Exploring the implications of m-commerce for markets and marketing. *Journal of the Academy of Marketing Science*, 30(4), pp.348–361.

Basole, R.C., 2008. Visualization of Interfirm Relations in a Converging Mobile Ecosystem. In *2008 7th International Conference on Mobile Business*. IEEE, pp. 65–74.

Bass, L., Clements, P. & Kazman, R., 2012. *Software Architecture in Practice (3rd Edition) (SEI Series in Software Engineering)*, Addison-Wesley Professional. Available at: http://www.amazon.com/Software-Architecture-Practice-Edition-Engineering/dp/0321815734 [Accessed September 19, 2013].

Bertram, J. & Kleiner, C., 2012. Secure *Web* Service Clients on Mobile Devices. *Procedia Computer Science*, 10(0), pp.696–704.

Bosch, J., 2009. From software product lines to software ecosystems. In *Proceedings of the 13th International Software Product Line Conference, Carnegie Mellon University, Pittsburgh, PA, USA*. pp. 111–119.

Bouwman, H., Bejar, A. & Nikou, S., 2012. Mobile services put in context: A Q-sort analysis. *Telematics and Informatics*, 29(1), pp.66–81. Available at: http://www.sciencedirect.com/science/article/pii/S0736585311000256 [Accessed October 4, 2013].

Buschmann, F. et al., 1996. *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*, Wiley. Available at: http://www.amazon.com/Pattern-Oriented-Software-Architecture-Volume-Patterns/dp/0471958697 [Accessed September 20, 2013].

Chen, M.-C., Chen, J.-L. & Chang, T.-W., 2011. *Android*/OSGi-based vehicular network management system. *Computer Communications*, 34(2), pp.169–183.

Choi, D., Kim, N. & Hung, D.T., 2012. Conceptual data modeling for realizing context-aware services. *Expert Systems with Applications*, 39(3), pp.3022–3030.

Clements, P. et al., 2010. *Documenting Software Architectures: Views and Beyond (2nd Edition)*, Addison-Wesley Professional. Available at: http://www.amazon.com/Documenting-Software-Architectures-Beyond-Edition/dp/0321552687 [Accessed September 19, 2013].

Corral, L., Sillitti, A. & Succi, G., 2012. Mobile Multiplatform Development: An Experiment for Performance Analysis. *Procedia Computer Science*, 10(0), pp.736–743.

Datta, L., 2010. A Taxonomy of Collaboration Tools. Available at: http://allcollaboration.com/home/2010/4/19/a-taxonomy-of-collaboration-tools.html [Accessed August 19, 2013].

Davenport, T. & Short, J., 1990. The New Industrial Engineering: Information Technology And Business Process Redesign. *Sloan Management Review*, 31(4), pp.11–27.

Delone, W.H. & Mclean, E.R., 2003. The DeLone and McLean model of information systems success: a ten-year update. *Journal of Management Information Systems*, 4(19), pp.9–30. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.88.3031 [Accessed February 7, 2014].

Dey, A.K. & Abowd, G.D., 2000. Towards a better understanding of context and context-awareness. In *Conference on Human Factors in Computing Systems*.

Dey, A.K., Abowd, G.D. & Salber, D., 2001. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 4(2), pp.97–166.

Dinh, H. & Lee, C., 2011. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless and Mobile Computing*, 13(13), pp.1–38. Available at: http://onlinelibrary.wiley.com/doi/10.1002/wcm.1203/full [Accessed August 16, 2013].

Ellis, C.A., Gibss, S. & Rein, G.L., 1991. Groupware: some issues and experiences. *Communications of the ACM*, 1(43), pp.38–58.

Esayas, S.Y., 2012. A walk in to the cloud and cloudy it remains: The challenges and prospects of “processing” and “transferring” personal data. *Computer Law & Security Review*, 28(6), pp.662–678.

Espada, J.P. et al., 2012. Extensible architecture for context-aware mobile *Web* applications. *Expert Systems with Applications*, 39(10), pp.9686–9694.

Espada, J.P. et al., 2013. Using extended *Web* technologies to develop Bluetooth multi-platform mobile applications for interact with smart things. *Information Fusion*, (0). Available at: http://dx.doi.org/10.1016/j.inffus.2013.04.008.

Fei, X. et al., 2011. Opportunities and Challenges in Running Scientific *Workflows* on the Cloud. In *2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. IEEE, pp. 455–462. Available at: http://dl.acm.org/citation.cfm?id=2082756.2083324 [Accessed August 8, 2013].

Fernández-López, Á. et al., 2013. Mobile learning technology based on *iOS* devices to support students with special education needs. *Computers & Education*, 61(0), pp.77–90.

Fernando, N., Loke, S.W. & Rahayu, W., 2013. Mobile cloud computing: A survey. *Future Generation Computer Systems*, 29(1), pp.84–106.

Ferreira, C.J. da R., 2009. *A simplified workflow-based system for publishing processes using CMS and WfMS standards*,

Flautero, J., 2012. Mobile Backend-as-a-Service MindMap 2012. *Blog*, p.1. Available at: http://quickblox.com/blog/2012/12/mobile-backend-as-a-service-mindmap-2012/ [Accessed October 8, 2013].

Friese, P., 2012. *Cross-Platform Mobile Development*, Available at: http://www.infoq.com/presentations/Cross-Platform-Mobile.

Gamma, E. et al., 1999. *Design Patterns. Elements of Reusable Object-oriented Software*, Addsion Wesley Longman, printed in India by Eastern Press. Available at: http://www.amazon.com/Patterns-Elements-Reusable-Object-oriented-Software/dp/0201455633 [Accessed September 20, 2013].

Garzonis, S., 2010. *MOBILE SERVICE AWARENESS VIA AUDITORY NOTIFICATIONS*.

Gavalas, D. et al., 2011. Status and trends of mobile-health applications for *iOS* devices: A developer's perspective. *Journal of Systems and Software*, 84(11), pp.2022–2033.

Greefhorst, D. & Proper, E., 2011. *Architecture Principles: The Cornerstones of Enterprise Architecture (The Enterprise Engineering Series)*, Springer. Available at: http://www.amazon.com/Architecture-Principles-Cornerstones-Enterprise-Engineering/dp/3642202780 [Accessed October 2, 2013].

Grefen, P., 2012. *Business Information System Architecture*, Available at: http://is.ieis.tue.nl/staff/pgrefen/research/publications/index.php?selection=Search&search=paul+grefen [Accessed October 2, 2013].

Gummerus, J. & Pihlström, M., 2011. Context and mobile services' value-in-use. *Journal of Retailing and Consumer Services*, 18(6), pp.521–533.

Hansen, D.L., Shneiderman, B. & Smith, M.A., 2011. Chapter 2 - Social Media: New Technologies of Collaboration. In *Analyzing Social Media Networks with NodeXL*. Analyzing Social Media Networks with NodeXL. Boston: Morgan Kaufmann, pp. 11–29.

Harman, M. et al., 2013. Cloud engineering is Search Based Software Engineering too. *Journal of Systems and Software*, 86(9), pp.2225–2241.

Henricksen, K. & Indulska, J., 2006. Developing context-aware pervasive computing applications: Models and approach. *Pervasive and Mobile Computing*, 2(1), pp.37–64.

Henten, A. et al., 2009. Exploring a heterogeneous and fragmented digital ecosystem: Mobile content. *Telematics and Informatics*, 26(3), pp.282–292.

Herskovic, V. et al., 2011. The iceberg effect: behind the user interface of mobile collaborative systems. *Journal of Universal Computer Science*, 17(2), pp.183–202.

Herskovic, V., Ochoa, S.F. & Pino, J.A., 2009. Modeling groupware for mobile collaborative work. In *Computer Supported Cooperative Work in Design, 2009. CSCWD 2009. 13th International Conference on*. pp. 384–389.

Hevner, A., 2007. A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems*, 19(2).

Hevner, A. & Chatterjee, S., 2010. Design Research in Information Systems. *Integrated Series in Information Systems*, 22, pp.9–22.

Hevner, A.R. et al., 2004. Design Science in Information Systems Research A. R. Hevner & S. Chatterjee, eds. *MIS Quarterly*, 28(1), pp.75–105.

Hollingsworth, D., 1995. *Workflow* Management Coalition The *Workflow* Reference Model. , (1), pp.1–55.

Hua, H., Yi-Lai, Z. & Min, Z., 2013. A Survey of Cloud *Workflow*. In *Proceedings of the 2nd International Conference On Systems Engineering and Modeling (ICSEM-13)*.

Hyrynsalmi, 2012. App Store, Marketplace, Play! An Analysis of Multi-Homing in Mobile Software Ecosystems. In *Proceedings of the Fourth International Workshops on Software Ecosystems*. pp. 59–72.

Jamal, S. & Deters, R., 2011. Using a Cloud-Hosted Proxy to support Mobile Consumers of RESTful Services. *Procedia Computer Science*, 5(0), pp.625–632.

Jansen, S., Brinkkemper, S. & Finkelstein, A., 2009. Business network management as a survival strategy: a tale of two software ecosystems. In *First International Workshop on Software Ecosystems (IWSECO-2009), Citeseer (2009)*. pp. 34–48.

Kavyanidhi, N., 2012. The Mobile App Ecosystem – Members & their functionality. *Telecominfo's Weblog*. Available at: http://telecominfo.wordpress.com/tag/ibm-worklight/ [Accessed October 8, 2013].

Keith, M., Demirkan, H. & Goul, M., 2010. The influence of collaborative technology knowledge on advice network structures. *Decision Support Systems*, 50(1), pp.140–151.

Khan, A.N. et al., 2013. Towards secure mobile cloud computing: A survey. *Future Generation Computer Systems*, 29(5), pp.1278–1299.

Kim, W., Jeong, O.-R. & Lee, S.-W., 2010. On social *Web* sites. *Information Systems*, 35(2), pp.215–236.

Kruchten, P., 1995. “Architectural Blueprints - The ”4+1“ View Model of Software Architecture.” *IEEE Software*, 12(6), pp.42–50.

Li, Q., 2013. A novel Likert scale based on fuzzy sets theory. *Expert Systems with Applications*, 40(5), pp.1609–1618.

Li, Q., Abel, M.-H. & Barthès, J.-P.A., 2013. Modeling and exploiting collaborative traces in *Web*-based collaborative working environment. *Computers in Human Behavior*. Available at: http://dx.doi.org/10.1016/j.chb.2013.04.028.

Lim, S.-H., Lee, S. & Ahn, W.H., 2013. Applications IO profiling and analysis for smart devices. *Journal of Systems Architecture*. Available

Lin, C.-C. & Tsai, C.-C., 2011. Applying social bookmarking to collective information searching (CIS): An analysis of behavioral pattern and peer interaction for co-exploring quality online resources. *Computers in Human Behavior*, 27(3), pp.1249–1257.

Liu, F. et al., 2011. NIST Cloud Computing Reference Architecture. *Recomendations of the National Institute of Standarts and Technology*, p.35. Available at: http://www.nist.gov/manuscript-publication-search.cfm?pub_id=909505 [Accessed August 16, 2013].

Liu, X. et al., 2012. Cloud *Workflow* System Architecture. In *The Design of Cloud Workflow Systems*. SpringerBriefs in Computer Science. New York, NY: Springer New York, pp. 13–18. Available at: http://www.springerlink.com/index/10.1007/978-1-4614-1933-4 [Accessed September 10, 2013].

Manikas, K. & Hansen, K.M., 2013. Software ecosystems – A systematic literature review. *Journal of Systems and Software*, 86(5), pp.1294–1306.

Mao, L., Yang, Y. & Xu, H., 2013. Design and Optimization of Cloud-Oriented *Workflow* System. *Journal Of Software*, 8(1).

McCrickard, D.S. & Chewar, C.M., 2003. Attuning notification design to user goals and attention costs. *Communications of the ACM*, 46(3), p.67.

McGinn, C.A. et al., 2012. Users’ perspectives of key factors to implementing electronic health records in Canada: a Delphi study. *BMC medical informatics and decision making*, 12(1), p.105.

Meilin, S. et al., 1998. *Workflow* management systems: a survey. In *ICCT’98. 1998 International Conference on Communication Technology. Proceedings (IEEE Cat. No.98EX243)*. Publising House of Constr. Mater, p. 6.

Mell, P. & Grance, T., 2011. The NIST Definition of Cloud Computing. *NIST Special Publication*, p.7. Available at: http://www.nist.gov/manuscript-publication-search.cfm?pub_id=909616 [Accessed August 13, 2013].

Microsoft Patterns & Practices, T., 2009. Chapter 3: Architectural Patterns and Styles. In *Microsoft ® Application Architecture Guide, 2nd Edition (Patterns & Practices)*. *Microsoft* Press, p. 45. Available at: http://www.amazon.com/*Microsoft* ®-Application-Architecture-Patterns-Practices/dp/073562710X [Accessed September 20, 2013].

Mikkonen, T. & Taivalsaari, A., 2013. Cloud computing and its impact on mobile software development: Two roads diverged. *Journal of Systems and Software*, 86(9), pp.2318–2320.

Minor, M., Bergmann, R. & Görg, S., 2011. Adaptive *Workflow* Management in the Cloud -Towards a Novel Platform as a Service. In *Proceedings of the ICCBR 2011 Workshops*. pp. 131–138.

Mohamed, K. & Wijesekera, D., 2012. Performance Analysis of *Web* Services on Mobile Devices. *Procedia Computer Science*, 10(0), pp.744–751.

Molder, J., van Lier, B. & Jansen, S., 2011. Clopenness of systems: The interwoven nature of ecosystems. In *Third International Workshop on Software Ecosystems (IWSECO-2011), CEUR-WS (2011)*. pp. 52–64.

Mühl, G., Fiege, L. & Pietzuch, P., 2006. *Distributed Event-Based Systems*, Springer. Available at: http://www.springer.com/computer/communication+networks/book/978-3-540-32651-9 [Accessed March 4, 2014].

Nardi, B.A., 1995. *Context and Consciousness: Activity Theory and Human-Computer Interaction*, The MIT Press. Available at: http://www.amazon.com/Context-Consciousness-Activity-Human-Computer-Interaction/dp/0262140586 [Accessed October 7, 2013].

Nasirifard, P., Peristeras, V. & Decker, S., 2011. Annotation-based access control for collaborative information spaces. *Computers in Human Behavior*, 27(4), pp.1352–1364.

Neyem, A. et al., 2012. A reusable structural design for mobile collaborative applications. *Journal of Systems and Software*, 85(3), pp.511–524.

Neyem, A., Ochoa, S.F. & Pino, J.A., 2008. Integrating service-oriented mobile units to support collaboration in ad-hoc scenarios. *Journal of Universal Computer Science*, 14(1), pp.88–122.

Nicolaou, A. & Geerts, G.L., 2011. *A design science research methodology and its application to accounting information systems research*, Available at: http://www.sciencedirect.com/science/article/pii/S1467089511000200 [Accessed November 18, 2013].

Niu, J., Song, W. & Atiquzzaman, M., 2013. Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications. *Journal of Network and Computer Applications*.

Noureddine, M. & Bashroush, R., 2013. An authentication model towards cloud federation in the enterprise. *Journal of Systems and Software*, 86(9), pp.2269–2275.

Pandey, S., Karunamoorthy, D. & Buyya, R., 2011. *Workflow* Engine for Clouds. In *Cloud Computing: Principles and Paradigms*. pp. 321–344.

Papadopoulos, C., 2006. Improving Awareness in Mobile CSCW. *IEEE Transactions on Mobile Computing*, 5(10), pp.1331–1346.

Park, S.C. & Ryoo, S.Y., 2013. An empirical investigation of end-users' switching toward cloud computing: A two factor theory perspective. *Computers in Human Behavior*, 29(1), pp.160–170.

Price, S. et al., 2013. SubSift *Web* services and *workflows* for profiling and comparing scientists and their published works. *Future Generation Computer Systems*, 29(2), pp.569–581.

Pura, M. & Heinonen, K., 2008. Exploring Mobile Service Business Opportunities from a Customer-Centric Perspective. In D. Taniar, ed. *Global Mobile Commerce: Strategies, Implementation and Case Studies*. IGI Global, pp. 111–132.

Riesner, M., Netter, M. & Pernul, G., 2013. Analyzing settings for social identity management on Social Networking Sites: Classification, current state, and proposed developments. *Information Security Technical Report*, 17(4), pp.185–198.

Rodríguez-Covili, J. et al., 2011. A communication infrastructure to ease the development of mobile collaborative applications. *Journal of Network and Computer Applications*, 34(6), pp.1883–1893.

Sallam, A.A. & Siba, U., 2011. Taxonomy for personalized Notification System for Improving the Quality Service of Mobile Marketplace. *International Journal of Distributed and Parallel Systems*, 2(1), p.116.

Sandra Nava-Muñoz, A.L.M., 2009. A Taxonomy of Notification Technology for Assisting the Caregivers of Elders with Cognitive Decline. In *HCI EA '09*. pp. 956–960.

Saucedo-Tejada, G., Mendoza, S. & Decouchant, D., 2013. F2FMI: A toolkit for facilitating face-to-face mobile interaction. *Expert Systems with Applications*, 40(15), pp.6173–6184.

Shao, G., 2008. Understanding the appeal of user-generated media: Uses and gratification perspective. *Internet Research*, 19(1), pp.7–25.

Smari, W.W., Clemente, P. & Lalande, J.-F., 2013. An extended attribute based access control model with trust and privacy: Application to a collaborative crisis management system. *Future Generation Computer Systems*.

Sridhar, S., 2012. Mobile Cloud Backend as a Service Ecosystem Map – All roads lead to BaaS. Available at: http://www.kinvey.com/blog/65/mobile-cloud-backend-as-a-service-ecosystem-map-8211-all-roads-lead-to-baas [Accessed October 8, 2013].

Stoitsev, V., 2012. *A Multi-aspect Reference Architecture for a Business Process Cloud Platform*.

Stoitsev, V. & Grefen, P., 2012. *Business Process Technology and the Cloud: defining a Business Process Cloud Platform*,

Subashini, S. & Kavitha, V., 2011. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 34(1), pp.1–11.

Székely, A., Talanow, R. & Bágyi, P., 2013. Smartphones, tablets and mobile applications for radiology. *European Journal of Radiology*, 82(5), pp.829–836.

Torroglosa-García, E. et al., 2013. Integration of the OAuth and *Web* Service family security standards. *Computer Networks*, 57(10), pp.2233–2249.

Valtonen, T. et al., 2010. Net generation at social software: Challenging assumptions, clarifying relationships and raising implications for learning. *International Journal of Educational Research*, 49(6), pp.210–219.

Wang, D., 2011. An Efficient Cloud Storage Model for Heterogeneous Cloud Infrastructures. *Procedia Engineering*, 23(0), pp.510–515.

Weiping, L., 2009. An analysis of new features for *workflow* system in the SaaS software. In *Proceedings of the 2nd International Conference on Interaction Sciences Information Technology, Culture and Human - ICIS '09*. New York, New York, USA: ACM Press, pp. 110–114.

West, A.G. et al., 2012. Trust in collaborative *Web* applications. *Future Generation Computer Systems*, 28(8), pp.1238–1251.

Worrell, J.L., Gangi, P.M. Di & Bush, A.A., 2013. Exploring the use of the Delphi method in accounting information systems research. *International Journal of Accounting Information Systems*, 14(3), pp.193–208.

Xu, X., 2012. From cloud computing to cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, 28(1), pp.75–86.

Yang, K. et al., 2005. Policy-based model-driven engineering of pervasive services and the associated OSS. *British Telecom Technical Journal (BTTJ)*, 23(3), pp.162–174.

Yang, Y., 2000. An architecture and the related mechanisms for *Web*based global cooperative teamwork support, international. *Journal of Computing and Informatics*, 24, pp.13–19.

Yeh, S.-C. et al., 2013. An efficient and secure approach for a cloud collaborative editing. *Journal of Network and Computer Applications*. Available at: http://dx.doi.org/10.1016/j.jnca.2013.05.012.

Zhao, Y. et al., 2012. Designing and Deploying a Scientific Computing Cloud Platform. In *2012 ACM/IEEE 13th International Conference on Grid Computing*. IEEE, pp. 104–113.

Zhu, Y., Shtykh, R.Y. & Jin, Q., 2012. A human-centric framework for context-aware flowable services in cloud computing environments. *Information Sciences*.

Anexos

Anexo A: Cuestionario del estudio *Delphi*

Referencias en la literatura para los factores:

a) colaborativos (Hansen et al. 2011; Kim et al. 2010; Riesner et al. 2013; Saucedo-Tejada et al. 2013; Neyem et al. 2012; Smari et al. 2013; Li et al. 2013; Herskovic et al. 2009; Rodríguez-Covili et al. 2011).

b) *cloud* (Subashini & Kavitha 2011; Mell & Grance 2011; Esayas 2012; Aceto et al. 2013; Xu 2012; Park & Ryoo 2013; Fernando et al. 2013; Azeemi et al. 2013; Harman et al. 2013, Liu et al. 2011; Mikkonen & Taivalsaari 2013; Dinh & Lee 2011).

c) móviles (Mühl et al. 2006; Sallam & Siba 2011; Arroqui et al. 2012; Bouwman et al. 2012; Pura & Heinonen 2008, Friese 2012; Flautero 2012).

Cuestionario:

Las funcionalidades de un WFMS son mejores utilizando/implementado Tecnologías Colaborativas.

Solución Tecnológica Colaborativa	**Factor a evaluar**
Utilizar aplicaciones móviles colaborativas con características cloud y sociales (*Dropbox, Google Docs, Evernote*, etc) para que las funcionalidades de un WfMS permitan a los usuarios realizar *Workflows* y tareas en forma colaborativa.	Colaboración
Utilizar medios sociales *(Twitter, Facebook, etc)* como sistemas de comunicación, para que las funcionalidades de un WfMS permitan el intercambio de mensajes, alarmas y notificaciones entre los usuarios y de esta manera puedan interactuar y colaborar.	Comunicación
Utilizar aplicaciones móviles colaborativas con servicios de coordinación *(gestión de sesiones, compartir información, gestión de usuarios y roles, etc)*, para que las funcionalidades de un WfMS permitan a los usuarios tener coordinación entre ellos.	Coordinación
Utilizar los servicios colaborativos que ofrecen las redes sociales, para que las funcionalidades de un WfMS permitan a los usuarios crear y gestionar *Workflows* de forma colaborativa con sus contactos de redes sociales.	Redes sociales abiertas

Utilizar la identidad que tiene el usuario en sus redes sociales, para que las funcionalidades de un WfMS la utilicen como método de autenticación e identificación y permitan al usuario gestionar *Workflows* y colaborar con otros usuarios.	Inicio de Sesión Única

Las funcionalidades de un WFMS son mejores utilizando/implementado tecnologías Cloud Computing

Solución Tecnológica Cloud	**Factor a evaluar**
Ejecutar como servicio en la nube la funcionalidad que se encarga de ejecutar y gestionar las instancias de *Workflows*, permitiendo que este servicio esté disponible sin importar el lugar y la hora en que se accede para gestionar *Workflows*.	Disponibilidad
Ejecutar como servicio en la nube, la funcionalidad que se encarga de ejecutar y gestionar las instancias de *Workflows*, utilizando recursos de cómputo dependiendo de la necesidad de procesamiento del servicio, permite que las funcionalidades del sistema gestor mantengan un nivel de calidad de servicio aceptable para la gestión de *Workflows*.	Elasticidad
Utilizar la comunicación como servicio en la nube, para que las funcionalidades de un WfMS puedan notificar e informar al usuario del estado de sus *Workflows* y tareas de forma más rápida y eficiente.	Servicios Cloud /Comunicación
Utilizar el almacenamiento como servicio en la nube, para que las funcionalidades de un WfMS permitan al usuario la gestión y acceso a los contenidos sin sufrir pérdidas o daños en los contenidos.	Consistencia
Utilizar el almacenamiento como servicio en la nube, para que las funcionalidades de un WfMS permitan la sincronización de contenidos sin importar la hora y el lugar y de esta forma el usuario puede gestionar y acceder a los contenidos en su versión más actual.	Sincronización

Las funcionalidades de un WFMS son mejores utilizando/implementado Tecnologías Móviles

Solución Tecnológica Móvil	**Factor a evaluar**
Implementar las funcionalidades de un WfMS en una versión móvil, para permitir al usuario gestionar Workflows sin importar el contexto o la situación de movilidad en que se encuentra.	Movilidad
Utilizar la conectividad del dispositivo móvil, para que las funcionalidades de un WfMS permitan gestionar *Workflows* sin importar la localización física o temporal del usuario.	Ubicuidad/ Disponibilidad
Utilizar diferentes formas de notificaciones en un dispositivo móvil *(push notifications, SMS, mensajería)*, para que las funcionalidades de un WfMS en versión móvil mantengan al usuario informado y actualizado sobre la gestión de *Workflows*, sin importar su localización física o temporal.	Comunicación
Utilizar servicios móviles *(almacenamiento, procesamiento, comunicaciones, etc.)* que extiendan las funcionalidades de un WfMS en versión móvil, para que la gestión de *Workflows* no se vea afectada por los recursos limitados del dispositivo móvil.	Servicios móviles

Anexo B: Resultados de las rondas de cuestionarios del estudio *Delphi*

B.1 Factores Colaborativos

Primera ronda	Segunda ronda	Tercera ronda
4	5	5
4	5	5
3	3	4
3	5	5
5	5	5
5	5	5
5	5	5
5	5	5
4	5	5
5	5	5

Tabla 7: Resultados para el factor Colaboración

Primera ronda	Segunda ronda	Tercera ronda
5	5	5
5	5	5
5	5	5
3	5	5
3	5	5
4	5	5
4	5	5
4	4	5
4	5	5
5	5	5

Tabla 8: Resultados para el factor Comunicación

Primera ronda	Segunda ronda	Tercera ronda
4	5	5
4	5	5
4	4	4
5	5	5
5	5	5
5	5	5
5	5	5
5	5	5
4	5	5
4	5	5

Tabla 9: Resultados para el factor Coordinación

Primera ronda	Segunda ronda	Tercera ronda
3	3	4
3	3	3
4	4	4
3	4	5
3	3	3
2	2	2
4	4	4
2	3	3
2	2	2
4	4	4

Tabla 10: Resultados para el factor Redes Sociales Abiertas

Primera ronda	Segunda ronda	Tercera ronda
5	5	5
4	5	5
5	5	5
5	5	5
5	5	5
4	5	5
5	5	5
5	5	5
5	5	5
5	5	5

Tabla 11: Resultados para el factor Inicio de Sesión Única

B.2 Factores *Cloud Computing*

Primera ronda	Segunda ronda	Tercera ronda
4	5	5
4	4	4
5	5	5
5	5	5
4	5	5
4	5	5
5	5	5
5	5	5
4	4	4
5	5	5

Tabla 12: Resultados para el factor Disponibilidad

Primera ronda	Segunda ronda	Tercera ronda
4	4	4
3	3	3
4	4	4
3	4	4
3	4	4
4	4	4
2	4	4
3	3	3
4	4	4
5	5	5

Tabla 13: Resultados para el factor Elasticidad

Primera ronda	Segunda ronda	Tercera ronda
4	4	4
4	4	4
3	4	4
3	5	4
3	4	4
4	4	4
2	4	4
4	4	4
4	5	5
5	5	5

Tabla 14: Resultados para el factor Servicios *Cloud*

Primera ronda	Segunda ronda	Tercera ronda
4	4	5
4	4	5
4	4	5
5	5	5
5	5	5
4	5	5
5	5	5
5	5	5
4	5	5
5	5	5

Tabla 15: Resultados para el factor Sincronización

Primera ronda	Segunda ronda	Tercera ronda
2	5	5
4	4	4
5	5	5
5	5	5
5	5	5
4	4	4
3	5	5
5	5	5
5	5	5
5	5	5

Tabla 16: Resultados para el factor Consistencia

B.3 Factores Móviles

Primera ronda	Segunda ronda	Tercera ronda
4	4	4
4	5	5
4	5	5
4	5	5
5	5	5
5	5	5
5	5	5
5	5	5
4	5	5
5	5	5

Tabla 17: Resultados para el factor Movilidad

Primera ronda	Segunda ronda	Tercera ronda
5	5	5
5	5	5
5	5	5
4	5	5
4	5	5
5	5	5
5	5	5
5	5	5
5	5	5
5	5	5

Tabla 18: Resultados para el factor Ubicuidad.

Primera ronda	Segunda ronda	Tercera ronda
3	3	3
4	4	4
4	4	4
5	5	4
4	5	4
5	4	4
4	4	4
3	3	3
5	4	4
4	4	4

Tabla 19: Resultados para el factor Comunicación.

Primera ronda	Segunda ronda	Tercera ronda
4	4	4
4	5	4
4	5	5
5	5	5
5	5	5
5	5	5
5	4	5
5	5	5
4	5	5
5	5	5

Tabla 20: Resultados para el factor Servicios Móviles.