UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 5

Centro de Consultoría y Desarrollo de Arquitecturas Empresariales.

Sistema para la informatización del modelo de madurez de tres perspectivas para evaluar y planificar la adopción de SOA.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Luis Miguel Pimentel González

Tutor: Dr.C. Arturo César Arias Orizondo

La Habana, Junio de 2014

“La actitud frente a la vida es mostrar con el ejemplo el camino a seguir, el llevar a las masas con el propio ejemplo cualesquiera que sean las dificultades a vencer en el camino, quien pueda mostrar el ejemplo de su trabajo repetido durante días sin esperar de la sociedad otra cosa que el reconocimiento a sus méritos, de constructor de esta nueva sociedad, tiene derecho a exigir a la hora del sacrificio.”

Che

DECLARACIÓN JURADA DE AUTORÍA

Declaro ser autor de la presente tesis y se reconoce a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Nombre del tesista

Luis Miguel Pimentel González

_____

Nombre del tutor

Dr.C. Arturo César Arias Orizondo

_____

AGRADECIMIENTOS

A la UCI, en específico al CDAE por la formación integral que recibí en sus proyectos.

A mi tutor Arturo por todo el conocimiento y apoyo que me ha brindado.

A mi hermano por ser siempre mi guía a seguir en la vida.

A mi mamá y a mi papá por el amor con que me han enseñado a caminar en la vida y por confiar siempre en mí.

A mi abuela Candy por enseñarme que todo en la vida es posible y enseñarme a confiar en mí.

A mi novia por apoyarme tanto y acompañarme siempre en todas las situaciones difíciles.

A mis tías, abuelas, primos, en fin, a toda mi familia a quienes le debo muchísimo de mi vida.

A mis suegros y a toda la familia de mi novia por hacerme sentir como uno más de su familia.

A todos mis amigos por su apoyo incondicional.

A mis profesores por formarme como profesional.

Y a la Revolución por darme la posibilidad de cumplir mis sueños.

DEDICATORIA

A todas las personas que hicieron posible la realización de este trabajo, en especial a mi tutor.

A mi familia.

A mis maestros y profesores de toda la vida, por su especial contribución a mi desarrollo humano y profesional. Esta tesis también es su obra.

RESUMEN

Los modelos de madurez, por el enfoque evolutivo que ofrecen, resultan útiles para abordar la complejidad inherente al proceso de adopción de Arquitecturas Orientadas a Servicios. Sin embargo, las insuficiencias detectadas en ellos, limitan las capacidades que ofrecen para evaluar y planificar de manera más efectiva la adopción de SOA en las organizaciones. Con el propósito de superar las principales limitaciones y empleando la experiencia internacional acumulada en la materia, fueron definidos los constructos y elementos de diseño que sustentan el “modelo de madurez de tres perspectivas para SOA” (MMSOA3P). Este integra los resultados esperados de la adopción de SOA y las capacidades técnicas y organizacionales necesarias para obtenerlos. Sobre los constructos y elementos de diseño definidos, resulta necesario construir una herramienta informática que brinde sustento al modelo y facilite su administración y aplicación. El presente trabajo está enfocado al desarrollo de una aplicación informática que permita la gestión y aplicación del modelo de madurez MMSOA3P, dicho modelo fue propuesto a través de una investigación por el centro de Consultoría y Desarrollo de Arquitecturas Empresariales perteneciente a la Universidad de las Ciencias Informáticas. Para ello se realizó un estudio del funcionamiento de las aplicaciones ya existentes para gestionar modelos de madurez. Como resultado se obtuvo una herramienta para gestionar y aplicar el modelo de madurez MMSOA3P.

Palabras clave: arquitectura orientada a servicios; modelo de madurez; adopción de SOA.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

INTRODUCCIÓN

Con la llegada del siglo XXI, la información se ha convertido en un recurso estratégico; las organizaciones necesitan de su disponibilidad y confiabilidad para la generación de conocimiento útil que favorezca la toma de decisiones. En la actualidad la mayoría de las instituciones, donde utilizan las tecnologías informáticas, gran parte de la información es generada, procesada y distribuida a través de toda una compleja y heterogénea infraestructura de Tecnologías de la Información (TI) (Arias-Orizondo 2013).

Las organizaciones son cada vez más conscientes de la necesidad de contar con la mejor y efectiva infraestructura tecnológica para operar sus negocios de una manera eficiente y competitiva, pero también del alto costo que esto puede representar (Rivero-Gutierrez y Bueno-Carrion 2009). Por ello que es necesaria la creación de sistemas informáticos soportados por estructuras sólidas y de calidad que les garantice competitividad y longevidad.

La Arquitectura Orientada a Servicios (SOA, por las siglas en inglés de Service Oriented Architecture), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos de negocios. De esta manera permite la creación de sistemas de información altamente escalables que reflejan el negocio de la organización (Nicolai M. 2007).

SOA propone una estrategia general de organización de los elementos de infraestructura tecnológica, de forma que sistemas distribuidos y aplicaciones complejas se puedan transformar en una red de recursos integrados, simplificada y sumamente flexible. Permite cambios más rápidos en los procesos, lo que brinda ventaja competitiva al crear nuevas aplicaciones dinámicas, ayuda a la automatización de los procesos de negocio, permite la integración de aplicaciones, la modernización incremental de sistemas legados[1], así como la consolidación de la información.

El desarrollo de un programa SOA ha resultado complejo y en muchas ocasiones no tributa a los beneficios esperados en su adopción. La adopción no ha tenido todo el éxito esperado en algunas de las empresas que lo han intentado, muchas entidades han tratado de asumir la arquitectura solo por tenerla y han fracasado. Se reconocen como principales causas el desconocimiento de las características de la entidad, la ausencia de un estudio de sus potencialidades, pobre implementación, infraestructura tecnológica inadecuada, la inexistencia de una estrategia general de desarrollo que permita tener en cuenta pasos escalonados, relacionados y ascendentes

[1] Se llama sistemas legados a los sistemas antiguos que funcionan en una organización

(Rivero-Gutierrez y Bueno-Carrion 2009), identificándose el factor humano como una de las principales causas. Es por ello que el modo en que se adopta este paradigma sigue siendo un factor determinante.

La adopción de SOA puede planificarse empleando un modelo de madurez. Estos se utilizan para ayudar a las organizaciones a saber hasta dónde han avanzado en el camino de adopción. Forman la base para comunicar y extender capacidades y ayudan en la construcción de planes. Los modelos de madurez para SOA constituyen un método para evaluar y medir la madurez de la adopción, además describen las capacidades de una arquitectura SOA efectiva y una visión sobre aspectos a mejorar.

Existen muchos modelos de madurez para su adopción en todo el mundo, pero presentan limitaciones, principalmente: el enfoque a características específicas, los beneficios que genera la adopción son enunciados en la mayoría, sin embargo no queda establecida una relación explícita entre ellos (Arias-Orizondo y Estrada-Senti 2013). Son poco precisos para abordar aspectos organizacionales que influyen en la adopción de SOA y en el contexto nacional constituyen un obstáculo para su aplicación por la dificultad para acceder al instrumento de evaluación, principalmente de los modelos propietarios (Arias-Orizondo 2013).

Con el propósito de superar las principales limitaciones y empleando la experiencia internacional acumulada en la materia, el Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE), perteneciente a la Universidad de las Ciencias Informáticas (UCI) a partir de investigaciones realizadas en estos temas deja definidos los constructos y elementos de diseño que sustentan un nuevo modelo de madurez, el “modelo de madurez de tres perspectivas para SOA” (MMSOA3P) (Arias-Orizondo 2013).

Este integra los resultados esperados de la adopción de SOA y las capacidades técnicas y organizacionales necesarias para obtenerlos. La información almacenada en el modelo constituye un conjunto de buenas prácticas asociadas al desarrollo de una iniciativa SOA y que pueden ser empleadas para tomar decisiones acertadas en el proceso de adopción de SOA. Sobre los constructos y elementos de diseño definidos, el CDAE no cuenta con ninguna herramienta informática que posibilite el proceso de creación, administración y aplicación del modelo de madurez para SOA MMSOA3P, por lo que resulta necesario construir una herramienta informática que brinde sustento al modelo y facilite su administración y aplicación.

Por lo anterior se define como **problema de la investigación:** ¿Cómo facilitar la administración y aplicación del modelo de madurez MMSOA3P?

La **idea a defender** plantea una posible solución al problema: Si se desarrolla una herramienta que informatice el modelo MMSOA3P se facilitará la administración y adopción del mismo.

Para darle solución a la problemática planteada se establece como **Objeto de Estudio**: Proceso de evaluación y planificación de SOA mediante modelos de madurez, y **campo de acción** la informatización del modelo de madurez MMSOA3P desarrollado por el CDAE.

Objetivo general:

Desarrollar una herramienta informática que facilite el proceso de administración y aplicación del modelo de madurez MMSOA3P.

Objetivos específicos:

- Realizar estudio del estado del arte de sistemas informáticos para el procesamiento de encuestas estructuradas.
- Seleccionar y valorar las herramientas necesarias para realizar el diseño e implementación del sistema.
- Implementar la solución propuesta a partir del modelo.
- Evaluar el sistema desarrollado mediante pruebas de calidad.

Tareas de investigación:

- Análisis de la estructura lógica del modelo de madurez MMSOA3P.
- Diseño de las estructuras de datos y algoritmos necesarios para soportar el modelo de madurez.
- Diseño de las interfaces de usuario y reportes de salidas de información del software.
- Análisis y selección de las plataformas, lenguajes de programación, herramientas y metodologías de desarrollo de software que se utilizarán para el desarrollo del software.
- Desarrollo de los módulos que conforman la solución informática.
- Poblado de la estructura del modelo de madurez desarrollado.
- Aplicación de pruebas para garantizar la calidad y funcionalidad del producto obtenido.

Métodos Científicos

- Análisis Histórico – Lógico para realizar un estudio de la evolución de la utilización de los sistemas para la administración y aplicación de los modelos de madurez, así como las técnicas usadas a nivel mundial, las tendencias actuales y su aplicación en la Universidad de las Ciencia Informáticas (UCI).

- Analítico-Sintético para el análisis de la documentación existente relacionada con el tema, extrayendo los elementos más importantes y necesarios para dar solución al problema existente.
- Inductivo-Deductivo para estudiar las principales técnicas utilizadas para la creación de sistemas de administración y aplicación de los modelos de madurez, tanto a nivel mundial como en Cuba y la aplicación de los mismos con el objetivo de determinar cuáles son las alternativas viables a incorporar en la presente investigación.
- Modelación para la representación explícita de la solución propuesta a través de la modelación de los procesos, las ideas y referentes teóricos extraídos de las fuentes bibliográficas consultadas.
- Análisis documental, en la consulta de la literatura especializada en las temáticas afines de la investigación.
- Entrevista a profundidad para obtener información referente a los procesos llevados en el modelo de madurez de tres perspectivas para SOA" (MMSOA3P), así como obtener las concepciones de especialistas con experiencias en su desarrollo en la UCI.

Estructura de la Tesis

El contenido de este trabajo de diploma está compuesto por tres capítulos, conclusiones, recomendaciones y bibliografía, donde se analiza todo el proceso de investigación y desarrollo de la herramienta informática que facilita el proceso de administración y aplicación del modelo de madurez para SOA MMSOA3P.

- **Capítulo 1**: "Marco teórico de la investigación relacionado con la creación del sistema para la administración y aplicación del modelo de madurez para SOA MMSOA3P". Se precisan un conjunto de elementos para conformar el marco teórico relacionado con los aspectos definidos en el objeto de estudio. Se identifican y analizan las tecnologías existentes, detectando sus mejores prácticas para tenerlas en cuenta en la solución a proponer.
- **Capítulo 2**: "Propuesta de solución del sistema para la administración y aplicación del modelo de madurez". Se precisan un conjunto de elementos para conformar la propuesta de solución de la presente investigación así como sus características y componentes, además de describir el funcionamiento general del sistema que se desea implementar. Se realiza una descripción de la arquitectura propuesta para la solución, así como el uso de patrones presentes en el mismo. Además, se abordan las fases de Exploración y Planificación propias de la metodología de desarrollo utilizada en la implementación del sistema informático, en las cuales se exponen las historias de usuario seccionada en

iteraciones, los planes de entrega y la duración de las iteraciones durante el desarrollo del sistema.

- **Capítulo 3**: “Implementación, pruebas y validación de la solución propuesta”. Se abordan contenidos relacionados con la descripción de la implementación del sistema, para darle solución a las historias de usuario definidas en el capítulo anterior, se realizan las tareas de ingeniería y se especifican las pruebas a las que fue sometida la aplicación. Por último se describe el proceso de validación para comprobar el cumplimiento de las historias de usuario definidas por el cliente.

CAPÍTULO 1. MARCO TEÓRICO DE LA INVESTIGACIÓN RELACIONADO CON LA CREACIÓN DEL SISTEMA PARA LA ADMINISTRACIÓN Y APLICACIÓN DEL MODELO DE MADUREZ.

Introducción

En el presente capítulo se precisan un conjunto de elementos para conformar el marco teórico relacionado con los aspectos definidos en el objeto de estudio. Se realiza un análisis del modelo de madurez de tres perspectivas para SOA (MMSOA3P). Se analizan las soluciones de software existentes para gestionar los modelos de madurez para SOA a través del estudio del estado del arte. La solución informática propuesta, tendrá funcionalidades que se comportarán muy similares a las de un sistema de encuestas estructuradas, por lo que el estudio del arte estará encaminado a las aplicaciones de encuestas. Se realiza un estudio de las tecnologías, metodología y herramientas, detectando las mejores prácticas para tenerlas en cuenta en la solución a proponer.

1.1. Conceptos y definiciones

Los siguientes epígrafes abordan los conceptos de Arquitectura Orientada a Servicios, Arquitectura Empresarial y modelos de madurez, así como la complejidad en las iniciativas basadas en SOA y sus fracasos.

1.1.1. Arquitectura Orientada a Servicios

El paradigma de la orientación a servicios se visualiza mejor como un método o una aproximación para alcanzar un estado objetivo específico, que se define por un conjunto de metas y beneficios estratégicos. Cuando se aplica SOA, se modelan programas de software y arquitectura de tecnologías en apoyo a la consecución de este estado objetivo. Esto es lo que califica a la arquitectura de tecnología como orientada a servicios (SOA-Manifiesto 2014).

SOA es un paradigma para organizar y utilizar capacidades distribuidas que pueden estar bajo el control de diferentes dominios de propiedad. En general, las entidades (personas y organizaciones) crean capacidades para resolver o apoyar una solución para los problemas que enfrentan en el transcurso del negocio. Es natural pensar en las necesidades de una persona satisfechas por las capacidades ofrecidas por otra persona. No hay necesariamente una correlación uno a uno entre las necesidades y capacidades, la granularidad de las necesidades y capacidades varían de fundamental a lo complejo y cualquier necesidad determinada puede requerir la combinación de numerosas capacidades mientras que una sola capacidad puede abordar más de una necesidad.

Visibilidad, la interacción y el fracaso son conceptos clave para describir el paradigma SOA. Normalmente, esto se hace proporcionando descripciones de aspectos tales como las funciones y los requisitos técnicos, las limitaciones y las políticas relacionadas, y los mecanismos para el acceso o la respuesta. Las descripciones tienen que estar en una forma (o se pueden transformar a una forma) en el que su sintaxis y la semántica son ampliamente accesibles y comprensibles (OASIS 2006).

Antes de existir las arquitecturas orientadas a servicios las TI tradicionales, los datos, las aplicaciones y las actividades de procesos empresariales a menudo se bloqueaban en "silos" independientes e incompatibles que resultaban costosos de mantener y hacían que los usuarios tuvieran que navegar en distintas redes, aplicaciones y bases de datos para realizar tareas empresariales específicas. Después que surge SOA los usuarios ya no tienen que iniciar sesión en varios sistemas, buscar los datos relevantes e integrar los resultados manualmente. Los datos de las actividades de los procesos de negocios se entregan como un servicio integrado, en una sola aplicación, en una sola pantalla, con un solo inicio de sesión (SOAction 2014).

Existe diversidad de definiciones sobre SOA (Hhoshafian 2007). Unas se enfocan en el aporte al negocio y otras en el aspecto tecnológico. Es una arquitectura desacoplada de componentes de software que proveen funciones específicas (proveedor) y que pueden ser invocadas por otros componentes (consumidor) independientemente de la plataforma en que se encuentren ambos (SOAction 2014). Es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio y permite la creación de sistemas altamente escalables que reflejan el negocio de la organización (Nicolai M. 2007).

1.1.2. Arquitectura empresarial

El concepto de arquitectura empresarial tiene su origen con la publicación del artículo titulado “Un marco para la arquitectura de sistemas de información” de J. Zachman (Zachman 1987). En ese documento, Zachman establece tanto el desafío como la visión de la arquitectura empresarial. La arquitectura empresarial es una metodología de mejora continua a mediano plazo, que basada en una visión integral, permite mantener actualizada la estructura de información organizacional alineando procesos, datos, aplicaciones e infraestructura tecnológica en cuatro dimensiones: negocios, datos/información, aplicaciones y tecnología (Amazing 2014).

Es un conjunto de representaciones gráficas y especificaciones textuales que permiten representar y entender cuáles son los procesos, organización, datos, sistemas informáticos, servicios, indicadores y demás recursos empresariales, además cómo gestionarlos y optimizarlos de forma que asegure el más alto grado de satisfacción al cliente, manteniendo un balance entre

nivel de calidad y costes. La arquitectura empresarial define las relaciones entre los principales activos de una empresa, incluyendo procesos, personas, productos, servicios, aplicaciones y tecnología (Club-BPM 2014).

El objetivo principal de la arquitectura empresarial es definir una forma ordenada de proveer a todos los niveles de la empresa un marco de trabajo definido, donde cada nivel participa, centrándose en los procesos y finalmente apoyando las estrategias y metas del negocio.

Dada esta organización, los departamentos de TI no basarán su preocupación en aplicaciones, sistemas o programas y se enfocarán en dar soporte a los procesos de negocio de la empresa, con base en una arquitectura técnica orientada a servicios y procesos.

La base técnica que hace posible dicho soporte es la arquitectura orientada a servicios, comúnmente llamada SOA. Las aplicaciones son descompuestas en pequeños servicios web (WebServices) que proveerán de las funcionalidades y lógicas de negocio básicas que alimentaran a herramientas de tipo BPM por sus siglas en inglés (Business Process Management) y así remplazar las aplicaciones por un flujo de procesos de negocio (NETSAC 2014).

1.1.3. Complejidad de las iniciativas basadas en SOA y sus fracasos

Para muchas organizaciones, la definición y ejecución de proyectos alineados a las iniciativas estratégicas, suele ser un proceso complejo por la falta de una visión integral que cubra la situación de los procesos de negocio, la tecnología para soportarlos y el establecimiento de una mejora conjunta para llegar a un estado deseado (Amazing 2014). La adopción por las organizaciones de SOA, muestra altas tasas de fracaso. Las principales causas apuntan al modo en que es adoptado este paradigma. La complejidad de este proceso debe ser abordada empleando el enfoque evolutivo que ofrecen los modelos de madurez de SOA; sin embargo, la diversidad de estos dificulta la selección y aplicación de alguno (Arias-Orizondo y Estrada-Senti 2013).

La adopción de una arquitectura basada en servicios es actualmente una necesidad para la mayoría de las empresas pues disponer de una infraestructura SOA, también es un requerimiento para que una empresa pueda lograr una implementación exitosa de sistemas que permita alcanzar las metas que propone la teoría de BPM (Weske 2007), lo que permite responder más rápido a las necesidades del negocio. Sin embargo, adoptar SOA no es una tarea sencilla, especialmente para aquellas organizaciones que no están acostumbradas a desarrollar sistemas distribuidos. Para aquellos que parten desde cero es usual sugerir que adopten la tecnología

paulatinamente, para reducir tanto la curva de aprendizaje como la probabilidad de incurrir en errores costosos.

En general se destacan cuatro áreas que imponen importantes retos y son fuentes de fracasos: la tecnología, la gestión del programa de adopción de SOA, la organización y la gobernanza. Estas dos últimas son consideradas las de mayor dificultad, pues requieren significativos cambios organizacionales que incluyen a: los métodos, modos de comunicación entre los involucrados, formas de cooperar y la cultura de la organización (Arias-Orizondo 2013).

1.1.4. Modelo de madurez

La adopción de SOA puede planificarse empleando un modelo de madurez. Estos se utilizan para ayudar a las organizaciones a saber hasta dónde han avanzado en el camino de adopción. Un modelo de madurez para SOA beneficiaría a las empresas que piensan o comenzaron a adoptar una iniciativa SOA. Les permitiría medirse y plantearse metas de mejoramiento, ubicándose objetivamente en un nivel.

Además permitiría tomar decisiones tecnológicas fundamentadas en las capacidades actuales y las capacidades deseadas. También podría ayudar a enfocar esfuerzos organizacionales ya que SOA no es solo una iniciativa exclusivamente tecnológica, gerenciales porque requiere un cambio de enfoque en las gerencias de TI y personales porque las personas de TI involucradas en este tipo de proyectos deben capacitarse y cambiar esquemas de pensamiento (Camilo Bustamante 2014).

1.2. Descripción y funcionalidades del modelo de madurez para SOA MMSOA3P

El modelo de madurez de tres perspectivas para SOA: MMSOA3P desarrollado en el CDAE. Como se ilustra en la *Figura 1*, facilita la evaluación y planificación de la adopción de SOA.

Las perspectivas de madurez agrupan las principales áreas a evaluar durante la adopción de SOA. La estructura de pirámide indica que las perspectivas inferiores brindan sustento a las superiores. Así, las capacidades de “gestión de la arquitectura” referidas a los aspectos organizacionales que inciden en la adopción de SOA, garantizan el desarrollo de la “arquitectura”, perspectiva que agrupa las capacidades técnicas. Finalmente, estas capacidades generan beneficios tanto a las TI como al negocio, agrupados en la perspectiva “resultados”. De esta forma, solo cuando se obtengan beneficios, las capacidades técnicas y organizacionales estarán confirmadas (Arias-Orizondo 2013).

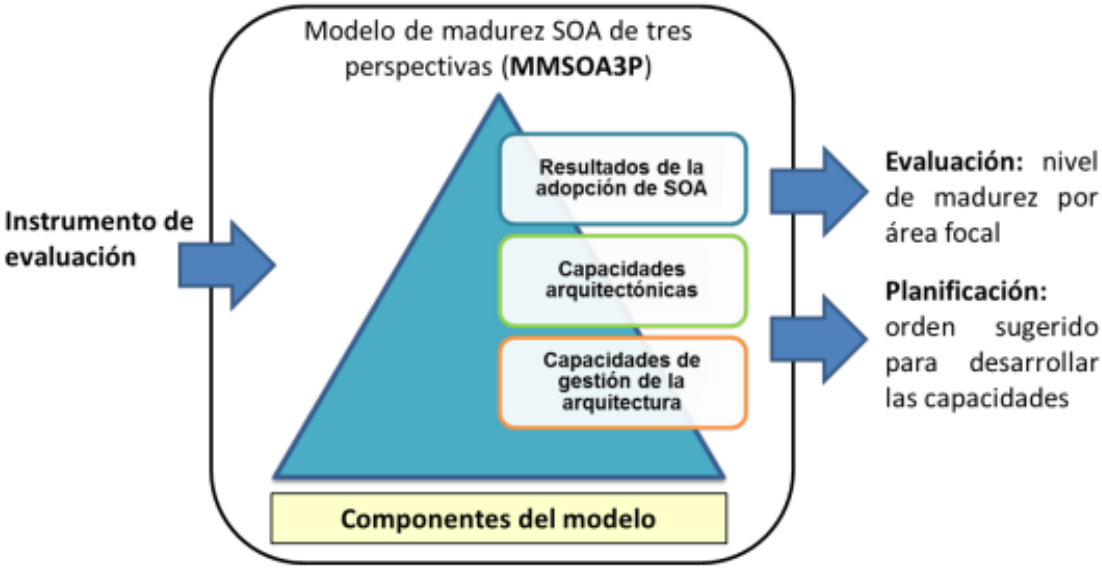


Figura 1. Estructura general de modelo de madurez (Arias-Orizondo 2013)

El modelo establece una relación causa-efecto entre capacidades y beneficios. Con ello se clarifica la forma en que SOA genera los resultados que promete.

1.2.1. Componentes del modelo de madurez y sus relaciones

Para especificar cómo se articulan las perspectivas anteriormente definidas, se establecieron los componentes lógicos que conforman el modelo de madurez propuesto y sus relaciones (Arias-Orizondo 2013). Estos se representan en la *Figura 2* y son descritos a continuación.

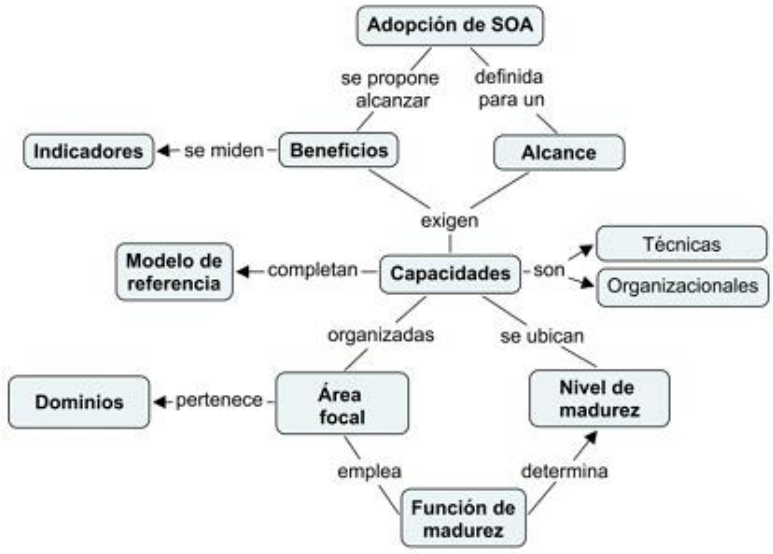


Figura 2. Elementos del modelo de madurez y sus relaciones (Arias-Orizondo 2013)

Capacidades: Una capacidad se define como una habilidad requerida para alcanzar una meta predefinida asociada a un determinado nivel de madurez. Las capacidades pueden expresarse en términos de procesos, objetos y personas. La adopción de SOA debe ser cuidadosamente analizada desde la perspectiva técnica y organizacional. Para ello, el modelo distingue capacidades técnicas y organizacionales, diferenciando así aspectos propios del desarrollo de la arquitectura y los de su gestión.

Dominios y áreas focales: Un dominio funcional se define como el conjunto de actividades, responsabilidades y actores involucrados en el cumplimiento de una función bien definida dentro de la organización. En cada dominio se distinguen varias áreas focales. Estas últimas responden a aspectos que deben ser implementados hasta cierto nivel para que sean efectivos dentro de un dominio funcional. Los niveles de madurez de cada área focal, dependen del análisis particular que se realice en cada una. Esto, a diferencia de los modelos tradicionales, permite distinguir distintos niveles de madurez entre las áreas focales, lo que resulta favorable para realizar análisis locales más profundos. Los dominios y áreas focales del modelo, cubren los aspectos relevantes de la adopción de SOA.

Modelo de referencia: Para identificar las capacidades de acuerdo a un patrón uniforme, es necesario contar con un modelo de referencia. En principio, las arquitecturas de referencia SOA son empleadas para este fin; sin embargo, aspectos particulares (ejemplo: gobierno SOA, arquitectura de datos, métodos y otros) pueden requerir referencias adicionales para profundizar en ellos y completarlos.

Beneficios o metas de adopción: La adopción de SOA tiene como propósito la obtención de beneficios para la organización. La forma en que estos son obtenidos se clarifica si se conectan con las capacidades que los habilitan, siguiendo la lógica causa-efecto.

Por otro lado, aunque en la adopción de SOA no resulta trivial cuantificar los beneficios mediante indicadores, hacerlo permite realizar una evaluación cuantitativa y más objetiva del proceso en función de los resultados que se obtengan. De esta forma se establece que las capacidades se ratifican sí generan beneficios.

Adoptar esta filosofía para estructurar el modelo, mejora el modo en que es evaluada la adopción de SOA. Esto ocurre debido a que la mayoría de los modelos de madurez son informativos (no cuentan con un modelo de evaluación asociado que exija datos objetivos y de calidad para determinar la madurez) y se enfocan a evaluar el desempeño de los procesos. En cambio, los procesos son solo habilitadores del resultado, no el resultado en sí y la contribución de los procesos como habilitadores es relativamente modesta.

Alcance del programa SOA: A un modelo de madurez debe definírsele el nivel de abstracción en que opera. SOA puede iniciarse como un proyecto piloto y luego ejecutarse como iniciativa de un departamento (áreas funcionales o unidades de negocio). Si el proyecto es validado, puede establecerse como iniciativa donde los servicios se convierten en activos empresariales compartidos para toda la organización, hasta lograr compartir funciones de negocio y datos con sus asociados, mediante entornos federados integrados por varias organizaciones.

No existe una relación uno a uno entre el alcance y la madurez. Es posible generar beneficios crecientes dentro de un mismo alcance y por tanto, lograr mayor nivel de madurez. Obviamente, estos beneficios tendrán mayor impacto en alcances mayores. Por lo anterior, definir el alcance del programa SOA es esencial, pues en este ámbito para obtener beneficios es necesario crear capacidades técnicas y organizacionales. Con la expansión progresiva de la iniciativa y el aumento del alcance, se incrementa el número de involucrados y beneficiarios, lo que complejiza el programa de adopción e impone nuevos requisitos técnicos y organizacionales para obtener mayores beneficios y mantener los ya obtenidos. Por lo anterior, un modelo de madurez es más preciso cuando se definen instancias de este para alcances específicos, determinando para cada uno las capacidades necesarias.

1.2.2. Estructura del modelo MMSOA3P y forma para representar la madurez

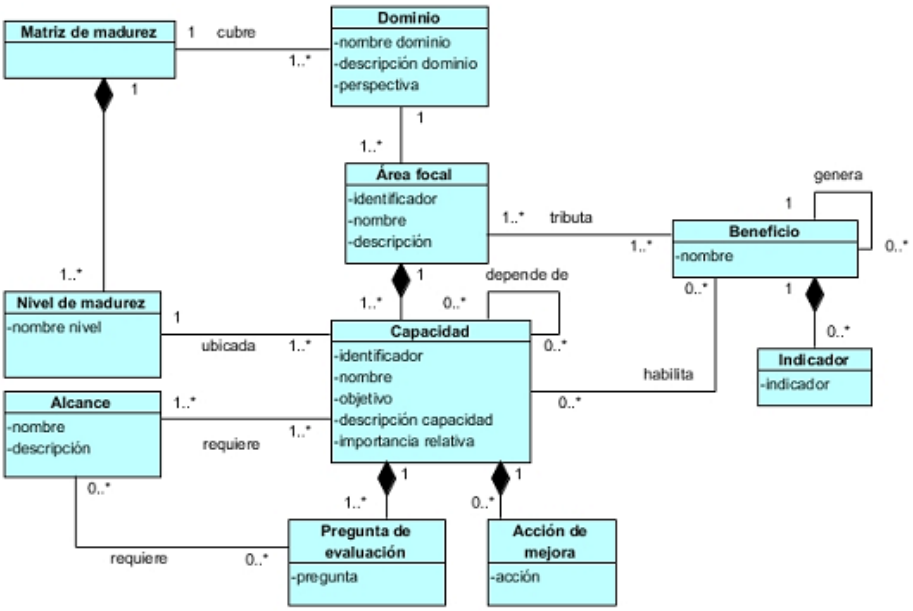


Figura 3. Representación formal de la estructura del modelo de madurez (Arias-Orizondo 2013)

Las perspectivas que evalúan el desarrollo de las capacidades son organizadas en dominios. En cada uno de ellos se distinguen las áreas focales y dentro de estas son ubicadas las capacidades.

A cada una de ellas se le asocian preguntas de evaluación cuya posible respuesta es Sí, No o Parcialmente. Para que la capacidad sea lograda, todas las preguntas deben ser respondidas positivamente. A las capacidades es posible asociarles acciones de mejora, presentadas como prácticas sugeridas para alcanzarlas.

Las capacidades se posicionan en una matriz de madurez de tal modo que reflejen relaciones de dependencia (relaciones dentro de una misma área focal y entre capacidades de distintas áreas focales). De esta forma el modelo muestra el orden en que las capacidades deben ser desarrolladas mediante pasos progresivos, lo que facilita la planificación de la mejora.

En la matriz las áreas focales son ubicadas en la columna izquierda y las capacidades por área focal son representadas por letras que muestran una progresión en la madurez. El nivel de madurez que ha alcanzado una organización en cada área focal se representa coloreando las celdas de cada fila hasta la capacidad aún no desarrollada. La columna más a la derecha que está completamente coloreada, indica la madurez que ha alcanzado la organización. El ejemplo que ilustra la *Figura 4* expresa que la organización evaluada se encuentra en el nivel 0, pues la capacidad A del área focal 8 (AF 8) no ha sido desarrollada.

Escala de Madurez / Área Focal	0	1	2	3	4	5	6	7	8	9	10	11	12	13
AF 1		A			B			C						
AF 2			A						C			D		
AF 3		A				B								
AF 4			A				B		C					
AF 5					A		B			C			D	
AF 6				A		B		C			D			
AF 7				A					C			D		
AF 8		A			B						D			
AF 9			A				B							D

Figura 4. Forma seleccionada para representar la madurez por el modelo (Arias-Orizondo 2013).

Para contribuir a la planeación de la mejora, en cada nivel de madurez se establecen **niveles de importancia** relativa a las capacidades allí ubicadas. Así el modelo no solo considera las dependencias entre las capacidades, sino que además incorpora el orden de prioridad en que deben crearse.

Las capacidades son habilitadoras de los **beneficios**. Igualmente un área focal está orientada a la obtención de resultados concretos. Por lo anterior, en el modelo se conectan los beneficios de

la adopción de SOA con aquellas capacidades que los habilitan. Ello permite clarificar la forma en que SOA genera resultados y además, establecer un mecanismo para ratificar con mayor objetividad el nivel de madurez alcanzado, pues de definirse los respectivos indicadores, podrían medirse los beneficios que generan las capacidades.

También es posible reconocer las dependencias que existen entre los beneficios, de modo que unos beneficios son consecuencia de haber alcanzado otros. Esto permite establecer un vínculo entre los beneficios que SOA genera a las TI y su impacto en el desempeño de la organización.

Para lograr mayor precisión en la evaluación, son definidas instancias del modelo para alcances bien establecidos (ejemplo: departamento, segmento de la organización, organización, entre organizaciones). Para cada una de ellas son determinadas las capacidades requeridas. Incluso, es posible establecer diferentes preguntas de control en el instrumento de evaluación del modelo, para evaluar una misma capacidad de manera diferenciada en los distintos alcances.

1.2.3. Funcionamiento del modelo: procedimientos de evaluación y planificación

Primero el modelo debe ajustarse a las características de la iniciativa SOA en curso. Para ello es determinado el alcance en que se adopta SOA y son identificadas las capacidades requeridas en ese escenario, a modo de obviar durante la evaluación aquellas capacidades propias de otros alcances. Posteriormente es aplicado el instrumento de evaluación, consistente en el cuestionario asociado a las capacidades. Con las respuestas obtenidas es posible determinar las capacidades logradas y con ello el nivel de madurez alcanzado en cada área focal. En este punto son comprobados los beneficios asociados a dichas capacidades. En los casos en que sea posible su cuantificación, quedarán ratificadas las capacidades para las que se confirmen los beneficios que prometen.

Una vez determinado el nivel de madurez alcanzado por cada área focal, se procede a planificar la mejora. El mecanismo emplea el valor de importancia relativa definido para las capacidades ubicadas dentro de un mismo nivel. De este modo el proceso de planificación es facilitado mediante el establecimiento de prioridades.

A partir de las capacidades ya logradas por cada una de las áreas focales, son identificadas aquellas no desarrolladas ubicadas en el próximo nivel de madurez. Estas serán las siguientes a desarrollar. Como en la matriz de madurez las capacidades se ubican teniendo en cuenta sus relaciones de dependencia, para establecer un orden son seleccionadas primero las capacidades no desarrolladas de menor nivel de madurez. Las de un mismo nivel son ordenadas de acuerdo a su importancia relativa. De esta forma se procede con el resto de las capacidades ubicadas en

los siguientes niveles, las que se van ordenando respetando la sucesión de niveles de madurez. La *Figura 5* ilustra los pasos antes descritos para completar los procedimientos de evaluación y planificación.

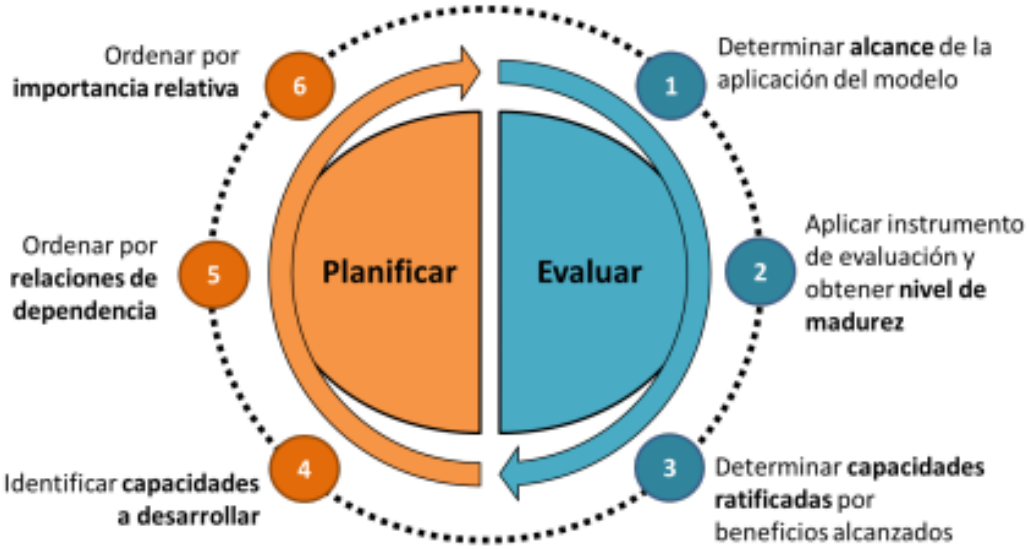


Figura 5. Representación del funcionamiento del modelo de madurez (Arias-Orizondo 2013).

1.2.4. Materialización del modelo de madurez

La materialización es el proceso que permite poblar la estructura del modelo de madurez. Para ello se ejecutan dos procesos en paralelo como muestra la *Figura 6*.

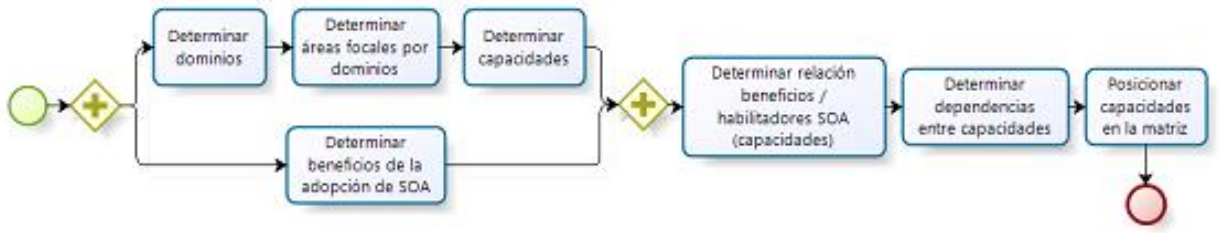


Figura 6. Actividades para habilitar la estructura del modelo de madurez (Arias-Orizondo 2013).

Por un lado, se determinan los dominios, áreas focales y capacidades. Por otro lado, se determinan los beneficios, los que permiten acotar la estructura del modelo bajo la siguiente premisa: “los beneficios están condicionados por las capacidades”; por tanto, cada dominio, área focal y las capacidades, deben tributar al logro de los beneficios identificados.

1.2.5. Perspectiva de madurez “resultados”

La perspectiva “resultados” está conformada por los beneficios fundamentales a las TI y al negocio que genera la adopción de SOA. Además, incluye las relaciones entre los beneficios para clarificar la forma en que se obtienen los resultados.

1.2.6. Perspectivas de madurez “arquitectura” y “gestión de la arquitectura”

El modelo organiza las capacidades en dos perspectivas: “arquitectura” y “gestión de la arquitectura”. Los dominios definidos en ellas, cubren los aspectos relevantes a considerar durante la adopción de SOA.

1.3. Análisis de aplicaciones existentes

Un modelo de madurez es un instrumento de evaluación que para gestionar y aplicar en alguna institución, funciona la mayoría de las veces como una encuesta estructurada. Por lo que surge la necesidad de estudiar sistemas informáticos que faciliten el proceso de gestionar una encuesta. En este epígrafe se realiza un análisis sobre la herramienta desarrollada en el CDAE que implementó un modelo de madurez y sobre sistemas informáticos para el procesamiento de encuestas estructuradas más utilizadas a nivel mundial, que aportarán conocimiento y servirán de sostén para la realización de la presente investigación.

1.3.1. Herramienta desarrollada por el CDAE que implementa un modelo de madurez.

Es un software de escritorio que permite gestionar la primera versión del modelo de madurez creado por el CDAE (Chacón Casas 2012), un modelo con otra estructura y funcionalidades diferentes a las del modelo de madurez MMSOA3P. Además por ser una aplicación de escritorio hay que instalarla previamente en cada estación de trabajo (PC) donde se tenga que utilizar y no puede ser integrada a herramientas web existentes en el CDAE como el portal del intranet.

Esta herramienta no facilita funcionalidades como la gestión de dominios, áreas focales, agrupar las áreas focales, definir acciones de mejoras, crear planificación y otras que no fueron necesarias en el modelo de madurez que implementa. Además cuenta con una estructura de datos diferente lo que lleva a nuevos requisitos y la necesidad de una nueva herramienta.

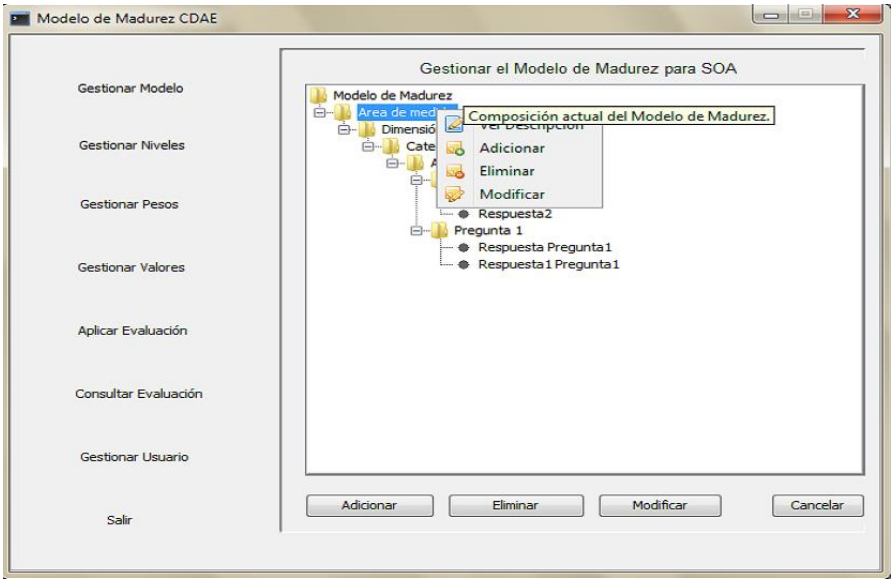


Figura 7. Herramienta desarrollada por el CDAE para gestionar un modelo de madurez

1.3.2. Gestor de encuestas Encuesta Tick

Encuesta Tick es el servicio que ofrece el portal de encuestas orientado especialmente para empresas, con el objetivo de facilitar información sobre el consumidor a los productores. Se encuentra en www.encuestatick.com donde se puede crear todas las encuestas que se quieran y empezar a recibir todos los resultados al instante. Una vez registrado se puede empezar a crear o editar plantillas o ejemplos de encuestas, enviarlas por correo electrónico o publicarlas en la web o blogs mediante un enlace, y cuando los encuestados rellenan la encuesta, las respuestas se guardan al instante y entonces se puede empezar a ver resultados (EncuestaTick 2014). Aunque cuenta con versiones que solo mediante pago se podrá personalizar la apariencia de las encuestas, crear la encuesta en varios idiomas y exportar resultados en PDF y en Excel, opciones que no se encuentran gratuitamente (EncuestaTickTarifas 2014).

1.3.3. Gestor de encuestas Polldaddy

Polldaddy es un sistema en línea que te permite realizar encuestas y cuestionarios de forma gratuita. Tiene tres versiones, de las cuales, la primera y más básica es gratis y las dos siguientes con capacidad mayor son de paga. Los costos para las versiones de paga oscilan entre $200 a $900 USD por año. La diferencia entre versiones son básicamente dos: El número de preguntas por cuestionario y el número de respuestas para cada cuestionario (PollDaddy 2014).

Cuenta con varías funcionalidades que pueden ser útiles como crear encuestas y cuestionarios impresionantes en cuestión de minutos. Recoge las respuestas a través de una página web, correo electrónico, IPAD, Facebook y Twitter, genera y comparte fácilmente los informes de

resultados y estadísticas presentándolos en tiempo real además de ser muy completos. La mayoría de las funcionalidades que posee solo se encuentran en las versiones de pago por lo que se necesitaría pagar licencias con altos precios.

1.3.4. Gestor de encuestas Opina

Opina permite modelar cuestionarios y/o encuestas. Con ella se pueden crear desde cualquier punto con acceso a internet (web), cuestionarios, gestionar usuarios (encuestados), realizar modificaciones, así como parametrizar y configurar todas las opciones de las que consta.

Algunas de las opciones más destacadas de la aplicación son:

- Creación de cuestionarios y encuestas de forma personalizada.
- Distintos tipos de cuestiones: numéricas, elección múltiple, gradientes, abiertas y matrices.
- Configuración de mensajes de validación.
- Configuración de la fecha de inicio y fin de los cuestionarios y encuestas.
- Número indefinido de cuestionarios por usuario encuestador.
- Número indefinido de cuestiones por cuestionario.
- Paginación de las cuestiones.
- Organización de los cuestionarios por carpetas.
- Portapapeles para copiar y mover cuestionarios
- Generación de informes globales e individuales.
- Tres modos de acreditación y 3 modos de registro para los usuarios encuestados.
- Asociación de usuarios a cuestionarios y encuestas.

Todas estas funcionalidades representan una fuente de valor y conocimiento para el estudio de las encuestas estructuradas pero no son suficientes para gestionar un modelo de madurez. Se necesita además gestionar el modelo de datos específico del modelo lo cual no se puede hacer con este sistema.

1.3.5. Gestor de encuestas Encuestafácil

Encuestafácil es una herramienta web de encuestas online. Permite a los usuarios elaborar por sí mismos, de una forma rápida y sencilla, encuestas internas y externas que ayuden en la toma de decisiones. Encuestafácil permite obtener información en tiempos muy rápidos y sin destinar muchos recursos aunque para configurar la estructura de las encuestas es necesario usar versiones de pago (Encuestafácil 2014).

1.4. Selección de tecnologías y herramientas que se utilizarán en el desarrollo de la solución.

Actualmente hay una tendencia fuerte que consiste en diseñar aplicaciones de gestión utilizando como software de cliente los exploradores de Internet y esto trae en sí muchas ventajas como: en cualquier sitio se tiene la oportunidad de usar los exploradores Web y todos tienen un comportamiento genérico y análogo y los diseños que se pueden hacer en Web son más atractivos y vistosos debido a la especialización del sistema Web para mostrar información, aunque existen muchas ventajas y desventajas entre las aplicaciones Web y las que poseen interfaz de usuario de escritorio como se pueden ver en la *Tabla 1*.

Tabla 1. Tabla comparativa entre aplicaciones web y de escritorio (OSD 2014).

Característica	Web	Escritorio
Personalización, actualización y soporte	Es suficiente con realizar los cambios en el servidor WEB	Hay que realizarlos en cada estación de trabajo (PC) donde se tenga la aplicación
Accesibilidad y cobertura	Cualquier lugar con acceso a Internet	Solo en el computador donde se haya instalado previamente el software
Capacidad de usuarios concurrentes	Alta debido a la arquitectura de clientes livianos que la pueden usar	Baja ya que la forma de de diseño es centrada en un único usuario local
Portabilidad	El sistema puede ser usado con cualquier navegador de Internet	Solo funciona en el sistema operativo para el cual fue creado
Infraestructura y movilidad	Solo se tiene que conectar a la Internet	Está restringido a la ubicación del computador local
Seguridad eléctrica y lógica	Es responsabilidad del proveedor del servicio	Es responsabilidad del administrador de la compañía y de cada usuario que usa el sistema localmente

Las aplicaciones Web para internet e intranet presentan muchas ventajas con respecto al software de escritorio, logrando así aprovechar y acoplar los recursos de su empresa de una forma más práctica que el software tradicional, por lo que teniendo en cuenta las necesidades del cliente, la elección del desarrollo de una aplicación Web es la respuesta más factible al proceso de gestionar de manera flexible la información referente al nuevo modelo de madurez para SOA y además se consideró la condición de que el sistema pueda ser accedido por muchos usuario en diferentes lugares y puedan ser integrados los resultados al portal de intranet del CDAE.

1.4.1. Servidor de aplicaciones

Un servidor de aplicaciones consiste en un contenedor que abarca la lógica de negocio de un sistema, y que provee respuestas a las peticiones de distintos dispositivos que tienen acceso a ella. Como consecuencia del éxito del lenguaje de programación Java, el término servidor de aplicaciones usualmente hace referencia a un servidor de aplicaciones J2EE[2].

Entre los servidores de aplicación Java EE privativos más conocidos se encuentran WebLogic de Oracle (antes BEA Systems) y WebSphere de IBM. EAServer de Sybase Inc. es también conocido por ofrecer soporte a otros lenguajes diferentes a Java, como PowerBuilder. Entre los servidores de aplicaciones libres se encuentran JOnAS del consorcio ObjectWeb, JBoss AS de JBoss (división de Red Hat), Geronimo de Apache, TomEE de Apache, Resin Java Application Server de Caucho Technology, Blazix de Desiderata Software, Enhydra Server de Enhydra.org y GlassFish de Oracle. Tomcat es solamente un contenedor de servlets (Apache 2014) aunque es la mejor opción para desplegar en un servidor el sistema propuesto por la gran cantidad de conocimiento y documentación que se tiene sobre este en el CDAE.

1.4.2. Lenguaje de programación.

Se elige Java como lenguaje de programación ya que el mismo es el más utilizado en el centro CDAE y se mantienen las políticas del departamento de Java del mismo. Es un lenguaje compilado de programación orientado a objetos, las clases que genera son interpretadas por una máquina virtual, siendo esta la que mantiene el control sobre las clases que se estén ejecutando. Es multiplataforma, independiente del sistema operativo en el que se trabaje y cuando la máquina virtual de java está ejecutando algún código realiza comprobaciones de seguridad, lo cual permite que los errores sean tratados en el momento que son percibidos.

[2] **Java Platform, Enterprise Edition** o **Java EE** (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4; traducido informalmente como **Java Empresarial**), es una plataforma de programación—parte de la Plataforma Java—para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java.

Por último, debido a la Máquina Virtual de Java (JVM) las aplicaciones desarrolladas en Java funcionan en Linux, Windows, Mac OS, y en cualquier sistema operativo para el cual exista una JVM, garantizando que no se dependa de un único sistema operativo para el funcionamiento de la solución.

1.4.3. Entorno de desarrollo integrado.

En la solución propuesta se decide utilizar **Eclipse** como entorno de desarrollo integrado (IDE por sus siglas en inglés) pues es un entorno de desarrollo integrado capaz de proveer un conjunto de herramientas para administrar espacios de trabajo, construir, correr y depurar aplicaciones, además posee una arquitectura de módulos llamados plug-ins que le permite incorporarle múltiples funcionalidades, como lo es la instalación de un plug-ins para el trabajo e integración con todas las herramientas utilizadas. Es un entorno de desarrollo diseñado para ser extendido con sofisticadas herramientas.

1.4.4. Framework

El concepto framework se emplea en muchos ámbitos del desarrollo del software. En general, se refiere a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Desde el punto de vista del desarrollo de software, un framework es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado (Alegsa 2014).

Los frameworks suelen incluir:

- Soporte de programas.
- Bibliotecas.
- Lenguaje de scripting.
- Software para desarrollar y unir diferentes componentes de un proyecto de desarrollo de programas.

Los frameworks permiten:

- Facilitar el desarrollo de software.
- Evitar los detalles de bajo nivel, permitiendo concentrar más esfuerzo y tiempo en identificar los requerimientos de software.

1.4.5. Framework Mapeo Objeto Relacional (ORM)

Cuando se trabaja bajo el paradigma “Orientado a objetos” y a la vez se utiliza el de sistemas de bases de datos relacionales, los desarrolladores afrontan las desavenencias de estos dos

enfoques de trabajo. Un ORM es un framework que propone una nueva forma de modelar los datos, permitiendo solucionar la diferencia que existe entre estos dos paradigmas.

Se decide utilizar Hibernate en su versión 3.5.2 pues es una herramienta ORM de código abierto muy madura y completa. Agiliza la relación entre la aplicación y la base de datos (Educación IT 2014). Tiene como objetivo facilitar el mapeo de atributos entre una base de datos relacional y un modelo de objetos, a través de archivos (XML) o con anotaciones en las entidades (Codecriticon 2014).

1.4.6. Framework Primeface v3.5 para el desarrollo de componentes visuales

Uno de los framework más exitosos hoy en día y que muchos programadores WEB tienen ánimo de ponerlo en práctica. PrimeFaces, es una librería de componentes visuales OPEN SOURCE para JSF[3], así como IceFaces o RichFaces.

Entre sus bondades se tienen:

- Librería con alrededor de 100 componentes Ajax de fácil uso.
- No requiere unas complicadas configuraciones.
- Showcase de ejemplo para descarga.
- Documentación.
- Temas pre-configurados.

1.4.7. Framework Spring v3.2

Spring es un framework que se puede emplear en todo tipo de aplicaciones java, ya sean pequeñas aplicaciones web o voluminosos sistemas que distribuyen su carga entre varios servidores.

Entre otras cosas permite independizar la configuración de la aplicación del servidor en que dicha aplicación se encuentre, evitando así tener que configurar recursos en cada uno de los servidores donde se despliegue o depender de descriptores específicos de determinados servidores comerciales (Consultoría Java 2014).

Spring proporciona:

- Una potente gestión de configuración basada en JavaBeans[4], aplicando los principios de Inversión de Control (IoC). Esto hace que la configuración de aplicaciones sea rápida y

[3] **JavaServer Faces** (**JSF**) es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE.
[4] Componentes de software reutilizables que se puedan manipular visualmente en una herramienta de construcción.

sencilla. Ya no es necesario tener singletons[5] ni ficheros de configuración, una aproximación consistente y elegante. Esta factoría de beans[6] puede ser usada en cualquier entorno, desde applets[7] hasta contenedores J2EE. Estas definiciones de beans se realizan en lo que se llama el contexto de aplicación.

- Una capa genérica de abstracción para la gestión de transacciones, permitiendo gestores de transacción enchufables, y haciendo sencilla la demarcación de transacciones sin tratarlas a bajo nivel. Se incluyen estrategias genéricas para JTA[8] y un único JDBC DataSource. En contraste con el JTA simple o EJB[9], el soporte de transacciones de Spring no está atado a entornos J2EE.
- Una capa de abstracción JDBC que ofrece una significativa jerarquía de excepciones (evitando la necesidad de obtener de SQLException los códigos que cada gestor de base de datos asigna a los errores), simplifica el manejo de errores, y reduce considerablemente la cantidad de código necesario.
- Integración con Hibernate, JDO e iBatis SQL Maps en términos de soporte a implementaciones DAO [10]y estrategias con transacciones. Especial soporte a Hibernate añadiendo convenientes características de IoC, y solucionando muchos de los comunes problemas de integración de Hibernate. Todo ello cumpliendo con las transacciones genéricas de Spring y la jerarquía de excepciones DAO.
- Funcionalidad AOP[11], totalmente integrada en la gestión de configuración de Spring. Se puede aplicar AOP a cualquier objeto gestionado por Spring, añadiendo aspectos como gestión de transacciones declarativa. Con Spring se puede tener gestión de transacciones declarativa sin EJB, incluso sin JTA, si se utiliza una única base de datos en un contenedor web sin soporte JTA.
- Un framework MVC (Model-View-Controller), construido sobre el núcleo de Spring. Este framework es altamente configurable vía interfaces y permite el uso de múltiples tecnologías para la capa vista como pueden ser JSP, Velocity, Tiles, iText o POI. De cualquier manera una capa modelo realizada con Spring puede ser fácilmente utilizada

[5] El patrón de diseño singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.
[6] Un Bean es un componente software que tiene la particularidad de ser reutilizable.
[7] Un *applet* es un componente de una *aplicación* que se ejecuta en el contexto de otro programa.
[8] JTA (del inglés Java Transaction API - API para transacciones en Java) es parte de Java EE APIs.
[9] Enterprise JavaBeans (también conocidos por sus siglas EJB) son una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE.
[10] Data Access Object (DAO, Objeto de Acceso a Datos).
[11] Programación orienta a aspectos.

con una capa web basada en cualquier otro framework MVC, como Struts, WebWork o Tapestry.

Todas estas funcionalidades pueden usarse en cualquier servidor J2EE, y la mayoría de ella ni siquiera requiere su uso. El objetivo central de Spring es permitir que objetos de negocio y de acceso a datos sean reusables, no atados a servicios J2EE específicos.

En la arquitectura en capas de Spring *Figura 8*, toda la funcionalidad está construida sobre los niveles inferiores. Por ejemplo, se puede utilizar la gestión de configuración basada en JavaBeans sin utilizar el framework MVC o el soporte AOP (Sourceforge 2014).

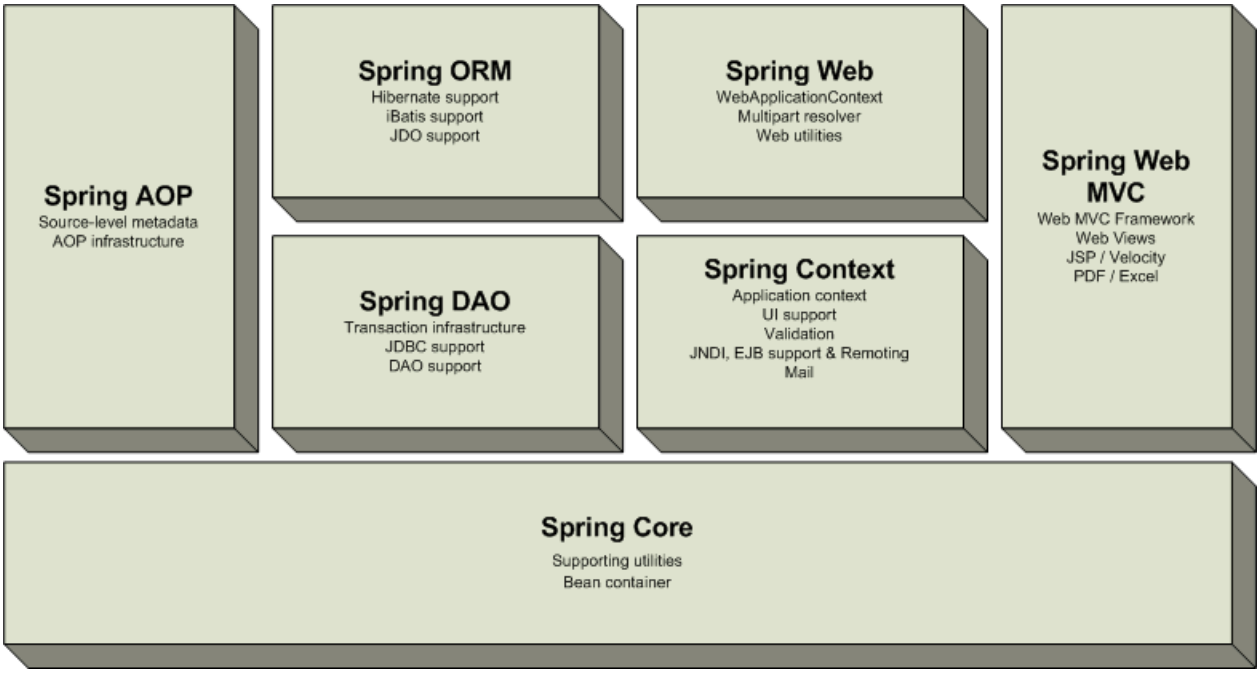


Figura 8. Spring: arquitectura en capas (Consultoría Java 2014).

1.4.8. Sistema Gestor de Bases de Datos PostgresSQL

PostgreSQL es un Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos. Es un gestor de bases de datos de código abierto, brinda un control de concurrencia multi-versión (MVCC por sus siglas en inglés) que permite trabajar con grandes volúmenes de datos; soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación, entre el que se encuentra Java.

Posee características significativas del motor de datos, entre las que se pueden incluir las subconsultas, los valores por defecto, las restricciones a valores en los campos (constraints) y los disparadores (triggers). El código fuente se encuentra disponible para todos sin costo alguno. Posee una integridad referencial e interfaces nativas para lenguajes como ODBC, JDBC, C, C++,

PHP, PERL, TCL, ECPG, PYTHON y RUBY. Funciona en todos los sistemas operativos Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows. Por todas estas características se decide su utilización en la solución planteada.

1.5. Metodología del desarrollo del software

Una metodología de desarrollo de software se refiere al entorno que se usa para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información. Una gran variedad de metodologías se han desarrollado a lo largo de los años, cada una de ellas con sus fortalezas y debilidades. Una determinada metodología no es necesariamente aplicable a todo tipo de proyectos, más bien cada tipo de proyecto tiene una metodología a la que se adapta mejor (Romero 2014).

1.5.1. Metodologías pesadas

Son las más tradicionales, se centran en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar, y requiere una extensa documentación, ya que pretende prever todo de antemano. Este tipo de metodologías son más eficaces y necesarias cuanto mayor es el proyecto que se pretende realizar respecto a tiempo y recursos que son necesarios emplear, donde una gran organización es requerida. Una de las metodologías pesadas más conocidas y utilizadas es la Metodología RUP (Rational Unified Process) que divide el desarrollo en 4 fases que definen su ciclo de vida (ProyectosAgiles 2014):

- Inicio: El objetivo es determinar la visión del proyecto y definir lo que se desea realizar.
- Elaboración: Etapa en la que se determina la arquitectura óptima del proyecto.
- Construcción: Se obtiene la capacidad operacional inicial.
- Transmisión: Obtener el producto acabado y definido.

1.5.2. Proceso Unificado de Desarrollo de Software

Actualmente no existe una metodología de desarrollo de software que sea global, es decir que encierre características que puedan aplicarse a cualquier tipo de proyecto. Las características de cada proyecto conjuntamente con su equipo de desarrollo, recursos, y requisitos exigen que se escoja una que se adapte en la mayor medida posible a estas características (ProyectosAgiles 2014).

Elementos de RUP

- Actividades: Procesos que se han de realizar en cada etapa/iteración.
- Trabajadores: Personas involucradas en cada actividad del proyecto.
- Artefactos: Herramientas empleadas para el desarrollo del proyecto.

1.5.3. Metodologías ágiles

Las metodologías tradicionales han intentado abordar la mayor cantidad de situaciones de contexto del proyecto, exigiendo un esfuerzo considerable para ser adaptadas, sobre todo en proyectos pequeños y con requisitos muy cambiantes. Las metodologías ágiles ofrecen una solución casi a medida para una gran cantidad de proyectos que tienen estas características. Una de las cualidades más destacables en una metodología ágil es su sencillez, tanto en su aprendizaje como en su aplicación, reduciéndose así los costos de implantación en un equipo de desarrollo (Beck, K. 1999).

Esto ha llevado hacia un interés creciente en las metodologías ágiles. Sin embargo, hay que tener presente una serie de inconvenientes y restricciones para su aplicación, tales como: están dirigidas a equipos pequeños o medianos, el entorno físico debe ser un ambiente que permita la comunicación y colaboración entre todos los miembros del equipo durante todo el tiempo, cualquier resistencia del cliente o del equipo de desarrollo hacia las prácticas y principios puede llevar al proceso al fracaso (el clima de trabajo, la colaboración y la relación contractual son claves) (Beck 1999).

1.5.4. Scrum

Scrum es un proceso en el que se aplican de manera regular un conjunto de mejores prácticas para trabajar en equipo y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos. Más que una metodología de desarrollo software, es una forma de auto-gestión de los equipos de programadores. Un grupo de programadores deciden cómo hacer sus tareas y cuánto van a tardar en ello. Ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro. Permite además seguir de forma clara el avance de las tareas a realizar, de forma que los "jefes" puedan ver día a día cómo progresa el trabajo. En Scrum se realizan entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor del proyecto (ProyectosAgiles 2014).

1.5.5. Extreme Programming

XP es una metodología ágil de desarrollo de software creada por Kent Beck para pequeños y medianos equipos de desarrollo. Está enfocada en ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto, siendo más realista que al intentar definir todos los requisitos al comienzo del proyecto y después invertir esfuerzos después en controlar los cambios en los requisitos. Entre las principales características de esta metodología se encuentran (Beck 1999):

- La incorporación del cliente al equipo de desarrollo
- La programación se realiza en parejas
- El proyecto inicia con un simple diseño
- Realiza pruebas e integra el sistema completo varias veces en el día

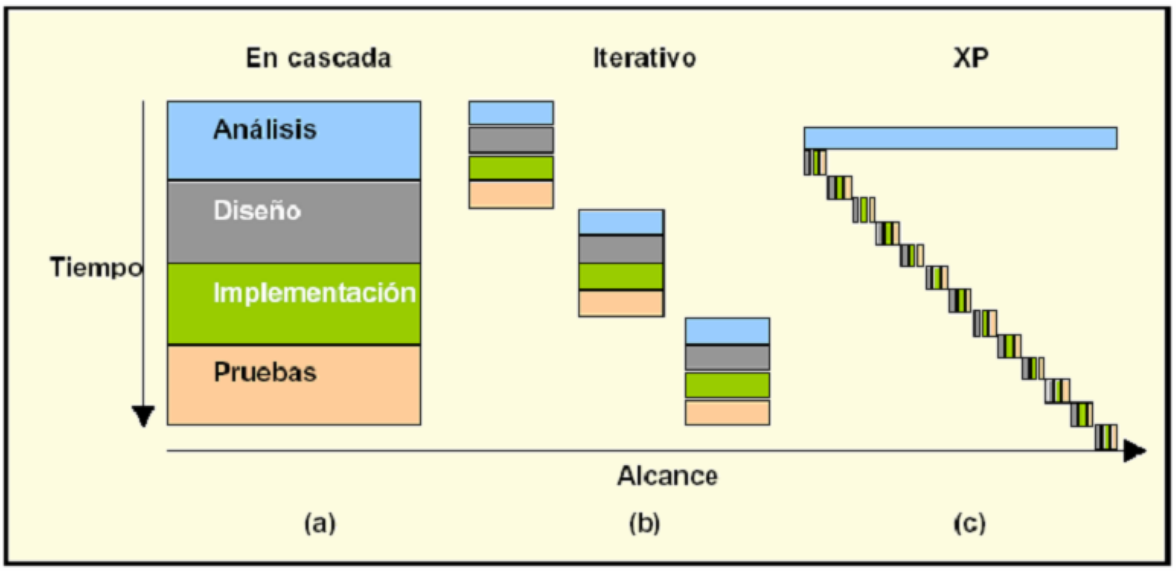


Figura 9: Comparación de los ciclos de desarrollo en cascada, iterativos y XP (Joskowicz 2008)

El ciclo de vida de un proyecto XP es muy dinámico (ver *Figura 9*), basado en pequeñas iteraciones en las cuales se llevan a cabo recursivamente los procesos de análisis, diseño, implementación y pruebas (Joskowicz 2008).

Los valores que rigen la programación extrema son: simplicidad, comunicación, retroalimentación (feedback), respeto y coraje (Beck, K. 1999).

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas.

Las historias de usuarios sustituyen a los documentos de especificación funcional, y a los casos de uso. Estas historias son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle

requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios. Las historias de usuarios deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia (Joskowicz 2008).

Quedan estructuradas por: código, nombre, referencia, prioridad, puntos de estimación, descripción y observación y sus principales características son (Beck, K. 1999):

- Independientes unas de otras: De ser necesario, combinar las historias dependientes o buscar otra forma de dividir las historias de manera que resulten independientes.
- Negociables: La historia en sí misma no es lo suficientemente explícita como para considerarse un contrato, la discusión con los usuarios debe permitir esclarecer su alcance y éste debe dejarse explícito bajo la forma de pruebas de validación.
- Valoradas por los clientes o usuarios: Los intereses de los clientes y de los usuarios no siempre coinciden, pero en todo caso, cada historia debe ser importante para alguno de ellos más que para el desarrollador.
- Estimables: Un resultado de la discusión de una historia de usuario es la estimación del tiempo que tomará completarla. Esto permite estimar el tiempo total del proyecto.
- Pequeñas: Las historias muy largas son difíciles de estimar e imponen restricciones sobre la planificación de un desarrollo iterativo. Generalmente se recomienda la consolidación de historias muy corta.
- Verificables: Las historias de usuario cubren requerimientos funcionales, por lo que generalmente son verificables.

Adicionalmente a cada historia de usuario se le asigna un número para facilitar su identificación por parte del equipo de desarrollo y se le puede incluir alguna información adicional que pueda ser útil para la comprensión de la misma.

Dada las características de XP y los valores en los cuales se sustenta esta metodología, es la se decide usar en el desarrollo del sistema, disminuyendo los riesgos por los requisitos cambiantes y ajustándose a equipos de desarrollo pequeños en los cuales la comunicación es de suma importancia (Schuh 2001).

El ciclo de vida de un proyecto con metodología XP es muy dinámico, por lo que se separa en fases. En el presente trabajo se abordarán cuatro fases que plantea esta metodología (Joskowicz 2008): exploración, planificación, iteración y producción.

1.6. Patrones de diseño

En la tecnología de objetos un Patrón es una descripción de un problema y la solución, a la que se le da un nombre, y que se puede aplicar a nuevos contextos. Un patrón de diseño puede considerarse como un documento que define una estructura de clases que aborda una situación particular. Los patrones de diseño se dividen en tres grupos principales (Kioskea 2014): patrones de creación, patrones estructurales y patrones funcionales.

Ejemplos de patrones de diseño:

- Patrón **MVC** (Modelo-Vista-Controlador), La tecnología Java Server Faces es un marco de trabajo de interfaces de usuario del lado de servidor para aplicaciones Web basadas en tecnología Java que es la que se usa en la solución propuesta. Útil con aplicaciones basadas en la arquitectura MVC (Oliver 2014): proviene del principio de que las aplicaciones se pueden dividir en tres áreas separadas (Kioskea 2014):
    - **Modelo**: los datos utilizados en la aplicación
    - **Vista**: cómo se representan los datos al usuario
    - **Controlador**: cómo se procesa la información en la interfaz del usuario
- **Proxy**: es el patrón que define el objeto intermediario que pide un objeto remoto y que es transparente para el usuario.

1.6.1 Patrones de asignación de responsabilidades (GRASP)

Los patrones GRASP[12] describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable (Saavedra 2014). Los principales patrones GRASP son: creador, experto, controlador, alta cohesión, bajo acoplamiento y fabricación pura.

Consideraciones parciales del capítulo

En el presente capítulo, se ha realizado un amplio análisis de los conceptos y aspectos fundamentales a tener presente durante todo el ciclo de vida de la solución propuesta, así como las herramientas y tecnologías que serán utilizadas, seleccionando la Metodología XP para el

[12] Acrónimo de General Responsibility Assignment Software Patterns y son patrones generales de software para asignación de responsabilidades.

desarrollo, utilizando java como lenguaje de programación y como entorno de desarrollo por las características que presenta, las cuales ofrecieran ventajas de su utilización para el grupo de desarrollo. Partiendo de lo planteado, que al realizar una evaluación a alguna empresa, haciendo uso de un modelo de madurez para SOA, el proceso que se desarrolla es muy similar al proceso de aplicar una encuesta estructurada, se realizó un estudio de algunas aplicaciones que permitan la gestión y aplicación de encuestas. Se obtienen conocimientos que serán aplicados en el desarrollo de la solución propuesta.

CAPÍTULO 2. PROPUESTA DE SOLUCIÓN DEL SISTEMA PARA LA ADMINISTRACIÓN Y APLICACIÓN DEL MODELO DE MADUREZ MMSOA3P.

Introducción

En el presente capítulo se precisan un conjunto de elementos para conformar la propuesta de solución de la presente investigación así como sus características y componentes, además de describir el funcionamiento general del sistema que se desea implementar. Se realiza una descripción de la arquitectura propuesta para la solución, así como el uso de patrones presentes en el mismo.

Se aplica la metodología de desarrollo seleccionada y se definen la documentación significativa que esta propone en sus primeras fases del ciclo de desarrollo. Además, se abordan las fases de Exploración y Planificación propias de la metodología de desarrollo utilizada en la implementación del sistema informático, en las cuales se exponen las historias de usuario seccionada en iteraciones, los planes de entrega y la duración de las iteraciones durante el desarrollo del sistema.

2.1. Necesidad del desarrollo de una nueva aplicación para gestionar el modelo de madurez SOA.

Después de realizar un detenido y profundo análisis se llega a la conclusión de que los sistemas informáticos estudiados representan una fuente de valor y conocimiento para la investigación. Sirven de sostén para la elaboración de la misma, aportando conocimiento sobre las estructuras de preguntas estructuras, así como el tipo de pregunta que debería gestionarse en el modelo y el tipo de respuestas que se deberían dar.

Pero no se toma ninguna como solución válida ya que para la gestión y aplicación de un modelo de madurez para SOA se requieren de otras funcionalidades como la gestión del modelo de datos que propone MMSOA3P y aplicación del mismo, las cuales no pueden ser desarrolladas por estos sistemas. Además para la visualización su se hace uso de una estructura de datos en forma de árbol, existe la necesidad de configurar encuestas estructuradas de una forma muy específica y dinámica, se necesitan muchas características específicas del modelo propuesto y la necesidad de una solución libre para mantener la soberanía nacional. Los elementos anteriores constituyen aspectos fundamentales en la solución para una mejor comprensión del modelo de madurez y estos sistemas estudiados no permiten dichas funcionalidades.

2.2. Descripción de la solución propuesta

Con el objetivo de dar solución al problema de investigación planteado en el capítulo anterior, se propone desarrollar un sistema que gestione el modelo de madurez para SOA de tres perspectivas propuesto por el CDAE, de manera flexible e intuitiva.

2.2.1. Principales interfaces de la aplicación

Para el desarrollo del sistema propuesto fueron diseñadas e implementadas varias interfaces de usuario para lograr la comunicación entre el usuario y el sistema. En la *Figura 10* se muestra la interfaz donde se encuentran los módulos existentes en la aplicación.

Figura 10: Módulos de la aplicación.

El módulo editar matriz facilita la edición del modelo y la conformación de la matriz de madurez. Ello permite crear y/o modificar las áreas focales, las capacidades asociadas, sus relaciones de dependencia, niveles de importancia relativa y las preguntas de control del instrumento de evaluación, lo que posibilita además la definición de instancias del modelo para alcances específicos.

El resto de los módulos viabilizan la aplicación del modelo. Estos permiten realizar ajustes al instrumento de evaluación para adaptarlo a entornos concretos. Además, facilitan el procesamiento de las respuestas obtenidas como resultado de la aplicación del instrumento de evaluación, para determinar y visualizar el nivel de madurez alcanzado y sugerir las próximas capacidades a desarrollar de forma ordenada.

En las figuras *Figura 11*, *Figura 12* y *Figura 13* se muestran las interfaces principales correspondientes al módulo Editar Matriz, Organización y Evaluación respectivamente donde se visualizan los datos más importantes de cada uno.

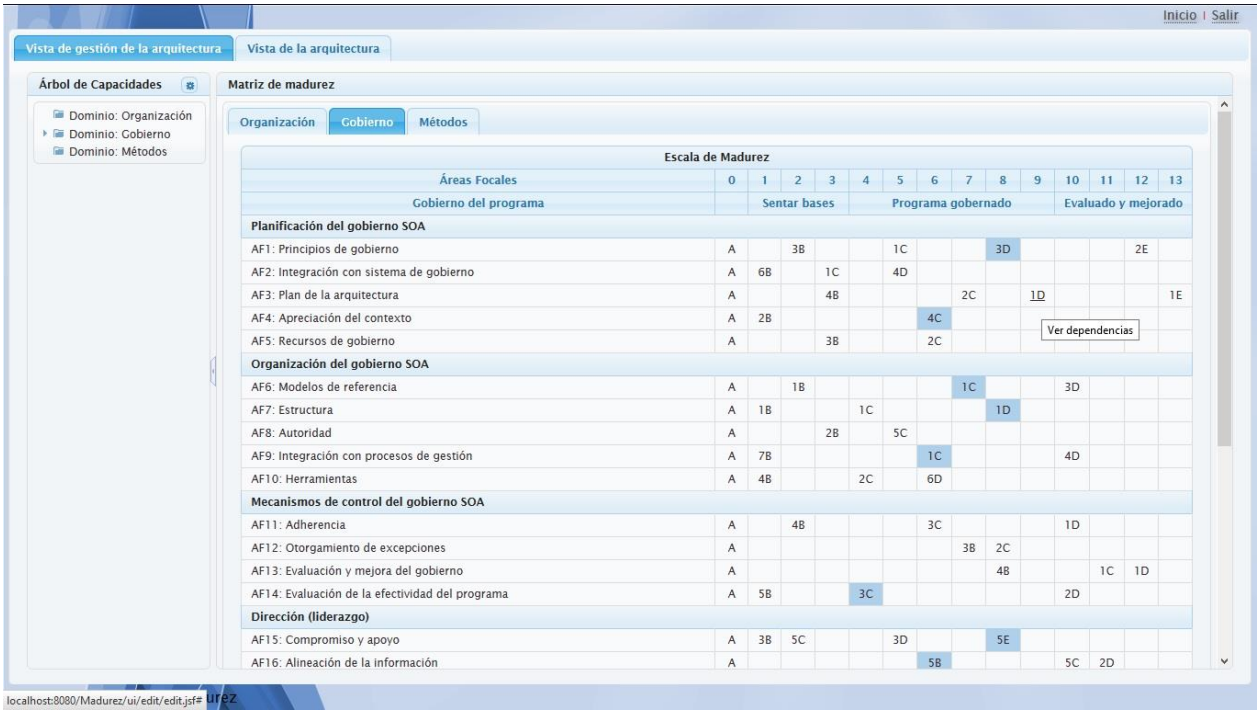


Figura 11: Interfaz principal del módulo Editar Matriz.

Figura 12: Interfaz principal del módulo Organización.

Figura 13: Interfaz principal del módulo Evaluación.

2.3. Historias de usuarios

En la fase de exploración el cliente define lo que necesita mediante la redacción de sencillas "historias de usuarios". Los programadores estiman los tiempos de desarrollo en base a esta información. Debe quedar claro que las estimaciones realizadas en esta fase son primarias (ya que estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más en detalle en cada iteración (Joskowicz 2008). Esta fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado.

Durante este proceso se identificaron 14 historias de usuarios, tomando para los puntos estimados la unidad como una semana de trabajo, cada una de ellas respondiendo a las diferentes funcionalidades solicitadas por el cliente y dando una idea al resto del equipo de desarrollo de cómo debe ser su posterior implementación. A continuación se describen algunas de las historias de usuarios identificadas con el cliente, seguidas de un prototipo de interfaz, para consultar las restantes historias de usuario dirigirse los anexos.

Tabla 2. HU7 Gestionar capacidad.

Historia de Usuario	
Código: HU7	Nombre: Gestionar capacidad
Referencia: HU6 y 5	Prioridad: Alta
Iteración Asignada: 1	Puntos Estimados: 1.0
Descripción: La aplicación debe permitir listar, adicionar, eliminar y modificar una capacidad y asignarle preguntas, acciones de mejora y capacidades relacionadas.	
Observaciones: 1. Las capacidades tienen asociadas preguntas, acciones de mejoras y capacidades relacionadas.	

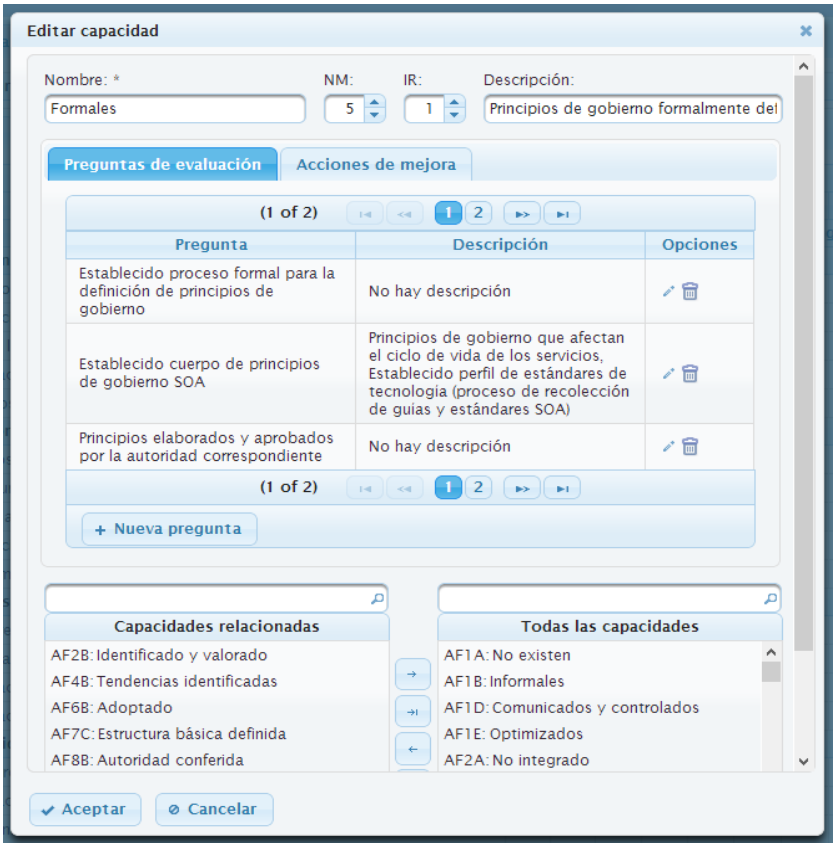


Figura 14. Prototipo de interfaz para la historia de usuario Gestionar capacidad

Tabla 3. HU5 Gestionar pregunta.

	Historia de Usuario
Código: HU5	Nombre: Gestionar pregunta
Referencia:	Prioridad: Media
Iteración Asignada: 1	Puntos Estimados: 0.5
Descripción: La aplicación debe permitir listar, adicionar, eliminar y modificar una pregunta.	
Observaciones: 1. Cada pregunta está asociada a una capacidad.	

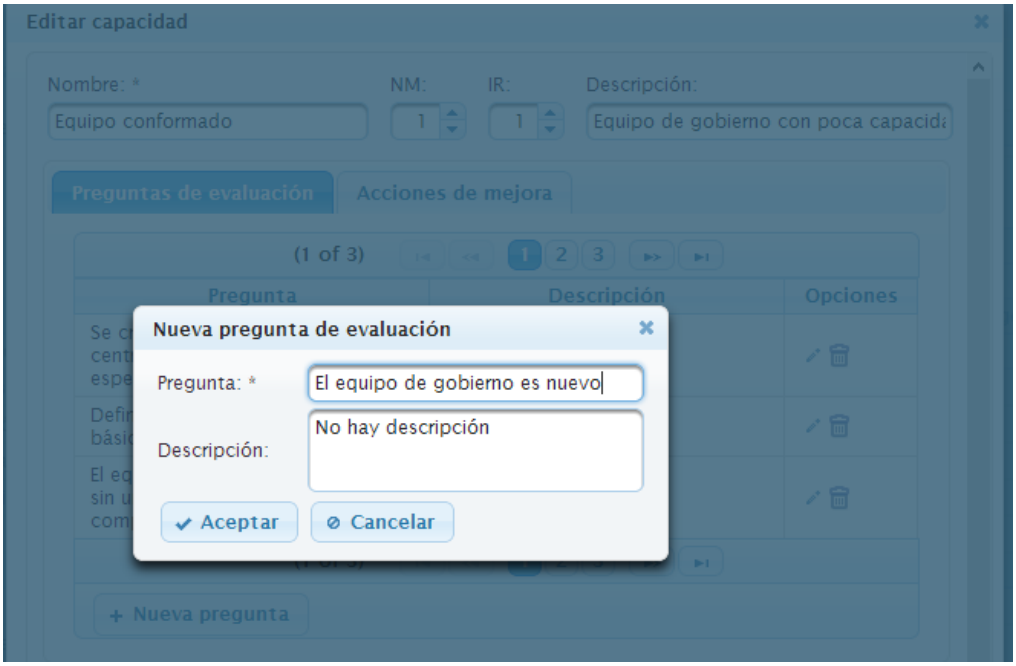


Figura 15. Prototipo de interfaz para la historia de usuario Gestionar pregunta.

2.4. Velocidad del proyecto

Es una medida de la capacidad que tiene el equipo de desarrollo para evacuar las HU en una determinada iteración. Esta medida se calcula totalizando el número de HU realizadas en una iteración. Para la iteración siguiente se podrá (teóricamente) implementar el mismo número de HU que en la iteración anterior (Joskowicz 2008).

Para el desarrollo del sistema se realizó una estimación del esfuerzo para cada una de las HU identificadas, llegándose a los resultados que se muestran en *Tabla 4*, existen HU independientes, las cuales se realizaron en un mismo período de tiempo ya que no existen dependencias entre ellas.

Tabla 4. Estimación de esfuerzo por HU

Historia de usuario	Puntos de estimación (semanas)
Gestionar dominio	1.0
Gestionar grupo de área focal	1.0
Autenticar usuario	0.5
Gestionar área focal	1.0
Gestionar pregunta	0.5
Gestionar acción de mejora	
Gestionar capacidad	1.0

Gestionar organizaciones	1.0
Gestionar evaluaciones	2.0
Realizar evaluación	2.5
Consultar evaluación	1.5
Consultar planificación	1.5
Reporte planificación	2.0
Consultar beneficios	1.0

El desarrollo estimado contó con 16 semanas y media, los valores estuvieron comprendidos 0.5 y 2.5 semanas en dependencia de la complejidad de cada HU.

2.5. Plan de iteraciones

Después de identificar y elaborar las HU y la estimación del esfuerzo necesario para realizarlas, se debe conformar el plan de iteraciones. Las HU seleccionadas para cada iteración son desarrolladas y probadas de acuerdo al orden preestablecido.

Cada HU se traduce en tareas específicas de programación. Asimismo, para cada HU se establecen las pruebas de aceptación. Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir (Joskowicz 2008).

La tarea de escoger las historias fue realizada por el grupo en conjunto, incluyendo al cliente. La clasificación de las historias no fue realizada estrictamente por su grado de importancia en el proyecto, se priorizó la dependencia entre los componentes a desarrollar. El proyecto fue dividido en tres iteraciones las cuales a continuación se describen cada una y la justificación de su selección:

- **Iteración 1:** Para esta iteración se desarrollarán las HU de la 1 a la 7 correspondientes a gestionar dominio, gestionar grupo de área focal, autenticar usuario, gestionar área focal, gestionar pregunta, gestionar acción de mejora y gestionar capacidad. Las HU seleccionadas para esta iteración permiten la gestión de los elementos básicos del sistema; los cuales no tienen dependencias durante su desarrollo y son de gran importancia para las futuras iteraciones. En esta iteración se elaboran todos los elementos necesarios para la creación y edición de la matriz de madurez del modelo. Se realiza una entrega funcional al finalizar la iteración en la cual se puede realizar la gestión de los elementos principales del modelo de madurez.

- **Iteración 2:** En esta iteración se desarrollan las HU de la 8 a la 11 correspondientes a gestionar organizaciones, gestionar evaluaciones, realizar evaluación y consultar evaluación. Cada una de las historias que se realizan, corresponden a la gestión de organización y evaluaciones a realizar en estas. Para su implementación se apoyan en las historias implementadas en la iteración anterior. En la iteración además de las pruebas de aceptación y unitarias, se realizan pruebas de integración para verificar que no han ocurrido afectaciones en las historias anteriores. Al finalizar la iteración se realiza una entrega funcional, la cual permite ya realizar una evaluación.
- **Iteración 3:** Para la iteración se desarrollan las HU de la 12 a la 14 pertenecientes a consultar planificación, reporte planificación y consultar beneficios. Cada una de las historias que se realizan, corresponden a la planificación que se realiza para la evaluación y los beneficios que genera esta. En la iteración, como en la anterior, además de las pruebas de aceptación y unitarias se realizan pruebas de integración para verificar que no han ocurrido afectaciones en las historias anteriormente implementadas. Al finalizar la iteración se realiza la entrega final del proyecto.

Para aproximar el tiempo de ejecución de cada iteración, se tomó como medida que cada semana constaba de 5 días (lunes, martes, miércoles, jueves y viernes). En la *Tabla 5* se muestra el plan de iteraciones, este incorpora a la iteración, las HU que se van a desarrollar y los puntos estimados para cada una de ellas.

Tabla 5. Plan de iteraciones.

Iteración	Historia de usuario	Puntos de estimación (semanas)
1	Gestionar dominio	5
	Gestionar grupo de área focal	
	Autenticar usuario	
	Gestionar área focal	
	Gestionar pregunta	
	Gestionar acción de mejora	
	Gestionar capacidad	
2	Gestionar organizaciones	7
	Gestionar evaluaciones	
	Realizar evaluación	
	Consultar evaluación	
3	Consultar planificación	4.5
	Reporte planificación	

	Consultar beneficios	

2.6. Plan de entrega

A partir del plan de iteraciones analizado en el acápite anterior, y en correspondencia con el mismo se realiza el plan de entregas, que se muestra en el Anexo 15. Plan de entrega. En el cual como resultado del mismo se harán versiones de la aplicación al finalizar cada iteración en la fecha aproximada que se indica.

2.7. Diseño del sistema

La metodología XP propone implementar el diseño más simple posible que funcione. Se sugiere nunca adelantar la implementación de funcionalidades que no correspondan a la iteración en la que se esté trabajando (Escribano 2002).

2.7.1. Tarjetas Clases-Responsabilidades-Colaboración

Las tarjetas CRC(Clases-Responsabilidades-Colaboración) son una técnica de modelado orientado a objetos que permite identificar las clases y sus responsabilidades, además de servir de herramienta de ayuda al refinamiento de clases (Escribano 2002). Los desarrolladores escogen tarjetas a medida que cada clase colabora con la HU. Conforme se van formando ideas sobre las responsabilidades, se pueden escribir en las tarjetas y se utilizan normalmente cuando primero se determinan las clases que se necesitan y cómo van a interactuar y después se implementa la solución (Kent 1999).

A continuación se muestra algunas de las tarjetas CRC que fueron confeccionadas:

Tabla 6. Tarjeta CRC de la clase GeneralService.

Tarjeta CRC	
Clase: GeneralService	
Responsabilidades	Colaboración
• Gestionar acciones de mejora • Gestionar áreas focales • Gestionar beneficios • Gestionar capacidades • Gestionar dominios • Gestionar evaluaciones • Gestionar preguntas de evaluación	• AcciónMejora • ÁreaFocal • Beneficio • Capacidad • Dominio • Evaluación • PreguntaEvaluación

• Gestionar grupos de áreas focales	• GÁreaFocal

Tabla 7. Tarjeta CRC de la clase DominioController.

Tarjeta CRC	
Clase: DominioController	
Responsabilidades	Colaboración
• Insertar dominio • Modificar dominio • Eliminar dominio	• GeneralService • DominioDAO

Tabla 8. Tarjeta CRC de la clase ÁreaFocalController.

Tarjeta CRC	
Clase: ÁreaFocalController	
Responsabilidades	Colaboración
• Insertar área focal • Modificar área foca • Eliminar área focal • Consultar áreas focales por dominio	• GeneralService • ÁreaFocalDAO • DominioDAO

2.7.2. Patrón arquitectónico

Los patrones de arquitectura son aquellos que expresan un esquema organizativo estructural fundamental para sistemas de software (Oliver 2014). El Modelo Vista Controlador es un patrón de diseño de arquitectura que está asociado a la idea de tres capas, se usa principalmente en aplicaciones que procesan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de conceptos para que el desarrollo este estructurado de manera eficiente, facilitando la programación en diferentes capas y de manera paralela e independiente (Ladd y Donald 2006).

La tecnología Java Server Faces es un marco de trabajo de interfaces de usuario del lado de servidor para aplicaciones Web basadas en tecnología Java. Es útil con aplicaciones basadas en la arquitectura MVC (Oliver 2014).

La clase de la capa **Vista (View)**, maneja eventos, captura y visualiza los datos de interés para el usuario, lo cual se realiza con páginas JSP para generar las vistas, añadiendo bibliotecas de

etiquetas propia de JSF para crear los elementos de los formularios HTML. Estos datos son manipulados, validados y controlados por la lógica del negocio, es decir, por las clases agrupadas en la capa **Controlador (Controller)** que son los llamados “managed beans”, estos se pueden ver en las clases DominioController, CapacidadController, AFController, GAFController, en general todas las clases que están dentro del paquete “controllers” de la aplicación que son las encargadas de manipular y validar todos los formularios. Después se toma la información necesaria de las clases persistentes y de las de acceso a datos, agrupada en la capa **Modelo (Model)** que son las clases existentes en el paquete de acceso a datos “dao” y en el paquete del modelo “model”**.**

2.7.3. Patrones de diseño

Para que la solución propuesta sea reusable, fácil de mantener, extensibles y además de reducir el tiempo de desarrollo, utilizando soluciones probadas se utilizaron diferentes patrones de diseño. Siguiendo el patrón arquitectónico MVC utilizado por JSF, a continuación se explica brevemente cuáes fueron utilizados y en qué parte de la implementación del sistema.

Patrones GRAPS

Alta cohesión: El diseño de clases se realizó de forma tal que cada una de ellas tengan responsabilidades moderadas en un área funcional y colaboran con las otras para llevar a cabo las tareas. Cada una de las clases posee solamente las responsabilidades imprescindibles. Un ejemplo donde se evidencia este patrón en la aplicación es mediante la clase “Planificartion”, que es la única encargada de generar las funciones necesarias para generar el reporte en PDF.

Bajo acoplamiento: El acoplamiento es definido por la cantidad de clases relacionadas a una clase, de manera que acoplamiento bajo significa que una clase no depende de muchas clases. Las clases controladoras en la aplicación no dependen de otras clases para realizar sus funcionalidades. Los modelos solo dependen de la clase controladora que lo usa, así se pueden realizar cambios en cada clase de forma independiente.

Creador: Se aplica para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto sólo pueda ser creada por el objeto que contiene la información necesaria para ello. De esta forma una clase controladora puede instanciar objetos, pero es el modelo quien sabe cómo crearlos.

Experto: Con la inclusión de Hibernate para mapear la base de datos y mediante esta realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases del modelo datos poseen un

grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan.

Controlador: Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. Según la utilización del patrón arquitectónico MVC, se cumple ya que las vistas y el modelo son separadas por un controlador el cual se responsabiliza y separa lógicamente la implementación del sistema.

Patrones GOF

Decorador (Decorator): Proporciona una interfaz unificada y de alto nivel para un conjunto de interfaces de un subsistema lográndose que sea más fácil de usar. El uso de este patrón se pone de manifiesto en el archivo "/edit/templates/edit.xhtml" la cual es una plantilla global almacena el código XHTML que unifica a todas las demás interfaces visuales utilizadas en el módulo de edición de la matriz de madurez.

Estrategia (Strategy): Perteneciente a los patrones de comportamiento, teniendo como propósito reducir el acoplamiento entre los objetos utilizando la herencia y el encapsulamiento. Permite disponer de varios métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución. Para hacer uso de este patrón es necesario utilizar la herencia. En las entidades existen hijos que poseen diferentes validaciones de los atributos y mediante este patrón puede ser solucionado.

2.8. Seguridad del sistema

Las aplicaciones web deben implementar mecanismos de seguridad para protegerse de un gran número de vulnerabilidades y amenazas, que pueden ocasionar daños en la integridad y confidencialidad de la información así como en la disponibilidad de los servicios brindados al público. Los controles que se aplican para lograr proteger la información nunca son suficientes pero si ayudan a mitigar las vulnerabilidades en la seguridad del sistema, y los niveles de protección están en dependencia de la privacidad de los datos que se manejen en el negocio.

La aplicación cuenta con un mecanismo de autenticación basada en un formulario Web, este mecanismo recoge las credenciales de un usuario y las confronta contra la base de datos del sistema. Se define además un control para autorizar a los usuarios con diferentes privilegios de acceso a la información, el cual se basa en roles definidos en el sistema. También presenta un gran número de aplicaciones interactivas por lo tanto su correcta validación es un tema que aporta

seguridad a la integridad del sistema, ya que una de las principales amenazas que enfrentan en la actualidad las aplicaciones Web es la explotación de vulnerabilidades.

Consideraciones parciales del capítulo

Luego del desarrollo del presente capítulo se puede llegar a las siguientes conclusiones:

- Se realiza una breve descripción de la herramienta propuesta y se definen 14 HU que debe cumplir dicha aplicación.
- Fueron generados los artefactos que concibe la metodología XP tales como la estimación de esfuerzo por HU y a partir de allí se define el plan de iteraciones, con tres iteraciones y el plan de entregas con tres entregas funcionales y tarjetas CRC.
- En la fase de iteraciones se analizó el diseño de la aplicación, la aplicación del patrón de arquitectura MVC, la utilización de patrones de diseño.

CAPÍTULO 3. IMPLEMENTACIÓN, PRUEBAS Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

Introducción

En el presente capítulo se abordan contenidos relacionados con la descripción de la implementación del sistema, para darle solución a las historias de usuario definidas en el capítulo anterior, se realizan las tareas de ingeniería y se especifican las pruebas a las que fue sometida la aplicación. Por último se describe el proceso de validación para comprobar el cumplimiento de las historias de usuario definidas por el cliente.

3.1. Implementación de la aplicación

La implementación tiene como objetivo principal desarrollar la arquitectura y el sistema como un todo, así como definir la organización del código. Para llevarla a cabo se desglosan en tareas de ingeniería las HU definidas en el capítulo anterior, las cuales guían la implementación, siendo así más fácil el desarrollo del sistema logrando una programación eficiente.

3.1.1. Tareas de ingeniería

Asociado a cada iteración se encuentra la planificación de las tareas de ingeniería o programación, cada HU se transforma en estas tareas que serán desarrolladas. Para cada iteración se realizó la distribución de tareas en correspondencia con las HU que se desarrollaron.

Tabla 9. Tareas de ingeniería de la iteración 2.

Historias de usuario	Tareas por HU
Gestionar organizaciones	• Desarrollo de la entidad que permita la adición, edición de una organización. • Desarrollo de una interfaz para la gestión de organizaciones. • Desarrollo del controlador que permita acceder a los datos de las organizaciones. • Desarrollo de las pruebas de aceptación.
Gestionar evaluaciones	• Desarrollo de la entidad que permita la adición, edición de una evaluación.

	• Desarrollo de una interfaz para la gestión de las evaluaciones. • Desarrollo del controlador que permita acceder a los datos de las evaluaciones. • Desarrollo de las pruebas de aceptación.
Realizar evaluación	• Desarrollo de una interfaz para llevar a cabo una evaluación con sus preguntas correspondientes. • Desarrollo de las pruebas de aceptación.
Consultar evaluación	• Desarrollo de una interfaz para consultar el estado de una evaluación. • Desarrollo de las pruebas de aceptación.

En la *Tabla 9* se muestran la relación de tareas correspondientes a la iteración 2. Cada tarea de desarrollo corresponde a un periodo de uno a tres días de desarrollo.

De la *Tabla 10* a la *Tabla 13* se describen las tareas de ingeniería de la HU9 que pertenece al proceso de desarrollo de una evaluación asociada a una organización.

Tabla 10. Tarea de ingeniería 26 de la iteración 2: Desarrollo de la entidad que permita la adición, edición de una evaluación.

Tarea de Ingeniería	
No. De la tarea: 26	No. De HU: 9
Nombre de la tarea: Desarrollo de la entidad que permita la adición, edición de una evaluación.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2
Fecha inicio: 9/2/2014	**Fecha fin:** 10/2/2014

Descripción: Se realiza la implementación de los métodos para adicionar, modificar y eliminar una evaluación. Así como los métodos para el acceso a la base de datos y el trabajo con los datos.

Tabla 11. Tarea de ingeniería 27 de la iteración 2: Desarrollo de una interfaz para la gestión de las evaluaciones.

Tarea de Ingeniería	
No. De la tarea: 27	No. De HU: 9
Nombre de la tarea: Desarrollo de una interfaz para la gestión de las evaluaciones.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.25
Fecha inicio: 10/2/2014	**Fecha fin:** 12/2/2014
Descripción: Se realiza el diseño e implementación de la vistas para adicionar, editar y mostrar los datos de una evaluación en el sistema. Se crea el diseño realizando una correspondencia entre los tipos de datos de entradas, los datos utilizados en la base de datos y los componentes visuales que más se adapten a estos.	

Tabla 12. Tarea de ingeniería 28 de la iteración 2: Desarrollo del controlador que permita acceder a los datos de las evaluaciones.

Tarea de Ingeniería	
No. De la tarea: 28	No. De HU: 9
Nombre de la tarea: Desarrollo del controlador que permita acceder a los datos de las evaluaciones.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.25
Fecha inicio: 13/2/2014	**Fecha fin:** 17/2/2014
Descripción: Se realiza la implementación de los métodos para validar los datos, así como el acceso a las clases entidades y a los métodos que permiten adicionar y modificar una evaluación. Además los métodos para obtener las evaluaciones por organizaciones.	

Tabla 13. Tarea de ingeniería 29 de la iteración 2: Desarrollo de las pruebas de aceptación.

Tarea de Ingeniería	
No. De la tarea: 29	No. De HU: 9
Nombre de la tarea: Desarrollo de las pruebas de aceptación.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.20
Fecha inicio: 18/2/2014	**Fecha fin:** 19/2/2014
Descripción: Se realiza las pruebas de aceptación a partir de los casos de pruebas propuesto por el usuario.	

En la primera iteración se realizaron un total de 26, en la segunda 12 y en la última 9. En el

*Anexo **16*** se puede observar la planificación final de las tareas por iteración y en correspondencia con las HU.

3.1.2. Estándares de codificación

Los estándares de codificación son aquellos que permiten entender de manera rápida, fácil y sencilla el código empleado en el desarrollo de un software. Además, garantizan un mantenimiento óptimo de dicho código por parte del programador y mejoran la legibilidad permitiendo que otras personas puedan colaborar, ya que la redacción del código no les resulta incómoda. A continuación se muestra el estándar de codificación que se llevó a cabo en la implementación del sistema propuesto.

Comentarios: A cada clase, método y variables se dejan comentadas para su mejor entendimiento. Generalmente se usan comentarios de una sola línea y se reservan los comentarios de bloques para la documentación formal o para comentar porciones de código.

```
377     //obtener capacidad por id
378     private Capacidad getCapById(String id) {
```

Figura 16. Ejemplo de estándar de comentarios.

Declaración de variables: Cada variable se declara en una línea y comentada. También deben aparecer ordenadas alfabéticamente y el nombre de las variables debe de comenzar con letras minúsculas y cada palabra relevante por la que esté compuesta debe ser con letra mayúscula.

```
private String nombre; //nombre del dominio
private String perspectiva; //perspectiva del dominio
```

Figura 17. Ejemplo de declaración de variables en la aplicación.

Declaración de funciones: No debe haber espacio entre el nombre de la función y el paréntesis izquierdo, ni entre este y la lista de parámetros. Debe haber un espacio entre el paréntesis derecho y la llave de comienzo del cuerpo de la función. Las sentencias del cuerpo deben estar en la línea siguiente. La llave que cierra debe estar alineada con la línea de declaración de la función. Los nombres de las funciones se rigen por las mismas características que el de las variables.

```
40      //guardar dominio en la base de datos
41      public void saveDom(ActionEvent actionEvent) {
```

Figura 18. Ejemplo de declaración de funciones.

Nombre de las clases: Las clases comenzarán con mayúscula, si la clase es compuesta empezarán con mayúsculas las dos letras iniciales de cada palabra y estarán separadas por un guion bajo.

```
public class Dominio {
```

Figura 19. Ejemplo de declaración de clases.

3.2. Validación de la solución

La validación es un proceso del desarrollo de software que permite asegurar que el software cumple las expectativas del cliente. Es importante llevar a cabo la validación de los requerimientos del sistema de forma inicial pues es fácil cometer errores y omisiones durante la fase de análisis de requerimientos del sistema y, en tales casos, el software final no cumpliría las expectativas de los clientes. A continuación se muestra cómo fue validada la herramienta desarrollada teniendo en cuenta las pruebas que realiza la metodología de desarrollo utilizada, en este caso XP.

3.2.1 Pruebas de Software

En la metodología XP se prueba el sistema a implementar continuamente como sea posible, reduciendo así el número de los errores, lo que permite aumentar la calidad de los sistemas. Esta metodología divide las pruebas del sistema en dos grupos las unitarias y las de aceptación. Las primeras son desarrolladas por los programadores para verificar que el código funciona correctamente, se diseñan antes de la implantación y se realizan constantemente durante el desarrollo de una HU, mientras que las segundas son utilizadas para evaluar si al final de la iteración se obtuvo las funcionalidades deseada por parte del cliente (Joskowicz 2008).

Pruebas unitarias

Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Que todo código liberado pase

correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código (Joskowicz 2008).

Las pruebas unitarias se realizaron a todo lo largo del desarrollo de cada una de las tareas de programación ejecutadas. Ellas se encargan de mostrarle al desarrollador si la tarea está cumplida. Permite el análisis del código, así como la toma de decisiones para realizar mejoras y la optimizar del mismo. Deben ser automatizable, completas, repetibles o reutilizables, independientes y profesionales

Cuando se encuentra un error (“bug”), debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto (Joskowicz 2008).

Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información (Joskowicz 2008).

Para la realización de cada una de las pruebas de aceptación se siguieron una serie de pasos que se muestran a continuación:

- Identificar todas las acciones en la HU.
- Para cada acción escribir al menos una prueba.
- Para algunos datos, reemplazar las entradas que hacen que la acción ocurra y llenar en la casilla resultados esperados los resultados obtenidos.
- Para otros datos, reemplazar las entradas que hacen que la acción falle, y registrar los resultados.

Se desarrollaron 33 casos de pruebas, un caso de prueba por cada tarea de investigación funcional. A continuación se muestran las pruebas de aceptación relacionadas a la HU 9 perteneciente a la segunda iteración del sistema. Los casos de pruebas se corresponden con el proceso de desarrollo de una evaluación asociada al Centro de Inmunología Molecular (CIM).

Tabla 14. HU9_P1: Listar evaluaciones.

Código: HU9_P1	**HU:** 9
Nombre: Listar evaluaciones	
Descripción: Vista de una tabla con las evaluaciones existentes.	
Condiciones de Ejecución:	
Entrada/Pasos de Ejecución: Lista las evaluaciones existentes en el sistema y la opción de agregar una nueva, debe existir las opciones de: editar, y de evaluar.	
Resultado Esperado: La aplicación muestra un listado con cada una de las evaluaciones en el sistema.	
Resultado Obtenido: 	
Evaluación de la Prueba: Satisfactoria.	

Tabla 15. HU9_P2: Crear nueva evaluación.

Código: HU9_P2	**HU:** 9
Nombre: Crear nueva evaluación.	
Descripción: Vista de una nueva evaluación.	
Condiciones de Ejecución:	

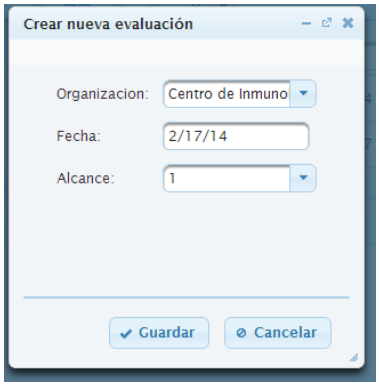
• Existan organizaciones para asociar la evaluación.
Entrada/Pasos de Ejecución: • Organización: CIM. • Fecha: 17 de febrero del 2014. • Alcance: 1.
Resultado Esperado: Si los datos de entrada son correctos la aplicación debe realizar la acción de crear una evaluación y mostrar los datos que conforman la nueva evaluación.
Resultado Obtenido: 
Evaluación de la Prueba: Satisfactoria.

Tabla 16. HU9_P3: Editar evaluación.

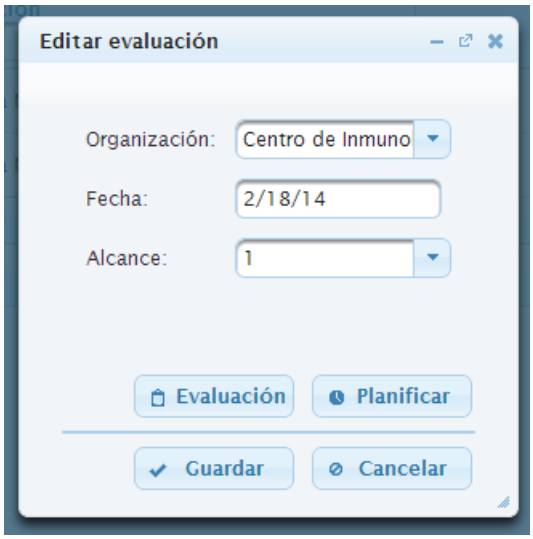
Código: HU9_P3	**HU:** 9
Nombre: Editar evaluación.	
Descripción: Vista editar de curso.	
Condiciones de Ejecución: • Existan organizaciones para asociar la evaluación. • Exista evaluación en el sistema.	
Entrada/Pasos de Ejecución:	

- Fecha: 18 de febrero del 2014.

Resultado Esperado:

Si los datos de entrada son correctos la aplicación debe realizar la acción de actualizar los datos de la evaluación y mostrar los datos que la conforman.

Resultado Obtenido:



Evaluación de la Prueba: Satisfactoria.

Se realizaron 3 iteraciones de pruebas donde en la primera se encontraron en total 5 no conformidades: 2 de validación, 2 de ortografía y 1 de interfaz. En la 2da iteración fueron encontradas 4 no conformidades: 2 de validación, 1 de ortografía y 1 de interfaz. En la 3ra iteración se obtuvieron resultados satisfactorios, encontrándose 0 no conformidades como se puede apreciar en la *Figura 20*.

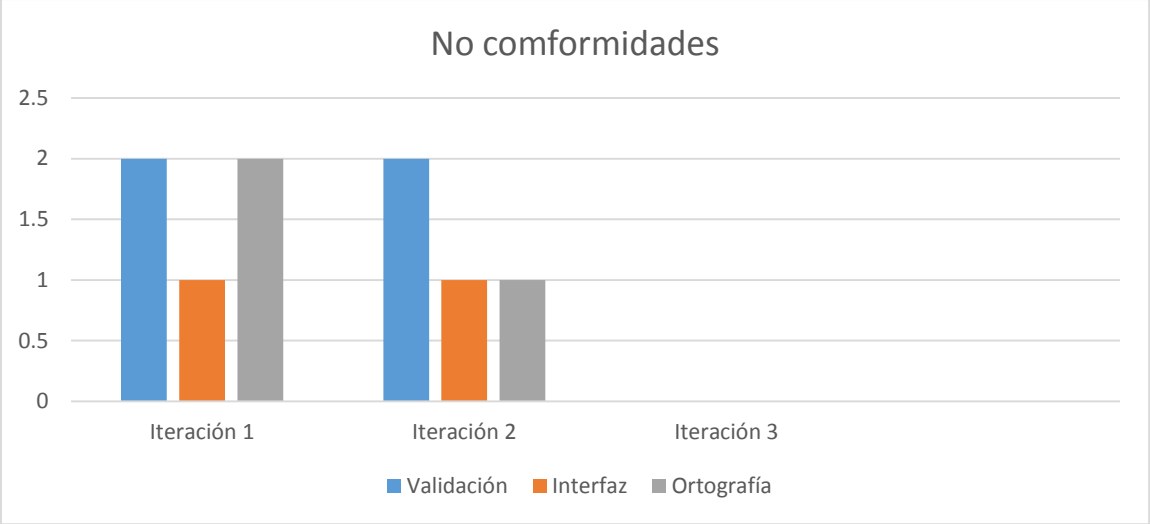


Figura 20. Representación de las no conformidades encontradas en cada una de las iteraciones de prueba.

Consideraciones parciales del capítulo.

Luego del desarrollo del presente capítulo se puede arribar a las conclusiones siguientes:

- La realización de las pruebas de caja negra permitió detectar, documentar y corregir las no conformidades existentes en el sistema implementado.
- Al concluir el período de pruebas se obtuvo una aplicación que cumple de forma correcta con la totalidad de las funcionalidades esperadas por el cliente.

Conclusiones generales

- El diagnóstico inicial apoyado en la aplicación de los métodos científicos permitió identificar las funcionalidades y las tecnologías libres a utilizar en el desarrollo del sistema de gestión para el modelo de madurez MMSOA3P.
- La implementación de la herramienta para la gestión del modelo de madurez SOAMM3P propuesto por el CDAE, facilita el proceso de administración y aplicación del modelo de madurez MMSOA3P.
- Las pruebas de aceptación aplicadas para la validación de la herramienta demostraron que el sistema cumple con los requisitos que garantizan su correcto funcionamiento.

RECOMENDACIONES

- Aumentar la variedad de reportes dinámicos que se generan.
- Aplicar el sistema en futuros proyectos de adopción de SOA.
- Se recomienda que la aplicación sea integrada al portal del intranet del CDAE.

BIBLIOGRAFÍA

1. ALEGSA 2014. Definicion de Framework - ¿qué es Framework? [en línea]. [Consulta: 9 abril 2014]. Disponible en: http://www.alegsa.com.ar/Dic/framework.php.

2. AMAZING 2014. Arquitectura Empresarial. [en línea]. [Consulta: 9 abril 2014]. Disponible en: http://www.amazing.com.co/arquitectura-empresarial.php.

3. APACHE 2014. Apache Tomcat 6.0 (6.0.39) - Documentation Index. [en línea]. [Consulta: 9 abril 2014]. Disponible en: http://tomcat.apache.org/tomcat-6.0-doc/index.html.

4. ARIAS-ORIZONDO, A. 2013a. *Modelo de madurez de tres perspectivas para evaluar y planificar la adopción de Arquitecturas Orientadas a Servicios en las organizaciones.* S.l.: Universidad de Ciencias Informáticas.

5. ARIAS-ORIZONDO, A.C. y ESTRADA-SENTI, V. 2013a. BASES PARA CREAR UN MODELO DE MADUREZ PARA ARQUITECTURAS ORIENTADAS A SERVICIOS. pp. 307-318.

6. BECK, K. 1999a. Embracing change with extreme programming. pp. 70-77.

7. BECK, K. 1999b. Embracing change with extreme programming. *Computer*. Vol. 32, no. 10, pp. 70-77. ISSN 0018-9162. DOI 10.1109/2.796139.

8. CAMILO BUSTAMANTE, A. 2014. Dia a dia: Modelos de Madurez para SOA (Service Oriented Architecture). [en línea]. [Consulta: 9 abril 2014]. Disponible en: http://dia10.blogspot.com/2005/12/modelos-de-madurez-para-soa-service.html.

9. CHACÓN CASAS, A.A. 2012. Desarrollo de un software que implemente en modelo de madurez para SOA. [en línea]. [Consulta: 20 marzo 2014]. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_05637_12.

10. CLUB-BPM 2014. Arquitectura Empresarial - Concepto Clave BPM - Autor Club-BPM. [en línea]. [Consulta: 9 abril 2014]. Disponible en: http://www.club-bpm.com/ConceptoClaveArquitecturaEmpresarial.htm.

11. CODECRITICON 2014. Frameworks de persistencia.Diferencias » Codecriticon. [en línea]. [Consulta: 9 abril 2014]. Disponible en: http://codecriticon.com/diferencias-frameworks-persistencia-i/.

12. CONSULTORÍA JAVA 2014. SPRING - Framework de aplicaciones Java. [en línea]. [Consulta: 10 abril 2014]. Disponible en: http://www.consultoriajava.com/tools/spring.shtml.

13. EDUCACIÓN IT 2014. ¿Qué es Java Hibernate? | Blog de EducacionIT - Cursos y Formacion Profesional IT. [en línea]. [Consulta: 9 abril 2014]. Disponible en: http://blog.educacionit.com/2013/02/07/que-es-java-hibernate/.

14. ENCUESTAFACIL 2014. encuestas online - software encuestas - crea y envia una encuesta facilmente - sondeos web - Prueba Gratis. [en línea]. [Consulta: 9 abril 2014].

Disponible en: http://www.encuestafacil.com/Mas_Informacion/Precios_Software_Encuestas.aspx.

15. ENCUESTATICK 2014. Encuestas en linea - Encuestas satisfaccion del cliente - EncuestaTick. [en línea]. [Consulta: 9 abril 2014]. Disponible en: http://www.eprowin.co/encuestas-en-linea.html.

16. ENCUESTATICKTARIFAS 2014. Versiones y tarifas. [en línea]. [Consulta: 9 abril 2014]. Disponible en: http://www.portaldeencuestas.com/versiones-y-tarifas.php.

17. ESCRIBANO, G.F. 2002. Introducción a Extreme Programming. [en línea]. [Consulta: 14 febrero 2014]. Disponible en: http://proyectoredox.googlecode.com/svn/bibliografia%20Informe/Introduccion%20a%20Extreme%20Programming.pdf.

18. HHOSHAFIAN, S. 2007. *Service Oriented Enterprises*. S.l.: s.n.

19. JOSKOWICZ, J. 2008a. Reglas y prácticas en Extreme Programming. [en línea]. [Consulta: 14 febrero 2014]. Disponible en: http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf.

20. JOSKOWICZ, J. 2008b. Reglas y prácticas en Extreme Programming. *Universidad de Vigo. España* [en línea]. [Consulta: 14 febrero 2014]. Disponible en: http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf.

21. KENT, B. 1999. *Extreme_Programming_Explained* [en línea]. S.l.: s.n. [Consulta: 2 febrero 2014]. Disponible en: http://software2012team23.googlecode.com/git-history/5127389d21813c2bd955c53999f66cede994578b/docs/literature/Extreme_Programming_Explained_Kent_Beck_1999.pdf.

22. KIOSKEA 2014. Patrones de diseño. [en línea]. [Consulta: 10 abril 2014]. Disponible en: http://es.kioskea.net/contents/224-patrones-de-diseno.

23. LADD, S. y DONALD, K. 2006. *Expert Spring MVC and Web Flows*. 2006: s.n.

24. NETSAC 2014. NETSAC - consulting - technology - outsourcing. [en línea]. [Consulta: 19 mayo 2014]. Disponible en: http://www.netsac.com/PaintServlet?node=006008001003003&articleId=1318&treeManagerId=46&treeId=46.

25. NICOLAI M., J. 2007a. *O'reilly SOA In Practice*. S.l.: s.n.

26. OASIS 2006. *Reference Model for Service Oriented Architecture 1.0*. 2006: 2006.

27. OLIVER, J. 2014. JSF y El Uso de Patrones de Diseno. [en línea]. [Consulta: 10 abril 2014]. Disponible en: http://www.scribd.com/doc/95328943/JSF-y-El-Uso-de-Patrones-de-Diseno.

28. OSD 2014. Comparativo software WEB vs. software de Escritorio. [en línea]. [Consulta: 9 abril 2014]. Disponible en: http://www.osdglobal.com/faq/desarrollo-software/comparativo-web-vs-escritorio.

29. POLLDADDY 2014. PollDaddy, encuestas y cuestionarios en línea gratis | internetica. [en línea]. [Consulta: 9 abril 2014]. Disponible en: http://www.internetica.com.mx/2008/10/11/polldaddy-encuestas-y-cuestionarios-en-linea-gratis/.

30. PROYECTOSAGILES 2014. Metodologías de desarrollo de software - EcuRed. [en línea]. [Consulta: 10 abril 2014]. Disponible en: http://www.ecured.cu/index.php/Metodolog%C3%ADas_de_desarrollo_de_software.

31. RIVERO-GUTIERREZ, Y. y BUENO-CARRION, K. 2009. Propuesta de Modelo Madurez para la adopcion de la Arquitectura Orientada a Servicios. [en línea]. [Consulta: 21 febrero 2014]. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_2027_09.

32. ROMERO, H. 2014. Metodologias de desarrollo. [en línea]. [Consulta: 10 abril 2014]. Disponible en: http://www.slideshare.net/MeneRomero/metodologias-de-desarrollo.

33. SAAVEDRA 2014. PATRONES GRASP (Patrones de Software para la asignación General de Responsabilidad).Parte II | El Mundo Informático. [en línea]. [Consulta: 10 abril 2014]. Disponible en: http://jorgesaavedra.wordpress.com/2007/05/08/patrones-grasp-patrones-de-software-para-la-asignacion-general-de-responsabilidadparte-ii/.

34. SCHUH, P. 2001. Recovery, redemption, and extreme programming. *IEEE Software*. Vol. 18, no. 6, pp. 34-41. ISSN 0740-7459. DOI 10.1109/52.965800.

35. SOACTION 2014. SOAction. [en línea]. [Consulta: 9 abril 2014]. Disponible en: http://soaction.sisorg.com.mx/definicion.html.

36. SOA-MANIFIESTO 2014. El Manifiesto SOA Comentado. [en línea]. [Consulta: 4 abril 2014]. Disponible en: http://www.soa-manifesto.com/annotated_spanish.html.

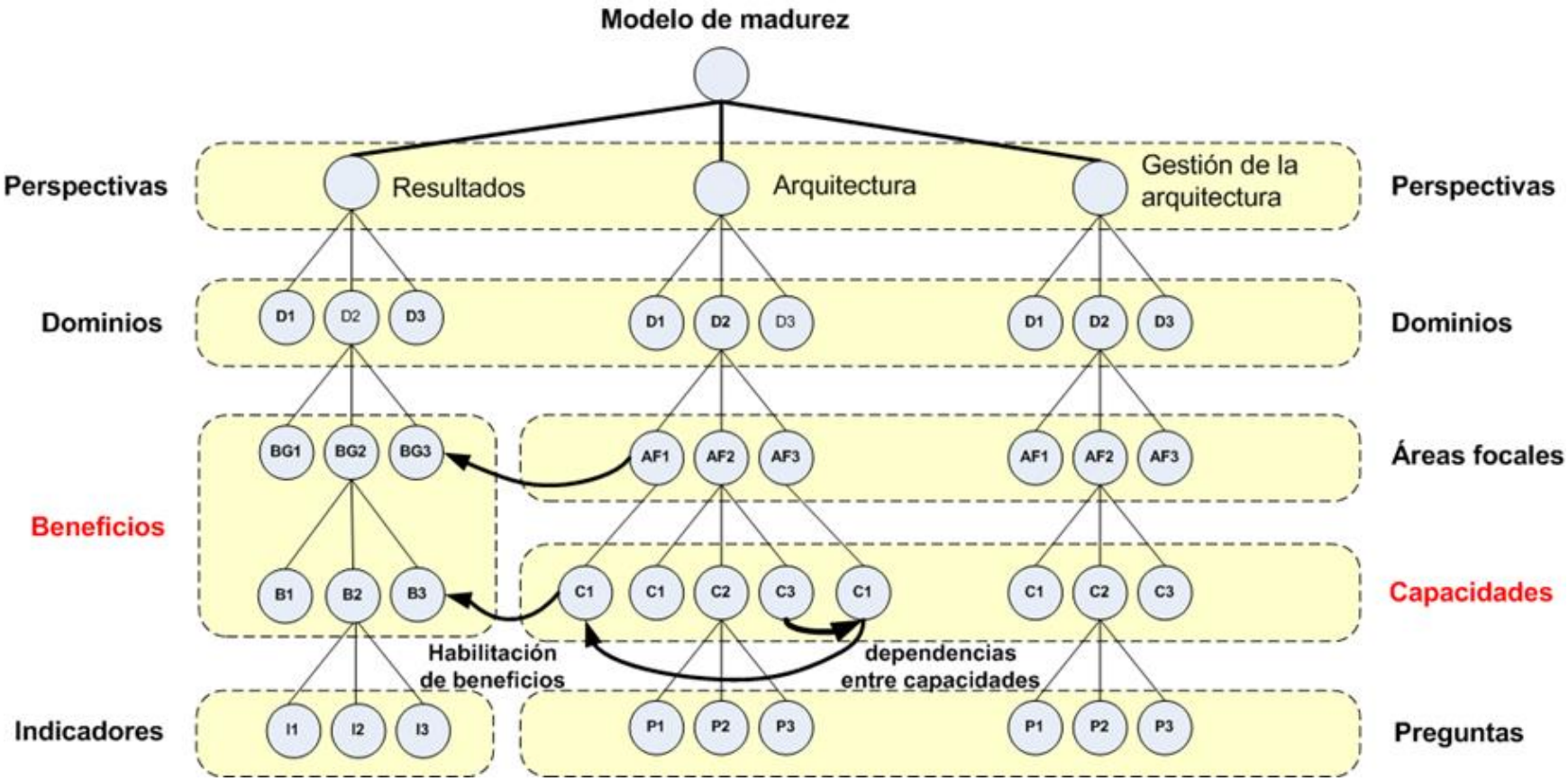
37. SOURCEFORGE 2014. Tecnologías. [en línea]. [Consulta: 10 abril 2014]. Disponible en: http://oness.sourceforge.net/proyecto/html/ch06s03.html.

38. WESKE, M. 2007. *Business Process Management. Concepts, Languages, Architectures*. S.l.: s.n.

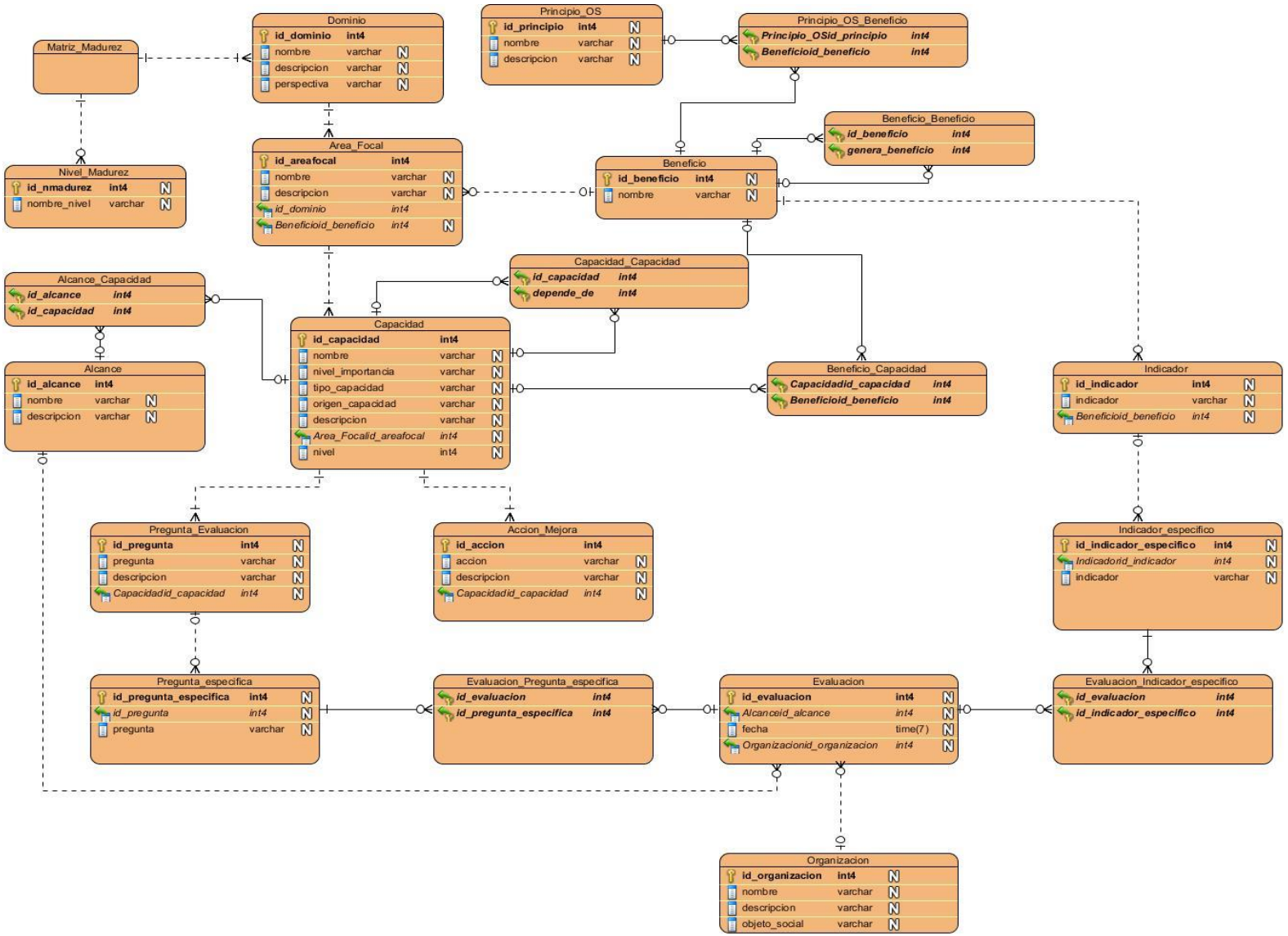
39. ZACHMAN, J.A. 1987. *A framework for information systems architecture*. S.l.: IBM System Journal.

ANEXOS

Anexo 1. Estructura de datos en forma de árbol del modelo de madurez MMSOA3P.



Anexo 2. Estructura de datos del modelo de madurez MMSOA3P.



Anexo 3. HU1 Gestionar dominio.

Historia de Usuario	
Código: HU1	Nombre: Gestionar dominio
Referencia:	Prioridad: Alta
Iteración Asignada: 1	Puntos Estimados: 1.0
Descripción: La aplicación debe permitir listar, adicionar, eliminar y modificar un dominio.	
Observaciones:	

Anexo 4. HU2 Gestionar grupo de área focal.

Historia de Usuario	
Código: HU2	Nombre: Gestionar grupo de área focal
Referencia:HU1	Prioridad: Alta
Iteración Asignada: 1	Puntos Estimados: 1.0
Descripción: La aplicación debe permitir listar, adicionar, eliminar y modificar un grupo de área focal.	
Observaciones: 1. Debe existir un dominio al cual asignarle el grupo de área focal.	

Anexo 5. HU3 Autenticar usuario.

Historia de Usuario	
Código: HU3	Nombre: Autenticar usuario
Referencia:HU1	Prioridad: Alta
Iteración Asignada: 1	Puntos Estimados: 0.5
Descripción: La aplicación debe permitir autenticar un usuario para acceder a la aplicación.	
Observaciones: 1. Solo existe un rol en el sistema que es el administrador, que es un experto para aplicar el modelo de madurez.	

Anexo 6. HU4 Gestionar área focal.

Historia de Usuario	
Código: HU4	Nombre: Gestionar área focal
Referencia:HU2	Prioridad: Alta
Iteración Asignada: 1	Puntos Estimados: 1.0
Descripción: La aplicación debe permitir listar, adicionar, eliminar y modificar un área focal.	
Observaciones: 1. Debe existir un grupo de área focal al cual asignarle un área focal.	

Anexo 7. HU6 Gestionar acción de mejora.

Historia de Usuario	
Código: HU6	Nombre: Gestionar acción de mejora
Referencia:	Prioridad: Media
Iteración Asignada: 1	Puntos Estimados: 0.5
Descripción: La aplicación debe permitir listar, adicionar, eliminar y modificar una acción de mejora.	
Observaciones: 1. Cada acción de mejora está asociada a una capacidad.	

Anexo 8. HU8 Gestionar organizaciones.

Historia de Usuario	
Código: HU8	Nombre: Gestionar organizaciones
Referencia:	Prioridad: Alta
Iteración Asignada: 2	Puntos Estimados: 1
Descripción: La aplicación debe permitir listar, adicionar, eliminar y modificar las organizaciones.	
Observaciones:	

Anexo 9. HU9 Gestionar evaluaciones.

Historia de Usuario	
Código: HU9	Nombre: Gestionar evaluaciones
Referencia: HU8	Prioridad: Alta
Iteración Asignada: 2	Puntos Estimados: 2
Descripción: La aplicación debe permitir listar, adicionar y modificar las evaluaciones.	
Observaciones: 1. Las evaluaciones se realizan a una organización determinada.	

Anexo 10. HU10 Realizar evaluación.

Historia de Usuario	
Código: HU10	Nombre: Realizar evaluación
Referencia: HU9	Prioridad: Alta
Iteración Asignada: 2	Puntos Estimados: 2.5
Descripción: La aplicación debe permitir responder un cuestionario de preguntas relacionado a una evaluación.	
Observaciones: 1. Debe existir la opción de guiarse por una planificación pera responder las preguntas. 2. Las preguntas se responden por capacidades en área focales.	

Anexo 11. HU11 Consultar evaluación.

Historia de Usuario	
Código: HU11	Nombre: Consultar evaluación
Referencia: HU9 y 10	Prioridad: Alta
Iteración Asignada: 2	Puntos Estimados: 1.5
Descripción: La aplicación debe permitir consultar las evaluaciones realizadas con el avance alcanzado hasta el momento.	
Observaciones: 1. Se muestra el nivel de madurez obtenido por área focal en forma de matriz.	

Anexo 12. HU12 Consultar planificación.

Historia de Usuario	
Código: HU12	Nombre: Consultar planificación
Referencia: HU9, 10 y 11	Prioridad: Alta
Iteración Asignada: 3	Puntos Estimados: 1.5
Descripción: La aplicación debe permitir consultar y generar una planificación para realizar las evaluaciones.	
Observaciones: 1. Se muestra un orden de las capacidades por área focal para realizar la evaluación.	

Anexo 13. HU13 Reporte planificación.

Historia de Usuario	
Código: HU13	Nombre: Reporte planificación
Referencia: HU12	Prioridad: Alta
Iteración Asignada: 3	Puntos Estimados: 2
Descripción: La aplicación debe permitir generar un reporte en formato pdf para consultar la planificación.	
Observaciones: 1. Se muestra un documento pdf generado por el sistema con la planificación.	

Anexo 14. HU14 Consultar beneficios.

Historia de Usuario	
Código: HU14	Nombre: Consultar beneficios
Referencia:	Prioridad: Alta
Iteración Asignada: 3	Puntos Estimados: 1
Descripción: La aplicación debe permitir generar una tabla con los beneficios generados por la aplicación del modelo.	
Observaciones: 1. Los beneficios que pueden ser generados ya se encuentran previamente en el sistema.	

Anexo 15. Plan de entrega.

Historia de usuario	1ra Iteración 6 de enero del 2014	2da Iteración 24 de febrero del 2014	3ra Iteración 1 de abril del 2014
Gestionar dominio	V0.1		
Gestionar grupo de área focal			
Autenticar usuario			
Gestionar área focal			
Gestionar pregunta			
Gestionar acción de mejora			
Gestionar capacidad			
Gestionar organizaciones		V0.2	
Gestionar evaluaciones			
Realizar evaluación			
Consultar evaluación			
Consultar planificación			V1
Reporte planificación			
Consultar beneficios			

Anexo 16. Tareas de ingeniería por historia de usuario.

Historias de usuario	Tareas por HU
Gestionar dominio	• Desarrollo de la entidad que permita la adición, edición de un dominio. • Desarrollo de una interfaz para la gestión de los dominios. • Desarrollo del controlador que permita acceder a los datos de los dominios. • Desarrollo de las pruebas de aceptación.
Gestionar grupo de áreas focales	• Desarrollo de la entidad que permita la adición, edición de un grupo de áreas focales. • Desarrollo de una interfaz para la gestión grupos de áreas focales. • Desarrollo del controlador que permita acceder a los datos de los grupos de áreas focales.

	• Desarrollo de las pruebas de aceptación.
Autenticar usuario	• Desarrollo de una interfaz que permita autenticar un usuario. • Desarrollo de las pruebas de aceptación.
Gestionar área focal	• Desarrollo de la entidad que permita la adición, edición de un área focal. • Desarrollo de una interfaz para la gestión de las áreas focales. • Desarrollo del controlador que permita acceder a los datos de las áreas focales. • Desarrollo de las pruebas de aceptación.
Gestionar pregunta	• Desarrollo de la entidad que permita la adición, edición de una pregunta. • Desarrollo de una interfaz para la gestión de las preguntas por capacidades. • Desarrollo del controlador que permita acceder a las preguntas de las capacidades. • Desarrollo de las pruebas de aceptación.
Gestionar acción de mejora	• Desarrollo de la entidad que permita la adición, edición de una acción de mejora. • Desarrollo de una interfaz para la gestión de las acciones de mejora por capacidades. • Desarrollo del controlador que permita acceder a las acciones de mejora de las capacidades. • Desarrollo de las pruebas de aceptación.
Gestionar capacidad	• Desarrollo de la entidad que permita la adición, edición de una capacidad. • Desarrollo de una interfaz para la gestión de las capacidades por área focal. • Desarrollo del controlador que permita acceder a los datos de las capacidades. • Desarrollo de las pruebas de aceptación.
Gestionar organizaciones	• Desarrollo de la entidad que permita la adición, edición de una Organización. • Desarrollo de una interfaz para la gestión de organizaciones. • Desarrollo del controlador que permita acceder a los datos de las organizaciones.

	• Desarrollo de las pruebas de aceptación.
Gestionar evaluaciones	• Desarrollo de la entidad que permita la adición, edición de una evaluación. • Desarrollo de una interfaz para la gestión de las evaluaciones. • Desarrollo del controlador que permita acceder a los datos de las evaluaciones. • Desarrollo de las pruebas de aceptación.
Realizar evaluación	• Desarrollo de una interfaz para llevar a cabo una evaluación con sus preguntas correspondientes. • Desarrollo de las pruebas de aceptación.
Consultar evaluación	• Desarrollo de una interfaz para consultar el estado de una evaluación. • Desarrollo de las pruebas de aceptación.
Consultar planificación	• Desarrollo de una interfaz para visualizar la planificación que sugiere la solución propuesta. • Desarrollo del controlador que permita calcular las prioridades de las capacidades para sugerir una planificación efectiva. • Desarrollo de las pruebas de aceptación.
Reporte planificación	• Desarrollo de una interfaz para visualizar la planificación que sugiere la solución propuesta en forma de reporte. • Desarrollo del controlador que permita generar un reporte en formato pdf a partir de la planificación calculada. • Desarrollo de las pruebas de aceptación.
Consultar beneficios	• Desarrollo de una interfaz para visualizar los beneficios generados por la aplicación del modelo. • Desarrollo del controlador que permita acceder a los datos de los beneficios. • Desarrollo de las pruebas de aceptación.