



TESINA DE LICENCIATURA

Título: Modelo de adopción SOA OSIMM: Una herramienta visual para planificar los objetivos de adopción

Autores: Nicolás González, Joaquín Díaz Vélez

Director: Lic. Patricia Bazán

Codirector:

Asesor profesional: Santiago Urrizola

Carrera: Licenciatura en Informática

Resumen

Tradicionalmente la precedencia de objetivos y su división es un proceso realizado manualmente. Toda la información y experiencia que se obtiene en cada una de las adopciones queda en manos del grupo de trabajo llevó a cabo el análisis y tomó las decisiones correspondientes al estado de la organización. Si bien existe documentación que respalda las decisiones tomadas, no es posible reconocer patrones de adopción que se repiten entre organizaciones. También la transferencia de conocimiento entre un proceso de adopción y otro es más difícil.

La motivación de nuestro trabajo es desarrollar una herramienta confiable siguiendo las políticas dictadas por el modelo de adopción OSIMM, que facilite la generación de la matriz de adopción y que permita la utilización de la información generada por el proceso de adopción a futuro.

Palabras Claves

SOA, Modelos de adopción, madurez, OSIMM, objetivos, nivel de madurez,

Trabajos Realizados

- Herramienta de software desarrollada sobre tecnologías web que permita la creación y gestión de adopciones SOA siguiendo el modelo OSIMM.

Conclusiones

- Una adopción SOA es un proceso transformador de una organización.
- La amplitud de estas transformaciones es necesario generar un plan de trabajo
- El plan de trabajo está asociado a un contexto
- OSIMM fija los límites del contexto y la adopción se vuelve modelable
- Representando la información de la adopción, la herramienta genera nuevos datos que ayudarán a la próxima adopción.

Trabajos Futuros

- Búsqueda de patrones de adopción SOA
- Creación de un modelo para gestionar la madurez tecnológica
- Extensión de la herramienta para llevar adelante la ejecución de los objetivos.

Agradecimientos

A Violeta por la paciencia, a mis padres Roberto y Marisa por ayudarme durante toda la carrera.

A mi hermano Federico por sus aportes

A toda mi familia y amigos por el apoyo.

Nicolás

A mi familia

A mis amigos

A vos

Joaquín

A la Lic. Patricia Bazan por su ayuda y paciencia en este recorrido hacia la finalización de la carrera

Contenido

| | | |
|--------|--|----|
| 1 | Motivación y objetivo | 6 |
| 1.1 | Introducción..... | 6 |
| 1.2 | Contexto histórico | 6 |
| 1.3 | Motivación | 7 |
| 2 | SOA..... | 9 |
| 2.1 | Introducción..... | 9 |
| 2.2 | Evolución de los sistemas..... | 9 |
| 2.2.1 | Introducción | 9 |
| 2.2.2 | Programación orientada a objetos | 15 |
| 2.2.3 | Programación orientada a componentes | 17 |
| 2.2.4 | Sistemas distribuidos | 20 |
| 2.2.5 | Remote Procedure Call (RPC)..... | 22 |
| 2.2.6 | Objetos distribuidos | 23 |
| 2.2.7 | Definición de SOA (Arquitectura orientada a servicios) | 24 |
| 2.2.8 | Actores principales y características de SOA..... | 25 |
| 2.2.9 | Principios de una arquitectura orientada a servicios | 27 |
| 2.2.10 | Desarrollo de aplicaciones orientadas a servicios | 31 |
| 2.2.11 | Arquitectura SOA separada por capas (SOA Layered Architecture) | 31 |
| 2.3 | Conclusión | 32 |
| 3 | SOA Governance..... | 34 |
| 3.1 | ¿Qué se entiende por un Gobierno SOA?..... | 34 |
| 3.2 | Ejecutando un Gobierno SOA | 36 |
| 3.2.1 | Dinámica de un Gobierno de SOA | 36 |
| 3.2.2 | Alcance de un Gobierno SOA..... | 38 |

| | | |
|-------|---|----|
| 3.2.3 | Framework de Gobierno SOA | 39 |
| 3.2.4 | SOA Governance Reference Model (SGRM) | 39 |
| 3.2.5 | SOA Governance Vitality Method (SGVM) | 40 |
| 3.3 | Conclusión | 40 |
| 4 | Esquemas de adopción SOA | 41 |
| 4.1 | OSIMM | 41 |
| 4.1.1 | Contexto y descripción inicial | 41 |
| 4.1.2 | Niveles de madurez | 42 |
| 4.1.3 | Dimensiones | 45 |
| 4.1.4 | Service Foundation Levels | 48 |
| 4.1.5 | Cuestionario del Assessment e indicadores de madurez por dimensión | 49 |
| 4.2 | Método OSIMM de Assessment | 54 |
| 4.2.1 | Descripción | 55 |
| 4.2.2 | Pasos para realizar un assessment | 55 |
| 4.3 | Conclusiones | 56 |
| 5 | Características de un modelo de madurez orientado a sistemas de información | 58 |
| 5.1 | Definición de modelos de madurez | 58 |
| 5.2 | Clasificación de un modelo de madurez | 58 |
| 5.1 | Conclusión | 59 |
| 6 | Modelando la madurez de una organización | 60 |
| 6.1 | Actores involucrados | 60 |
| 6.2 | Descripción del modelo | 60 |
| 6.3 | Forma de trabajo | 62 |
| 6.4 | Conclusión | 63 |
| 7 | Herramienta de Gobierno de Adopción SOA | 64 |
| 7.1 | Descripción | 64 |
| 7.2 | Arquitectura | 65 |

| | | |
|--------|---|----|
| 7.3 | Cuestionario | 67 |
| 7.3.1 | Nombre del assessment | 68 |
| 7.3.2 | Creación desde la base de conocimiento | 68 |
| 7.3.3 | Selección de preguntas de la encuesta | 68 |
| 7.3.4 | Salvado de la encuesta | 68 |
| 7.3.5 | Ayuda | 69 |
| 7.3.6 | Administración de preguntas | 69 |
| 7.4 | Matriz de objetivos | 71 |
| 7.4.1 | Ayuda | 71 |
| 7.4.2 | Administración de objetivos | 71 |
| 7.4.3 | Incorporar objetivos al assessment | 71 |
| 7.4.4 | Relacionando y moviendo objetivos del assessment | 72 |
| 7.4.5 | Mover objetivos | 72 |
| 7.4.6 | Relacionar objetivos | 72 |
| 7.4.7 | Borrar objetivos | 73 |
| 7.4.8 | Marcar objetivos como cumplidos | 73 |
| 7.4.9 | Marcar objetivos como no cumplidos | 73 |
| 7.4.10 | Generar objetivos adyacentes | 73 |
| 7.4.11 | Ver objetivos adyacentes | 74 |
| 7.5 | Base de conocimiento | 75 |
| 7.5.1 | Contexto | 75 |
| 7.5.2 | Generación de la matriz a partir de la base de conocimiento | 75 |
| 7.5.3 | Paso 1, cuantificación: | 76 |
| 7.5.4 | Paso 2, asignación de objetivos: | 76 |
| 7.5.5 | Paso 3, relación de objetivos: | 76 |
| 7.5.6 | Generación de objetivos adyacentes | 77 |
| 7.6 | Aprendizaje a través de los assessments | 78 |

| | | |
|------|---|----|
| 8 | Conclusiones | 79 |
| 9 | Futuros trabajos..... | 80 |
| 9.1 | Búsqueda de patrones de adopción SOA | 80 |
| 9.2 | Hacia un modelo de madurez tecnológico..... | 80 |
| 9.3 | Integración con otras herramientas de gestión | 80 |
| 10 | Referencias..... | 81 |
| 11 | Figuras..... | 82 |
| 12 | Anexo I – Experiencias de adopción SOA personalizando el modelo OSIMM | 84 |
| 12.1 | Acerca de Flux IT..... | 84 |
| 12.2 | Framework de adopción SOA de Flux IT | 84 |
| 12.3 | Modificaciones al modelo OSIMM | 84 |
| 12.4 | Experiencias..... | 85 |
| 12.5 | Aplicación y problemática SOA | 85 |
| 12.6 | Conclusión | 85 |

1 Motivación y objetivo

1.1 Introducción

En este capítulo se presenta una descripción general del trabajo, junto con la motivación, los objetivos planteados y una explicación general de la distribución de los temas estudiados en cada capítulo

1.2 Contexto histórico

Las soluciones de software planteadas en un contexto empresarial han evolucionado notablemente de la mano de la tecnología, tanto en complejidad como en prestaciones. SOA [8] es una estrategia tecnológica para que la funcionalidad expuesta por las aplicaciones sea interoperable y pueda ser combinada en forma de servicios. Con la llegada de SOA, el mundo tecnológico presenta una ampliación del horizonte de lo posible [2]. Esto ha provocado que soluciones centralizadas se dirijan a la obsolescencia debido a las dificultades que presentan a la hora de escalar su funcionamiento. Por lo tanto surge como necesidad imperiosa la utilización de soluciones distribuidas de software dentro de una organización.

Una arquitectura orientada a servicios (SOA) es por definición una arquitectura distribuida. Ya que su objetivo es comunicarse remotamente con otro servicio para tener acceso a ciertos datos [5]. Su unidad básica de funcionamiento son los servicios y SOA es más una forma de estructurar el software pensando cada funcionalidad como un servicio [6].

Adoptar SOA no implica en ningún momento desechar los sistemas que ya están dentro de la organización. SOA adopta estos sistemas existentes y propone una visión integral de la situación. Estos sistemas realizan funciones elementales para el funcionamiento de la organización. El paradigma busca aprovecharlos para modernizarlos y definir estándares para comunicar e integrar las soluciones. De esta manera la información se expone mediante servicios y puede ser utilizada para definir procesos de alto nivel que representan procesos de negocio de la organización.

Al ser SOA un paradigma y no un producto, es difícil comprar SOA. Por lo tanto tener un producto u otro no es equivalente a utilizar SOA como paradigma para la integración de aplicaciones. Cuando una organización empieza el recorrido para adoptar SOA surge la necesidad de plantear un esquema de adopción, es decir una forma de programar y definir de qué manera se continuará hacia la adopción del paradigma. Existen diferentes alternativas para definir el esquema de adopción. Entre estas alternativas encontramos a OSIMM [7], un modelo de código abierto de adopción SOA.

OSIMM presenta un esquema de adopción iterativo y se pueden distinguir dos tipos de etapas: la etapa de análisis donde se realiza un trabajo de relevamiento en toda la organización. Esto involucra entrevistas y reuniones con cada responsable involucrado de alguna manera con el proceso de adopción. También se suele consultar documentación asociada a cada sistema provista por la organización. La segunda etapa es donde se genera un matriz de adopción reflejando el estado actual de la organización y se plantean objetivos a cumplir para avanzar al siguiente nivel.

La matriz que define el modelo OSIMM es una matriz donde las dimensiones son aspectos o categorías de los sistemas de la organización: proceso, metodologías, infraestructura, arquitectura. Los niveles corresponde a las filas donde se asocian con el estado de la organización: sistemas monolíticos, aplicaciones orientadas a objetos, aplicaciones orientadas a componentes, diseño orientado a servicios, aplicaciones orientadas a servicios son ejemplos de niveles. La definición de objetivos por cada uno de los entrecruzamientos genera un roadmap o una sucesión de tareas a realizar para lograr que la organización llegue al próximo nivel de adopción. El proceso de análisis y definición de objetivos se repite cada vez que se completan los objetivos de un nivel.

Plantear los objetivos de cada entrecruzamiento no es una tarea trivial, ya que los objetivos están definidos con una granularidad muy alta. La granularidad implica un alto grado de abstracción y la ejecución de estos objetivos suele llevar un tiempo considerable. Al ser muy alto el tiempo de implementación de cada objetivo, se torna necesario contar con experiencia necesaria sobre cómo implementar este objetivo en la organización. Esta experiencia aporta mucho valor agregado y se obtiene a través del tiempo y los procesos de adopción encarados. Por lo tanto surge la necesidad de contar con una herramienta que permita modelar las condiciones y los objetivos a ejecutar por cada nivel de adopción. De manera de facilitar el acceso a la información subyacente en cada adopción. Esto genera un desafío, ya que actualmente no existe un mecanismo que nos aconseje o al menos advierta en base a experiencias previas qué objetivos deberían ejecutarse o cómo se debería plantear los objetivos de una adopción en particular siguiendo los resultados del análisis preliminar.

1.3 Motivación

Ejecutar un proceso de adopción SOA dentro de una organización es un proceso difícil, complicado y generalmente prolongado en el tiempo. El proceso consta de una etapa de recolección de datos cuyo objetivo es conocer el estado actual de la organización. Esto permite definir luego los objetivos y distribuirlos a través de las dimensiones y niveles que plantea la matriz de adopción.

La adopción presentada por el modelo OSIMM es representada como una matriz cuyas filas se denominan dimensiones y las columnas denotan los niveles. La maduración de una dimensión es dada por el nivel en que se encuentra la organización. El nivel en una dimensión se determina a través de herramientas, prácticas, infraestructura que se estén utilizando en la organización. La utilización, definición o puesta en marcha de cada uno de los ítems anteriormente nombrados se denominan objetivos. Se dice que una organización siempre está al menos en el primer nivel al comenzar el proceso de adopción.

Entre cada uno de los objetivos existen dependencias implícitas, ya que generalmente se deben seguir una serie de pasos para cumplirlo. Estos pasos que determinan el cumplimiento del objetivo pueden ser o bien otros objetivos de niveles inferiores u objetivos que marcan tareas con una granularidad más fina que el objetivo original. Las relaciones existentes entre los objetivos determinan un flujo de trabajo para avanzar en el proceso de adopción.

Tradicionalmente la precedencia de objetivos y su división es un proceso realizado manualmente. Toda la información y experiencia que se obtiene en cada una de las adopciones queda en manos del grupo de trabajo llevó a cabo el análisis y tomó la decisiones correspondientes al estado de la organización. Si bien existe documentación que respalda las decisiones tomadas, no es posible reconocer patrones de adopción que se repiten entre organizaciones. También la transferencia de conocimiento entre un proceso de adopción y otro es más difícil.

La motivación de nuestro trabajo es desarrollar una herramienta confiable siguiendo las políticas dictadas por el modelo de adopción OSIMM, que facilite la generación de la matriz de adopción y que permita la utilización de la información generada por el proceso de adopción a futuro.

2 SOA

2.1 Introducción

En este capítulo se trata la evolución de los sistemas de información pero enfocándose en los mecanismos de integración utilizados en ese entonces. Se realiza un breve repaso de cómo fueron evolucionando las distintas prácticas para la construcción de aplicaciones o sistemas hasta la llegada de SOA.

El objetivo de este capítulo es demostrar cómo se llega hasta la versión de SOA que tenemos hoy en día, motivado por los cambios tecnológicos cada vez más frecuentes y por ende, los cambios más frecuentes en el negocio asociado a una organización

2.2 Evolución de los sistemas

2.2.1 Introducción

Dentro del mundo de los sistemas se pueden tomar dos factores que ocasionaron un impacto muy grande en la forma que los sistemas o aplicaciones colaboran. El primero es la continua mejora del hardware. Mediante la caída de precio de los componentes y la constante expansión del horizonte del rendimiento los sistemas fueron capaces de realizar cada vez más tareas. La segunda fue la forma que todas estas computadoras estaban conectadas: las redes. Gracias a la conectividad alcanzada con las redes, surgieron nuevas formas de implementar los sistemas. Además obligó a las personas encargadas de desarrollar estos sistemas a adaptarse a las nuevas posibilidades que abrían las redes de computadoras.

Las redes han evolucionado en los últimos años a través de los años de manera vertiginosa. Inicialmente conformadas por algunas computadoras conectadas a través de líneas de comunicación dedicadas [9], hasta llegar a nuestros días donde una cantidad enorme de computadoras (y por ende recursos) está disponible a través de Internet.

En las Fig 1 y 2 se puede ver el contraste con respecto al nivel de conectividad en 1977 y al que está disponible hoy en día.

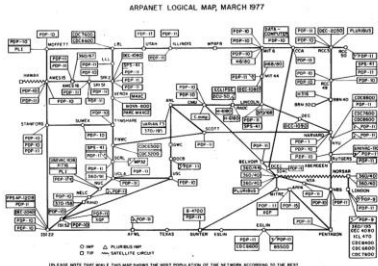


Fig. 1 Mapa lógico de Arpanet circa 1977

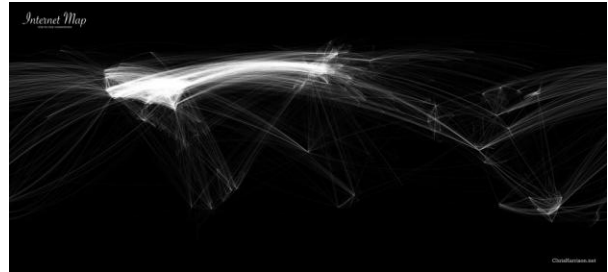


Fig. 2 Conectividad disponible a nivel global en nuestros días

Acompañando a este fenómeno, las aplicaciones han evolucionado desde sistemas monolíticos, hasta sistemas distribuidos geográficamente separados. Debido al crecimiento de las redes de computadoras, los sistemas han incrementado su tamaño, complejidad y sus capacidades, por nombrar algunos puntos. Como es de esperar las prácticas utilizadas para desarrollar aplicaciones han ido cambiando con el pasar del tiempo y obligando a todo el personal técnico (desarrolladores, arquitectos, expertos) a reemplazar los antiguos enfoques para el desarrollo de aplicaciones y encarar las soluciones adaptándose al marco tecnológico presente en ese momento.

Uno de los primeros enfoques para desarrollar aplicaciones fueron los sistemas monolíticos, conocidos como mainframes. Las aplicaciones monolíticas son consecuencia de los primeros sistemas con un sólo procesador, donde el procesamiento y administración de los datos está completamente centralizado. En la Fig. 3 podemos ver el aspecto físico de este tipo de dispositivos en 1990.



Fig. 3 Mainframe Honeywell-Bull DPS 7 de 1990

El primer paso para separar el tratamiento de datos fue dado a través de los lenguajes procedurales o programación procedural. Estos permitían manipular los datos a través de operaciones, además de un nivel de reúso de porciones de código muy prematuro. Este enfoque terminaba generando dependencias entre los segmentos de código que se volvían complejos de mantener. Muchas veces era difícil determinar el grado de impacto en la aplicación de un cambio en el código fuente.

FORTRAN, COBOL, C, Pascal y BASIC son lenguajes para desarrollar software siguiendo el paradigma procedural. El desarrollo de una aplicación requería un conocimiento detallado de cómo eran

representados los datos en los dispositivos físicos. Por último aunque las aplicaciones podían estar separadas en módulos y se empezaba a incorporar el reúso de código a través de librerías, todavía seguían existiendo muchas dependencias internas y lo que complicaba la aplicación de un cambio o modificación sobre el código.

Las dificultades con los lenguajes procedurales expresadas en el párrafo anterior merecían algún tipo de trato para mitigar los efectos. Con el pasar del tiempo surge una práctica que dividía los problemas en problemas más pequeños. Nace el diseño estructurado. El diseño estructurado [10] pregonaba la idea de descomponer los problemas en problemas más pequeños. La Fig 4. Muestra un ejemplo de diseño estructurado en acción, donde el problema de “Actualizar un archivo” se divide en problemas más pequeños.

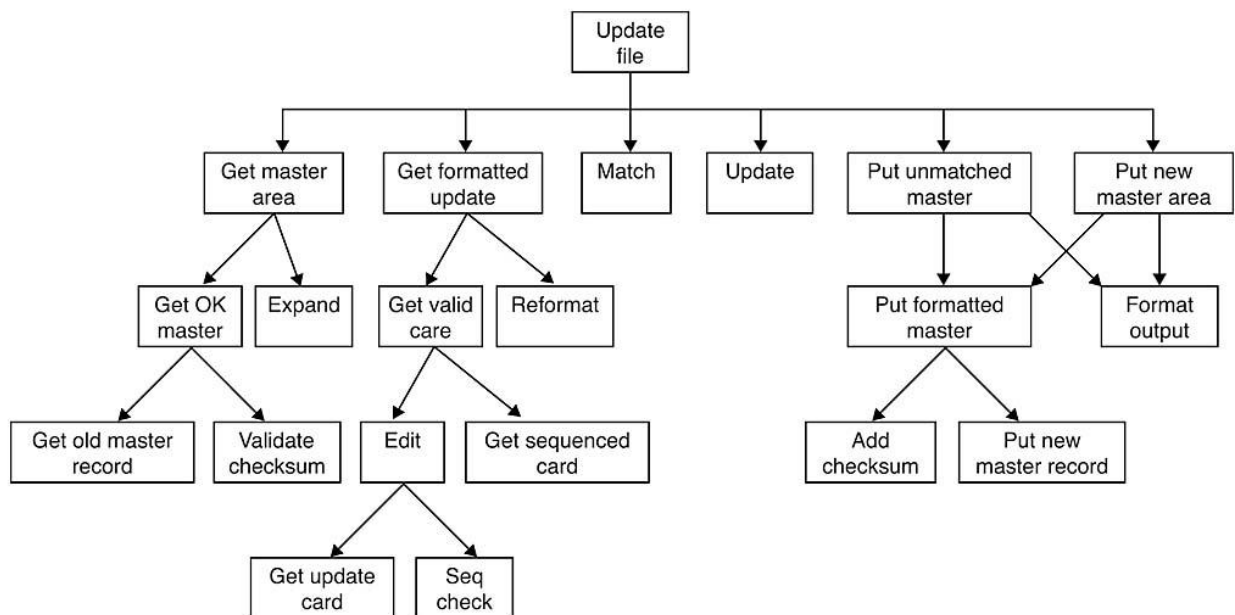


Fig. 4 El diseño estructurado pregonaba la idea de descomponer los problemas en problemas más pequeños

De esta manera, la solución de un problema en general constaba de atacar varios problemas de menor complejidad. Al estar separados y diferenciados, se obtenía módulos realmente reusables. Por último, con el diseño estructurado se puede obtener una clara distinción entre el comportamiento o flujo de la aplicación y el manejo de datos asociado a la aplicación como se puede observar en la Fig 5.

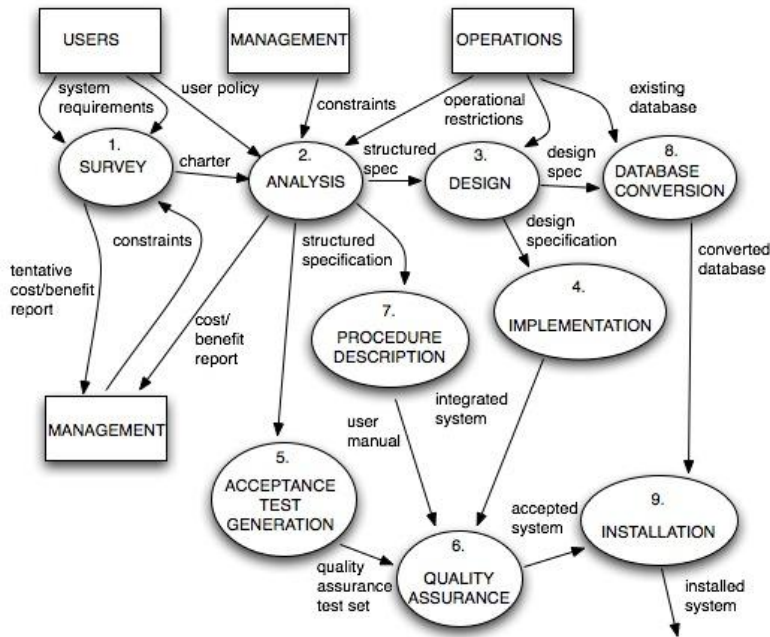


Fig. 5 Con el diseño estructurado se puede obtener una clara distinción entre el comportamiento o flujo de la aplicación y el manejo de datos a asociado a la aplicación

El diseño estructurado permitió el desarrollo de software más flexible y aún más complejo. De todas maneras seguía existiendo una gran dependencia entre los módulos y los datos manejados por la aplicación.

La dependencia existente entre el comportamiento y los datos de los sistemas se empieza a romper con el nacimiento del esquema cliente-servidor. Un diseño que respeta los principios de una arquitectura cliente servidor implica la separación del cliente y la lógica de negocio del repositorio de datos. Esto fue posible gracias a los avances que presentaba por aquel entonces, la conectividad entre aplicaciones a través de los protocolos de red. Algunos persisten hasta el día de hoy (y están lejos de ser reemplazados) como el protocolo HTTP o cualquier protocolo utilizado en la comunicación de una aplicación y la base de datos. Las arquitecturas cliente-servidor pueden considerarse como las primeras arquitecturas distribuidas.

Siguiendo con la evolución de los sistemas, la arquitectura cliente-servidor era el camino más natural a seguir. Esta arquitectura presenta en el momento del diseño una clara separación de responsabilidades entre las capas que la integran. La lógica de negocio, el almacenamiento de datos y la presentación de la información pasaron a ser responsabilidades con cierta independencia entre cada una de ellas. Esta separación fue posible en parte por el crecimiento que tuvo el protocolo HTTP, transportando la información entre cada uno de los componentes distribuidos de la solución. Las bases de datos, brindando métodos de accesos remotos también colaboraron en gran parte a la adopción de este estilo de arquitectura. En la Fig. 6 vemos un diagrama que muestra un ejemplo de arquitectura cliente-servidor, donde los servidores y clientes se comunican a través de un canal de comunicación común.

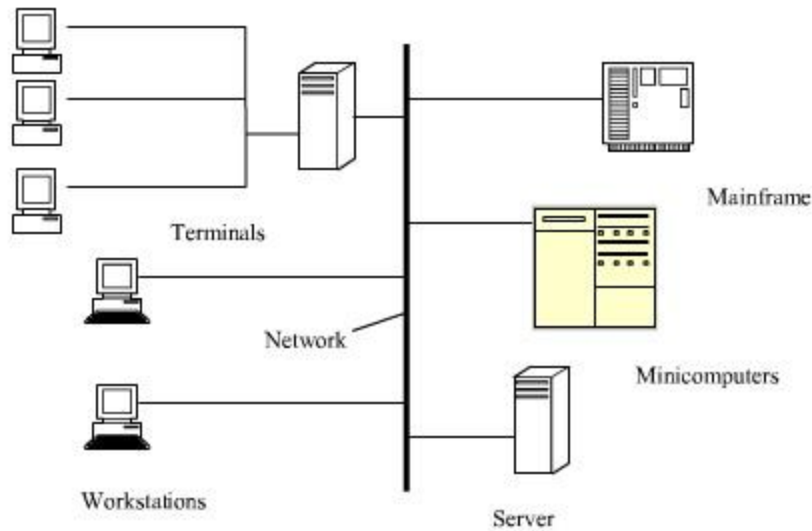


Fig. 6 Integrantes de una arquitectura cliente-servidor

La década de los 1990's trajo consigo la necesidad de separar claramente el acceso a datos y la lógica de la aplicación de la presentación de la información. Por lo tanto, las arquitecturas cliente-servidor empiezan a ser reemplazadas por arquitecturas orientadas a componentes reusables. Como máximo exponente de este paradigma, surge la arquitectura de N-capas o (N-tier). Al agregar más capas, se reduce el acoplamiento entre cada una de ellas, por la separación natural de responsabilidades. Esto permitió que existan diferentes tipos de clientes accediendo o interactuando con la capa de lógica de negocio. La Fig.7 nos muestra un ejemplo de arquitectura n-capas dentro de una aplicación.

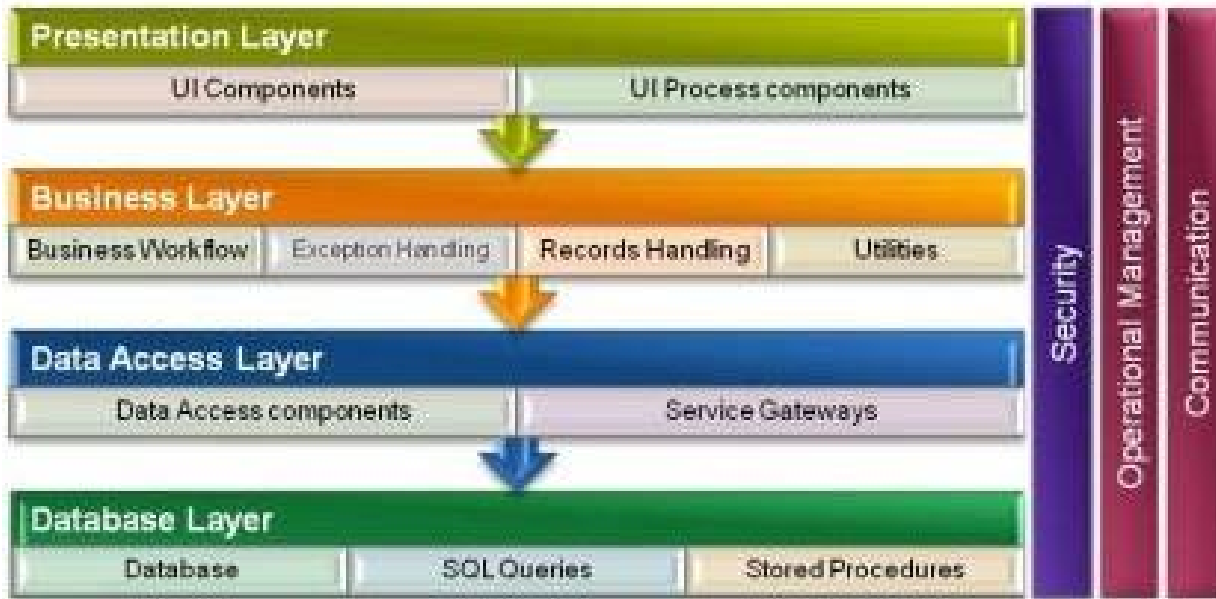


Fig. 7 Ejemplo de arquitectura n-capas donde se distinguen la capa de presentación, negocio, acceso y datos.

Por último, se debe distinguir dos tipos de sistemas o aplicaciones que son utilizadas en organizaciones. El primer tipo de aplicaciones corresponde a las que son producidas por empresas, terceros o privados. Poseen su propio protocolo de comunicación y las modificaciones o extensiones sólo son ingresadas por el proveedor del software. A este tipo de productos se les denomina “software propietario”. Junto con este tipo de aplicaciones está asociada una infraestructura para dar soporte a los clientes o usuarios del sistema. Por otro lado también están los sistemas denominados abiertos que en posición opuesta al software propietario, se trabaja entre los distintos proveedores en pos de definir un estándar al que todos los productos luego se sumarán. Al compartir la tecnología o las definiciones entre la comunidad, se genera una sinergia importante para nuevas mejoras o en el desarrollo de nueva funcionalidad o integración con sistemas existentes. La mayor ventaja de este tipo de sistemas es la interoperabilidad, al no estar afectados por lo que se denomina “vendor lock”. Si bien puede ocurrir que exista software propietario basado en estándares abiertos, por la naturaleza cerrada de estos productos, suele ser costoso tanto en tiempo como monetariamente, ampliar la funcionalidad de estos productos. Este costo a veces se vuelve prohibitivo contra las necesidades de la organización, lo que dificulta el camino a la madurez y a veces provoca la adopción de planes alternativos y poco convenientes en el largo plazo para alcanzar el objetivo planteado.

Si bien puede ocurrir que existan sistemas propietarios basados en estándares abiertos, los proveedores de software generalmente restringen las extensiones a contratos o acuerdos de trabajos contra la empresa u organización que está utilizando el software.

Podría considerarse que una gran ayuda para los avances en interoperabilidad entre sistemas fue dado por Internet, la web y varios de sus derivados. Entre ellos se destaca XML que se define como un meta lenguaje y el estándar para compartir información entre sistemas heterogéneos.

Como XML es indicado para describir los datos de manera independiente a la plataforma que lo haya generado, posibilitó y facilitó en gran medida el intercambio de datos entre distintas aplicaciones. Reusando la infraestructura ya montada (Internet) y un metalenguaje neutral (XML) la integración entre aplicaciones se facilitó de manera significativa. Hoy en día en el desarrollo de cualquier aplicación no es extraño encontrarse con funcionalidad que exponga servicios que puedan interactuar con clientes y que realizan sólo una función determinada. En conjunto y colaborando cada uno de estos servicios ayuda a formar lo que luego será la infraestructura de la aplicación.

2.2.2 Programación orientada a objetos

La programación orientada a objetos promueve el encapsulamiento de datos y comportamiento dentro de una unidad fundamental conocida como clase. Las instancias de las clases junto con sus relaciones forman objetos. Cualquier cambio en la representación de la información afecta sólo al objeto que contiene ese dato. Las clases tienen un ciclo de vida que puede ser hasta el fin de los tiempos, mientras que el ciclo de vida de una instancia puede ser mucho más breve.

Los objetos se comunican entre sí mediante el envío de mensajes. Con esta premisa el ocultamiento de información y de comportamiento es fácil de lograr. El éxito o la posibilidad de aplicar estas técnicas en nuestro desarrollo, implica el grado de acoplamiento que existirá entre las instancias de los objetos.

Idealmente en un lenguaje orientado a objetos todo es un objeto. Las principales características de un lenguaje de este tipo son las siguientes:

Encapsulamiento

Un objeto puede contener datos (estado) y funcionalidad (comportamiento) formando una unidad, como se puede observar en la Fig. 8.

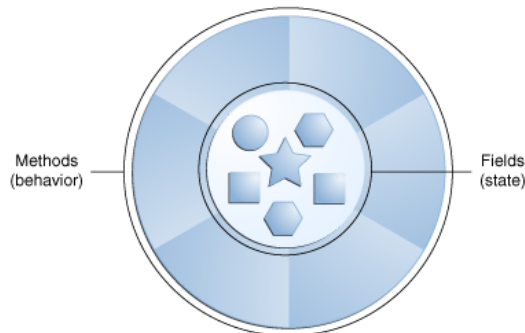


Fig. 8 Un objeto puede definirse como una unidad compuesta de métodos (comportamiento) y propiedades (estado)

Ocultamiento de información

Un objeto mantiene su estado interno oculto de los demás objetos y no revela información específica de su estado interno a los demás. En la Fig. 9 se puede observar un ejemplo de ocultamiento de información.

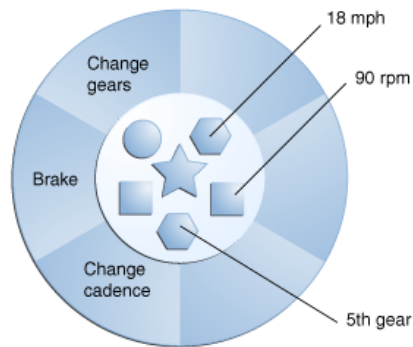


Fig. 9 Ejemplo de la representación de una moto como objeto, donde se identifican los métodos y el estado actual del objeto

Herencia y asociación

Los objetos pueden asociarse entre sí. La herencia es una forma de asociación entre clases y define una relación de tipo “es un” entre sus participantes. Permite que las clases sean extendidas agregando más comportamiento y estado aprovechando lo definido en la superclase, como se puede observar en el ejemplo de los rodados en la jerarquía de la Fig 10.

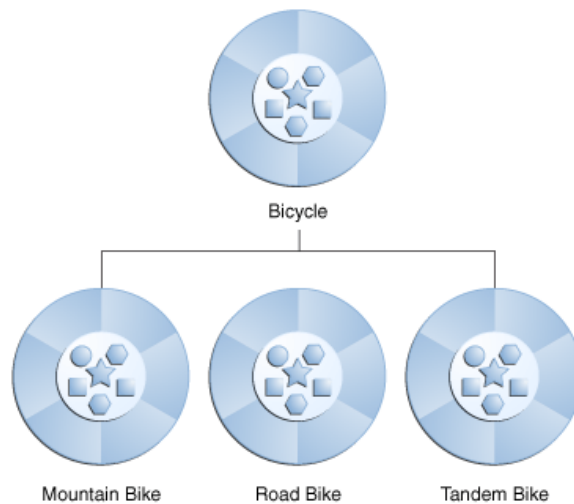


Fig. 10 Ejemplo de la representación de una moto como objeto, donde se identifican los métodos y el estado actual del objeto

Polimorfismo

Los lenguajes orientados a objetos permiten que distintos objetos puedan responder al mismo mensaje, obteniendo resultados diferentes. En la Fig. 11 vemos como dos figuras responden de diferente manera al mensaje Area().

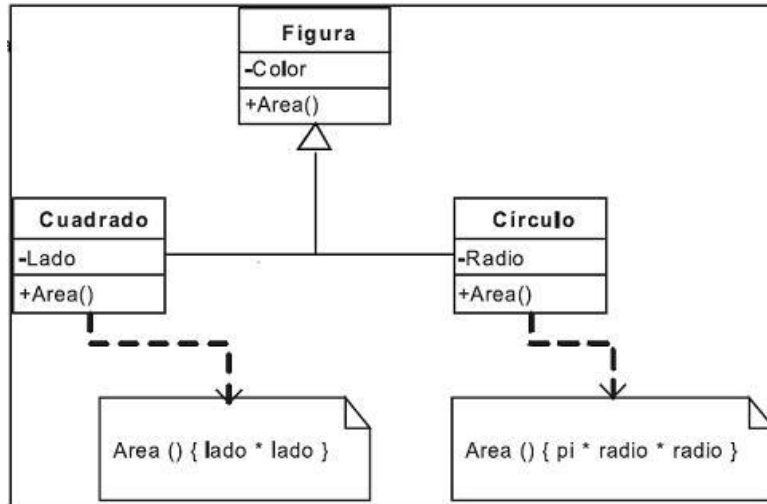


Fig. 11 En este ejemplo de polimorfismo, dos figuras responden diferente al mensaje “Area()”

Comparado con el estilo procedural, el diseño y desarrollo orientado a objetos es un gran avance teniendo en cuenta la capacidad de abstracción que provee a la hora de modelar procesos de negocio del mundo real. En el diseño procedural se basaba en control de flujos y representación de datos. En cambio, en el diseño orientado a objetos es común ver los participantes del problema modelados como clases con su respectivo comportamiento y estado.

El diseño orientado a objetos ha superado la etapa del éxito comercial, pasando la prueba en soluciones extensas y por demás complejas. Permite planificar la estructura interna del software, mejorando su calidad encontrando anticipadamente las deficiencias que una implementación puede llegar a tener. Además facilita la creación de prototipos de software. En cuanto a las herramientas de trabajo, hoy en día existe una variedad inmensa de lenguajes orientados a objetos o que presentan algún tipo de soporte para objetos. Además de los tradicionales exponentes del ramo: Smalltalk, Java, C++, C# hoy en día la mayoría de los lenguajes incluye el soporte para objetos.

Las entidades del mundo real se pueden representar de manera inmediata utilizando metodologías orientadas a objetos. Tal fue el crecimiento en los últimos tiempos, que esta tecnología ha sido adoptada como el paradigma por defecto para desarrollar software en varios lenguajes de programación. Cuando se aísla un requerimiento o funcionalidad específica, definiendo un fin concreto y un contexto de aplicación cerrado, se está definiendo un componente de software.

2.2.3 Programación orientada a componentes

Los componentes emergen como módulos de software con una mayor complejidad inherente. Desarrollar sistemas utilizando componentes requiere de un cambio en la manera de implementar los sistemas y la forma de plantear las soluciones. Por último, como los componentes generalmente atacan un problema

específico y común a varios escenarios como se puede observar en el ejemplo planteado en la Fig. 12 donde se ven dos componentes que colaboran para realizar un pago de una compra.

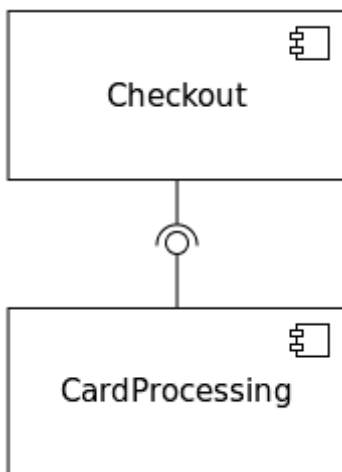


Fig. 12 El componente de “Checkout” encargado de implementar la compra de un cliente requiere que el componente de procesamiento online de tarjetas de crédito pueda procesar la compra.

Un componente de software está definido por su contrato o interface. Se puede definir como un conjunto de objetos que colaboran para cumplir con un rol bien definido. Este contrato tiene validez en el contexto en el cual se utiliza el componente. En algunos casos los componentes pueden ser instalados dentro de un servidor de aplicaciones, solucionando así algún tema funcional o técnico en particular. Un claro ejemplo podemos encontrarlo en cualquier servidor de aplicaciones J2EE del mercado, donde nos podemos encontrar con manejo de transacciones, pool de conexiones entre varios componentes. Cada uno de ellos fue diseñado como de propósito general y sin embargo cada una de las aplicaciones instaladas puede utilizarlos siguiendo el mecanismo definido por la plataforma.

El desarrollo orientado a componentes toma varios principios del desarrollo orientado a objetos como encapsulamiento y polimorfismo, dejando de lado en lo posible a la herencia. Los componentes reúsan funcionalidad existentes invocando a otros objetos o componentes, antes que heredar de los mismos. Estas invocaciones se denominan “delegaciones”.

Los componentes poseen una especificación pública para integrarse con otros componentes o ser orquestados para satisfacer un requerimiento. Las especificaciones de estos componentes deben ser lo más amplias posibles para que puedan ser reusados en cualquier otra situación. Ya sea de la misma aplicación, organización o la industria en general. En la Fig. 13 se ve un diagrama de componentes donde cada uno colabora a través de interfaces bien definidas para implementar un programa de recompensas a clientes.

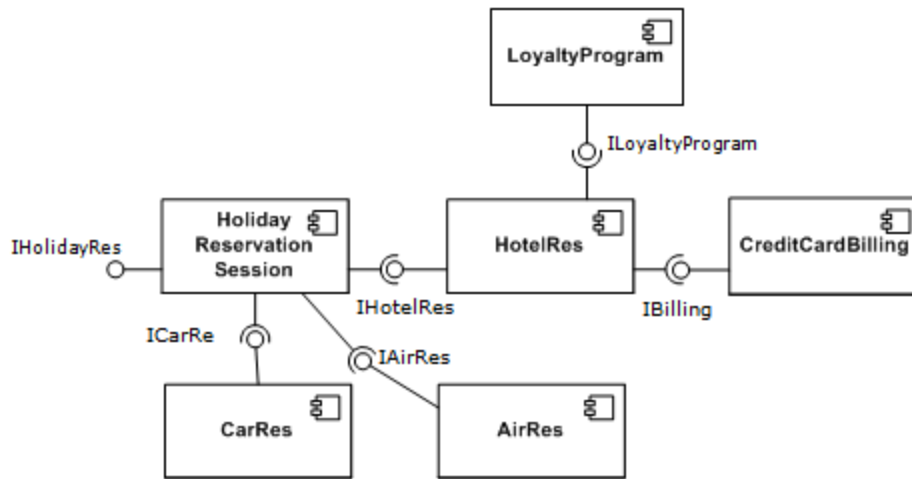


Fig. 13 Componentes interactúan a través de interfaces bien definidas

Cuando se integran varios componentes se forma una nueva entidad, que dependiendo de sus características puede ser definida como otro componente en sí, un framework o inclusive una aplicación. Esta práctica se denomina composición. Cuando los componentes tienen una especificación común, es decir definida como estándar, puede reemplazar un componente por otro, sin que esto afecte el comportamiento de la aplicación o en el peor de los casos, reduciendo el impacto a puntos muy específicos.

Un ejemplo de buen diseño de software es una arquitectura orientada a componentes. Los nuevos componentes cumplen una función bien definida en el contexto de la aplicación o del problema a solucionar. Al ser componentes, se provee una interface y una especificación que indica cómo utilizarlos. Esto también provoca que los límites de acción o el alcance del componente estén bien definidos, haciendo más sencilla la tarea de realizar pruebas de unidad u cualquier otro tipo de validación del componente. Reutilizando los componentes y formando nuevo software se llega a nivel de plataforma o infraestructura, donde un conjunto de componentes de uno o varios vendedores se junta para brindar servicios corporativos. Con XML y WS estos componentes pasan a estar distribuidos, formando una arquitectura corporativa orientada a componentes. El diagrama de la Fig. 14 muestra que si bien cada uno de los componentes esté físicamente distribuido, esto es transparente para la aplicación.

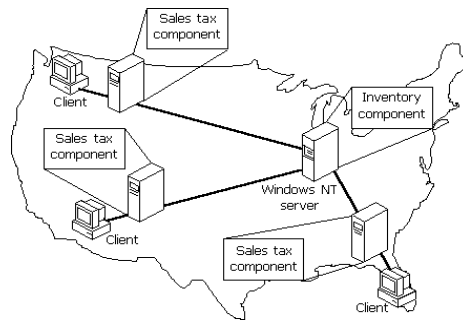


Fig. 14 Ejemplo de arquitectura orientada a componentes distribuidos, donde no necesariamente todas las piezas de software se ejecutan en el mismo contexto

2.2.4 Sistemas distribuidos

Comunicar dos sistemas directamente a través de un enlace físico y un protocolo ad-hoc o particular para tal fin es el primer enfoque de lo que se denomina sistemas distribuidos y tal vez su forma más elemental. Las API de programación provista por los protocolos de red como SNA, TCP/IP, IPX ayudaron a mitigar el esfuerzo necesario para comunicar dos o más sistemas. Con el pasar del tiempo se fue madurando hasta llegar a un middleware de comunicación que permitía acceder remotamente a recursos expuestos por aplicaciones sin conocer detalles como el sistema operativo, implementación del protocolo de red y tener idea sobre direcciones de red físicas. En la Fig. 15 se observa un ejemplo donde las áreas de Fabricación, Finanzas, Ventas y la base de datos están distribuidos y se comunican a través de redes.

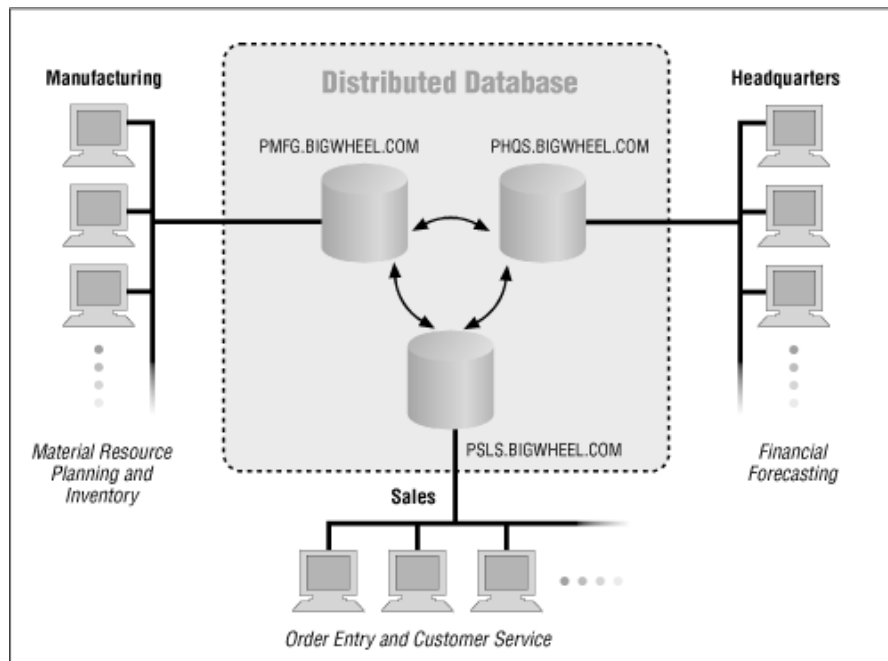


Fig. 15 Ejemplo de sistemas distribuidos

A medida los sistemas distribuidos maduraban técnicamente, fue necesario que el medio físico y el transporte lógico (red y protocolos) satisfagan varios requerimientos como: performance, entrega asegurada, seguridad entre otros requerimientos no funcionales.

Infraestructuras de objetos distribuidos permiten acceder a los recursos remotos como si fueran locales. Permiten elegir el protocolo adecuado para llevar a cabo una comunicación que cumpla con los requerimientos dados. Por lo tanto, se obtuvo como resultado la interoperabilidad de componentes de software corporativos y distribuidos.

La comunicación entre aplicaciones se puede realizar de dos maneras: asincrónica y sincrónica. Sin embargo en la realidad existen varias formas de comunicación derivadas de estas dos.

En la comunicación sincrónica en intercambio de mensajes requiere un compromiso simultáneo en ambos extremos de la comunicación. Este mecanismo de comunicación se suele denominar “request-response”.

Por otro lado, en la comunicación asincrónica el cliente envía un mensaje y continúa con su procesamiento sin esperar por una respuesta. Luego, por un canal de comunicación definido el receptor de mensaje puede enviar la respuesta al mensaje. El contraste entre los dos esquemas de comunicación se puede ver en el Fig. 16.

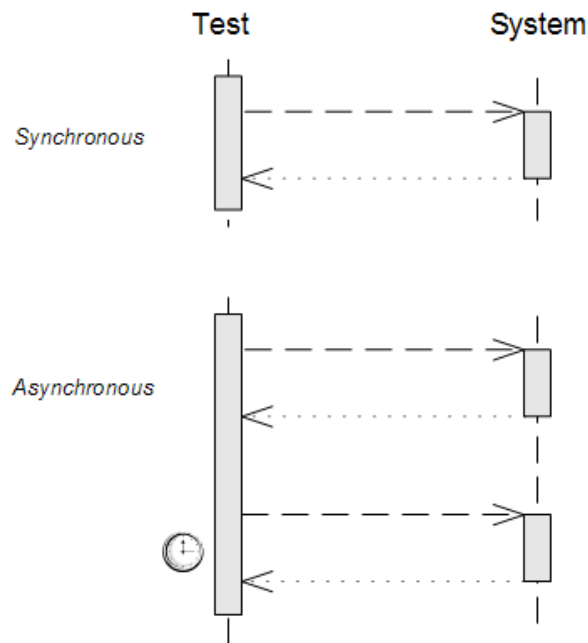


Fig. 16 Diagrama de secuencia presentando las alternativas de comunicación sincrónica vs asincrónica

2.2.5 Remote Procedure Call (RPC)

Creado en la década de 1980 por Sun, RPC se define como un mecanismo para invocar a procedimientos en aplicaciones remotas. RPC toma la definición de una función o procedimiento de una aplicación distribuida y la expone para que sea consumida remotamente por otras aplicaciones. La invocación remota llega hasta la otra aplicación a través de la red, donde es ejecutada. Luego el resultado es retornado al cliente que originó la invocación. Gran parte de las implementaciones RPC estaban basadas en protocolos sincrónicos de comunicación, siguiendo el modelo request-response. Este modelo de comunicación obliga que el cliente permanezca bloqueado hasta que el servidor envíe la respuesta, como se puede observar en la secuencia representada en la Fig. 17.

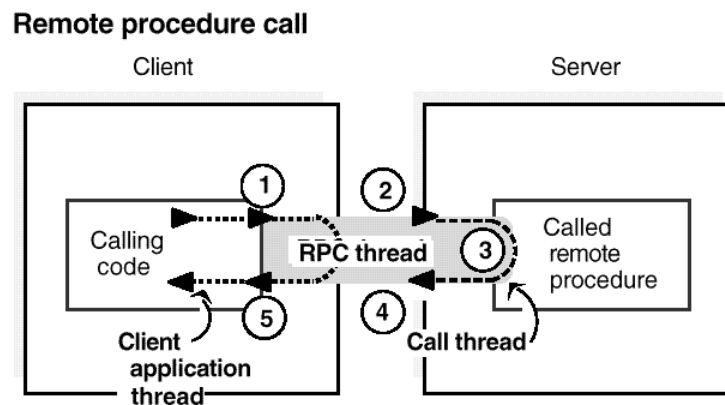


Figure 6-1 Execution Phases of an RPC Thread

Fig. 17 Fases en la ejecución de un hilo RPC

El modelo RPC se divide en varios componentes. Un protocolo de red que provee enrutamiento, secuencialidad de paquetes, direccionamiento y forwarding de paquetes para transmitir los datos entre nodos. La implementación de RPC generalmente es específica para cierto protocolo de red y además de transmitir información entre distintos nodos, se encarga del manejo de error y control de flujo. Los componentes instalados de ambos lados de una comunicación entre aplicaciones (server stub y client stub) son los encargados de administrar las conexiones, creando, cerrando y terminándose según cómo esté implementado el comportamiento. El último eslabón de la cadena está formado por el layer correspondiente a la aplicación, donde la funcionalidad de negocio está implementada y por lo tanto, es el contexto en donde se ejecutan las llamadas. La relación entre las capas se muestra en la Fig. 18

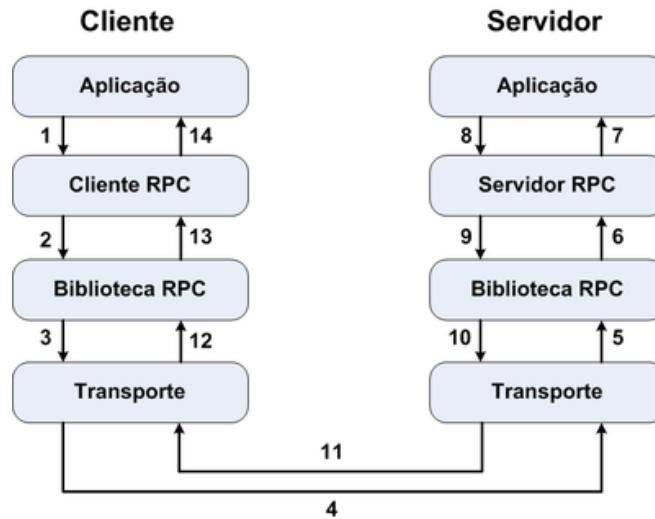


Fig. 18 Secuencia de interacciones en una llamada RPC

La necesidad de aplicaciones independientes y distribuidas fue la principal razón para la implementación de protocolos de comunicación al estilo RPC. Cerca de 1990, se trató de estandarizar sin éxito varias tecnologías similares en una iniciativa denominada DCE (Distributed Computing Environment). Eso dio lugar a que otras tecnologías emerjan como las más utilizadas: CORBA, DCOM y Java RMI son algunas de las tecnologías que todavía se siguen utilizando para implementar comunicación al estilo RPC entre plataformas. En las Fig. 19, Fig. 20 y Fig. 21 se constrastan cada uno de los esquemas de comunicación RPC-Like.

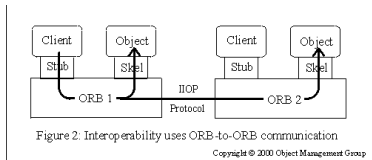


Fig. 19 Diagrama de una solución CORBA

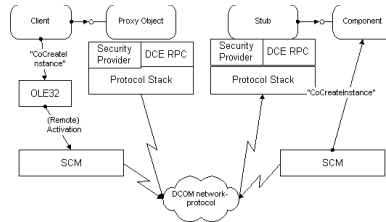


Fig. 20 Diagrama de una solución DCOM

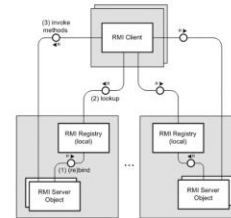


Fig. 21 Diagrama de una solución RMI

2.2.6 Objetos distribuidos

El concepto de objetos distribuidos surge a partir de los años 1990's. Es la evolución natural de la programación orientada a objetos en un ambiente remoto o distribuido. La solución se implementa con un ORB (Object Request Broker) quien es el encargado de realizar la comunicación y el intercambio de datos entre los objetos remotos. Con esto se materializa una plataforma de objetos distribuidos con un

concepto que se denomina “transparencia de ubicación”. Además permite que los objetos oculten los detalles de su implementación a los clientes. El ejemplo característico de esta solución lo encontramos en CORBA que es una solución RPC basada en un Object Request Broker.

2.2.7 Definición de SOA (Arquitectura orientada a servicios)

SOA en una definición amplia, es un estilo de arquitectura de software empresarial que permite que las aplicaciones de diferentes negocios y diferentes plataformas se comuniquen entre sí dinámicamente. SOA provee una unidad básica de trabajo, genérica y confiable denominada servicio. Estos servicios son la base fundamental de SOA y permite generar aplicaciones más complejas a través de su composición u orquestación. Un ejemplo de las tecnologías y conceptos que están asociados a SOA lo podemos ver en la Fig.22.



Fig. 22 Tecnologías y conceptos asociados a SOA

Aunque SOA no es un concepto reciente, tomó impulso de la mano de la adopción de Web Services en el desarrollo de aplicaciones. Esto provocó una nueva tendencia a la hora de plantear la arquitectura de aplicaciones y un nuevo enfoque que permite integrar a los viejos sistemas legacy a la nueva plataforma de la organización. Una forma de exponer sistemas legacy está representada en la Fig. 23.

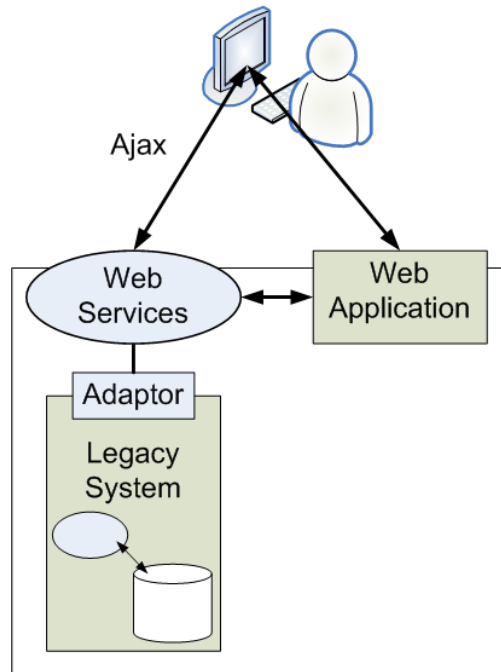


Fig. 23 Exponiendo sistemas legacy a través de Web Services

A pesar de sus beneficios, adoptar la metodología SOA para una organización sigue siendo una tarea desafiante por los siguientes motivos:

1. SOA propone cambios en la arquitectura de software de las aplicaciones. En ocasiones suele ser un enfoque muy radical y la transición no es fácil.
2. Las aplicaciones necesitan respetar ciertos estándares técnicos para ser aptas para exponer o consumir servicios.
3. Será frecuente que el ambiente de la organización deba ser accedido por terceros, lo que supone desafíos técnicos para rutear los pedidos.
4. Administrar la definición y composición de nuevos servicios para crear nuevos servicios de más granularidad también es una tarea que necesita de mucha sincronización dentro de la organización.

2.2.8 Actores principales y características de SOA

SOA es un paradigma de arquitectura de software que define las interacciones entre tres actores principales: el proveedor de un servicio que se encarga de exponer la funcionalidad que provee y el consumidor del servicio que a través de un registro de servicios busca el más adecuado para el requerimiento funcional que debe completar. Implícitamente en la descripción fue nombrado el registro de servicios, en cual podemos ver cómo interactúa con los demás integrantes de la solución SOA en la Fig. 24.

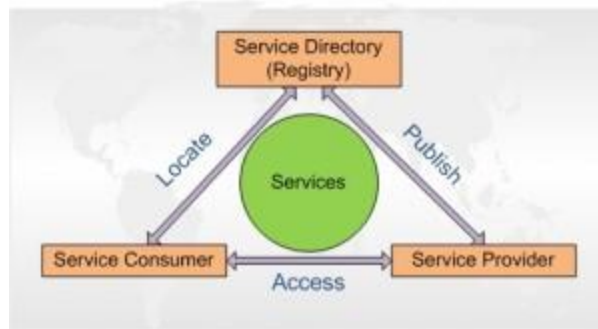


Fig. 24 Relaciones entre participantes de una solución SOA donde se destaca el rol del Registro de Servicios

Si realizamos un modelo conceptual de SOA podemos encontrar seis actores en total, que aumentan a los tres nombrados en el párrafo anterior:

2.2.8.1 Consumidor del servicio

Es la entidad o actor que debe buscar por un servicio que le permita resolver alguna funcionalidad en particular. Un consumidor puede ser una aplicación, otro servicio o algún componente de software que requiera de un servicio. La ubicación física del servicio se obtiene consultando al registro, pero también puede ser conocida de antemano. En este último caso, se dice que el consumidor del servicio interactúa directamente con el proveedor del servicio.

2.2.8.2 Proveedor del servicio

Es una entidad que puede ser ubicada a través de la red que ejecuta los requerimientos que llegan desde los consumidores. Se encarga de crear el descriptor del servicio (su definición) y también de implementarlo para que los demás puedan accederlo.

2.2.8.3 Registro de servicios

Es un directorio al que se puede acceder a través de una red local y contiene todos los servicios disponibles en la organización. Su función principal es tomar la definición de los servicios de los proveedores y exponerla para que los consumidores puedan acceder a ella.

2.2.8.4 Contrato del servicio

Es la definición que marca la forma en la cual el consumidor y el productor del servicio interactuarán. Define la estructura de datos a compartir, tiempos de respuestas y en la manera que se espera que el servicio sea invocado.

Cuando se está en la tarea de construir una infraestructura orientada a servicios (un conjunto de aplicaciones que colaboren exponiendo servicios) salen a la luz especificaciones y principios propios de una arquitectura orientada a servicio, los cuales se describen en el siguiente punto.

2.2.9 Principios de una arquitectura orientada a servicios

2.2.9.1 Los servicios son descubiertos y asociados de manera dinámica



Fig. 25 Los servicios se descubren de manera dinámica a través de un Registro

Los servicios pueden ser descubiertos en tiempo de ejecución. El consumidor del servicio, primero ubica el servicio utilizando el Registro de Servicios donde se encuentra toda la información necesaria para invocar el servicio. La Fig. 25 muestra cómo interactúan las tres partes. No existe una dependencia en tiempo de compilación con el servicio, excepto del contrato expuesto por el Registro de Servicios.

2.2.9.2 Los servicios son autocontenidos y autosuficientes

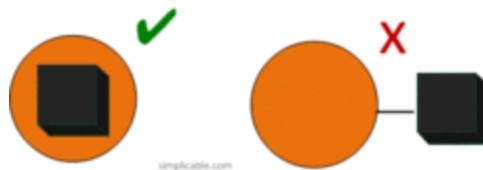


Fig. 26 Se debe evitar la dependencia con otros módulos o servicios

Un servicio una interface para realizar una funcionalidad de negocio concreta y específica. Estas

interfaces se relacionan una con otras en el contexto de “módulos” y contiene toda la información suficiente, sin necesidad de dependencias con otros módulos, como se presenta en la Fig. 26

2.2.9.3 Los servicios son interoperables



Fig. 27 Los servicios deben ser interoperables o capaces de trabajar con consumidores heterogéneos

Los servicios son una forma de comunicación entre integrantes de la organización sin importar la plataforma y el lenguaje utilizado dentro de cada integrante. Los servicios deben estar definidos bajo tecnologías abiertas e interoperables y soportar los protocolos de comunicación disponibles, como así también las estructuras de datos a utilizar en el intercambio de mensajes. La Fig. 27 presenta un servicio que puede ser consumido por clientes heterogéneos.

2.2.9.4 Los servicios poseen un bajo acoplamiento

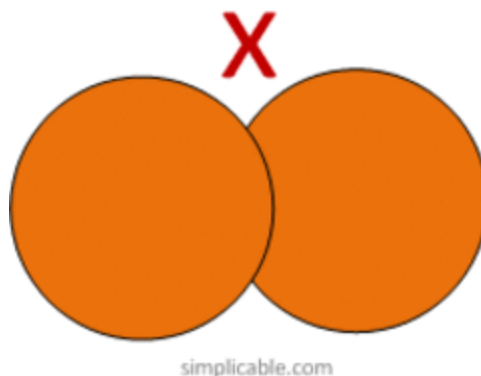


Fig. 28 Se debe minimizar el acoplamiento entre los servicios

El acoplamiento es una métrica que expone el nivel de dependencia entre módulos de software. Los módulos con bajo nivel de acoplamiento son flexibles y tienen dependencias bien definidas (y sobre todo, restringidas en su alcance). Módulos con un alto nivel de acoplamiento son más difíciles de configurar, ya que al agregar una dependencia entre módulos, estamos trayendo implícitamente todas las restricciones o configuraciones de ese módulo, lo que inevitablemente vuelve terminando haciendo más complejo al módulo en cuestión.

Una arquitectura orientada a servicios debe basarse definitivamente en un conjunto de servicios con un bajo nivel de acoplamiento. A través del Registro de Servicios se obtiene toda la información necesaria para invocar a un servicio.

2.2.9.5 Los servicios deben ser ubicados a través de una dirección de red



Fig. 29 Los servicios publican su contrato en un Registro

Un servicio debe hacer pública su interface siguiendo los principios dictados dentro de una arquitectura orientada a servicios. De esta manera el consumidor es capaz de invocar al servicio a través de un protocolo de red. Por lo tanto, en un momento dado puede haber más de un consumidor del servicio activo.

2.2.9.6 Los servicios poseen interfaces de granularidad gruesa

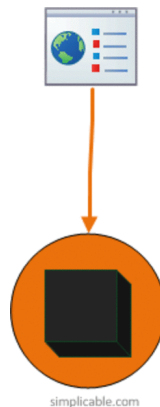


Fig. 30 Se debe promover la creación de interfaces de granularidad gruesa

La granularidad de una interface del servicio está definida en base a la funcionalidad de negocio que implementa. Si un método alcanza para implementar toda la funcionalidad de negocio, se dice que la granularidad es gruesa. En cambio si se necesitan varios métodos de una interface para implementar una funcionalidad de negocio, se dice que esta interface es de granularidad fina. Una arquitectura orientada a servicios soporta los dos tipos de interfaces. En lo posible se debe ir hacia interfaces de grano grueso como se ve en la Fig. 30. Debido a esta definición, dentro de una organización conviven interfaces con diferente nivel de granularidad.

2.2.9.7 La ubicación del servicio es transparente

SOA promueve dentro de su definición la transparencia en cuanto a la ubicación física del servicio. Esto es, hasta el momento de invocación el consumidor no sabe cuál es la ubicación física del servicio. La ubicación se resuelve cuando se obtiene la definición del servicio a través del Registro de Servicios. Por lo tanto, se puede transportar el servicio de una ubicación a otra sin afectar a los demás consumidores y en tiempo de ejecución.

2.2.9.8 Los servicios pueden componerse dentro de aplicaciones

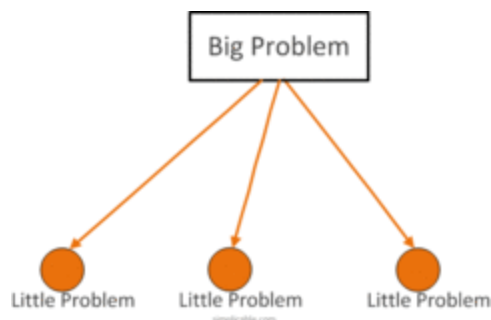


Fig. 31 Los servicios pueden ayudar a resolver problemas más complejos

Otras de las características principales de SOA es la habilidad de construir nuevas aplicaciones componiendo servicios existentes y por ende reusándolos.

La composición de servicios es más efectiva o se realiza con un menor esfuerzo, siempre y cuando se hayan respetado principios de bajo acoplamiento entre servicios. En definitiva termina siendo un factor preponderante, ya que es un mecanismo sencillo que aumenta el reúso de servicios y por ende, aumenta el éxito de una arquitectura orientada a servicios. La Fig. 31 muestra como se puede resolver un problema separándolo en invocaciones a servicios más pequeños.

2.2.10 Desarrollo de aplicaciones orientadas a servicios

La evolución natural de un componente de software es un servicio. El camino a seguir es tomar todos los métodos que ofrece un componente y definir una nueva unidad de trabajo, el servicio siempre teniendo en cuenta que se deberá realizar una tarea muy específica y concreta. Además los servicios pueden ser utilizados para ejecutar dos tipos de tareas: tareas técnicas o ejecutar actividades o funciones asociadas al negocio.

Además, un desarrollo orientado a servicios supone una evolución en la manera que se deberán implementar estos servicios. El contexto es más amplio (comparándolo con un desarrollo orientado a componentes reusables) y se deben considerar consumidores heterogéneos, independencia en la ubicación de los servicios y el potencial reúso del mismo. Para mitigar este esfuerzo se deben seguir estándares abiertos y cumplir con los principios de las arquitecturas orientadas a servicios.

Cada servicio debe ser implementado cumpliendo con los siguientes principios:

1. Cada servicio implementa una funcionalidad de negocio en particular, que debe estar asociada con una actividad de la vida real.
2. Un servicio puede exponer varias operaciones
3. Un servicio puede colaborar a través del envío de mensajes con otros servicios para satisfacer la función de negocio implementada.
4. Un servicio posee una interface bien definida y puede ser utilizado por cualquier tipo de clientes, ya sean otros servicios o aplicaciones.

2.2.11 Arquitectura SOA separada por capas (SOA Layered Architecture)

Los primeros sistemas estaban divididos en dos capas bien definidas, en la cual una capa era responsable de acceder a los datos guardados en una base de datos a través de algún protocolo de red. En este tipo de aplicaciones se caracterizan por no tener modelados funcionalidades de negocio, como también así la ausencia de un modelo lógico. Las aplicaciones definidas utilizando este tipo de arquitecturas presentan dificultades a la hora de aislar las responsabilidades de cada capa, lo que posteriormente impide exportar o reutilizar los componentes para que sean consumidos por otras aplicaciones.

Hoy en día el modelo arquitectónico de aplicaciones más común es el denominado de “tres capas” donde existe una capa de presentación, una de lógica de negocio y otra de acceso a datos. La capa de lógica de negocio es la que permite compartir lógica de negocio entre distintas aplicaciones, ya que en ella se modelan la funcionalidad de negocio que provee o implementa la aplicación.

SOA está basado en el desarrollo de aplicaciones con n-capas en las cuales, los servicios expuestos están ubicados en la primer capa de la aplicación. Estos son los responsables de exportar la funcionalidad de negocio que la aplicación implementa y exporta. También es la capa donde se implementan algunos conceptos técnicos como SLA, seguridad y trazabilidad.

Cada capa de una aplicación SOA tiene un objetivo específico. Las capas presentes en aplicaciones son las siguientes:

Operativa: contiene aplicaciones como CRM, ERP y aquellas dedicadas a realizar BI (Business Intelligence). También incluye a sistemas o aplicaciones orientadas a objetos. Estas aplicaciones son las que exponen los servicios nativos implementados por algún esquema propietario (cada aplicación posee su base de datos y posee sus estructuras propietarias). Dentro de un diagrama de arquitectura de referencia SOA a este conjunto de aplicaciones se le denomina “backend”.

Capa de componentes empresariales: componentes que implementan una funcionalidad común a todos los servicios. Son considerados activos de negocio, ya que a través de estos componentes se implementan o desarrollan los siguientes conceptos: administración centralizada, disponibilidad y balanceo de carga. Dentro del diagrama de arquitectura corporativa, estos componentes se denominan “activos de middleware”.

Capa de servicios: en esta capa se encuentran los servicios que las aplicaciones pueden invocar y a través de ellos implementar la funcionalidad de negocio o su objetivo en particular. Los servicios implementados son la realización de un descriptor o contrato y están disponibles para ser invocados a través de la red. Estos servicios suelen exponerse a través de un ESB para centralizar y administrar cada requerimiento no funcional (SLA) definido por el servicio.

Capa de proceso de negocio: los servicios pueden componerse a través del uso de técnicas como orquestación o coreografía, para implementar casos de uso o procesos de negocios más complejos y así reutilizar funcionalidad existente.

Concluyendo, SOA busca que los servicios puedan reutilizarse y componerse, siguiendo o validando los aspectos no funcionales (o técnicos) que debe proveer la plataforma (middleware) donde están implementados los servicios. Ruteo, disponibilidad, el cumplimiento de los contratos, seguridad, SLA y monitoreo de los servicios son algunos de aspectos técnicos que se deben tener en cuenta a la hora de diseñar una solución SOA dentro de una empresa.

2.3 Conclusión

SOA se podría considerar como una consecuencia de la evolución natural de los sistemas informáticos. Buscando adaptarse constantemente a los cambios propuestos por el negocio, define propiedades o restricciones sobre su unidad básica de trabajo, el servicio.

Las características de este paradigma involucran grandes cambios en las organizaciones. Estos cambios por lo tanto necesitan estar planeados de manera adecuada considerando la organización como un todo y donde se necesita de cada sector o grupo para llevar adelante las transformaciones de manera adecuada.

Las transformaciones generan nuevas estructuras o activos que son compartidos. Al ser compartidos, se deben promover objetivos y formas de utilización de estos activos. Surge la necesidad de gobernar los componentes tecnológicos y de negocio que posea la organización.

3 SOA Governance

3.1 ¿Qué se entiende por un Gobierno SOA?

El gobierno de SOA establece una serie de prácticas o estándares que administran los activos SOA de la organización o compañía.

En particular, el gobierno de SOA establece lineamientos para los siguientes puntos [15]:

1. Gestión de las autorizaciones para llevar a cabo una tarea. Cadena de mando bien definida.
2. Medición de las prácticas o procesos, para lograr una mayor efectividad en su ejecución
3. Políticas que guíen a la organización a su objetivo
4. Mecanismos de control para asegurar el cumplimiento de las políticas y lineamientos
5. Comunicación para mantener a todos los interesados informados.

En resumidas cuentas, a través del gobierno se define quién es el responsable de tomar las decisiones, qué decisiones se deben tomar y prácticas y políticas para que la implementación de esas políticas sea consistente. El Gobierno establece un plan y una serie de decisiones que deberán ser ejecutadas con el tiempo para la maduración de la plataforma. Se fijan las políticas y luego a través de la administración diaria de los objetivos, se la lleva adelante.

Aplicar un Gobierno IT, es básicamente, dictar las reglas con las cuales las personas, procesos, información y recursos disponibles serán coordinadas para cumplir los objetivos de negocio establecidos por la organización.

Tomando en cuenta un Gobierno IT, un gobierno SOA es una especialización, donde los activos principales a gobernar serán el ciclo de vida de servicios y procesos de negocio. La correcta administración del ciclo de vida es vital para el éxito de SOA dentro de una organización [4].

Un Gobierno SOA bien formado, se focalizará sobre los siguientes aspectos del ciclo de vida de los servicios:

1. Planning

Dentro de cada sector o área de negocio se ejecuta un relevamiento en busca de servicios candidatos. Con un criterio de selección, que puede ser el retorno de inversión o simplemente la necesidad existente de algún actor (ya sea externo o interno) se priorizan estos servicios candidatos para su desarrollo.

Un gobierno debe definir un mecanismo para seleccionar los servicios candidatos que se encuentren en la organización y priorizar aquellos que en base a algún criterio resulten los más prioritarios.

2. Publishing

Una vez desarrollados los servicios deben ser publicados para que cualquier potencial consumidor pueda utilizarlo. La publicación de un servicio generalmente involucra generar un contrato para que los clientes puedan accederlo y la definición de un SLA que el proveedor del servicio se compromete a cumplir.

Por lo tanto, un buen Gobierno SOA debe definir todas las etapas que tendrá un proceso de publicación de un servicio, para que los potenciales consumidores sepan tanto las especificaciones técnicas como la funcionalidad de negocio que implementa.

3. Discovery

Las aplicaciones o componentes de la organización deben ser capaces de ubicar los servicios. Esto generalmente se realiza a través de un catálogo de servicios corporativos, donde cada uno de los servicios de la organización se encuentra publicado.

Por lo tanto, un Gobierno SOA debe definir cómo los interesados en una funcionalidad de negocio en particular ubican al servicio que la implementa. Por esta razón se dicen que los clientes descubren los servicios corporativos.

4. Versioning

Los servicios publicados sufren cambios, mejoras o correcciones [3]. Estos cambios son los naturales a cualquier pieza de software que esté en funcionamiento. El versionado de servicios permite que los clientes del servicio puedan seguir funcionando sin necesidad de cesar su funcionamiento.

Por lo tanto, un Gobierno SOA debe establecer qué se entiende por versionado de servicios. Como cambiará el número de versión y definir cualquier tipo de activo, procedimiento o herramienta que ayude a los clientes a ser notificados del cambio de versión de un servicio y a planificar, en caso de ser necesario, la integración con su nueva versión.

5. Management

La administración de servicios es una actividad constante una vez que el servicio entra en producción [3]. Ayuda a detectar fallas y problemas en tiempo real, cuando es acompañado de un monitoreo activo de la salud del servicio. También este monitoreo suele vigilar el cumplimiento de los parámetros definidos en el SLA del servicios.

Por lo tanto, un Gobierno SOA deberá especificar qué cosas se deben monitorear de los servicios y cómo se notificará a los interesados cuando un servicio presente indicadores de salud que no sean satisfactorios. Con estos indicadores se podrá buscar la raíz del problema y ayudar a los responsables del servicio a corregir dicha implementación.

6. Security

Algunos servicios necesitan que existan ciertas restricciones a la hora de acceder a la funcionalidad que exponen. Esto previene que consumidores que no tengan autorización accedan a información o ejecuten ciertos procesos restringidos por las políticas de seguridad.

Un Gobierno SOA debe determinar las condiciones que dictarán que un servicio deba implementar algún tipo de restricción de acceso. También será responsable en definir un mecanismo de seguridad unificado para todos los servicios.

El Gobierno dentro de SOA es mucho más crítico que dentro de otros aspectos del Gobierno IT. SOA por naturaleza alienta a que diferentes aplicaciones, en silos de desarrollo o gerencias diferentes publiquen sus servicios de la forma que dicta el Gobierno. Por lo tanto se requiere de mucha coordinación y esfuerzo para lograr las alianzas necesarias, para administrar el Gobierno SOA de una organización. Este sigue siendo un gran desafío para SOA, tanto en organizaciones con sistemas monolíticos, como así también cuando ya encontramos los primeros componentes reutilizables dentro de la organización.

A medida que las organizaciones adoptan SOA para alinear el negocio con IT, pueden utilizar el Gobierno de SOA para mejorar el Gobierno IT. Utilizar un Gobierno SOA es clave para el éxito de una iniciativa SOA, ya que se deben gobernar aspectos técnicos y relacionados con el negocio.

3.2 Ejecutando un Gobierno SOA

En la práctica, el Gobierno SOA guía el desarrollo de servicios reutilizables, estableciendo cómo los servicios serán diseñados y desarrollados y cómo estos servicios podrán cambiar con el tiempo. Establece acuerdos entre los proveedores de servicios y los consumidores de esos servicios, diciendo a los consumidores lo que pueden esperar y los proveedores de lo que están obligados a proporcionar.

3.2.1 Dinámica de un Gobierno de SOA

El Gobierno de SOA no se limita solamente al diseño de los servicios, sino estableciendo políticas o definiciones sobre cómo esos servicios serán diseñados. Ayuda a responder a muchas cuestiones relacionadas con SOA:

- ¿Qué servicios están disponibles?
- ¿Quién puede utilizarlos?
- ¿Qué tan confiables son?
- ¿Por cuánto tiempo se dará soporte?
- ¿Se puede esperar que no cambien esos servicios?
- ¿Qué pasa si los servicios necesitan cambiar, como por ejemplo para corregir un error?
- ¿Si necesitan cambiar para añadir una nueva función?
- ¿Qué pasa si dos consumidores quieren que el mismo servicio trabaje de manera diferente?
- El proveedor del servicio, ¿está obligado dar soporte para siempre?
- Si alguien decide consumir un servicio, ¿puede estar seguro de que no se dará de baja mañana o en el corto plazo?

El gobierno de SOA se basa en las técnicas y prácticas existentes de gestión de IT. Un aspecto clave del Gobierno IT cuando se utilizan tecnologías orientadas a objetos como Java 2 Platform, Enterprise Edition (J2EE) es la reutilización de código. La reutilización de código también un ejemplo que expone las dificultades del Gobierno IT. Todo el mundo piensa que los activos reutilizables son buenos, pero en la práctica es común encontrarse con los siguientes desafíos:

- ¿Quién va a asumir el costo de su desarrollo?
- ¿Los equipos de desarrollo se comprometerán a usarlos? ¿En reusarlos?
- ¿Todo mundo puede realmente puede utilizar una única versión del componente? ¿O cada uno tendrá su propia versión del componente?

El impacto que tiene cada una de estas preguntas sobre un ecosistema SOA es lo que hace que el Gobierno SOA tenga en cuenta estos interrogantes y sus consecuencias en la organización.

El Gobierno es más un problema político que tecnológico o empresarial. Tecnología se centra en la coincidencia de las interfaces y protocolos de invocación. El negocio se enfoca en la funcionalidad necesaria para atender a los clientes. Tecnología y negocios se centran en las necesidades. Mientras el gobierno se involucra en estos aspectos, pero se centra más en asegurar que todos trabajen juntos y que los esfuerzos por separado no se contradigan. El Gobierno no determina cuáles son los resultados de las decisiones, pero si define qué decisiones se deben tomar y quién debe ejecutarlas.

Las dos partes, los consumidores y los proveedores, tienen que ponerse de acuerdo sobre la forma de trabajo en conjunto. Como resultado, puede generarse un acuerdo de nivel de servicio (SLA), que no es otra cosa que un contrato que define lo que un proveedor de servicios se compromete a cumplir y lo que un consumidor de servicios se compromete está dispuesto a tolerar. Este acuerdo es como un contrato entre las partes, y puede, de hecho, ser un contrato legal. Por lo menos, el SLA articula lo que el proveedor debe hacer y lo que el consumidor puede esperar.

El Gobierno de SOA es llevado adelante por un centro de excelencia SOA (COE), una junta de profesionales con conocimientos de SOA que establecen y supervisan las políticas para asegurar el éxito de la organización que está adoptando o utilizando SOA. El COE establece las políticas para la identificación y desarrollo de los servicios, definición de SLAs, gestión de registros de servicios, y otras iniciativas que ayudan a lograr un gobierno más eficaz. Luego, los miembros del COE, ponen en práctica esas políticas, asistiendo y orientando a los equipos de desarrollo de servicios compuestos.

Una vez que el COE gobierno elabora las políticas, se pueden utilizar herramientas tecnológicas para administrar esas políticas. Estas herramientas no definen un SLA, si no que son utilizadas para ejecutar y medir el cumplimiento de los contratos asumidos. Por ejemplo, se pueden limitar la cantidad de consumidores en un momento dado. También se puede advertir al consumidor de que el servicio ha quedado en desuso. Como así también medir la disponibilidad del servicio y tiempo de respuesta.

A través de la combinación de un bus de servicios empresariales (ESB) y un registro de servicios se hacen cumplir las políticas de gobierno SOA, al ser puntos neurálgicos y centrales de una arquitectura SOA. Un servicio puede estar expuesto de modo que sólo ciertos ESB puedan invocarlo. Entonces la combinación ESB / registro puede ayudar a controlar el acceso de los consumidores a los servicios, a monitorear su uso y de validar el cumplimiento de los SLA entre otras cosas. De esta manera, el servicio

mantiene en enfoque exclusivo sobre la función o procedimiento que busca implementar y el par ESB / registro se enfocan sobre aspectos relevantes para el Gobierno SOA.

Al momento de los problemas, el Gobierno SOA (a través de sus herramientas) puede convertirse en un chivo expiatorio. A menudo ocurren situaciones donde el Gobierno SOA es el primer culpable para cada problema y una justificación para cada solución que se quiere llevar a cabo de manera unilateral. Un desafío para el Gobierno SOA es utilizar con mucho juicio o cuidado cada una de las herramientas o prácticas que posee.

El Gobierno como disciplina no es un término nuevo en lo que respecta a IT, pero con la llegada de SOA obligó a las empresas a tomar de manera más seria el Gobierno SOA.

La definición de un Gobierno SOA siempre se ve afectada por una serie de factores comunes en la vida de una organización:

1. Proveedores de software
2. Organismos de control y auditoría
3. Analistas externos
4. Opiniones de referentes externos

La variedad de las opiniones puede hacer tambalear a un Gobierno SOA sin un plan de trabajo y los objetivos bien claros.

Como las obligaciones de un Gobierno SOA en la práctica deben extenderse un poco más allá de la definición del ciclo de vida de los servicios, usualmente un Gobierno SOA termina teniendo cierto nivel de influencia sobre los siguientes Gobiernos:

1. Gobierno de Aplicaciones Empresariales (modificando u obligando a adaptar la Arquitectura de Referencia de Aplicaciones)
2. Gobierno de Procesos de Negocio (coordinando tareas sobre cómo integrar procesos de negocios y servicios)
3. Gobierno IT, dictando el uso de frameworks y nuevas tecnologías.

3.2.2 Alcance de un Gobierno SOA

Aunque SOA en base es un conjunto de prácticas de perfil tecnológico, existe la tentación de querer resolver todo el Gobierno SOA a través de la tecnología. Un Gobierno SOA que busque el éxito deberá también focalizarse sobre personas, procesos, activos y aspectos tecnológicos de una solución SOA en una organización. La amplitud de la definición, hace de la tarea de definición de Gobierno, una tarea desafiante.

El gobierno de SOA debe extenderse a los Gobiernos IT y de Aplicaciones Empresariales para coordinar las fuerzas de trabajo y evitar el trabajo por cuenta propia, aislado del contexto y de la iniciativa SOA que se esté ejecutando dentro de la organización.

Esta ampliación de responsabilidades es todo un reto desde el punto de vista político, ya que se deben coordinar esfuerzos y tiempo entre áreas que no tienen puntos en común.

3.2.3 Framework de Gobierno SOA

El objetivo de un Framework de Gobierno SOA es permitir que las organizaciones definan y ejecuten una instancia personalizada de un modelo de Gobierno SOA.

El modelo de Gobierno SOA requiere cambios culturales dentro de una organización. Por lo tanto, al momento de definir un régimen de Gobierno SOA deben evitarse enfoques de big-bang o creación absoluta de cada una de las políticas que serán aplicadas dentro del Gobierno. El framework de Gobierno SOA define un método de implementación incremental para que las organizaciones puedan seguir cumpliendo sus demandas mientras avanzan hacia sus metas de largo plazo para SOA.

No existe un modelo único de “buen Gobierno SOA” debido a las particularidades que se dan dentro de cada organización. Ejemplo de alguna de estas variantes son pueden ser:

1. Gobiernos existentes en la organización
2. Nivel de madurez de SOA
3. Tamaño de la organización

Un modelo de Gobierno SOA adecuado para una organización es el que define:

1. Qué decisiones deben tomarse en su organización para tener una Gobierno SOA efectivo
2. Quién debe tomar estas decisiones de Gobierno SOA en la organización
3. Cómo serán monitoreadas estas decisiones sobre el Gobierno SOA de la organización (medir los resultados)
4. Qué estructuras, procesos y herramientas deben ser instaladas en la organización
5. Qué indicadores son necesarios para asegurar que iniciativa SOA de la organización se ajusta a los objetivos estratégicos de la organización

Las organizaciones deben evaluar sus regímenes de Gobierno actuales de manera franca analizando el estado actual contra los objetivos planteados. A partir de esto, generar un roadmap de trabajo para garantizar un correcto Gobierno de los activos.

El Framework de Gobierno SOA se compone de un Modelo de Referencia de Gobierno SOA (SGRM) que se utiliza como punto de partida y un método de feedback y mejora de calidad continua entre iteraciones denominado SGVM (SOA Governance Vitality Method)

3.2.4 SOA Governance Reference Model (SGRM)

Es un modelo que establece las bases y es utilizado en el proceso de personalización del Gobierno SOA de la organización. Todos los puntos o aspectos que define el SGRM deberán ser evaluados contra la realidad de la organización y / o adaptados a ella.

3.2.5 SOA Governance Vitality Method (SGVM)

Es un proceso que comienza al mismo tiempo que SGRM y logra a través de una separación en fases, adaptar las actividades de la organización al Gobierno SOA. En el contexto de un Gobierno SOA, SGVM puede verse también como un ciclo de mejora continua, donde el progreso es medido y el camino adoptado dentro del Gobierno SOA puede ser modificado o corregido si fuera necesario.

3.3 Conclusión

El Gobierno sobre activos tecnológicos de una organización es un factor determinante para el éxito que pueda tener cualquier iniciativa. En particular, los desafíos para llevar adelante un Gobierno SOA traen a discusión el concepto de “puja de poder” o “fuerzas políticas” dentro de la organización.

Al no existir un modelo de Gobierno SOA único, si no que más bien, cada Gobierno debe tener la capacidad de adaptarse a la realidad de la organización, existen lineamientos o frameworks que dictan cómo se debería comportar un Gobierno SOA, pero en la práctica son los propios integrantes de este Gobierno que deberán en algunos casos ceder y en otros ganas para llevar adelante la iniciativa.

El aspecto político de una adopción SOA es un factor que propone transformaciones que pueden obligar a los sectores de una organización a llevar adelante tareas a la que no estaban acostumbrados o que no consideran prioritarias.

4 Esquemas de adopción SOA

4.1 OSIMM

4.1.1 Contexto y descripción inicial

El “Open Group Service Integration Maturity Model” (OSIMM) especifica cómo medir el nivel de integración de los servicios en una organización, tanto en los sistemas IT como así también en sus aplicaciones de negocio. Además sirve de guía para lograr una mayor madurez en lo que a servicios respecta, siguiendo los objetivos planteados desde el negocio.

Este modelo presenta siete dimensiones con siete niveles de madurez. Cada nivel de madurez representa un avance significativo en la madurez necesaria para llegar a la orientación a servicios. Dentro de OSIMM este concepto se denomina “madurez de los servicios de integración”. A su vez, OSIMM puede denominarse “modelo de madurez SOA”. El modelo OSIMM es inclusivo y extensible. Esto se basa en que no busca solamente lograr una orientación a servicios en la organización, sino que también busca incluir nuevas tecnologías como cloud computing y a través de OSIMM Framework, proveer los mecanismos para que este modelo pueda ser aumentado cuando la ocasión lo requiera. El modelo OSIMM define un conjunto de dimensiones, que son en realidad vistas de diferentes conceptos de una organización:

1. Business
2. Organization & Governance
3. Method
4. Application
5. Architecture
6. Information
7. Infrastructure & Management

Además los niveles de madurez que define son los siguientes:

1. Silo
2. Integrated
3. Componentized
4. Service
5. Composite Services
6. Virtualized Services
7. Dynamically Re-Configurable Services

Cada nivel de madurez presenta una serie de indicadores que determina en el caso que se esté evaluando una dimensión, el estado actual de la organización en ese aspecto. La matriz, una vez completado en análisis, es representativa del estado actual de la organización en cuanto a la orientación a servicios.

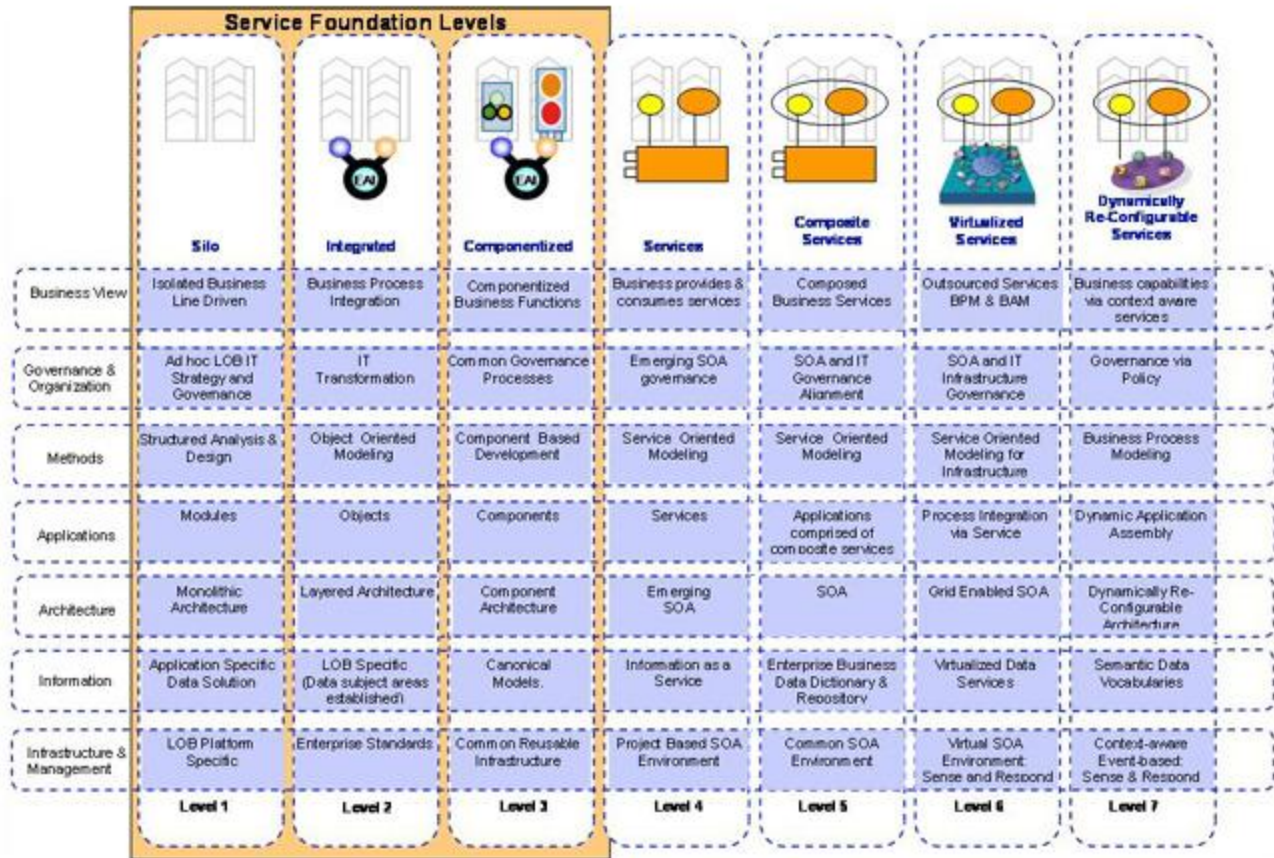


Fig. 32 Matriz de adopción SOA de OSIMM

Las columnas de la matriz corresponden a los distintos niveles de madurez. Las filas a su vez representan las dimensiones. Cada celda de esta matriz define un nivel de madurez para una dimensión a través de objetivos a cumplir. La madurez SOA de una organización se identifica al evaluar cada una de las celdas.

4.1.2 Niveles de madurez

Anteriormente se hizo mención a los niveles de madurez OSIMM de una organización, que definen el estado de la organización con respecto al estado de los servicios de integración y la madurez SOA de la organización. Cada nivel es una acumulación de objetivos o características que la organización debe cumplir. Por lo tanto, la maduración es un factor progresivo a medida que se aumenta el nivel de madurez OSIMM.

A continuación se describen cada uno de los niveles propuestos por OSIMM:

4.1.2.1 Nivel 1: Silo

Cada grupo de trabajo en la organización está desarrollando sus sistemas de manera independiente, sin procesos, datos, lineamientos compartidos o estándares para la organización. Esto limita la posibilidad de implementar procesos de negocios que requieran la intervención de varias partes, ya que la heterogeneidad tecnológica es un factor limitante. Los procesos de negocio se implementan con mucha intervención manual, ya sea transformando o reinterpretando los datos generados entre aplicaciones.

4.1.2.2 Nivel 2: Integrated

Existe cierto nivel de coordinación tecnológica dentro de la organización. Esto permite la comunicación entre distintos silos utilizando metodologías o estándares compartidos. En este contexto, la intercomunicación entre silos es posible. Sin embargo, la estandarización técnica todavía no alcanza a los datos manejados por los procesos de negocio. Con lo cual, la intercomunicación de distintos silos, frecuentemente involucra conversiones o adaptaciones a los datos. Con lo cual para implementar cada una de las integraciones dentro de distintos sistemas, se generan nuevas piezas de software que realizan la tarea de adaptar la información, que luego se vuelve difícil de mantener o administrar. Lo que termina limitando la reutilización de estas integraciones y por ende, la creación de nuevos procesos de negocio.

4.1.2.3 Nivel 3: Componentized

Los sistemas existentes dentro de cada uno de los silos, han sido analizados y separados en componentes reusables. Toda esta tarea se ejecuta con la ayuda de un framework de desarrollo de aplicaciones. Dentro de este framework que dentro del framework estén implementadas algunas funciones de negocio. Estos componentes todavía presentan altos niveles de acoplamiento, lo que dificulta la madurez y la interoperabilidad entre distintas partes de la organización. Esto lleva a que el desarrollo de aplicaciones que utilicen estos componentes sea más complejo. Lo que finalmente provoca es la replicación y redundancia de piezas de software que realizan la misma tarea dentro de la organización.

4.1.2.4 Nivel 4: Service

Las aplicaciones se van creando en base a servicios y presentando un nulo o bajo acoplamiento entre las partes. Los servicios se invocan de manera estándar y definida y este mecanismo es independiente de la tecnología presente en cada una de las aplicaciones. Estos servicios corren sobre una infraestructura que soporta los protocolos de comunicación definidos, los mecanismos de seguridad, la transformación de datos y la administración de servicios. Por lo tanto, los servicios son invocados desde cualquier punto de la organización y en ocasiones desde terceras partes (actores externos) y se administra asignando responsabilidades definidas en el contrato del servicio (Service Level Agreements) a cada segmento de la organización. La funcionalidad de negocio ha sido analizada en detalle y posteriormente separada en servicios que se encuentran dentro de una arquitectura de negocio que asegura que estos servicios serán interoperables a nivel de negocio. Además estos servicios pueden ser definidos utilizando algún lenguaje de especificación como WSDL o SCA que elimine toda ambigüedad de la operación ejecutada por el servicio, colaborando así con la construcción del catálogo o registro de servicios. A este nivel de madurez es posible la composición de servicios en nuevos de mayor valor agregado, pero esta tarea a menudo se sigue realizando a través de código fuente -con desarrolladores

trabajando explícitamente en esa construcción- y no se utilizan que modelen el flujo de un proceso de negocio de manera declarativa. Esto atenta contra la agilidad y el desarrollo de nuevos procesos de negocio como servicios de la organización.

4.1.2.5 Nivel 5: Composite Services

En este nivel de madurez de servicios es posible construir procesos de negocio en base a servicios existentes utilizando lenguajes de modelado de procesos de negocio como BPEL. También se pueden utilizar la clásica composición a nivel de servicios para crear nuevos servicios de mayor valor agregado para la organización. Estos nuevos servicios pueden ser calificados o categorizados como: estáticos, de negocio o de actividad. Esta calificación facilita su composición formando así nuevos procesos de negocio con poca cantidad de código. Además el diseño y desarrollo de nuevos servicios ya es un proceso ágil y dinámico.

4.1.2.6 Nivel 6: Virtualized Services

Los servicios técnicos y de negocio son provistos a través de una interface virtual, un nivel de indirección más. El consumidor no invoca al servicio directamente, sino que utiliza un “servicio virtual”. La infraestructura donde corre este servicio realiza la tarea de convertir esa invocación virtual en una llamada real, cambiando la dirección física, el protocolo, el patrón de sincronismo utilizado para la invocación del servicio, como así también la transformación de datos en caso de ser necesario. Esto provoca que el servicio virtual tenga aún menos acoplamiento de la infraestructura en la cual está corriendo. Esto aumenta las posibilidades de reuso o del servicio a través de la composición.

4.1.2.7 Nivel 7: Dynamically Re-Configurable Services

Antes de completar este nivel las interacciones de los procesos de negocio son definidas en tiempo de diseño en un trabajo realizado de manera conjunta por desarrolladores y analistas de negocio utilizando una herramienta adecuada.

La finalización de esta implica que el diseño y modificación de estos procesos de negocio puede ser realizada en tiempo de ejecución sin necesidad de dar de baja el servicio. Para lograr esto, se requiere acceso al Repositorio de Servicios para ubicar cada uno de los servicios necesarios para implementar la funcionalidad o proceso de negocio requerido. El repositorio debe ser capaz de localizar la mejor instancia del servicio, en base a los requisitos o especificaciones definidos.

4.1.3 Dimensiones

El nivel de madurez SOA de una organización puede ser evaluada a través de un conjunto de dimensiones que son indicadores esenciales para una correcta adopción SOA.

4.1.3.1 Business

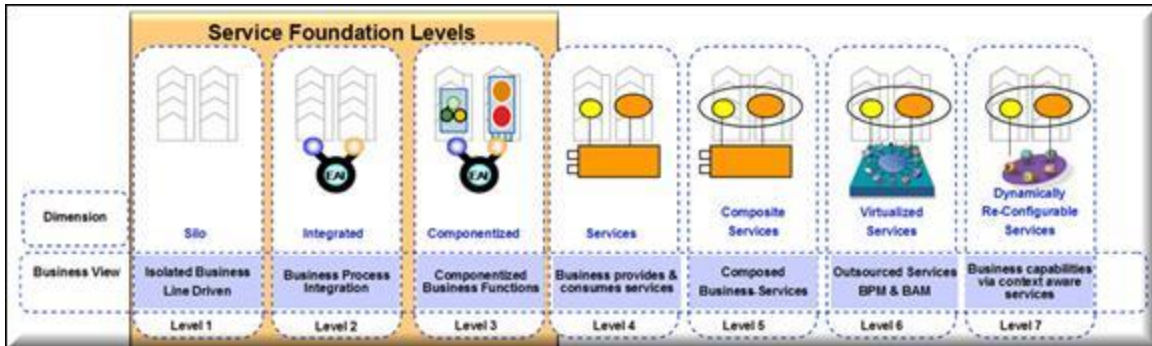


Fig. 33 - Dimensión business en la matriz de adopción SOA de OSIMM

La dimensión de negocio (Fig. 33) está centrada en la arquitectura de negocio, esto es: el estado actual de las prácticas y políticas de negocio de la organización, como los procesos de negocio son diseñados, estructurados, implementados y ejecutados. La dimensión de negocio también se encarga de distribuir el costo de aumentar la capacidades de la estructura IT de la organización y cómo esas capacidades deben soportar las características del negocio: flexibilidad, agilidad, SLA, etc. La dimensión de negocio también incluye la estrategia IT de la organización. Y se encarga de determinar el valor agregado que le dará a la organización avanzar hacia un nivel de madurez mayor en la matriz de adopción.

4.1.3.2 Organization & Governance

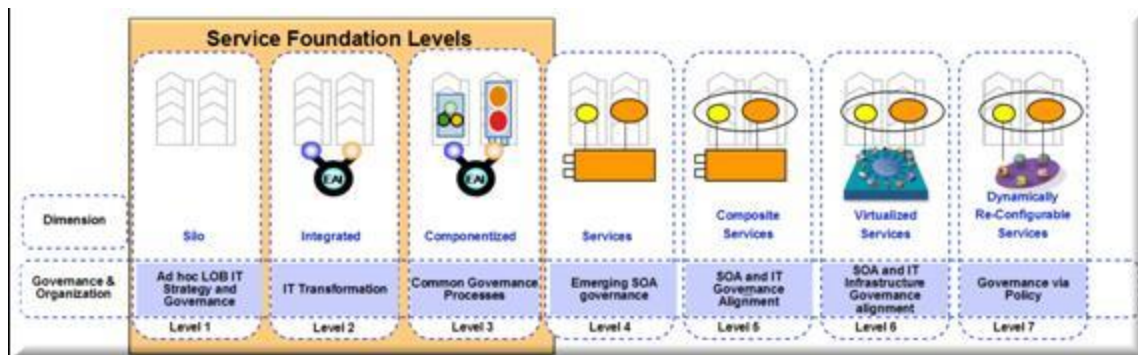


Fig. 34 - Dimensión Organization & Governance en la matriz de adopción SOA de OSIMM

Esta dimensión (Fig. 34) se enfoca sobre el diseño y la estructura de la organización en sí y sobre las medidas que se deberán tomar para aumentar la efectividad de la organización en el contexto de SOA y el Gobierno de SOA.

Se debe tener en cuenta todo lo que corresponda a la organización como estructura jerárquica, relaciones, roles y los cambios necesarios que se deberán ejecutar en miras de adoptar una estrategia de orientación a servicios. Para esto se debe evaluar las habilidades existentes dentro de la organización y la capacidad de formar el personal para encarar este proceso. Por otro el Gobierno está asociado con la administración formal de los procesos de la organización, para mantener las actividades, capacidades de servicios y soluciones SOA alineadas con los requerimientos del negocio. El Gobierno también influye en la forma de administrar los recursos y cómo y cuándo gestionar los gastos.

4.1.3.3 Method

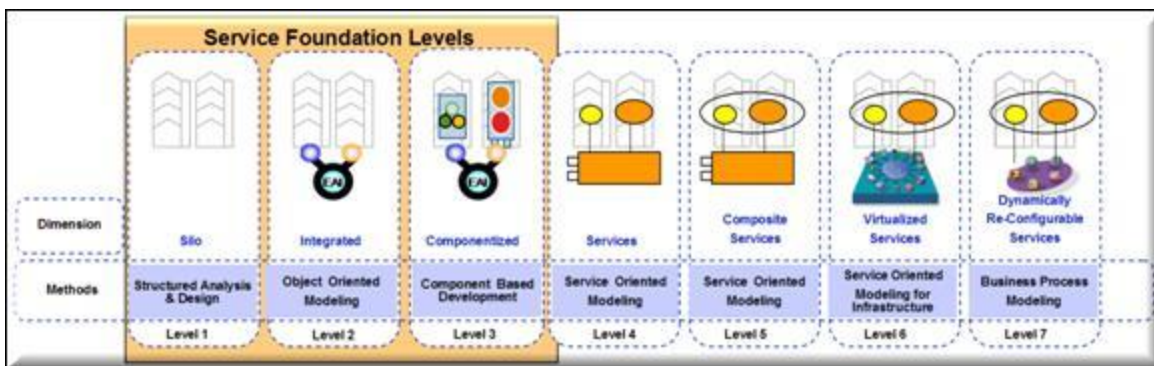


Fig. 35 - Dimensión Method en la matriz de adopción SOA de OSIMM

Esta dimensión (Fig. 35) se centra sobre los métodos y procesos utilizados en la organización para las evoluciones técnicas y de negocio. También sobre la madurez de la organización en lo que respecta al ciclo de vida del software: administración de requerimientos, técnicas de estimación, administración de proyectos, procesos de validación de calidad y las técnicas de diseño de software, como así también las herramientas utilizadas para cumplir con estos objetivos.

4.1.3.4 Application

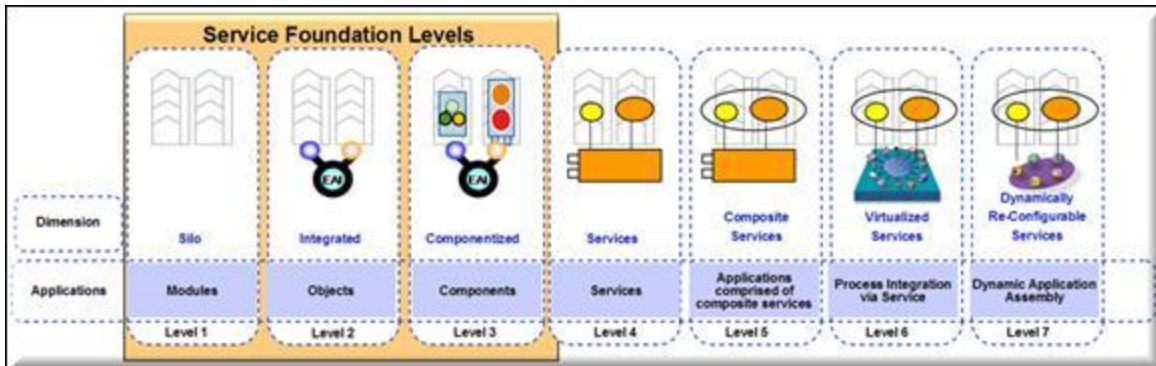


Fig. 36 - Dimensión Application en la matriz de adopción SOA de OSIMM

Esta dimensión (Fig. 36) pone el foco sobre características de las aplicaciones de la organización. Cómo están estructuradas, como se descomponen funcionalmente, reusabilidad, flexibilidad, disponibilidad y extensibilidad. Tratando de llevar a un concepto uniforme, definiendo y utilizando buenas prácticas diseño y desarrollo.

4.1.3.5 Architecture

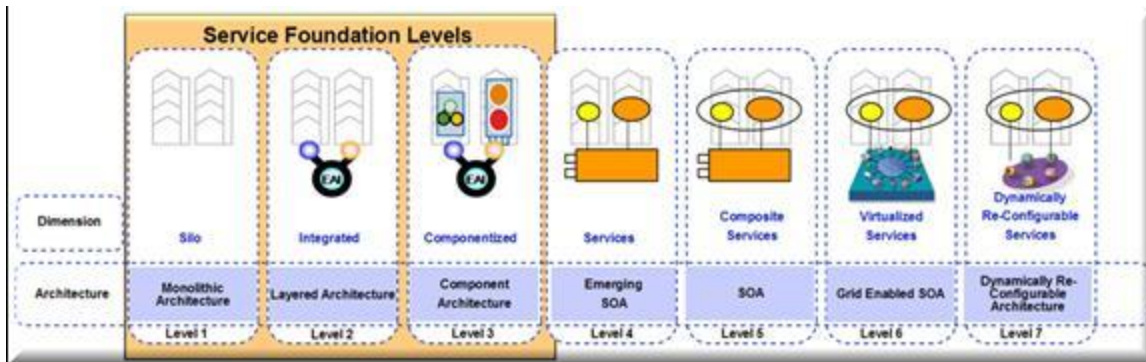


Fig. 37 - Dimensión Architecture en la matriz de adopción SOA de OSIMM

Esta dimensión (Fig. 37) se enfoca sobre la estructura de la arquitectura de la organización. Esto incluye conceptos como topología (cómo están distribuidas aplicaciones por su tipo), técnicas de integración de aplicaciones, decisiones sobre la arquitectura de la organización, definición de estándares y políticas, nivel de adopción de Web Services, experiencias en implementaciones SOA y el criterio de aceptación SOA.

4.1.3.6 Information

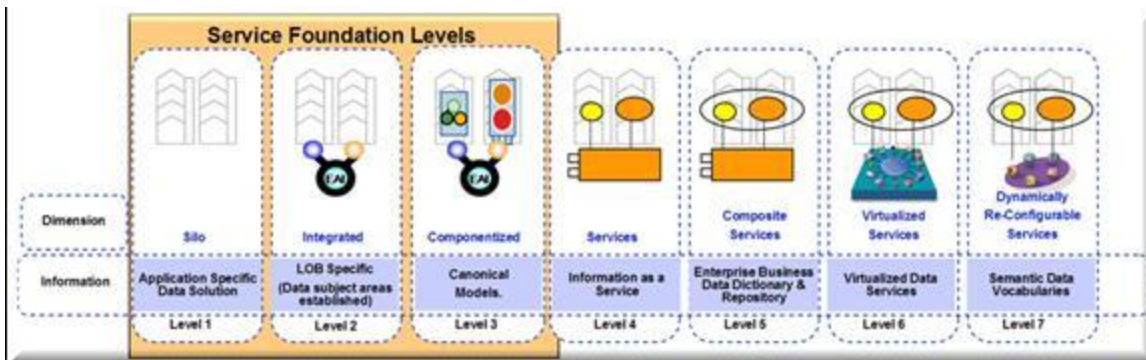


Fig. 38 - Dimensión Information en la matriz de adopción SOA de OSIMM

En esta dimensión (Fig. 38) se define como se estructura y modela la información, los métodos de acceso a los datos corporativos, abstracción del acceso a datos desde el punto de vista funcional, características de los datos, capacidades para la transformación de los datos, definiciones de servicios y procesos, manejo de identificadores, credenciales de seguridad, administración del conocimiento, modelo de información corporativo y administración de contenidos.

4.1.3.7 Infrastructure & Management

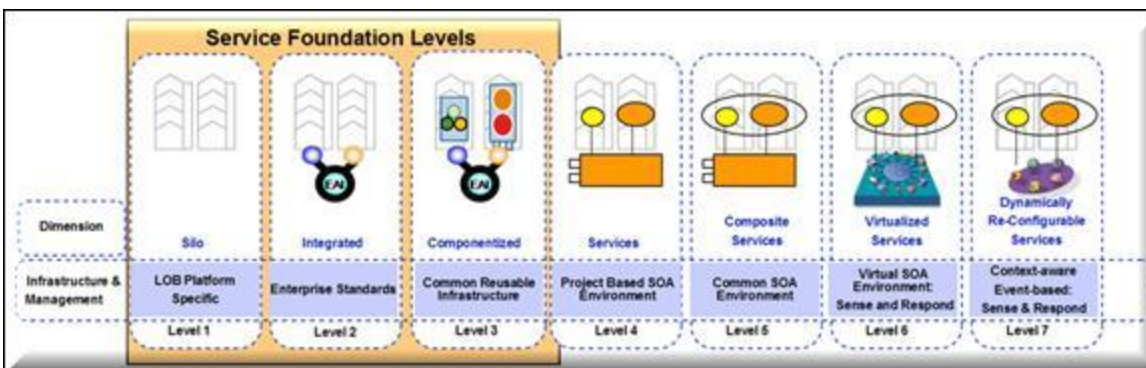


Fig. 39 - Dimensión Infrastructure & Management en la matriz de adopción SOA de OSIMM

Capacidad de la infraestructura corporativa, administración de los servicios, operaciones y administración de la infraestructura tecnológica. Cómo se cumplen con los SLAs establecidos, como se monitorean los servicios y qué tipos de plataformas de integración estarán disponibles dentro de la organización.

4.1.4 Service Foundation Levels

Los tres primeros niveles del modelo de madurez OSIMM se denominan “Service Foundation Levels”. Esta denominación se debe a que generalmente en esta etapa se establecen las bases para el resto de la adopción SOA. En esta etapa se transforman en servicios varias aplicaciones “legacy” o ya asentadas

en la organización. También se suele exponer a través de servicios, aplicaciones cuyo acceso esté restringido por la dificultad de la comunicación que existe entre aplicaciones (por ejemplo, componentes de comunicación punto a punto a través de sockets).

4.1.5 Cuestionario del Assessment e indicadores de madurez por dimensión

Para evaluar el estado actual de la organización con respecto a SOA, se utiliza un cuestionario que a través de cada una de sus preguntas nos ayuda a conocer el estado de la organización.

4.1.5.1 Preguntas e indicadores de madurez por dimensión

El estado actual de los servicios de negocio y servicios técnicos (IT) de la organización son evaluados a través de una serie de indicadores o preguntas de madurez. El modelo base de OSIMM incluye un conjunto de preguntas (Fig.40) que puede ser utilizado tal cual, o extendido para obtener una definición más precisa de la madurez corporativa en lo que a integración a través de servicios respecta.

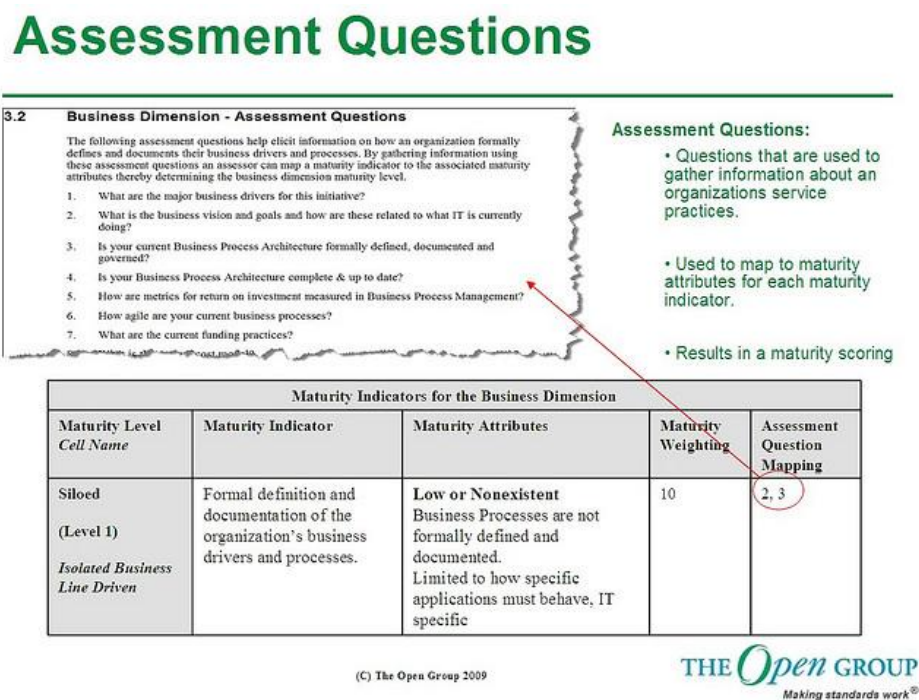


Fig. 40 - Ejemplo que muestra la relación entre el cuestionario y la evaluación de impacto en el assessment

4.1.5.2 Cuestionario sobre madurez de servicios

Las preguntas del cuestionario se utilizar para seleccionar los atributos de madurez de los servicios que se asocian con cada dimensión de madurez. El modelo OSIMM utiliza este cuestionario para conducir un estudio sobre los responsables de la definición de servicios tanto en la parte IT como en la parte de negocio. Se puede ejecutar un relevamiento sobre el siguiente grupo de personas para determinar qué nivel de madurez existe en la organización:

1. Equipo de infraestructura IT
2. Equipo de desarrollo de servicios
3. Grupo de desarrollo que da soporte a un área de negocio
4. Arquitectos corporativos
5. CIO o CTO de la organización

4.1.5.3 Asignación de indicadores de madurez a preguntas del cuestionario

Las preguntas de la evaluación del estado actual mantienen una relación con los atributos de madurez para cada indicador de madurez de una dimensión. A través de estas preguntas se facilita el proceso de detección de información relevante para establecer el nivel de madurez de los servicios. Las preguntas se clasifican por dimensión.

4.1.5.4 Business Dimension: Base Model

1. ¿Cuáles son los mayores promotores desde el punto de vista del negocio para esta iniciativa?
2. ¿Cuál es la visión y metas de negocio y cómo se relacionan estos puntos con las tareas que se están desarrollando desde IT?
3. ¿Existe una Arquitectura de Procesos de Negocio Corporativa, definida formalmente, documentada y gobernada?
4. ¿Está la Arquitectura de Procesos de Negocio completa y actualizada?
5. ¿Cómo se mide el retorno de inversión en la Administración de Procesos de Negocio (BPM)?
6. ¿Qué tan ágiles son los procesos de negocio actuales?
7. ¿Cómo se financia un proyecto?
8. ¿Cuál es el modelo de costos?
9. ¿Quién es el dueño de procesos, aplicaciones y servicios?
10. ¿Dentro del modelo de costos existente, es posible cobrar a los consumidores de los servicios por el uso de los mismos?
11. ¿Cómo se define el costo total de estos activos? (incluyendo software, hardware y mantenimiento a futuro)?
12. ¿Qué nivel de cooperación existe entre los promotores IT y los promotores de negocio?
13. ¿Cómo se mide actualmente el nivel de los servicios de negocio?
14. ¿Cuál es el mecanismo para trasladar los contratos asumidos por el negocio (SLA) a contratos técnicos (SLA técnico)?

15. ¿Existe una arquitectura de referencia corporativa?
16. ¿Existe un gobierno formal sobre la arquitectura de referencia corporativa?
17. ¿Existen varias líneas de negocio diferentes? ¿Cada una necesita o define sus propios procesos de negocio?
18. ¿Las líneas de trabajo, manejan un modelo común de información? ¿Existen representaciones compartidas? ¿Está replicado ese modelo?
19. ¿Comparten proveedores, clientes o partners estas líneas de trabajo?

4.1.5.5 Organization & Governance Dimension: Assessment Questions

1. ¿Cuáles son los talentos más comunes en el personal IT de la organización?
2. ¿Cómo se relaciona el Gobierno IT con la iniciativa SOA?
3. ¿De qué manera está alineado el Gobierno IT con SOA, la Arquitectura Corporativa o el Gobierno de la organización?
4. ¿Los procesos del Gobierno SOA existen? ¿Están documentados? ¿Se utilizan para trabajar con servicios, tanto en tiempo de diseño como en tiempo de ejecución?
5. ¿Cada uno de las organizaciones involucradas dentro del proceso SOA tiene roles y responsabilidades bien definidas?
6. ¿Cuáles son las responsabilidades y funciones de gobierno?
7. ¿Cómo describía al esquema de costos IT?
8. ¿Está disponible algún tipo de capacitación SOA dentro de la organización? ¿De qué tipo?
9. ¿Cómo es la relación entre los equipos de desarrollo e infraestructura dentro de la organización?
10. ¿Existe alguna autoridad SOA dentro de la organización? ¿Existe alguna autoridad de Gobierno IT?
11. ¿La solución SOA sólo afecta a actores internos a la organización? ¿Afecta a terceros?

4.1.5.6 Method Dimension: Assessment Questions

1. ¿Cómo se administran actualmente los requerimientos en los sistemas de software de la organización?
2. ¿Cuáles son las metodologías y buenas prácticas utilizadas o que se están adoptando?
3. ¿Se utilizan algunas de las prácticas de diseño SOA?
4. ¿Qué herramientas de diseño se utilizan?

5. ¿Cuál es la práctica actual para el diseño y administración de servicios?
6. ¿Cómo se administran actualmente los proyectos?
7. ¿Cómo está organizada la administración de proyectos IT?
8. ¿Existen procesos de calidad en la organización? ¿Cómo están organizados estos procesos?
9. ¿Existe una comunidad dentro de la organización dedicada a mejorar las prácticas SOA?
10. ¿Existe un repositorio para almacenar los activos y buenas prácticas definidas dentro de la organización?

4.1.5.7 Application Dimension: Assessment Questions

1. ¿Qué estilo de aplicaciones se utilizan / desarrollan dentro de la organización?
2. ¿Qué tan común es el reúso dentro de la organización?
3. ¿Qué tipo de reúso de activos se promueve y como se mide la reusabilidad efectiva?
4. ¿Cómo se integran las aplicaciones o sistemas dentro de la organización?
5. ¿Qué tipo de lenguajes de programación se utilizan dentro de la organización?
6. ¿Qué tipo de técnicas de integración se han utilizado dentro de la organización?
7. ¿Cómo se representa la lógica de negocio dentro de las aplicaciones de la organización?
8. ¿Qué tan confiables son las aplicaciones de negocio críticas para la organización?
9. ¿Qué tan utilizado es XML dentro de la organización? ¿Cuán complejo es el uso que se le da?
10. ¿Cuál es la tasa de cambios y el time-to-market de las aplicaciones de la organización?
11. ¿Se utiliza alguna tecnología asociada con SOA como ESB, datos compartidos?

4.1.5.8 Architecture Dimension: Assessment Questions

1. ¿Cómo definiría a la topología arquitectónica de software?
2. ¿Qué tipos de repositorios de datos se utilizan en la organización?
3. ¿Cuál es el mecanismo estándar de comunicación en la arquitectura de referencia de la organización?
4. ¿Cómo se logra una integración en la arquitectura actual?
5. ¿Qué métodos se utilizan en el desarrollo de arquitectura?
6. ¿Qué tan maduras están las implementaciones de servicios?
7. ¿Qué tan extensiva es la adopción de SOA?

8. ¿Qué principios arquitectónicos o lineamientos están definidos?
9. ¿Qué tan común y sofisticado es el uso de frameworks de desarrollo en la arquitectura de la organización?
10. ¿Cómo se toman las decisiones de arquitectura en la organización?
11. ¿Se utilizan arquitecturas de referencia en la organización?

4.1.5.9 Information Dimension: Assessment Questions

1. ¿Existe un modelo de datos común a todas las aplicaciones?
2. ¿Existen modelos de datos independientes para diferentes aplicaciones?
3. ¿Se utilizan reglas para convertir los datos desde un modelo al otro?
4. ¿Existen dificultades en el traslado de información de una aplicación a otra? ¿Para todas las aplicaciones? ¿Para algunas?
5. Si existe un modelo de datos común (ya sea corporativo o a través de conversiones o mapeos) ¿cómo está definido este modelo? ¿Programando aplicaciones o componentes? ¿A través de esquemas XSD? ¿Formalmente definido en documentos? ¿A través de herramientas de modelado?
6. Los modelos de datos representados como entidades corporativas o objetos de negocio, ¿son activos corporativos entendidos y administrados por el negocio y utilizados por IT? ¿son activos corporativos entendidos, administrados y utilizados sólo por IT?
7. Si existen reglas de transformación de datos entre distintos modelos, ¿estas reglas son administradas y mantenidas por el negocio o por la gente de IT? ¿La transformación ocurre en la infraestructura de integración disponible en la organización?
8. ¿Los modelos de datos están definidos en un lenguaje que define taxonomías, ontologías u otras representaciones lógicas de alto nivel?
9. ¿Existe un repositorio donde se mantengan estos objetos con identificadores globales? ¿O existen mecanismos para mapear esos objetos a distintos dominios / repositorios? ¿Esos mecanismos, son electrónicos o manuales? ¿Se hace para todas las aplicaciones o para algunos casos puntuales?
10. ¿Existe un mecanismo para buscar objetos corporativos a través de sus características?
11. ¿Cómo se ejecutan las transformaciones de datos entre aplicaciones? ¿Existe un intermediario que realiza esta tarea (ESB)? ¿Existen adaptadores que se encargan de esta tarea? ¿A través de una API? ¿A través de un servicio?
12. ¿Existen mecanismos para la transformación de datos utilizando ontologías? ¿Existe un servicio maestro de datos?

13. ¿Se está trabajando en la unificación de las representaciones de datos internas de la organización?

4.1.5.10 Infrastructure & Management Dimension: Assessment Questions

1. ¿Cuáles son los lineamientos de uso de la infraestructura de la organización?
2. ¿Cómo se derivan los SLA técnicos desde los SLA de negocio?
3. ¿Se han definidos SLA sobre QoS? ¿Cómo se monitorea y mide el cumplimiento?
4. ¿Se han definidos SLA sobre seguridad y privacidad? ¿Cómo se monitorean y mide el cumplimiento?
5. ¿Qué nivel de monitoreo se está ejecutando actualmente? ¿Qué herramientas de administración de infraestructura están disponibles en la organización?
6. ¿Qué plataformas se utilizan para la integración de aplicaciones?
7. ¿Sobre qué activos se utiliza versionado?
8. ¿Cuál es el proceso para administrar los cambios en la infraestructura?
9. ¿Qué herramientas de configuración se utilizan?
10. ¿Qué cosas son considerados activos IT de la organización? ¿Cómo se administran esos activos?
11. ¿Qué aspecto tiene la arquitectura operativa de la organización?
12. ¿Cómo se da soporte a requerimientos no funcionales a través de la arquitectura operativa?

4.2 Método OSIMM de Assessment

No necesariamente el método OSIMM debe aplicarse a toda la empresa, también puede ejecutarse sobre una sola línea de negocio o a un ecosistemas de servicios ya existentes. El objetivo de OSIMM es evaluar el estado actual del contexto analizado, determinar la madurez objetivo de la adopción y definir las tareas necesarias y los objetivos intermedios para llegar a ese estado de madurez buscado. La extensión o modificación del modelo OSIMM si bien es algo opcional, debería ser casi una obligación. Mayor precisión en la información representada por el modelo, nos llevan a una adopción más controlada y precisa, donde los objetivos de negocio tienen una representación más fuerte. El método de assessment OSIMM es iterativo y evolutivo, en donde se busca entrar en un círculo virtuoso con la organización / segmento donde se está ejecutando la adopción: a medida que se encuentran más familiarizados con la metodología, se va ganando experiencia en las ejecuciones de las iteraciones y se van planificando tareas que agregan más valor al negocio; ya sea por su propia definición o agregando indicadores de madurez al modelo OSIMM. OSIMM provee el control y guía sobre el proceso de adopción SOA en una organización y establece las bases para el proceso de Gobierno SOA. Ese es el principal valor agregado.

4.2.1 Descripción

Durante la etapa de análisis de contexto se puede dividir en tres grandes actividades:

1. Evaluar el nivel de madurez actual del negocio, la organización en sí y de todo lo que refiere a IT.
2. Identificar y definir un nivel de madurez meta u objetivo. Construir una visión de cómo se verían los procesos de negocio, la organización, el personal y las soluciones IT dentro de una solución SOA de alto desempeño.
3. Generación de un documento que describe el estado actual de la organización a través de todos los niveles de OSIMM. En este documento se debe especificar el estado meta o ideal, definir un plan de trabajo para determinar qué tareas se deberán ejecutar para llegar a ese estado ideal.

Estas tres actividades a su vez pueden descomponerse en tareas más granulares. Estas actividades se ejecutan en lo que se denomina un “análisis OSIMM” que se detallan a continuación:

1. Identificar los objetivos de negocio que sean relevantes para la evaluación o assessment.
2. Extender el modelo OSIMM agregando nuevos indicadores de madurez.
3. Agregar los atributos de madurez para cada nivel de madurez. Extendiendo el modelo base OSIMM agregando indicadores adicionales de madurez crea una relación entre la adopción SOA y los puntos en conflicto dentro de la organización, ya sea atacándolos o agregando capacidad para atender los requerimientos del negocio.
4. Evaluar el nivel de madurez actual SOA de la organización, comparando los indicadores de madurez contra los correspondientes indicadores de madurez.
5. En base a los objetivos de negocio, determinar cuál es el estado de madurez SOA buscando en la adopción que se está planificando.
6. Comparar el nivel de madurez actual y el especificado como meta. Identificar los puntos que todavía no están resueltos y definir un roadmap que lleve adelante estas transformaciones en la organización.
7. Documentar el resultado de la evaluación y el roadmap de trabajo.

4.2.2 Pasos para realizar un assessment

4.2.2.1 Identificar los puntos conflictivos, el alcance y los objetivos de negocio

Los puntos conflictivos se denominan a aspectos donde la organización considera que se debe mejorar el proceso y por lo tanto deben ser incluidos en el análisis OSIMM. Se debe recolectar material que ayude a determinar el nivel de madurez SOA buscado por la organización. Documentos de estrategia

tecnológica, requerimientos del usuario, documentos de arquitectura corporativa pueden ayudar a determinar este nivel de madurez meta. En esta etapa se confecciona una lista inicial de puntos conflictivos o metas estratégicas. Una vez que se finaliza con el alcance de la adopción, se puede preparar el roadmap SOA y las transformaciones que se deberán ejecutar. El dominio OSIMM planteado a través de las dimensiones que define ayuda a definir y contextualizar cada una de estas transformaciones.

4.2.2.2 Extender el modelo OSIMM

Tomando como base el alcance y los objetivos definidos en el punto anterior, se debe crear una matriz de adopción, basada en la matriz OSIMM, pero enfocada en los puntos conflictivos definidos por la organización. El modelo OSIMM puede extenderse agregando indicadores de madurez personalizados que harán foco sobre estos puntos conflictivos en cuestión o sobre requerimientos estratégicos.

4.2.2.3 Evaluar y determinar el estado actual

Se utiliza el modelo producido en etapas anteriores y se entrevista a puestos claves dentro del personal de la organización con el objetivo de determinar fehacientemente el estado actual de la organización y su estado de madurez. Las preguntas pueden estar basadas en el cuestionario base OSIMM, junto con otras que puedan asociarse con algunos puntos conflictivos que tengan un indicador de madurez asignado. En base a las respuestas, se establece el nivel de madurez actual y se define un puntaje del 1 al 10 para cada aspecto o dominio evaluado.

4.2.2.4 Determinar el estado futuro (evolución)

En base a la documentación, entrevistas y otras acciones se determina el estado a futuro. Las entrevistas deben llevarse a cabo con personal que tenga cargos clave en la toma de decisiones o en el conocimiento del contexto, que entienda la visión de negocio y conozca la estrategia de negocio de la organización. A partir de esto se puede tener un panorama de cómo la estrategia de la organización se relaciona con la iniciativa SOA a través de los servicios y de la infraestructura disponible. El estado futuro, meta o se determina estimando el ROI (retorno de inversión) de las tareas a ejecutar para llegar a determinado nivel de madurez y teniendo en cuenta los requerimientos de negocio.

4.2.2.5 Identificar las tareas a realizar y definir el roadmap

En los pasos anteriores se identificaron los niveles de madurez SOA meta de la adopción y se representan utilizando la matriz de adopción. Es momento de identificar las tareas que separan el estado actual de madurez de la organización del deseado. A través de los objetivos a ejecutar se deben alcanzar los objetivos corporativos o de negocio y aliviar algunos puntos de conflicto existentes en la organización. Se debe considerar en este análisis, restricciones propias del contexto como pujas de poder entre distintos sectores o áreas de la empresa, ya sea que formen parte de IT o respectivas a Gerencias de Negocio. El roadmap generado es un roadmap de alto nivel que sirve de hoja de ruta para los objetivos que se ejecutarán. Periódicamente se debe ejecutar una revisión de la madurez SOA de la organización, validando contra la documentación generada y revisando los objetivos planteados en la adopción.

4.3 Conclusiones

El modelo de adopción SOA planteado por OSIMM es un modelo que permite trabajar con idea de “incompletitud” pero con objetivos bien definidos. Es un esquema evolutivo que va mejorando a medida que se van ejecutando tareas sobre la organización objetivo de la adopción SOA.

El contexto inicial de la adopción se fija a través del cuestionario y del análisis que realice el equipo de trabajo encargado de llevar adelante la iniciativa. Este análisis es vital para el éxito de la iniciativa, ya que se genera en base a las experiencias previas del equipo y la capacidad técnica y política de cada uno de los integrantes.

La planificación de las tareas se realiza dentro del modelo de adopción propuesto por OSIMM. Este modelo de adopción deberá ser aumentado para ajustarse al contexto de la organización. Al aumentar la cantidad de objetivos y decidir el orden de aplicación se está tomando una decisión.

El marco que brinda el modelo de adopción OSIMM, sumado a la convergencia tecnológica basada en estándares y tecnologías que convergen sobre esos estándares abiertos permiten enfocar el trabajo sobre cómo lograr que esos cambios se produzcan de manera sustentable.

El tiempo es un recurso cada vez más escaso mirado desde una perspectiva de negocio. Los cambios deben ejecutarse con más velocidad y los sistemas, visto de una perspectiva global, deben responder a esos cambios cada vez con plazos más reducidos. Como una adopción SOA es por naturaleza un proceso largo donde se viven y se razona sobre un sinfín de factores que afectan su desarrollo natural, es necesario registrar las decisiones que se llevaron adelante para mitigar estos factores. La representación de estas decisiones a través de objetivos de una matriz de adopción en un sistema expone o modela de alguna forma experiencia o conocimiento. La repetición de esas experiencias construye un patrón y ese patrón puede ser utilizado más adelante, reduciendo así el tiempo o el esfuerzo necesario para llevar adelante el proceso.

5 Características de un modelo de madurez orientado a sistemas de información

Los modelos de madurez, han estado presentes dentro del ámbito de la informática desde la década de 1970. Los modelos son utilizados para comparar la madurez (en este contexto, la madurez puede ser cualquier cosa: capacidad, disponibilidad de ciertas características, complejidad) de un dominio en particular o algún elemento.

Con la introducción de los modelos de madurez en los años 1990 por CMM [13] se produjo un notable crecimiento en la cantidad de modelos de madurez disponibles en el mercado, en parte por los beneficios que este tipo de modelos tiene para los altos mandos ya que ayudan a balancear objetivos divergentes manteniendo una ventaja competitiva, reduciendo costos y “time to market” y mejorando la calidad del producto ofrecido. Este crecimiento se produjo de manera descontrolada y sin un modelo o esquema que lo generalice y controle.

La catalogación de modelos de madurez fue el primer intento para controlar su crecimiento. Sin embargo, cada modelo en sí es subjetivo ya que se creó en base al criterio personal del grupo de personas encargados de su ejecución. Por lo tanto, los modelos entre sí carecían de puntos de comparación y no se podía establecer métricas entre los modelos.

5.1 Definición de modelos de madurez

Madurez se define como “el estado de estar completo, perfecto o listo” [14]. Llegar un estado de madurez determinado implica un proceso evolutivo donde se han ejecutado ciertas actividades para llegar a ese estado. Los modelos de madurez son una guía para ir de un estado inicial a un estado objetivo de madurez. Todos los modelos comparten ciertas características:

1. Dimensiones que atacan un aspecto del problema
2. Niveles de madurez progresivos
3. Descripciones para los niveles, condiciones que se deben cumplir
4. Elementos para cada uno de las áreas o dimensiones, que serán las actividades a realizar para alcanzar un grado de madurez determinado.

5.2 Clasificación de un modelo de madurez

La clasificación de los modelos de madurez en base a su contenido no es una tarea fácil debido a la subjetividad que existe en cada definición o afirmación del modelo [14]. Una alternativa es agregar atributos a los modelos que faciliten su interpretación y descripción. Los atributos definen algún comportamiento o característica del modelo. En [14] se propone agregar a cada modelo tres grupos de atributos:

1. Generales
2. Diseño del modelo de madurez
3. Uso del modelo de madurez

Estos atributos ayudan a determinar quién es el responsable de la definición del modelo, dónde se puede obtener más información acerca del modelo, de qué forma está diseñado y para qué fin.

La clasificación nos ayuda a determinar en qué contexto fue pensado este modelo de madurez y en base a los atributos brinda ayuda al usuario del modelo de madurez a elegir un modelo según la situación que deba enfrentar. Los atributos surgidos sin embargo no atacan la ocurrencia de varias instancias del mismo modelo de adopción. Las actividades generadas en estos modelos pueden ser similares y repetitivas. Una alternativa de solución a este problema es la creación de recomendaciones explícitas, es decir, pasos que seguramente deban ser ejecutados. Para esto el modelo de madurez debe estar asociado a un contexto: ya sea metodología, práctica, entorno.

5.1 Conclusión

Los resultados destacados por [14] nos permiten conocer que existe una subjetividad implícita en los modelos de adopción que impide en cierta forma su generalización. Una manera de proponer una generalización o reúso de los conocimientos adoptados durante la ejecución de un modelo de adopción es fijando lo más posible el contexto donde se ejecutó la evaluación. Utilizando el modelo de adopción OSIMM se fija el contexto. Las preguntas del cuestionario OSIMM nos definen el contexto. Los objetivos a ejecutar durante la adopción SOA transforman ese contexto en un nuevo contexto. Estos cambios se producen gracias al análisis y un plan de trabajo y esa tarea es representable y modelable a través de un sistema de información o aplicación.

6 Modelando la madurez de una organización

Existen iniciativas que buscan mejorar la capacidad de evaluar la madurez de una arquitectura SOA utilizando la información que se puede encontrar en los documentos de arquitectura de software de la organización. La iniciativa busca reducir el costo (ya sea en tiempo o recursos) de realizar las entrevistas con cada uno de los afectados por una adopción SOA.

6.1 Actores involucrados

Una forma de tratar con el problema es representar todos los involucrados como un modelo siguiendo las pautas del Desarrollo Dirigido por Modelos, donde se debe definir claramente:

- **Modelo:** un conjunto de premisas acerca de un sistema que se encuentra bajo estudio.
- **Meta-modelo:** es la descripción del modelo utilizando un lenguaje de meta-modelado, describiendo un problema para un dominio específico.
- **Transformación:** es una operación basada en un modelo que da como resultado un nuevo modelo, siempre y cuando haya un meta-modelo de entrada y uno de destino.

Siguiendo estos lineamientos y utilizando ArchivoL [11], EMF[19] y un DSL[12] específico para el problema se pueden representar y modelar situaciones comunes de una adopción SOA siguiendo el modelo OSIMM. El objetivo es crear una relación entre un cuestionario OSIMM y una serie de artefactos de arquitectura de software.

6.2 Descripción del modelo

El meta-modelo propuesto es el siguiente, donde se pueden identificar fácilmente cada uno de los involucrados en un modelo OSIMM y las relaciones existentes entre cada uno de los modelos [11] como se puede observar en la Fig. 41:

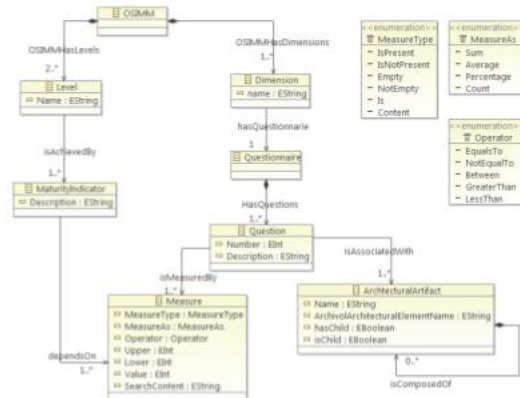


Fig. 41 Meta-modelo para representar un assestment OSIMM

Levels: OSIMM presenta siete niveles de madurez, un nivel debe ser descrito como una posible abstracción del estado de la organización, este concepto debe existir porque a través de los niveles es que pretendemos clasificar la madurez de la arquitectura.

MaturityIndicator: este concepto es opcional y depende de la forma en la cual queremos medir la arquitectura, es posible utilizarlo como parte de la evaluación o simplemente como un punto descriptivo de que indicadores de madurez pertenecen a qué nivel.

Dimensions: OSIMM presenta una línea particular de negocio o arquitectura que debe ser medida como parte de la madurez de toda la arquitectura, este concepto tiene que existir ya que un cuestionario debe obtener información de la arquitectura para una dimensión específica.

Questionnaires: OSIMM presenta una clasificación basada en encuestas por lo tanto los cuestionarios son claves y un concepto imprescindible.

Questions: el concepto de evaluación más importante de nuestro meta-modelo, todas las preguntas mapeadas deben poseer un elemento de arquitectura.

ArchitecturalArtifact: va a representar los elementos que pueden ser descritos en una arquitectura candidata de archivo, este es el corazón del mapeo y entendimiento entre los cuestionarios de OSIMM y la descripción de la arquitectura de la organización hecha en archivo.

Measure: este concepto es nuestro clasificador, debemos medir de alguna manera los artefactos arquitecturales y los indicadores para cada nivel de madurez.

Además para trabajar con este meta-modelo se define un DSL específico utilizado para introducir el resultado del cuestionario OSIMM ejecutado sobre una adopción SOA. Un ejemplo se expone en la Fig. 42

```
test.osimm
OSIMM {
  create new Dimension "OSIMM Test for Governance" {
    create new Questionnaire {
      define Question {
        Number=1;
        Description="What are the skills of your IT Staff?";
        using ArchitecturalArtifact {
          Name = "Portfolio.Team";
          ArchivolName = "portfolio.teamSkills";
          HasAChild = false;
          isChild = true;
          Using Measure for "SILO" {
            MeasureType = Empty;
            MeasureAs = Percentage;
            Operator = Between;
            Lower = 5;
            Upper = 10;
          }
          Using Measure for "INTEGRATED" {
            MeasureType = NotEmpty;
            MeasureAs = Percentage;
            Operator = EqualsTo;
            Value = "20";
          }
        }
      }
    }
  }
}
```

Fig. 42 - Ejemplo de un cuestionario OSIMM representado con un DSL

6.3 Forma de trabajo

La dinámica de trabajo persigue el objetivo de definir el cuestionario utilizando el DSL y a través de transformaciones obtener el meta-modelo asociado a los artefactos de arquitectura de software correspondientes a esta adopción.

La secuencia de paso a ejecutar es la siguiente:

Se genera la entrada del proceso

Describir la arquitectura actual, que será la entrada principal del proceso. Utiliza como base el catálogo de servicios que existen en la organización denominados también “service portfolio”. Este inicio de considera el “más importante” por las prácticas de gobierno y las políticas que se aplican sobre los servicios en SOA. Por lo tanto, se busca modelar esta necesidad sobre algún servicio y a partir de ahí conseguir todos los atributos necesarios para trabajar con modelos.

Se introduce el cuestionario OSIMM

Se crea el DSL del clasificador, creando los cuestionarios, definiendo las preguntas y asociando los componentes de arquitectura a mediciones correspondientes para cada nivel de madurez. De esta manera las mismas métricas pueden ser aplicadas a diferentes casos utilizando los mismos estándares y mismas medidas.

Se aplican transformaciones y se generan los reportes

Con los elementos descritos, se aplican una serie de transformaciones y se ejecutan las consultas sobre el modelo de arquitectura original. Luego se generan los reportes que detallan la situación actual en cuanto a componentes de arquitectura de la adopción OSIMM, utilizando para esto Query2 de EMF.

6.4 Conclusión

Los trabajos realizados por [11] y [1] marcan el camino para representar la matriz de adopción SOA OSIMM como un conjunto de objetos en una aplicación. Esta experiencia será tomada en cuenta para la definición del modelo de objetos de nuestra herramienta y evitar algunos problemas como lo subjetivo que pueden ser algunas preguntas del cuestionario OSIMM, lo que dificulta darle una ponderación discreta.

7 Herramienta de Gobierno de Adopción SOA

7.1 Descripción

La herramienta desarrollada facilitara la creación de assessments, tanto en la generación de encuestas como en la creación de la matriz de adopción.

Desde la misma se podrán administrar tanto las preguntas de las encuestas como los objetivos de la matriz de adopción. De esta manera se permite ir incrementando la complejidad de los assessments a medida que se van ingresando datos al sistema.

La principal ventaja de la aplicación es la de proveer la capacidad de generar los assessment utilizando una base de conocimiento. La misma incluirá todos los assessment previamente cargados, de manera que al momento de la creación de un nuevo assessment se puede elegir el porcentaje que debe cumplir un objetivo dentro del total de assessments para ser incluido en la nueva matriz.

La segunda funcionalidad que se proveerá a partir de la base de conocimiento es la de generar los objetivos adyacentes a uno seleccionado en la matriz de adopción. Con solo presionar un botón, se generaran y conectaran los objetivos anteriores y próximos al seleccionado, tomando como referencia los assessments previamente ingresados.

7.2 Arquitectura

La arquitectura a usar se puede observar en la Fig. 35:

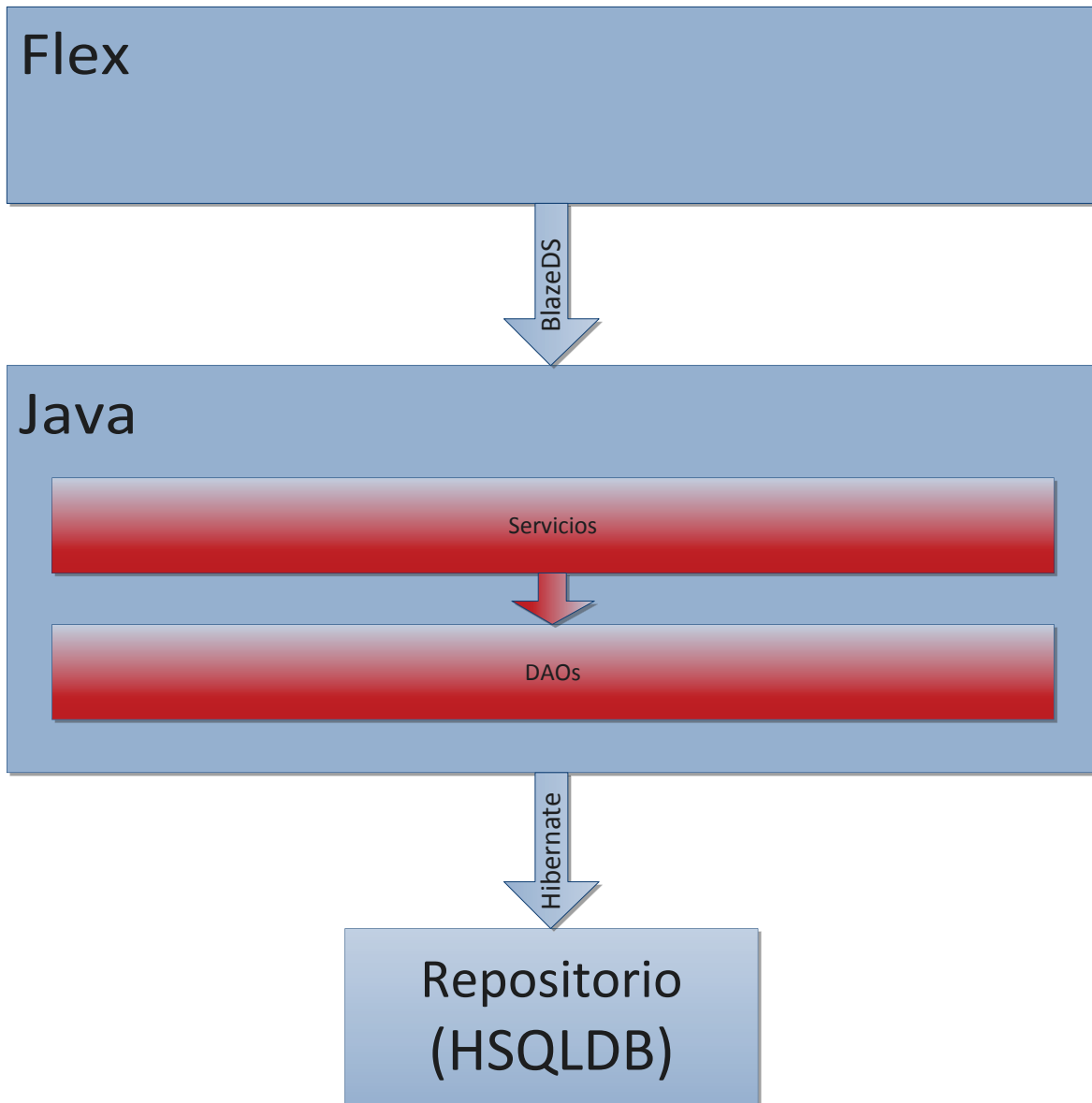


Fig. 43 - Diagrama arquitectura mostrando las distintas capas de la herramienta

Como se puede observar, para la capa de presentación al usuario se utilizó Flex y dentro de Flex la microarquitectura Cairngorm para manejar el envío de eventos al back end.

La decisión del uso de Flex en la capa de presentación radica en la facilidad de uso en múltiples navegadores, dado que al utilizar un player de Flash, se independiza del navegador que se utilice, sin tener que programar excepciones para cada uno de los navegadores disponibles en el mercado.

Otro punto fundamental en la decisión del uso de Flex es el manejo de drag and drop que posee, dando al usuario una manera muy natural de realizar la creación y relación de objetivos en la matriz de adopción del assessment.

La conexión de la capa de presentación con el back end se realiza mediante BlazeDS, de manera que se le delega a este framework la responsabilidad de serializar y enviar la información desde Flex a Java.

Ya en el backend, se poseen 2 capas bien definidas. La capa de servicios que es el punto de entrada, desde el cual se invoca a la capa de daos, los cuales acceden al repositorio mediante el uso de Hibernate para realizar el mapeo objeto relacional.

Para la configuración del proyecto java se utilizo Spring, ya que da claridad al momento de poder configurar los distintos actores mediante archivos de configuración muy sencillos de modificar y adaptar en caso de ser necesario.

En la capa de repositorio se utiliza una base de Datos HSQLDB, debido a su fácil utilización. Aunque con el uso de hibernate como mapeador, es muy sencillo poder cambiar este repositorio por otro como por ejemplo Oracle o MySQL.

A continuación, en la Fig. 44 se muestra un diagrama de clases de los elementos que interactúan en la aplicación:

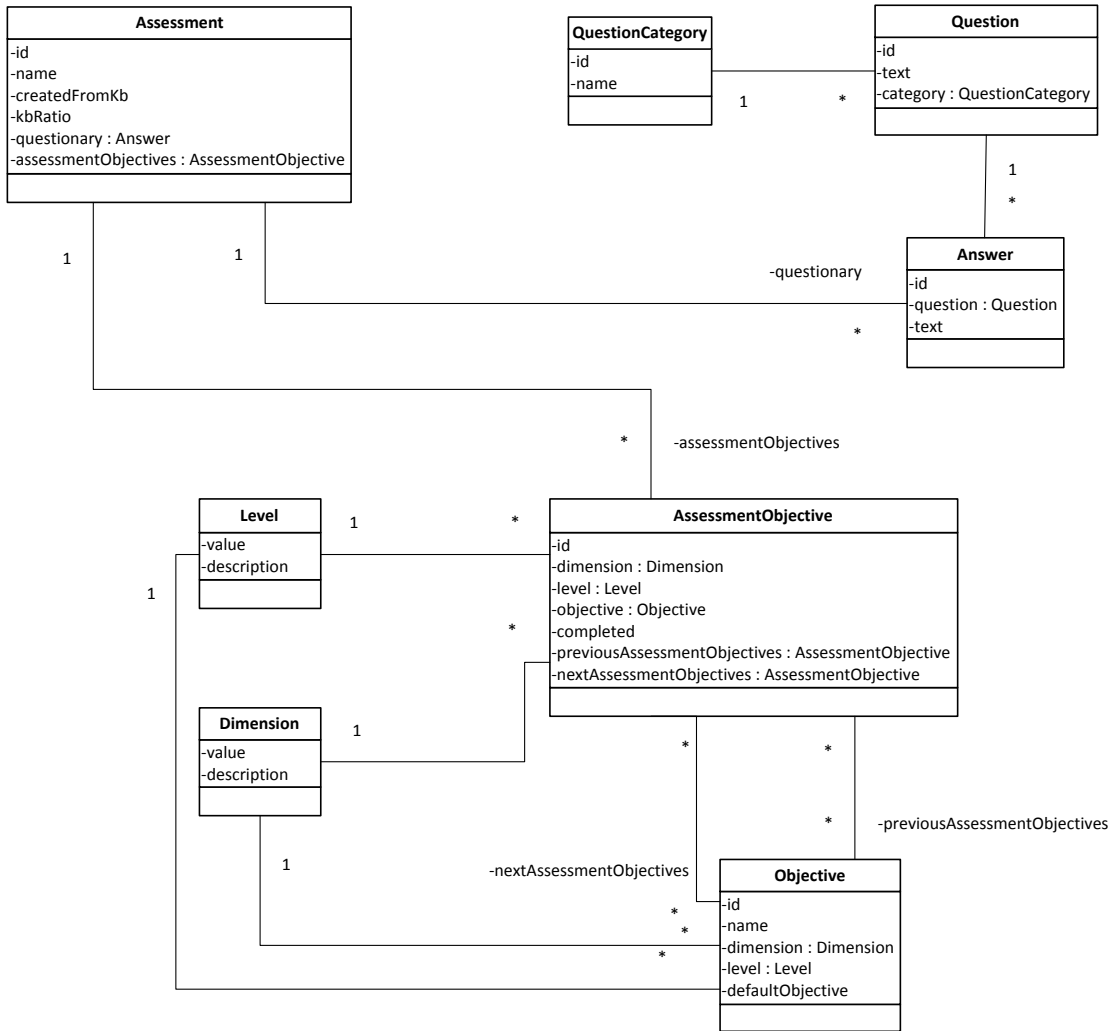


Fig. 44 - Modelo de clases de la herramienta

Como se puede observar, se modeló como clase central el Assessment, relacionando con este tanto al cuestionario así como los AssessmentObjectives que forman parte de la matriz de adopción.

7.3 Cuestionario

El primer punto importante de la aplicación es la generación del cuestionario del assessment. Al crear un nuevo assessment o al editar uno existente, se abrirá la pantalla del cuestionario.

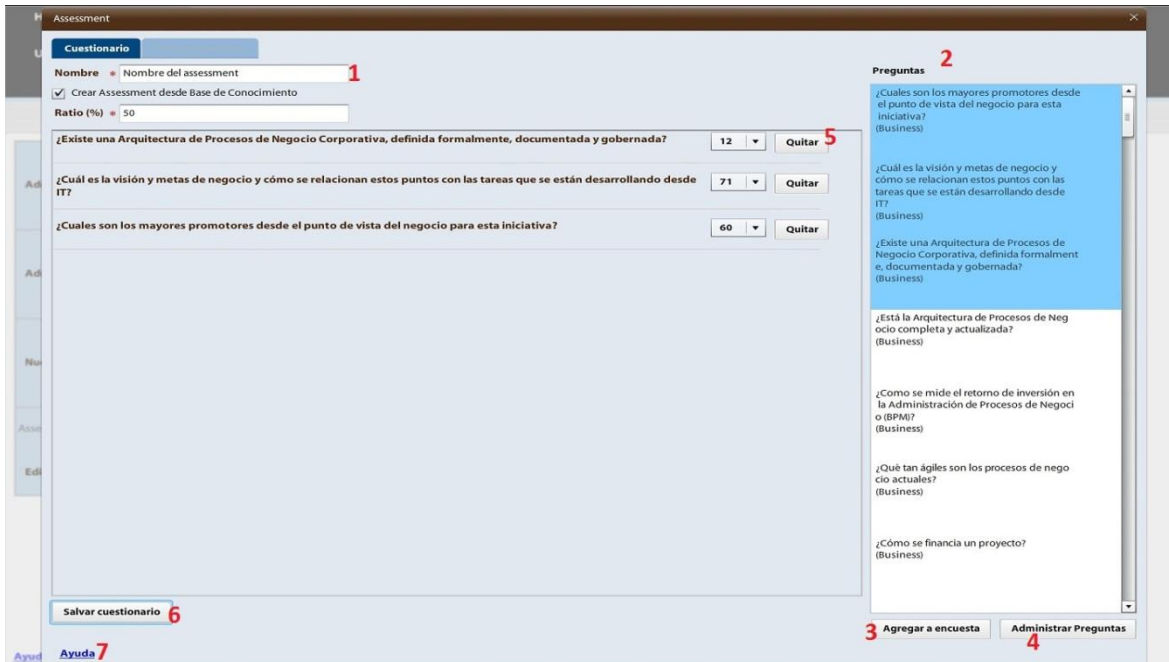


Fig. 45 - Completando un cuestionario OSIMM a través de la herramienta

7.3.1 Nombre del assessment

El primer campo a completar es el nombre del assessment (Marcado con el numero 1 en la Fig. 45)

7.3.2 Creación desde la base de conocimiento

En el caso de la creación de un nuevo assessment, se deberá decidir si el assessment se creara desde la base de conocimiento o si por el contrario se usaran los objetivos por defecto para la creación de la matriz. Los pormenores de como se realiza la creación de la matriz en este caso se abordara en el capítulo 7.5 Base de Conocimiento.

7.3.3 Selección de preguntas de la encuesta

En este momento se comenzara la carga de las preguntas de la encuesta, seleccionando de la lista de preguntas (marcado con el numero 2 en la Fig. 45) las que se desean incluir en este assessment en particular y presionando el botón de “Agregar a encuesta” (marcado con el numero 3 en la Fig. 45).

En caso que se haya agregado una pregunta que no corresponda, se podrá quitar la misma del cuestionario solo presionando el botón “Quitar” que se encuentra junto con la pregunta (Marcado con 5 en la Fig. 45).

7.3.4 Salvado de la encuesta

Una vez que se haya finalizado con la carga de las preguntas y sus correspondientes valores, se procede a salvar al cuestionario mediante el botón correspondiente (marcado con el numero 6 en la Fig. 45). Esta acción creara el assessment, así como su cuestionario y la matriz de adopción, habilitando la opción de poder comenzar a modificar la misma.

7.3.5 Ayuda

Cabe aclarar que en esta pantalla se dispone de un link de ayuda (marcado con el numero 7 en la Fig. 45), el cual abrirá la ventana de ayuda que se muestra en la Fig. 47 y se posicionara en el tópicó correspondiente.

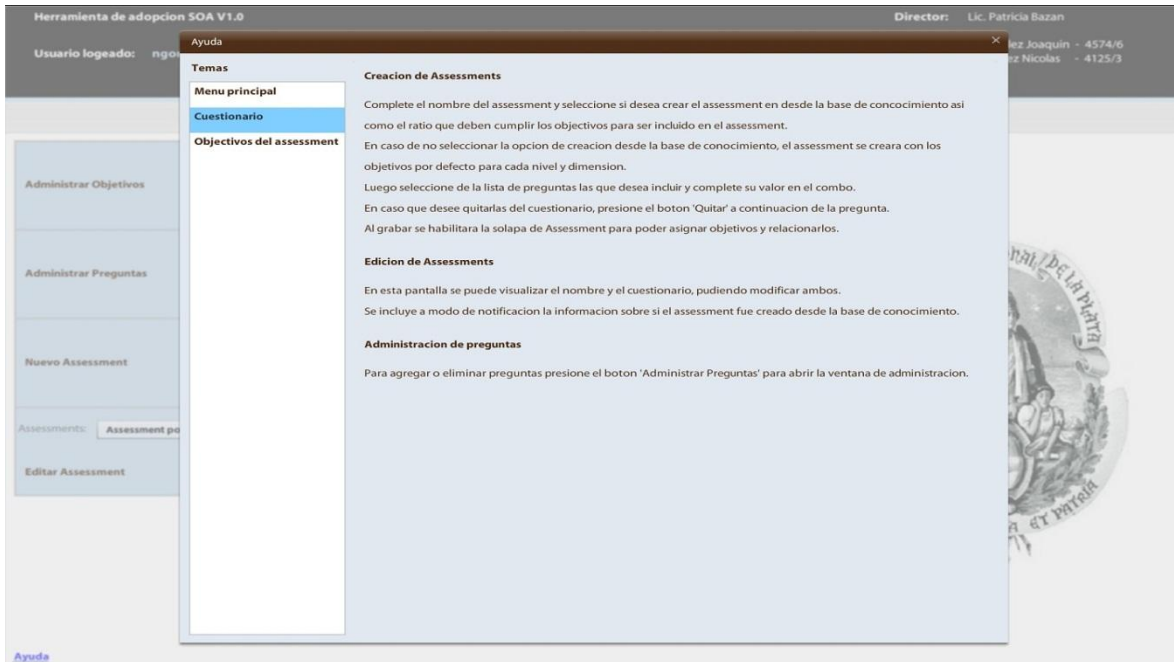


Fig. 46 - Ayuda contextual de la herramienta

7.3.6 Administración de preguntas

En caso que la pregunta no se haya cargado previamente en la aplicación, se podrá acceder a la pantalla de administración de preguntas presionando el botón correspondiente (marcado con el numero 4 en la Fig. 45). Se abrirá la pantalla que se muestra en la Fig. 48

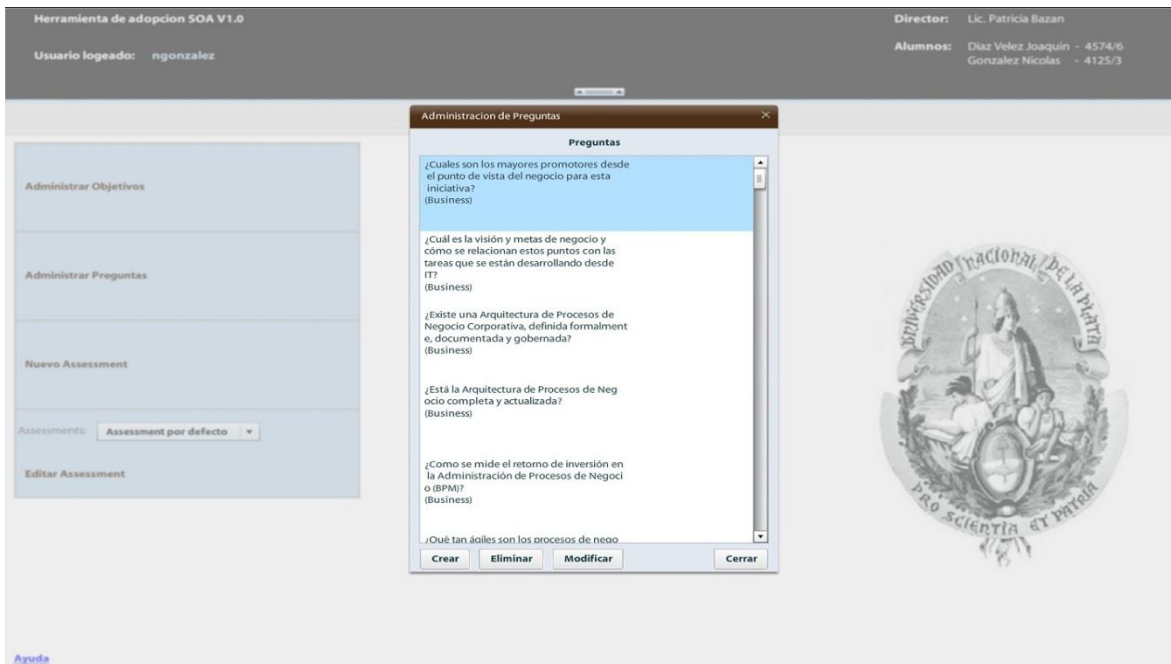


Fig. 47 - Administración de preguntas de la herramienta

En esta pantalla se podrán administrar las preguntas, pudiendo crear nuevas en caso de ser necesario para el ingreso de la encuesta del assesment en cuestión. Para esto se presionara el botón de Crear el cual abrirá la pantalla de ingreso de nuevas preguntas, la cual se muestra en la Fig. 49.

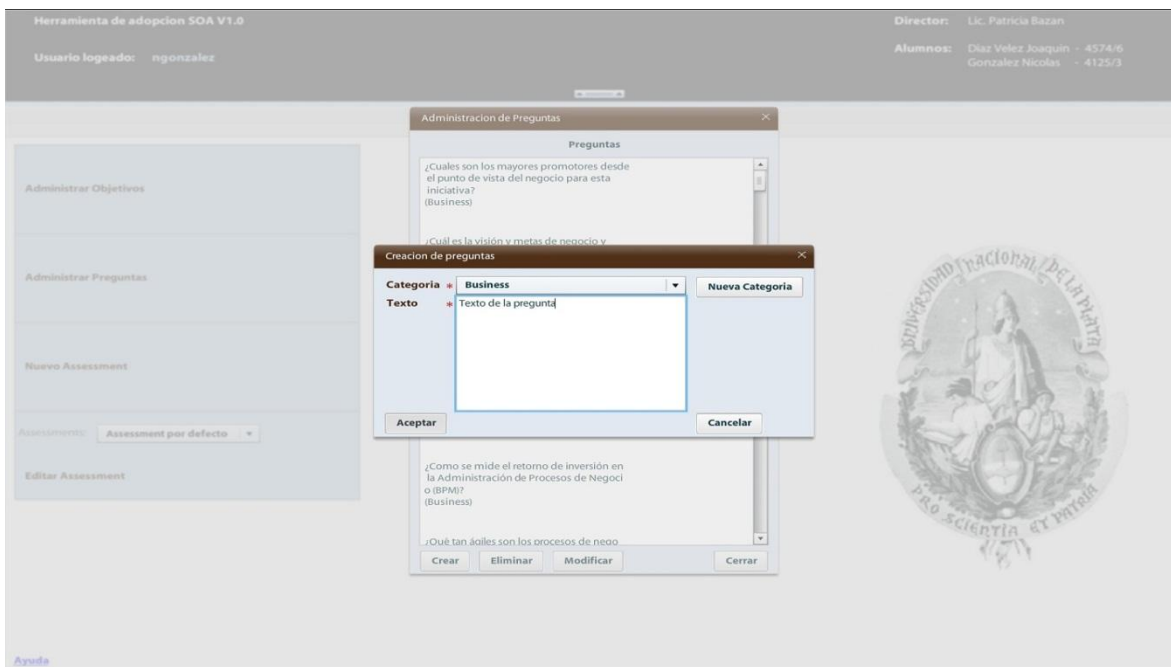


Fig. 48 - Creando una nueva pregunta en la herramienta

7.4 Matriz de objetivos

Como se comentó en el punto 7.3 al grabar el assessment se habilitara la solapa de administración de la matriz. La misma se muestra en la Fig. 50.

The screenshot displays the 'Assessment' tool interface. At the top, there are tabs for 'Cuestionario' and 'Matriz de adopción'. The main area is a grid with columns representing service models: Silo, Integrated, Componentized, Services, Composite Services, Virtualized Services, and Dynamically Re-Configurable. The rows represent different dimensions: Business, Organization, Methods, Applications, Architecture, Information, and Infrastructure. A search list on the right side shows a list of objectives, with one objective selected and marked with a red '2'. At the bottom, there is a toolbar with numbered buttons: 'Ayuda 4', 'Mover 5', 'Relacionar', 'Borrar objetivos 6', 'Marcar como cumplidos 7', 'Marcar como NO cumplidos 8', 'Generar adyacentes 9', and 'Ver Adyacentes 10'.

Fig. 49 - Vista de la matriz de objetivos OSIMM en la herramienta

7.4.1 Ayuda

En esta pantalla, al igual que en el resto de las principales pantallas, se cuenta con un link de ayuda, el cual está marcado con el número 4 en la Fig.50. Al presionarlo se abrirá la ayuda, en la cual se indica cómo realizar las operaciones relacionadas con la creación de la matriz de adopción.

7.4.2 Administración de objetivos

En caso de que no se encuentre el objetivo que quiere agregarse a la matriz en la lista de Objetivos, se puede acceder al administrador de objetivos presionando el botón marcado con 1 en la Fig.50.

7.4.3 Incorporar objetivos al assessment

Una vez identificado el objetivo de la lista de objetivos marcado con 2 en la Fig.50, se debe seleccionarlo y arrastrarlo a la lista en la dimensión y nivel deseados como por ejemplo la marcada con 3, siendo la lista de dimensión Business y nivel Silo. Al soltar el objetivo dentro de la lista, se grabara el assessment agregando el objetivo.

7.4.4 Relacionando y moviendo objetivos del assessment

Una vez que se hayan creado los objetivos de la matriz de adopción, se puede proceder a relacionarlos entre sí o en caso de ser necesario mover los mismos de una lista a otra. Para esto se cuenta con un conjunto de radio buttons marcados con el numero 5 en la Fig.50. Seleccionando Mover o Relacionar se podrá modificar el comportamiento de la aplicación al arrastrar un objetivo de una lista a otra.

7.4.5 Mover objetivos

En caso de que este seleccionado *Mover*, al arrastrar un objetivo de una lista a otra se moverá el mismo. Cabe aclarar que en el caso de que las relaciones con otros objetivos dejen de ser validas debido al orden de precedencia entre las mismas, se mostrara una alerta informando que se eliminaran las relaciones que dejaran de ser validas.

7.4.6 Relacionar objetivos

En caso de que este seleccionado *Relacionar*, al arrastrar un objetivo de una lista a otra, se abrirá una pantalla que nos permitirá seleccionar con qué objetivo de la lista de destino se quiere crear la relación así como la dirección de la misma en caso de que sea necesario. En las Fig. 51 y 52 se muestran las pantallas de relación de objetivos en donde, en la primera, la dirección es inferida por el Nivel de los objetivos y en la segunda se presenta al usuario la elección de la dirección de la relación.

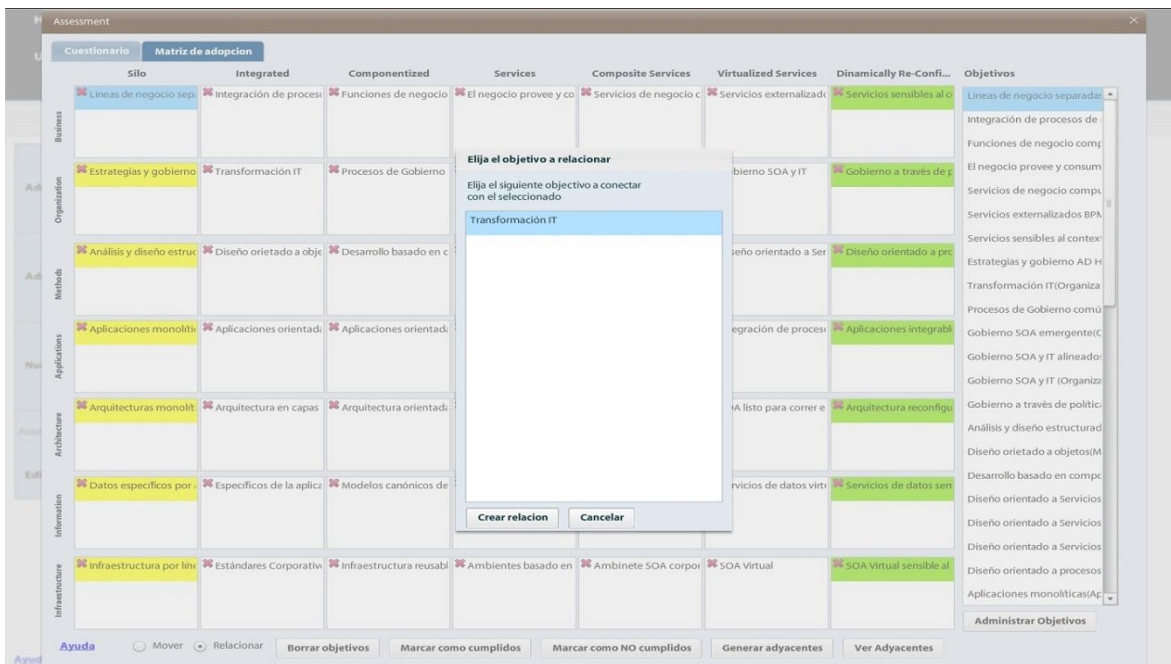


Fig. 50 - Relacionando objetivos de la matriz de adopción

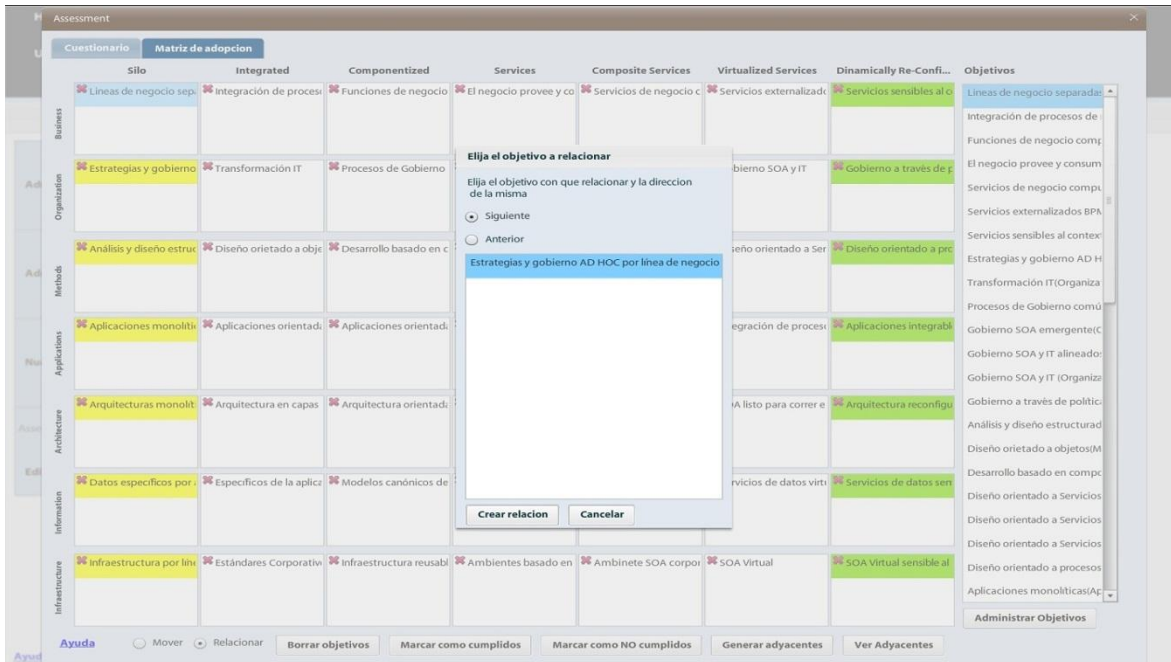


Fig. 51 - Estableciendo la relación entre objetivos


7.4.7 Borrar objetivos

En caso de que se desee borrar objetivos del assessment, sencillamente se procede a seleccionar los mismos y presionar el botón *Borrar Objetivos* marcado con el número 6 en la Fig. 50. Luego de confirmar que se desean eliminar los objetivos seleccionados estos serán eliminados del assessment.

7.4.8 Marcar objetivos como cumplidos


A modo de control sobre el avance de la adopción de la matriz en el actual assessment, a cada objetivo se lo puede marcar como cumplido. Para esto se deberá seleccionar los objetivos que se deseen y presiona el botón *Marcar como cumplidos* señalado con el número 7 en la Fig.50.

7.4.9 Marcar objetivos como no cumplidos

Al igual que en el punto anterior y a modo de control sobre el avance de la adopción de la matriz en el actual assessment, a cada objetivo se lo puede marcar como no cumplido. Para esto se deberá seleccionar los objetivos que se deseen y presiona el botón *Marcar como NO cumplidos* señalado con el número 8 en la Fig.50. Al ser marcados como cumplidos, se mostrar la imagen  a la izquierda del objetivo.

7.4.10 Generar objetivos adyacentes

Otra funcionalidad que provee la base de conocimiento es la de generar, en un nuevo assessment, los objetivos relacionados a uno en particular, tomando como base las relaciones que posee este mismo objetivos en ese mismo nivel y dimensión en el resto de los assessments cargados en la aplicación. Para esto se debe seleccionar un objetivo y presionar el botón *Generar adyacentes* marcado con el número 9 en la Fig.50. Se detallara la manera en que se obtienen los datos de la base de conocimiento y como se

generan los objetivos adyacentes en la sección 7.5 Base de Conocimiento. Al ser marcados como cumplidos, se mostrar la imagen  a la izquierda del objetivo. Cabe destacar que al crear un assessment, todos sus objetivos se inician como no cumplidos.

7.4.11 Ver objetivos adyacentes

Con el fin de poder observar los objetivos relacionados con uno en particular, se debe seleccionar el mismo en la matriz y presionar el botón *Ver adyacentes* marcado con el número 10 en la Fig. 50. Esto abrirá la pantalla que se muestra en la Fig. 53.

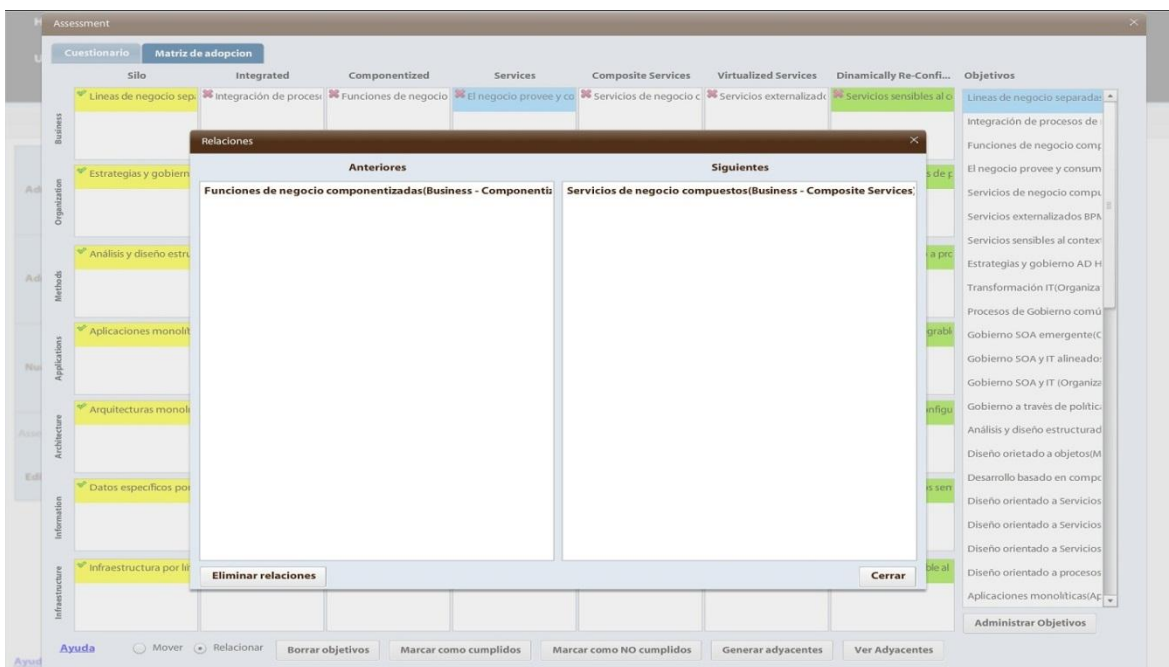


Fig. 52 - Detalle de los objetivos adyacentes

En esta pantalla se pueden observar los objetivos relacionados con el seleccionado, así como también se pueden eliminar las relaciones. Una aclaración que es necesaria hacer es que al eliminar una relación no se elimina el objetivo de la matriz, solo se elimina la relación entre los objetivos.

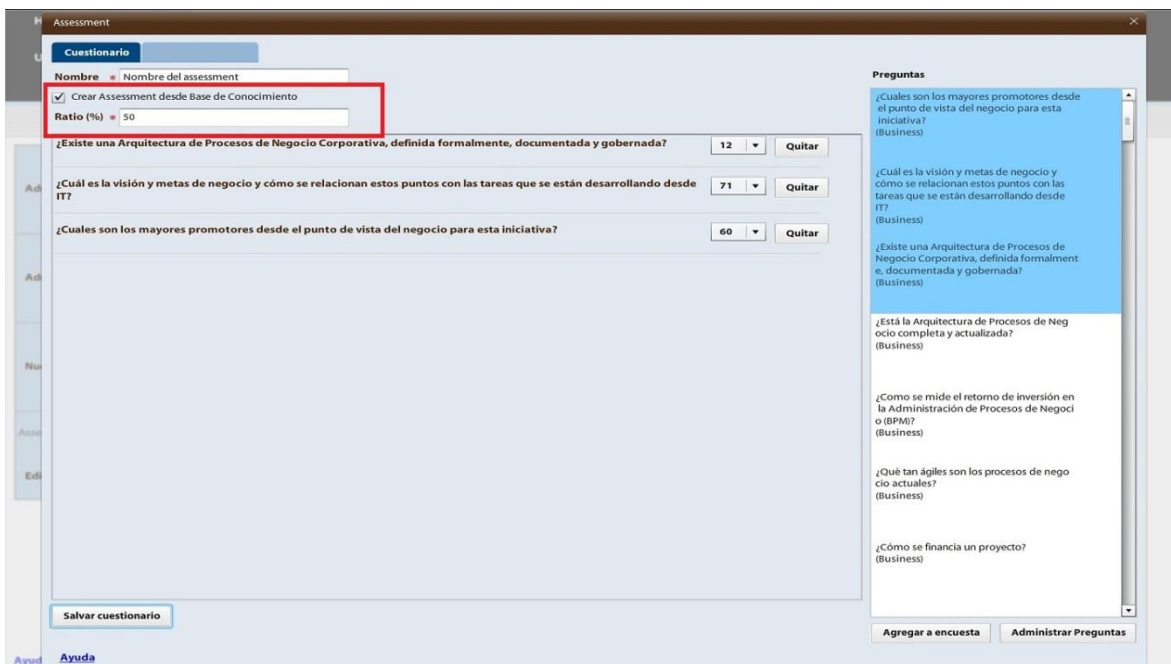
7.5 Base de conocimiento

7.5.1 Contexto

Al momento de la carga de assessments durante los procesos de adopción, se detectó que en la mayoría de los casos los mismos poseen muchos objetivos en común. Es por esto que la principal utilidad que provee la herramienta que se modeló es la de poder tomar la información de esta base de conocimiento subyacente y generar los objetivos de un assessment de manera automática. Para esto se generaron dos herramientas, la generación de la matriz completa a partir de la matriz de conocimiento y la generación de los objetivos y relaciones adyacentes a un objetivo en particular en una dimensión y nivel específicos. A continuación se detalla cómo se implementaron ambas herramientas:

7.5.2 Generación de la matriz a partir de la base de conocimiento

Como se detalló en el punto 7.3 Cuestionario, al momento de la creación de un nuevo assessment, existe la posibilidad de crear la matriz a partir de la base de conocimiento.



The screenshot shows the 'Assessment' application window. The 'Cuestionario' tab is active. In the 'Nombre' section, the checkbox 'Crear Assessment desde Base de Conocimiento' is checked, and the 'Ratio (%)' is set to 50. Below this, there are three questions with associated scores and 'Quitar' buttons:

- ¿Existe una Arquitectura de Procesos de Negocio Corporativa, definida formalmente, documentada y gobernada? (Score: 12)
- ¿Cuál es la visión y metas de negocio y cómo se relacionan estos puntos con las tareas que se están desarrollando desde IT? (Score: 71)
- ¿Cuales son los mayores promotores desde el punto de vista del negocio para esta iniciativa? (Score: 60)

The 'Preguntas' list on the right contains the following questions:

- ¿Cuales son los mayores promotores desde el punto de vista del negocio para esta iniciativa? (Business)
- ¿Cuál es la visión y metas de negocio y cómo se relacionan estos puntos con las tareas que se están desarrollando desde IT? (Business)
- ¿Existe una Arquitectura de Procesos de Negocio Corporativa, definida formalmente, documentada y gobernada? (Business)
- ¿Está la Arquitectura de Procesos de Negocio completa y actualizada? (Business)
- ¿Cómo se mide el retorno de inversión en la Administración de Procesos de Negocio (BPM)? (Business)
- ¿Qué tan ágiles son los procesos de negocio actuales? (Business)
- ¿Cómo se financia un proyecto? (Business)

Buttons at the bottom include 'Salvar cuestionario', 'Ayuda', 'Agregar a encuesta', and 'Administrar Preguntas'.

Fig. 53 - Creando una matriz de adopción utilizando la base de conocimiento

Tal como se muestra en la Fig. 54 se debe seleccionar el checkbox Crear Assessment desde *Base de Conocimiento* y luego ingresar el ratio de aceptación de objetivos. Este ratio, expresado en porcentaje, indica el porcentaje de assessments en que debe estar presente un objetivo para ser incluido en la matriz generada automáticamente.

Una vez ingresados estos datos, así como las preguntas de la encuesta y grabado el cuestionario, la aplicación calculara la matriz con el ratio de aceptación indicado, tomando los objetivos desde la base de conocimiento de la aplicación.

Para el cálculo de la matriz en base al ratio de aceptación, se desarrolló un algoritmo de tres pasos:

7.5.3 Paso 1, cuantificación:

El primer paso es cuantificar los objetivos en una dimensión y nivel específicos, de manera de poder tener una cuenta de en cuantos assessments está incluido ese objetivo en ese nivel y dimensión.

7.5.4 Paso 2, asignación de objetivos:

Una vez que se tienen todos los objetivos cuantificados, se procede a comparar el ratio de aceptación con la cantidad de apariciones de un objetivo sobre la cantidad total de assessments. En caso de que esta cantidad de apariciones sea mayor al ratio de aceptación se incluye el objetivo en esa dimensión y nivel como parte de la matriz.

7.5.5 Paso 3, relación de objetivos:

Luego de asignar los objetivos a la matriz, estos se encuentran sin ninguna relación entre sí, por lo que se recorren los objetivos que se utilizaron para la cuantificación y en caso de que dos objetivos hayan estado relacionados en algún assessment de la base de conocimiento, se procede a relacionarlos en esta nueva matriz generada.

7.5.6 Generación de objetivos adyacentes

La herramienta de generación de objetivos adyacentes se puede utilizar en un assessment ya creado. Esta herramienta permite que, tomando un objetivo en una dimensión y nivel específicos, se generen los objetivos y relaciones que ya se encuentran en otros assessments de la base de conocimiento.

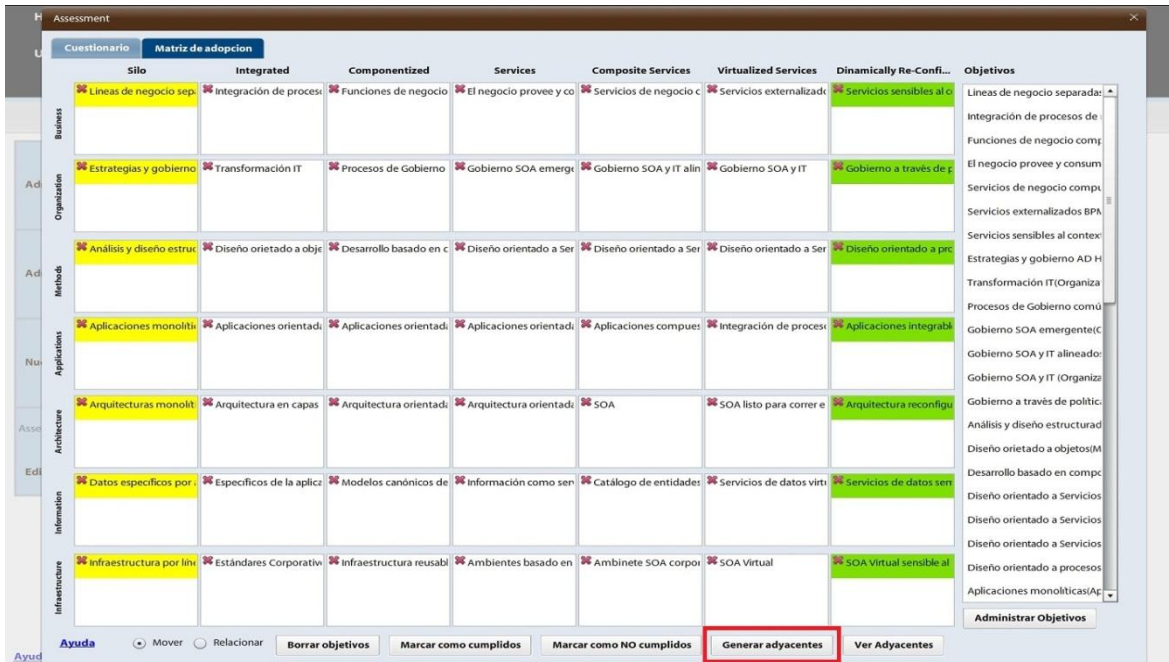


Fig. 54 - Creando los objetivos adyacentes a partir de la base de conocimiento

Al seleccionar un objetivo en la matriz y presionar el botón *Generar adyacentes* que se encuentra remarcado en la Fig. 55 la aplicación ejecutara el algoritmo encargado de generar y relacionar los objetivos adyacentes al seleccionado. Es decir que se buscara en la base de conocimiento este objetivo en esa dimensión y nivel específico y se obtendrán cuáles son los objetivos que se encuentran relacionados con este en toda la base de conocimiento con el fin de poder agregar esos objetivos a la matriz o (en caso de que los objetivos ya se encuentren en la misma pero no se encuentren relacionados con el seleccionado) crear la relación correspondiente.

El algoritmo que realiza esta tarea se puede desglosar en dos grandes tareas:

7.5.6.1 Paso 1, determinar relaciones:

El primer paso es el de tomar el objetivo seleccionado, así como la dimensión y el nivel del mismo y, utilizando la base de conocimiento, recopilar los objetivos con los que se encuentra relacionado este en el resto de los assessments.

7.5.6.2 Paso 2, asignación y relación de objetivos:

Una vez que se recopiló la información sobre las relaciones, se procede a recorrer la lista generada. Se puede dar dos casos:

1. El objetivo recopilado no es parte de la matriz del assessment:
 - a. En este caso se generara el objetivo en la matriz para la dimensión y nivel correspondientes y se creara la relación entre ambos.
2. El objetivo recopilado ya es parte de la matriz del assessment:
 - a. En este caso se procede a validar si ya existe una relación entre ambos objetivos y en caso de no existir se creara la relación correspondiente.

7.6 Aprendizaje a través de los assessments

Con el uso de la base de conocimiento, la herramienta posee una forma de generar aprendizaje a medida que más assessments son incluidos en la misma. Esto se debe a que con el correcto uso de la base de conocimiento y teniendo un buen relevamiento del assessment, las opciones de generación de objetivos automáticamente irán evolucionando de manera que el trabajo manual de inserción de objetivos en un nuevo assessment se minimizara. Esto conlleva una reducción significativa del tiempo empleado en la generación de la matriz de adopción para un assessment, pudiendo realizar la adopción de una manera más ágil.

8 Conclusiones

Una adopción SOA es un proceso transformador de una organización. Esas transformaciones atacan cada uno de los aspectos de la organización y no sólo se restringen a lo técnico o a lo que respecta a IT. Por la amplitud de estas transformaciones es necesario generar un plan y buscar alianzas dentro de la misma organización para llevar adelante cada una de los objetivos planteados en el roadmap de adopción.

El modelo OSIMM brinda el marco necesario para orientar cada uno de las tareas a realizar. Al asignar dimensiones y niveles se puede categorizar y se logra distinguir cada una de estas tareas.

Una adopción, vista desde el punto de vista técnico es un conjunto de preguntas que determinan el estado inicial y el nivel de madurez de la organización junto con un conjunto de tareas u objetivos que se deben ejecutar con un orden definido para llevar a la organización al nivel de madurez buscado. Estos dos mundos se encuentran en base al ojo de experto o al análisis de un grupo de personas encargado de definir el por qué de los objetivos planteados. Es decir, dentro de un análisis de adopción SOA existen criterios que son difíciles de modelar:

Fuerzas políticas internas de la organización, tendencias tecnológicas y por sobre todas las cosas la experiencia del grupo que lleva adelante la adopción.

Cualquier decisión sin importar la naturaleza puede quedar dentro del modelo de adopción SOA como un objetivo o tarea de un nivel. Esta presencia implica que en base a la información del contexto (cuestionario de adopción) se tomó una decisión a nivel estratégico a la hora de planificar la adopción. Esta decisión en las adopciones SOA tradicionales queda reflejada en la documentación del proceso, representada en formato digital, pero sin una semántica asociada que permita tomar decisiones y realizar deducciones a partir de la información disponible.

El desarrollo busca modelar y disponibilizar el contexto y las decisiones tomadas sobre cada adopción. Facilitar la construcción de la matriz de adopción en base a ejemplos existentes y reducir el esfuerzo que tradicionalmente se ejecuta de manera manual. Compartir con otros interesados los resultados de la adopción SOA y controlar de manera sencilla y rápida el nivel de madurez de la organización.

A través de las adopciones SOA la herramienta enriquece su base de conocimiento. A través de la base de conocimiento estos datos se analizan y se convierten en experiencia. La aplicación utiliza estos datos y provee ayuda al momento de comenzar una matriz de adopción utilizando la experiencia recolectada en otras adopciones.

9 Futuros trabajos

9.1 Búsqueda de patrones de adopción SOA

Al tener los datos almacenados dentro de una base de datos, se puede extender la aplicación en busca de patrones de adopción SOA, situaciones que se repitan en varias adopciones SOA. Un patrón de adopción SOA para el modelo OSIMM se podría definir como: ante un contexto determinado, es recomendable ejecutar una serie de objetivos en un orden establecido. Esto implica extender el desarrollo logrando lo siguiente:

9.2 Hacia un modelo de madurez tecnológico

El modelo OSIMM abarca varios puntos y excede lo tecnológico. Pero, la dinámica de la aplicación permite que cualquier componente tecnológico pueda ser planificado y controlada su madurez a través de una serie de preguntas o indicadores de madurez y una serie de objetivos o tareas a realizar. Realizada esta modificación, se podría seguir la misma dinámica para proponer los patrones de adopción de cierta tecnología o componente tecnológico en una organización.

9.3 Integración con otras herramientas de gestión

La herramienta desarrollada permite controlar la ejecución de un plan de trabajo asociado a SOA. Es probable que cada uno de los objetivos sea ejecutado por un grupo de trabajo en colaboración con otras áreas de la empresa siguiendo la metodología de trabajo preferida de la organización (ágiles, tradicionales, etc). Por lo tanto, una posible extensión de la herramienta pasa a ser el soporte a la gestión de cada una de las tareas, siendo capaz de asignar tareas a responsables y medir el cumplimiento y el grado de ejecución de la adopción de forma más granular.

10 Referencias

- [1]A. Díaz, D. Correal, Service Oriented Architecture: A Model Driven Governance Maturity Classifier. 2011
- [2]Enterprise SOA: Designing IT for Business Innovation, Dan Woods,Thomas Mattern. 2006
- [3]SOA governance: Examples of service life cycle management processes - <http://www.ibm.com/developerworks/webservices/library/ws-soa-governance/index.html>
- [4]SOA Governance. Todd Briske. Packt Publishing. 2009
- [5]The emergence of distributed SOA infrastructure - <http://searchsoa.techtarget.com/tip/The-emergence-of-distributed-SOA-infrastructure>
- [6]Enterprise Architecture & Service Oriented Architecture (SOA). J. Schekkerman - http://www.enterprise-architecture.info/Images/Services%20Oriented%20Enterprise/EA_Service-Oriented-Architecture1.htm
- [7]OSIMM "The Open Group Service Integration Maturity Model" - <https://collaboration.opengroup.org/projects/osimm/>
- [8]Open Source SOA. Jeff Davis. Manning. 2009
- [9]History of the Internet - <http://www.inetdaemon.com/tutorials/internet/history.shtml>
- [10]Diseño estructurado - http://es.wikipedia.org/wiki/Dise%C3%B1o_estructurado
- [11]Análisis Dirigido por Modelos para Identificar Reutilización de Sistemas Legado en Arquitecturas Orientadas por Servicios. Yeimi Yazmín Peña López. Universidad de los Andes. 2010
- [12]Domain Specific Language - http://es.wikipedia.org/wiki/Lenguaje_espec%C3%ADfico_del_dominio
- [13]Capability Maturity Model - http://en.wikipedia.org/wiki/Capability_Maturity_Model
- [14]Towards a Classification of Maturity Models in Information Systems - Mettler, Rohner, Winter. University of St. Gallen. 2010
- [15]SOA Governance Technical Standard - <https://www.opengroup.org/soa/source-book/gov/gov.htm>
- [16]SOA Adoption Levels - SOA Governance Flux IT Framework
- [17]SOA Initial Assessment - SOA Governance Flux IT Framework
- [18]SOA Initial Assessment - Relevamiento - SOA Governance Flux IT Framework
- [19]EMF - <http://www.eclipse.org/modeling/emf/>

11 Figuras

1. <http://en.wikipedia.org/wiki/ARPANET>
2. <http://www.chrisharrison.net/index.php/Visualizations/InternetMap>
3. <http://es.wikipedia.org/wiki/Mainframe>
4. <http://www.mcs.anl.gov/~itf/dbpp/text/node17.html>
5. <http://www.drdoobs.com/windows/software-complexity-bringing-order-to-ch/199901062>
6. http://yourdon.com/strucanalysis/wiki/index.php/Chapter_5
7. http://www.slideshare.net/suks_87/client-server-architecture-presentation
8. <http://tgarhwal.wordpress.com/2009/05/07/logical-building-blocks-in-n-tier-application-architecture/>
- 9, 10. <http://docs.oracle.com/javase/tutorial/java/concepts/object.html>
11. <http://docs.oracle.com/javase/tutorial/java/concepts/inheritance.html>
12. <http://progjava-ala.wikispaces.com/Polimorfismo+en+Java>
- 13, 14. http://en.wikipedia.org/wiki/Component-based_software_engineering
15. http://lifesoftsolutions.com/GEP/Distributed_Applications.aspx
16. <http://oreilly.com/catalog/ordistsys/chapter/ch01.html>
17. <http://www.codeproject.com/Articles/200085/Frictionless-WCF-service-consumption-in-Silverligh>
18. <http://pubs.opengroup.org/onlinepubs/9629399/chap6.htm>
19. http://pt.wikipedia.org/wiki/Chamada_de_procedimento_remoto
20. <http://www.omg.org/gettingstarted/corbafaq.htm>
21. <http://wiki.open-esb.java.net/Wiki.jsp?page=DCOMBC>

22. <http://www.drdoobbs.com/jvm/a-remote-java-rmi-registry/212001090>
23. http://docs.oracle.com/cd/E17904_01/integration.1111/e10223/suite_01.htm
24. http://docs.oracle.com/cd/E17904_01/integration.1111/e10223/02_service_infrastructure.htm
25. http://cscie153.dce.harvard.edu/lecture_notes/2009/20091117/handout.html
26. <http://www.tekspotlight.com/2008/09/27/soa/>
27. <http://simplicable.com/new/the-9-principles-of-soa-design>
28. <http://blog.dlvr.it/wp-content/uploads/2011/03/silo.jpg>
29. <http://www.textalibrarian.com/mobileref/wp-content/uploads/2012/03/integratedlibrarysystems.jpg>
30. http://docs.oracle.com/cd/E16764_01/integration.1111/e10223/images/gsoa_019_0.gif
31. <http://assets.okfn.org/files/talks/media/ext/lego.jpg>
32. <http://i.msdn.microsoft.com/dynimg/IC140003.gif>
33. <http://www.opengroup.org/soa/source-book/osimmv2/business.htm>
34. <http://www.opengroup.org/soa/source-book/osimmv2/organization.htm>
35. <http://www.opengroup.org/soa/source-book/osimmv2/method.htm>
36. <http://www.opengroup.org/soa/source-book/osimmv2/application.htm>
37. <http://www.opengroup.org/soa/source-book/osimmv2/architecture.htm>
38. <http://www.opengroup.org/soa/source-book/osimmv2/information.htm>
39. <http://www.opengroup.org/soa/source-book/osimmv2/infrastructure.htm>
40. <http://www.flickr.com/photos/skemsley/3850431157/sizes/z/in/photostream/>
41. Análisis Dirigido por Modelos para Identificar Reutilización de Sistemas Legado en Arquitecturas Orientadas por Servicios. Yeimi Yazmín Peña López. Universidad de los Andes. 2010
42. Análisis Dirigido por Modelos para Identificar Reutilización de Sistemas Legado en Arquitecturas Orientadas por Servicios. Yeimi Yazmín Peña López. Universidad de los Andes. 2010.

12 Anexo I – Experiencias de adopción SOA personalizando el modelo OSIMM

12.1 Acerca de Flux IT

Flux IT es una empresa que entre otras cosas se dedica a brindar servicios de consultoría en Arquitectura y Middleware. Entre las iniciativas que lleva adelante, está trabajando activamente para llevar SOA a corporaciones y empresas donde lo tecnológico tenga un rol preponderante en el negocio, como empresas del rubro telefónico, bancario y de servicios de salud.

12.2 Framework de adopción SOA de Flux IT

Para llevar adelante adopciones SOA dentro de organizaciones, Flux IT llevó adelante una especialización del modelo de adopción SOA de OSIMM [16] y [17]. Se tomó este modelo como base debido a su naturaleza abierta y además por tener un enfoque “darwiniano” de las organizaciones, donde no es posible conocer toda la información en el inicio y durante la ejecución misma del proceso se van ajustando las tareas a realizar. Es modelo es muy apto para las situaciones de baja certidumbre con las cuales se debe manejar un consultor SOA.

12.3 Modificaciones al modelo OSIMM

En base a las experiencias obtenidas, se determinó que los niveles superiores de la matriz de adopción “Virtualized Services” y “Dynamically Re-Configurable Services” no sumaban valor a la propuesta, al proponer objetivos muy lejanos a la realidad de las organizaciones donde se estaba ejecutando la iniciativa.

Por lo tanto se tomó la definición del modelo OSIMM y se creó una versión personalizada del modelo que se denominó “Flux IT SOA Governance Framework”.

La extensión del modelo modificó la matriz de adopción y extendieron las preguntas del cuestionario fijándole un perfil tecnológico orientado al stack tecnológico J2EE y productos open source. Se amplió el espectro social o político del modelo, incluyendo preguntas que describían o evaluaban las relaciones del área que llevaba adelante la iniciativa con el resto de la organización [18]. En estas preguntas se medía el poder político y los contactos o alianzas que se podían llevar a cabo para facilitar la adopción SOA.

Con respecto al negocio si bien SOA es un paradigma muy relacionado con los requerimientos de negocio, se agregaron nuevas preguntas que describen la relación de la tecnología con el negocio de la organización y se trabajó para dotar al modelo de indicadores de retornos de inversión.

12.4 Experiencias

Durante las consultorías llevadas adelante en Flux IT se detectaron ciertos patrones de comportamiento recurrentes, como negación a colaborar con otras áreas, cierto temor a presentar el caso a los actores de negocio de la organización y disputas de poder internas que afectaban a la iniciativa.

Una forma de promover que los interesados o los actores de la compañía rompan con su aislamiento es promover objetivos de tipo “político” en la adopción SOA. A partir de ese momento y formando consensos se iban logrando avances para involucrar a las demás áreas en la iniciativa.

Por el lado tecnológico los objetivos también buscaban atender los síntomas o dolores más recurrentes de las grandes organizaciones. Por ejemplo, estandarización del stack tecnológico, adoptar técnicas de calidad de código y promover la mejora continua de estas métricas, promover la participación de expertos técnicos para el análisis de las aplicaciones mejorando así su calidad técnica.

12.5 Aplicación y problemática SOA

Durante las instanciaciones correspondientes al Flux IT SOA Governance Framework se detectó que gran parte de las soluciones tecnológicas eran recurrentes en diferentes clientes, ya que en las tres primeras etapas de maduración denominadas “Service Foundation Levels” la evolución es similar.

Dentro de Flux IT se sigue una política de rotación de responsabilidades de las personas. Con lo cual los encargados de llevar adelante un “assessment SOA” o una adopción SOA no eran siempre los mismos. Una vez terminado el proyecto o superada cierta etapa se debía promover la transferencia de conocimiento de una persona a otra. Con lo cual el reemplazante debía adquirir una cantidad importante de conocimientos sobre la organización en poco tiempo. Todas las vivencias no podían ser transmitidas de manera efectiva y la documentación existente no reflejaba fielmente el por qué de estas decisiones. Además hay que considerar que una vez que finalizado el proyecto, el material es catalogado y almacenado, pero la información que existe dentro de estos documentos no puede ser utilizada de manera activa.

12.6 Conclusión

El modelo OSIMM fue adaptado para tener un horizonte más corto y por lo tanto un roadmap más acotado con respecto a los objetivos a cumplir dentro de la organización. Se promovieron indicadores reales y discretos de negocio para que la iniciativa SOA dentro de la organización presente resultados tangibles acerca de las mejoras que propone. Las modificaciones buscan facilitar la adopción SOA por sobre todo en la parte política y social, donde las personas e intereses contrapuestos entran en juego, pero sin descuidar lo tecnológico soporte vital para este tipo de proyectos.