Universidad Politécnica de Cataluña
Departamento de Lenguajes de Sistemas de Información LSI.

Estudio Ágil Aplicado a los Métodos de Selección de Componentes Commercial Off-The-Shelf (COTS)

Autor:
Navarrete Ramirez Fredy Javier

Con la colaboración:
Pere Botella
Xavi Franch

Enero 2006

Resumen

En la actualidad, las organizaciones tienden a construir sus Sistemas de Información obteniendo componentes adquiridos o licenciados desde el mercado. Esta práctica es cada vez más usual en la industria, ya que son pocas las compañías que cuentan con el tiempo y la capacidad para competir con el mercado y su constante evolución. Por otra parte, es cada vez más creciente la aceptación de los métodos ágiles como eXtreme Programming (XP), dentro del contexto del desarrollo a medida, por parte de la comunidad informática e investigativa.

El contexto de estudio de los componentes COTS, esta siendo en la actualidad motivo de talleres y arduas sesiones de trabajo por parte de la comunidad científica. Son varias las propuestas metodológicas que al respecto se han sugerido, y que están basadas en procesos en cascada, o desarrollos iterativos adaptados a contextos de negocio específicos. Dentro de este contexto COTS, es muy poco lo que se ha estudiado acerca de la agilidad de las propuestas metodológicas para seleccionar e integrar componentes COTS, así como la posible extrapolación de los conceptos ágiles al domino de los sistemas basados en componentes COTS.

Esta propuesta se enfoca en esta dirección. Busca extrapolar los conceptos del mundo ágil como lo son sus valores y principios, al mundo y dominio de los componentes de software COTS, queriendo ser un punto de partida, para una propuesta formal de estudio de la agilidad dentro del contexto COTS.

Tabla de contenido

1 Introducción

A continuación en esta sección se pretende establecer una parte introductoria para el lector, con el fin de familiarizarlo a los conceptos que en este trabajo se tratan.

1.1 Introducción a los componentes, Commercial-Off-The Shelf (COTS)

Con el paso del tiempo, el significado de componente reutilizable, se ha visto ligado al significado de componente de software COTS. Aunque difieren en varias características, los COTS, han emergido como resultado de diferentes factores que han impulsado el desarrollo de los sistemas de información basados en componentes de software. Retrocediendo en el tiempo, vemos como en [18], se comienza a introducir el concepto de componente de software, pero en [19], se destaca que hasta los mediados de los 80's, el incremento de la complejidad en las aplicaciones de los sistemas de software, obliga a que se comience a dar la importancia relevante a los componentes de software. Aparte del factor complejidad, destacado anteriormente, existieron otros factores y actores que dieron impulso a la idea de contar con componentes de software dentro de los sistemas de información. Un factor fundamental ha sido la estructura cambiante del mercado. Heineman T., se refiere al mercado de los componentes, como un mercado que se asemeja mucho a un bazar de mercado, en ves de ser un verdadero mercado de componentes [20]. Lo cual ha significado una manera diferente de obtener piezas de software desde el mercado. Este cambio supone nuevas dependencias y nuevas formas de seleccionar e integrar el componente de software adecuado.

Otro factor que ha contribuido a la inclusión de los componentes COTS dentro de la industria, ha sido la masiva información de la cual se puede tomar cuenta desde los servicios de información basados en Web y las aplicaciones e-commerce [19].

En este punto es importante encontrar las diferencias y similitudes entre un componente COTS y un componente de software. En [19] se recopilan definiciones aceptadas, acerca del significado de componente de software, pero para nuestro objetivo de identificar un componente que puede ser adquirido desde un segmento de mercado independiente y ajeno a la organización, tomaremos la definición citada en [8]:

> *...defining COTS components as those in which no source code is provided with the product (separate from any required API software).*

A continuación citamos diferentes términos y nomenclaturas relacionadas al contexto de los componentes COTS, extraídas también de [8]:

- API – Aplplication Program Interface.
- Black Box COTS – componente COTS, cuyo código interno no puede ser modificado.
- COTS – Commercial-Off-The-Shelf
- COTS Assessment/Qualifying – determinación de la viabilidad del potencial de un componente COTS para ser utilizado en una aplicación del sistema.
- GFE – Government Furnished Equipment.
- GFS – Government Furnished Software.
- GOTS – Government-Off-The-Shelf.
- MOTS – Modified-Off-The Shelf.
- NDI – Non-Developmental item.
- NOTS – Not-Off-The-Shelf.
- OTS – Off-The-Shelf.
- Reuse software – son los componentes de software hechos a medida, u obtenidos de terceros, que son provistos con el código fuente.
- ROTS – Research- Off-The-Shelf.
- White Box COTS – son componentes que permiten realizar modificaciones internas ha su código fuente.

Para nuestra fuente de estudio, nos basaremos en los componentes *Black Box COTS*, que son los componentes COTS que más abundan en el mercado.

1.2 Introducción a los métodos ágiles

Cuando se culmina un proyecto de software, gran parte de su fracaso se otorga a la falta de procesos definidos, que no son capaces de guiar y reducir esfuerzos dentro del proyecto de desarrollo [21]. Pero en la actualidad son muchos los métodos, modelos y estándares propuestos para ser aplicados en el ámbito de la ingeniería del software, los cuales soportan modelos de calidad, y frameworks que enriquecen a los procesos de desarrollo, por ejemplo dentro de los estándares definidos por la IEEE podemos encontrar varios ejemplos [22]. Un amplio número de estos estándares y propuestas no son aplicados en su totalidad por la industria por diferentes motivos, bien sea por desconocimiento de los mismos, por su dificultad para implementarlos, o por que

simplemente no se adaptan a la realidad de la empresa [23]. Por consiguiente, siempre se ha pretendido buscar desarrollar métodos que se adapten a las necesidades de la industria, con el fin de que sean acogidos por ella para probarlos y contribuir así a su mejoramiento y ajuste de los procesos internos de cada organización.

Esta puede ser una de las causas del por que, a comienzos del año 2001, se reunió a un grupo de expertos de la industria del software, para identificar valores y principios que permitieran a los equipos de desarrollo de software, generar rápidamente sus aplicaciones, y preparar a los equipos de trabajo para que hicieran frente a los continuos cambios que sufren los sistemas de información, con tal de poder adaptarlos a las exigencias de la industria. El resultado de esta reunión fue el manifiesto de la "Agile Alliance" [7].

En la actualidad los métodos ágiles que se basan en los principios y valores ágiles están jugando un papel importante en las prácticas de la ingeniería del Software [21, 24], y métodos como eXtreme Programming (XP) [25], y Scrum [26], han sido adoptados por muchas organizaciones y desde diferentes equipos de desarrollo se han reportado resultados satisfactorios de diferentes experiencias y resultados negativos, como suele suceder en todos los métodos.

2 Alcance de la propuesta

Este estudio pretende analizar los métodos de selección de componentes COTS, con respecto a los métodos ágiles. Se analizará si las propuestas metodológicas de selección de COTS, están influenciadas de alguna manera por los valores y principios ágiles, consignados en el manifiesto ágil [7].

Además, se pretende con este trabajo, poder ser un punto de referencia para la creación de un método ágil que abarque la totalidad de los ciclos de vida de un sistema de información de componentes COTS. Es así, como con el análisis posterior, y las conclusiones obtenidas de este trabajo, se hará mención a la necesidad que puede existir de extrapolar los contenidos ágiles dentro del contexto de selección de componentes COTS para generar dicho método.

3 Lectura del trabajo

El análisis comienza con la extrapolación de los valores y principios ágiles al contexto de los componentes COTS. A continuación, se entrará a analizar una a una, las propuestas de selección de componentes COTS más citadas, con el fin de observar la influencia que tienen sobre ellas los principios ágiles. En seguida, se expondrá el resultado final de la evaluación de los 6 métodos de selección de componentes COTS estudiados. Finalmente se encontrarán las conclusiones de este trabajo.

4 Métodos ágiles desde la perspectiva de COTS

El objetivo de este estudio es analizar los conceptos ágiles desde la perspectiva de los componentes COTS, extrapolando los valores y principios ágiles al dominio de los componentes COTS. Este análisis será un referente para conocer como están influenciados los métodos de selección de componentes COTS por los conceptos ágiles, y si puede llegar a ser necesario crear un método nuevo que este fundamentado por los principios ágiles. A continuación observaremos la influencia de estos conceptos ágiles analizando como primera medida los valores ágiles y a continuación los principios ágiles, los cuales están consignados en el manifiesto ágil [7].

4.1 Valores Ágiles aplicados al contexto de los procesos de selección de componentes COTS

Los valores ágiles son el punto de partida de nuestro análisis, hacia el estudio de la extrapolación del mundo ágil al contexto de los componentes COTS. En este apartado se pretende analizar los 4 valores ágiles desde una perspectiva orientada al dominio de los COTS:

V.1 *"**Individuals and interactions** over processes and tools"*

Los sistemas son usados por los individuos de una organización, y son ellos quienes explotarán todas las ventajas que ofrezca la arquitectura del sistema. Por esta razón se hace imprescindible tener en cuenta el factor humano dentro el desarrollo del proyecto COTS.

El humano participa en casi la totalidad de las fases dentro de un procesos de selección de un componente COTS (selección, evaluación, integración y mantenimiento), por esta razón es un aspecto relevante dentro de un proyecto de integración de COTS, el cual que no debe ser tomado únicamente con la etiqueta de "*agente*", o ser visto simplemente como una herramienta para lograr un objetivo. La capacidad del individuo dentro de un equipo de trabajo, y su socialización dentro del entorno operativo, influirá sobre un desarrollo satisfactorio del proyecto. Preocuparse por crear un equipo de trabajo que sea capaz de adaptarse a los cambios y a un desarrollo sostenible debe ser relevante dentro del proyecto, contando con integrantes que posean un conocimiento verificable, vital para que los procesos de selección de COTS se desarrollen bajo

bases técnicas, aportando calidad a la ejecución de los procesos. Además, los integrantes del equipo deben poseer la habilidad de comunicarse e interactuar con los demás integrantes del equipo, dejando a un lado conductas individualistas que impacten negativamente sobre el proyecto.

V.2 "***Working software*** *over comprehensive documentation*"

Se podría pensar que este valor no se puede extrapolar en su totalidad al contexto de los COTS, debido al poco código que puede llegar a desarrollarse durante el proyecto de selección del componente, con respecto al código desarrollado durante los proyectos de desarrollo de software. Pero en los proyectos de selección de COTS, se debe tener en cuenta el "*glue-code*" necesario para integrar el componente COTS a la arquitectura del sistema de información [8]. Victor Basili y Barry Boehm en [9], aportan una tabla comparativa de esfuerzo, donde se aprecia como la actividad de desarrollo del *glue-code* requiere un esfuerzo superior en comparación de diferentes actividades dentro de los procesos de selección de componentes COTS. Incluso, se llega a formular una hipótesis donde se considera que una línea de *glue-code* requiere un esfuerzo 3 veces superior a una línea de código de una aplicación de desarrollo a medida. Esto indica que el proceso de desarrollar *glue-code* puede estar influenciado perfectamente por las prácticas ágiles, con el fin de realizar desarrollo y pruebas continuas, para reducir el esfuerzo del desarrollo.

En general la idea de este valor, dentro del contexto de los COTS no se centra en trabajar en software, en cambio de trabajar en integrar, ya que este es el objetivo principal de un proyecto COTS. El pensar en integrar requiere no pensar demasiado en el mañana, pretendiendo que el sistema de información sobre el cual se incorporará el componente se adapte a las necesidades actuales, evitando integrar pensando en el futuro. Para este fin debemos realizar iteraciones al momento de seleccionar, evaluar, integrar y mantener al sistema de información. Los proyectos COTS pueden manejar rangos amplios de integración de componentes, los cuales ejercen dependencias entre si. Por eso las iteraciones deben ser controladas y deben de manejar estas dependencias con el fin de poder entregar resultados tempranos

V.3 "***Customer collaboration*** *over contract negotiation*"

Los métodos ágiles consideran vital la relación entre los desarrolladores y los clientes o usuarios del sistema [10], por lo cual esta relación es manejada por contratos para asegurar los cambios y las necesidades de los usuarios "*User Story*". En un proyecto COTS también es vital trabajar de la mano con el usuario final del sistema, ya que es la persona que solicita la incorporación de un nuevo componente/s dentro de su sistema de información, y finalmente será un usuario final quien use y opere el sistema final. Además, otra de las principales razones para trabajar de la mano con el usuario final, es la de poder realizar la negociación de los cambios de los requisitos dentro del proyecto de selección del COTS. Este proceso es vital, ya que los segmentos del mercado donde se desarrollan los componentes COTS, están en constante dinamismo y evolución, por lo cual los requisitos de los usuarios se enfrentan a una constante negociación.

Por otra parte, dentro del proyecto debemos tener en cuenta al proveedor de los componentes. Esto es debido a que la funcionalidad de los componentes COTS, en su mayor medida es provista por los proveedores de los COTS. Si contamos con el vendedor en la fase de integración del componente, podremos tener más opciones de personalizar nuestro componente y de obtener por ende una mejor asistencia al momento de realizar su mantenimiento. Así mismo, debemos tener en cuenta la importancia de trabajar con un rol que sea capaz de defender los intereses de la organización, al momento de adquirir o licenciar un componente COTS desde el mercado.

V.4 "**Responding to change** over following a plan"

Los segmentos del mercado donde se desarrollan los componentes COTS y el sistema de información con el que cuentan las organizaciones están en constante evolución. Por una parte, el mercado evoluciona rápidamente, por las presiones de la competencia y por las presiones del tiempo a la que se ven sometidos los distribuidores o proveedores de componentes COTS. Así pues, los avances tecnológicos proveen desarrollo y evolución a los componentes fabricados, lo cual sugiere continuas *releases* o versiones constantes de los productos desarrollados, que son puestos en oferta en los correspondientes segmentos del

mercado de componentes COTS. Y por otra parte los sistemas de información, se actualizan con una frecuencia regular, adquiriendo o licenciado los componentes que provienen del mercado, ya que las presiones de la competencia y las necesidades de la industria obligan a las organizaciones a contar con un sistema de información capaz de soportar la evolución del negocio.

Entre tanto dinamismo, se torna difícil tomar decisiones al momento de realizar un cambio que afecte directamente sobre el desarrollo del proyecto. Es por lo cual, se debe contar con un equipo de trabajo que responda a estos cambios con determinación, en el momento de tomar decisiones que impacten en la dirección del proyecto de integración del componente COTS [11].

4.2 Principios Ágiles aplicados al contexto de los procesos de selección de componentes COTS

De los valores ágiles estudiados en la sección anterior, se derivaron 12 principios ágiles, que aportan un significado importante a los métodos ágiles. Dentro de un primer análisis y una primera revisión de estos 12 principios, se optó por descartar aquellos principios ágiles bien sea por que no aplicaban, o no mostraban dependencia sobre el contexto de selección de los Componentes COTS que aquí se cita. Esos principios ágiles descartados son:

"*Our highest priority is to satisfy the customer through early and continuous delivery of valuable software*".

Dentro de los procesos de selección de componentes COTS, el desarrollo y las entregas constantes de Software no influencian en gran medida la evolución y el desarrollo de estos procesos. Esta es la razón por la cual este principio no influencia en gran medida el contexto de selección de un componente COTS, a pesar de que en fases posteriores a la selección, se requiera ensamblar el componente seleccionado dentro del sistema, desarrollando código g*lue-code* para su integración.

"*The most efficient and effective method of conveying information to and within a development team is face-to face conversation*".

La naturaleza semántica de este principio no parece tener relevancia importante dentro del contexto de selección que estamos analizando. Aunque dentro del estudio se hace patente la necesidad de trabajar con los diferentes actores y seres humanos que participan dentro de los procesos de selección de nuestro proyecto. Además, en

el contexto de este estudio, los equipos de trabajo están más enfocados a la selección que al desarrollo.

"*Working software is the primary measure of progress*".

Al igual que el primer principio analizado, este principio por su naturaleza de desarrollo de Software no parecer influenciar a los procesos de selección del componente COTS. Dentro de el contexto de estudio podrían existir deferentes índices para medir nuestro progreso, bien sea como el trabajar en integrar o en seleccionar adecuadamente un componente candidato. Pero es claro que dentro de un proyecto de selección de COTS, el trabajar en software no es una medida para analizar nuestro progreso con respecto a la selección del componente.

"*The best architectures, requirements, and designs emerge from self-organizing teams*".

Al igual que el Segundo principio aquí citado, la naturaleza semántica de este principio no parece influenciar el contexto de selección de un componente adecuado. Los factores externos a los que se ve sometido un proyecto COTS (p.e., las restricciones del mercado, el trato con los vendedores, las disposiciones legales de los productos, entre otros factores), hacen evidente que para seleccionar un componente que se ajuste a las necesidades del usuario, no se depende estrictamente de la organización para obtener un resultado satisfactorio. Aunque dentro de este análisis, se remarca la necesidad de conformar un equipo especializado para llevar a cabo el proyecto COTS.

A continuación, se analizará el resto de los principios ágiles que pueden ser extrapolados y aplicados dentro de los procesos de selección de componentes COTS. Para esta parte del análisis se han remarcado en negrita aquellas palabras que se han considerado claves dentro de los principios ágiles, para ser tomadas como referente del análisis. Además, cada principio es identificado con la letra *P* seguida de un índice, que refiere al orden de aparición dentro del estudio.

P1 *"Welcome* ***changing requirements****, even late in development. Agile processes harness change for the customer's competitive advantage"*

Los segmentos del mercado donde son desarrollados los componentes COTS, y la tecnología inmersa dentro de un sistema de información, están en constante

evolución y dinamismo, cambiando rápidamente. En la actualidad, los componentes COTS son adquiridos o licenciados desde un segmento del mercado especializado, y sus requisitos pueden ser tomados desde diferentes fuentes de información [12], razón por la cual, los requisitos deben ser flexibles, con el fin de poder adaptarse al estado actual del mercado. En [13], se menciona que el 31% de los proyectos estudiados apuntan a la necesidad de hacer más flexible los requisitos en la fase de definición. Esto se debe a la necesidad que existe de ajustar los requisitos de los usuarios, a las características reales de los componentes dentro del mercado. Por otra parte en los proyectos basados en COTS, los objetivos de los *Stakeholders* tienden a permanecer estables, y los requisitos a través de los cuales podemos alcanzar los objetivos, sufren modificaciones durante el desarrollo del proyecto [14]. Es por esto que una parte importante de los proyectos COTS, es entender el "por que" del proyecto, por encima del "que" de las características requeridas de los sistemas basados en componentes COTS [15, 16], con el fin de poder adaptarse a la evolución del mercado.

***P2** "**Deliver** working software **frequently**, from a couple of weeks to a couple of months, with a preference to the shorter time scale."*

En [17], Larman y Basili demuestran que la idea de la iteración y el desarrollo incremental (IID) no es un aporte exclusivo de las nacientes metodologías ágiles. El IID ha sido base de varios procesos bien fundamentados y conocidos que datan de décadas pasadas. Dos muestras claras son los Unified Process (UP), y el UML, donde el modelo de desarrollo de los procesos es iterativo [27], y donde por cada iteración se incluyen prácticas tales como el modelado del negocio, la captura de los requisitos de los usuarios, el análisis y el diseño, la implementación, las pruebas funcionales y el desarrollo. La aplicación de cada una de estas prácticas, cuentan con diferentes porcentajes de aplicación, lo cual aportan madurez al artefacto producido. Así mismo, las entregas frecuentes pueden ser aplicables a los sistemas basados en componentes COTS, donde los procesos de selección puedan contar con iteraciones bien fundamentadas, para realizar la exploración del mercado, analizar los requisitos de los usuarios, evaluar y seleccionar los COTS candidatos, y donde podamos integrar y mantener el sistema por completo. Más sin embargo, para integrar un componente COTS al sistema existe una fuerte restricción que se debe tener en cuenta, y es la dependencia generada ente los diferentes componentes COTS que se

han de integrar, ya que por lo general en un proyecto COTS se integran más de un componente. Con el propósito de realizar un *deliver frequently*, debemos procurar que las iteraciones tengan en cuenta estas dependencias, además, de los requisitos de integración, ya que estos también juegan un papel importante al momento de seleccionar un componente adecuado [28].

P3 *"Business people and developers must* ***work together*** *daily throughout the project."*

En los proyectos de software convencionales de desarrollo a medida, las metodologías ágiles han identificado la necesidad de trabajar en conjunto con los clientes a través del proyecto, donde actividades tales como la validación y la negociación de los requisitos son importantes para obtener el resultado deseado. Dentro de los proyectos COTS esta necesidad también esta presente. Además, podemos identificar otro actor que es importante, debido a la fuerte dependencia que ejerce sobre la funcionalidad del componente COTS: el vendedor o proveedor del componente. Los repositorios de información de características de componentes COTS, obtienen sus datos en gran parte por la información que proveen los vendedores del componente. Por está razón, resulta conveniente el interactuar y trabajar conjuntamente con los proveedores de los componentes, incluso debemos tener en cuenta que en la fase de integración de los componentes al sistema, se debería integrar al vendedor al equipo COTS. Trabajar conjuntamente con el proveedor, puede proveer beneficios conjuntos tanto para la organización como para los vendedores. Los vendedores pueden aprender sobre su producto y sobre la capacidad de su componente al momento de integrarse a un sistema de información [13], y la organización puede tener la opción de personalizar el componente y de recibir una mejor asistencia por parte del proveedor [29], especialmente si el cliente del proveedor es importante. Además, de la inclusión del vendedor, se hace indispensable contar con un rol que sea capaz de manejar los aspectos contractuales y legales del contracto con el vendedor, con el fin de velar por la funcionalidad ofrecida por el vendedor, y cuidar por la asistencia e intereses de la organización

P4 *"Build projects around* ***motivated individuals****. Give them the environment and support they need, and trust them to get the job done"*

Los sistemas de información son diseñados y usados por humanos, es por esto que las metodologías ágiles se centran en el factor humano como punto clave de desarrollo de sus procesos, donde aspectos como: el talento, las habilidades, cordialidad, sociabilidad del individuo y la comunicación son resaltados dentro de los equipos ágiles [24]. Así mismo, la cultura de la organización y la gente que la compone son la piedra angular del uso del sistema [30]. Por esta razón, las buenas relaciones dentro de un equipo de trabajo, las relaciones internas, la capacidad de comunicación y las diferentes interacciones con otros componentes del equipo, influencian en un porcentaje importante el desarrollo del proyecto de selección, donde el factor humano y el factor organizacional juegan un rol importante [31]. En este orden, son pocas las metodologías de selección de componentes COTS que tienen el factor humano como objetivo preponderante del éxito de sus procesos, así como también son pocas las metodologías que proponen roles específicos para conformar un equipo especializado, olvidando que en los proyectos de selección de componentes COTS, se generan nuevas responsabilidades y nuevas actividades, que deben ser cumplidas por nuevos roles [32, 33]. En la medida de lo posible estos roles propuestos para un equipo COTS, deben poseer la habilidad y el conocimiento verificable de todos los miembros que componen el equipo, con el fin de asegurar su calidad y la de los procesos que desarrollan.

P5 *"Agile processes promote* ***sustainable development****. The sponsors, developers, and users should be able to maintain a constant pace indefinitely."*

Los procesos de selección de componentes COTS no debes ser excesivos ni agotadores, esto con el fin de conservar un flujo de trabajo que no debilite ni deteriore la paz interna del equipo de trabajo. Es por esto que estos flujos internos de trabajo deben poseer procesos constantes e iterativos, capaces de realizar constantes *feedbacks*, buscando de aprender de los procesos desarrollados, por ejemplo, realizar selecciones iterativas de componentes, evaluaciones iterativas, actualizaciones iterativas del sistema, y demás procesos donde sea posible aplicar y reutilizar el conocimiento adquirido a través del tiempo.

Para lograr un desarrollo sostenible dentro del proyecto, debemos tener en cuenta la arquitectura del sistema y el repositorio de información. La arquitectura del sistema es utilizada como pilar sobre el cual los componentes serán integrados al sistema, generando restricciones y posibilidades al momento de integrar y seleccionar un componente adecuado. Razón por la cual, al momento de realizar las primeras fases de planeación y desarrollo del proyecto, se debe trabajar en paralelo en la descripción y actualización del sistema de información. Por otra parte el repositorio de información deberá contener la experiencia acumulada durante los proyectos, buscando ajustar la conducta y comportamiento del equipo de trabajo durante el desarrollo del proyecto. El repositorio será analizado en ***P8***.

P6 *"Continuous attention to* ***technical excellence*** *and good design enhances agility."*

"High quality is the key to high speed" [21]. En cada fase de los procesos de integración de componentes COTS, la calidad técnica de los miembros que componen el equipo de trabajo, es vital para obtener un resultado de alta calidad en cada fase de los procesos respectivos donde participan, o donde su rol exige responsabilidades. Por ejemplo para obtener una especificación clara de los requisitos de los *Stakeholders*, debemos contar con un ingeniero de requisitos dentro del equipo de trabajo, que sea capaz de esclarecer y guiar este proceso de especificación de requisitos. Al igual sucederá, si deseamos obtener una evaluación satisfactoria de un componente, tendremos que contar con evaluadores expertos en la herramienta a evaluar, y que además, controlen el dominio y la funcionalidad del componente deseado por los usuarios. Por otra parte, si nuestros procesos son guiados por técnicas que brinden un background, y que estén estandarizadas, aportarán calidad a nuestros diferentes procesos. Si llegamos a utilizar técnicas como el modelado orientado a objetivos [15, 16] y la negociación Win-Win [34], en nuestra ingeniería de requisitos, y técnicas de decisión de multi-criterio como AHP [35], para realizar la evaluación de los componentes, podremos obtener resultados de alta calidad. El problema de apoyar a los procesos con un conjunto determinado de técnicas, puede surgir si estas son complejas y requieren un adiestramiento especial para su entendimiento. Además, si para llevarlas a cabo se requiere de una

documentación excesiva, podemos afectar el rendimiento y los objetivos del proyecto.

P7 *"**Simplicity**--the art of maximizing the amount of work not done--is essential."*

Este principio es uno de los que puede entrar en conflicto con otro de los principios que aquí se analizan. El nivel de simplicidad puede entrar en conflicto con la excelencia técnica. Como se mencionó en el principio anterior ***P6***, la excelencia técnica puede estar guiada de modelos ricos pero complejos y robustos, lo cual pueden dificultar su uso y el rendimiento del proyecto, debido al adiestramiento necesario para su utilización. Un punto de partida para lograr que la simplicidad y la excelencia técnica trabajen juntas y no compitan entre si, es centrarnos en los requisitos relevantes, aquellos que realmente discriminen entre los candidatos potenciales. Estos requisitos deben aporten significado al criterio de evaluación y al desarrollo del proyecto, descartando o clasificando con menor prioridad, aquellos criterios inaplicables que pueden entorpecer el desarrollo de la selección.

Otro conflicto que aparece al aplicar este principio de simplicidad, se encuentra al considerar el desarrollo sostenible ***P5*** y las reflexiones regulares ***P8***. Para poder cumplir con estos dos principios, requerimos invertir tiempo y esfuerzo para alcanzarlos, y el desarrollo de estos dos principios se basa en procesos que van más allá de la simplicidad. El punto clave, está en no pensar demasiado en el mañana, trabajar en el futuro únicamente cuando el futuro pueda ocurrir, sin intentar anticiparnos al mañana y preocuparnos por lo que suceda con el día a día. Esto se puede lograr si hemos concebido una arquitectura y un sistema flexible, que se pueda adaptar fácilmente a los cambios y avances del mercado que evoluciona rápidamente, y que hace que la descripción de un componente pueda ser obsoleta en corto tiempo. En adición, para lograr un grado óptimo de simplicidad, debemos evitar en la medida de lo posible prácticas y actividades que entorpezcan el desarrollo del proyecto, y que no tengan un impacto sobre los procesos, por ejemplo utilizar una documentación excesiva que no será consultada.

P8 *"At **regular** intervals, the team **reflects** on how to become more effective, then tunes and adjusts its behavior accordingly"*

La reflexión regular es un principio que se adapta perfectamente al mundo de los COTS. Los procesos y métodos de selección de componentes COTS están en proceso de maduración y aprendizaje. Aprovechar el conocimiento y el aprendizaje de estos nuevos procesos y métodos, es una necesidad que nos permitirá utilizar este conocimiento en favor de nuestros proyectos, ayudando a nuestro equipo de trabajo, a ajustar su comportamiento y conducta frecuentemente, de cara al proyecto. En el modelo de COCOTS [36], así como en otros métodos, se propone utilizar un repositorio de información, con el fin de almacenar este conocimiento que se va adquiriendo, durante el proyecto. Además, este repositorio cumpliría una función importante, ya que almacenaría información realizando feedbacks constantes, de la descripción de los requisitos de los *Stakeholders*, de la descripción de los componentes del mercado, de los resultados de las evaluaciones, del por que de rechazo de un componente COTS, de las valoraciones a los proveedores, ente otros datos. En este punto la simplicidad se vera afectada si nuestros procesos para almacenar esta información en el repositorio, son lentos y están exageradamente documentados. Además, hay que evaluar concienzudamente que información será imprescindible para el proyecto, y cual no lo será. Esto es con el fin de no almacenar ni documentar información que nunca se utilizará. Como apunta Martin Fowler en [50], la decisión de utilizar diagramas o cualquier tipo de documentación, depende de cual útil sea para el proceso, si brinda comunicación, o si el diagrama o documento elaborado es utilizado por alguien, en caso contrario, lo mejor es no invertir tiempo en actualizar, ni utilizar dicho diagrama o documento. En adición, se puede incluir un rol en el equipo de trabajo, que sea responsable de realizar el almacenamiento de esta información en el repositorio, con el fin de agilizar el proceso de almacenamiento de la información.

5 Análisis de los métodos actuales de selección de componentes COTS

Actualmente en el campo de investigación de los componentes de software COTS, existen varias metodologías que han sido propuestas: CARE [12], SCARLET [37], OTSO [38], PECA [39], STACE [40], EPIC [41], entre otras. En su gran mayoría estas

propuestas están orientadas a los procesos de selección, son muy pocas las se enfocan en los procesos de integración y mantenimiento, específicamente en un ciclo total de vida del sistema de información. Estas metodologías sugieren procesos y prácticas para la selección de componentes COTS, como lo son los métodos de evaluación, técnicas de toma de requisitos, de adaptación e integración, entre otros procesos que están basados en procesos de metodologías de ciclo de vida tradicional y procesos iterativos. Debido a la naturaleza de los componentes COTS que aún está por madurar, se puede deducir que nuevas actividades y nuevos roles son necesarios en contraposición de las metodologías tradicionales [32, 33]. Las empresas de hoy en día, están invirtiendo más tiempo del necesario en los procesos de integración de componentes COTS al sistema [42]. A continuación en esta sección, veremos un breve repaso de las metodologías COTS más aceptadas, y abarcaremos el análisis de estos métodos. Este análisis consta de una evaluación de los 8 principios ágiles, frente a las metodologías de componentes COTS. El enfoque principal se centra en considerar si los métodos actuales de selección de componentes COTS tienen en cuenta los principios ágiles estudiados previamente, y si estos pueden influenciar positivamente o negativamente a estos métodos. Para esta evaluación se ha analizado si los principios están presentes en las metodologías de selección de componentes COTS, o si existe una carencia o ausencia de estos principios ágiles. Además, el análisis esta apoyado en el diseño de una tabla que consta de 9 columnas, las cuales están marcadas en su inicio con la letra ***P*** que refiere al principio estudiado, y en seguida se encuentra el número del principio al cual se hace referencia según el orden visto en la sección de principios ágiles. Las filas de esta tabla, extractan palabras claves de que son encontradas en cada metodología, las cuales tratan de resumir los fundamentos de cada método. La idea de esta tabla es comparar cada principio frente a los fundamentos de cada método COTS, si se encuentra presencia del principio ágil en el método la celda será marcada con '+', si en cambio encontramos una ausencia del principio ágil en el método la celda será marcada con '-', pero en cambio si el impacto no es fácilmente diferenciado se marcará con '+ - '

5.1 COTS-Aware Requirements Engineering Technique (CARE)

La propuesta metodológica CARE [43], se basa en las metodologías RUP [44], MBASE [45] y PORE [46]. CARE cuenta con una metodología de ingeniería de requisitos que es

orientada a agentes y orientada a objetivos, además, está basada en el conocimiento y su reutilización.

CARE es "*Agent-oriented*" (orientado a Agentes), ya que considera propiedades dentro de los actores de los procesos, que caracterizan a los agentes. Así mismo, el método CARE es "*Goal-oriented*" (orientado a Objetivos), por que los requisitos de alto nivel son tratados como objetivos de alto nivel para el sistema, donde se tienen en cuenta diferentes maneras para lograr los objetivos. La propuesta CARE está basada en el conocimiento, por que utiliza un repositorio que es alimentado con la descripción de los componentes. Se pretende almacenar y mantener el repositorio en dos niveles de abstracción: por los objetivos (*softgoals* y *hardgoals*) y por la especificación detallada de los componentes (funcional y no funcional). Con este enfoque, CARE pretende proveer un nivel alto de descripción del componente, teniendo en cuenta que el repositorio debe contar con un mantenimiento para obtener un resultado satisfactorio. Así, CARE utiliza ambos tipos de descripciones, la de los productos y la descripción de los objetivos.

5.1.1 Análisis de CARE

P1. El método reconoce que los objetivos y los requisitos de los *Stakeholders* pueden ser modificados durante el desarrollo de sus procesos. Además, permite que algunos de los roles propuestos negocien los requisitos en diferentes etapas de sus procesos durante el desarrollo del sistema;

P2. Organiza los procesos de elección, análisis, corrección y variación de objetivos iterativa mente, pero no es claro si se puede realizar evaluaciones o múltiples selecciones frecuentemente, con el fin de observar resultados tempranos;

P3. Aunque almacene y mantenga datos de los vendedores en su repositorio de datos, no tiene en cuenta la interacción en el equipo de trabajo del vendedor, o no destaca la importancia de interactuar con él;

P4. CARE no da importancia al factor humano en sus procesos. Considera al humano como un agente software o agente hardware, el cual posee intencionalidad. Aunque CARE, propone tres roles dentro de su método, no recomienda la conformación de un equipo para el desarrollo de un proyecto COTS;

P5. Este método cuenta con una arquitectura céntrica, la cual provee un medio para realizar un desarrollo sostenible, al igual que esta soportada por unos roles técnicos, los cuales interactúan entre si secuencial mente, y realizan una negociación de los requisitos continua;

P6. El método propone tres roles técnicos, los cuales deben aportar experiencia según el perfil del rol. Además, dentro de la propuesta se tienen en cuenta *Frameworks* como el *NFR* y el *i**, que aportan excelencia técnica;

P7. La simplicidad en este método puede verse afectada, a causa de la documentación que puede llegar a ser excesiva, y la complejidad de los *Framework* que se utilizan dentro del método;

P8. El método CARE cuenta con un repositorio como base de conocimiento, en el cual se almacenan todas las características de los componentes COTS, junto con sus objetivos y especificaciones. Sin embargo al no proponer un equipo de trabajo, llega hacer discutible que se pueda aprender de este repositorio para ajustar la conducta de los roles sugeridos por el método;

	P1	P2	P3	P4	P5	P6	P7	P8
Goal negotiation	+							
Arquitecture exists	–				+			
Iterative process		+ –						
Vendor data Gathered			+					
No interaction with vendors			–		–			
3 technical roles			+ –	+ –		+		
Logical interact-tion sequence guided by roles					+			
NFR, *i**		–				+	–	
Repository					+		–	+
Human factors not considered			–	–				

Tabla. Influencia de los principios ágiles en CARE.

5.2 Integrated Process and Tool Support for selecting Software Components (SCARLET/BANKSEC/PORE)

El método SCARLET (*Selecting Components Against Requirements*) [47], es formalmente es conocido como BANKSEC [37], y es sucesor del método PORE [48]. La propuesta de SCARLET es la adaptación del método PORE al dominio de la banca. SCARLET provee procesos guiados para equipos de procedimiento durante el

desarrollo de procesos en sistemas concurrentes, donde los requisitos de los *Stakeholders*, la arquitectura del sistema y los componentes candidatos son todos determinados al mismo tiempo. Los requisitos de los *Stakeholders* informan de la arquitectura del sistema y aportan a la selección de los componentes candidatos, y el diseño de la arquitectura y la selección del componente COTS, generan restricciones a los requisitos de los *Stakeholders*.

El método depende de la información que se obtenga de los componentes, con el fin de que el equipo de procedimiento determine si el componente cumple con los requisitos de los clientes. El método SCARLET considera que durante la ejecución de estos procesos varias situaciones pueden emerger desde cualquier punto. Es por esto que el método provee un modelo de situación, donde pretende controlar los diferentes estados de los requisitos de los usuarios, con respecto a las características de los componentes. Como parte del producto software desarrollado por el consorcio BANKSEC, SCARLET soporta la ingeniería de software basada en componentes, en la cual integra la administración de los datos, con el soporte de herramientas para la toma de decisiones. SCARLET cuenta con una herramienta para integrar la administración del sistema (*The System Integration Manager SIM*), la cual toma los resultados de los procesos de SCARLET y guía la integración de componentes seleccionados dentro de la aplicación, permitiéndole a los usuarios diseñar modelos de la aplicación comprendidos de los componentes seleccionados, donde se proveen pautas para futuras implementaciones.

5.2.1 Análisis de SCARLET

P1. SCARLET reconoce la naturaleza cambiante de los requisitos, con el fin de definir procesos iterativos de adquisición de requisitos, donde se realiza una evaluación sobre el producto COTS;

P2. En este método se descarta la inclusión gradual de los componentes, además, no se tienen en cuenta la entrega de resultados parciales cuando se tienen que seleccionar múltiples componentes. En [46], se ha propuesto como un trabajo futuro, contar con un prototipo plug&plan del sistema, con el fin de soportar pruebas al integrar los nuevos componentes, pero esta idea no ha sido implementada aún;

P3. El método SCARLETT reconoce dentro de sus procesos la importancia de contar con los proveedores. Los proveedores influyen sobre los procesos de

selección, en la elaboración del contrato al adquirir y asegurar la funcionalidad del componente, en las pruebas de demostración del componente. Sin embargo, en este método el proveedor no hace parte formal, del equipo de procedimiento sugerido por la metodología, por lo cual se deduce que existe una barrera técnica que no permite obtener provecho de trabajar conjuntamente con el proveedor;

P4. Este método no tiene en cuenta el factor humano dentro de sus procesos, a pesar de que recomienda la conformación de un equipo de procedimiento para llevar a cabo los procesos propuestos. Además, el hombre en SCARLET tiene el mismo nivel que un componente software;

P5. Se proveen procesos guiados para los equipos de procedimientos durante el desarrollo de procesos de sistemas concurrentes, donde son determinados a la vez los requisitos de los *Stakeholders*, la arquitectura del sistema y la selección de los componentes candidatos;

P6. Esta propuesta integra métodos, artefactos y técnicas tales como el proceso de análisis jerárquico AHP, Templates de Volere, etc. los cuales proveen un alto grado de excelencia técnica; SCARLET al centrarse en el dominio de la banca, y contar con una adquisición específica de los requisitos, hace que el método cuente con procesos confiables para su entorno;

P7. Aunque en SCARLET se quiere la simplicidad dentro de los procesos, con el fin de tomar la decisión más simple y correcta para guiar su equipo de procedimiento, el método cuenta con 3 tipos de plantillas, que dependen de la cantidad de trabajo que se tiene pensado invertir en el proceso de evaluación, donde la simplicidad puede verse afectada, si el proceso de selección es progresivo, porque necesitará más esfuerzo para completar las plantillas;

P8. En este método no se propone un repositorio real de datos, a excepción de [49], que se propone la elaboración de un repositorio como trabajo futuro, más sin embargo, el método cuenta con una herramienta para soportar y almacenar las historias de evaluación, con el fin de brindar soporte a los equipos de procedimientos y aprender así, de procesos de evaluación pasados;

	P1	P2	P3	P4	P5	P6	P7	P8
Intertwining processes	+	+						
Several types of templates		+					+	
Vendor left outside			–		–			
AHP, Volere and other techniques						+	–	
Process guidance					+			
No repository					–		+	–
Evaluations Files								+
Roles not defined				–		–		
Specialized Process						+		
Contracts and supplier managed			+ –		+			
Architecture Exists	–				+			

Tabla. Influencia de los principios ágiles en SCARLET.

5.3 Off-The-Shelf Option (OTSO)

El método OTSO [38, 51], puede ser considerado como el primer método formal de selección de componentes adquiridos desde el mercado. El método OTSO incluye: procesos documentados y definidos, una descomposición de criterios de evaluación jerárquicos y detallados, un modelo para realizar la comparación de diferentes alternativas comparables en término de costos y valor agregado. Además, cuenta con el uso de técnicas apropiadas para consolidar la evaluación de los datos.

OTSO separa la fase de análisis con el fin de enfatizar en la importancia de interpretar los datos de la evaluación efectuada por cada alternativa. En muchos casos puede ser necesario usar un criterio múltiple para tomar decisiones con el fin de ordenar las diferentes alternativas, por lo cual una de las opciones se considera seleccionar y desarrollar (*deployed*). OTSO sugiere diferentes orígenes o recursos de búsqueda para las alternativas, como son: reutilizar librerías hechas en la organización que se adapten al componente, utilizar los recursos disponibles en la Web, contar con magazines y revistas especializadas, asistir a Workshops y conferencias especializadas, recurrir a vendedores, a colegas expertos y consultantes, y a otras organizaciones. OTSO recomienda realizar pequeñas búsquedas incrementales donde se descubran nuevas alternativas por incremento, esto, con el fin de saber cuando terminar el proceso de búsqueda. El método utiliza el proceso de análisis jerárquico AHP, como técnica de toma de decisión de criterio múltiple. La decisión de utilizar un software COTS está

basada en la estimación de costos de una alternativa y el valor estimado sobre el proyecto. OTSO usa una aproximación que permite balancear la comparación de las alternativas con respecto a los costos y los valores que proveen. La estimación de los costos depende del *baseline* que proviene de la especificación de los requisitos y de su entendimiento.

5.3.1 Análisis de OTSO

P1. El método OTSO tiene en cuenta que al realizar el filtrado de los requisitos frente a las características de los componentes candidatos, estos requisitos pueden estar sujetos a cambios por parte de los *Stakeholders*. Sin embargo, como resultado de estos procesos de filtrado y entendimiento de los requisitos, el método sugiere crear un punto de referencia común para todas las alternativas, con el fin de contar con una base fundamentada para realizar la evaluación de costo y valor. A este punto de referencia se le llama *baseline*, y el cual no se considera que pueda cambiar durante la ejecución del proyecto;

P2. El método sugiere conducir las actividades de búsqueda y descubrimiento en pequeños incrementos, donde se pueda revisar la frecuencia de descubrir las nuevas alternativas por cada incremento;

P3. El método reconoce la importancia de buscar un beneficio al trabajar conjuntamente con el proveedor, para obtener beneficios conjuntos entre las partes. Más sin embargo, el método no sugiere la inclusión del proveedor en un equipo de trabajo;

P4. OTSO no centra sus fundamentos en el factor humano, pero considera que para reutilizar satisfactoriamente el conocimiento y lo aprendido a través del proyecto, no se depende exclusivamente de factores técnicos, también se depende de factores de la organización, de motivación y cuestiones legales;

P5. El método promueve el desarrollo sostenible, con el fin de asegurar una reutilización a través de procesos bien fundamentados apoyados de un repositorio;

P6. OTSO define formalmente un criterio de evaluación, donde las diferentes alternativas de la evaluación pueden ser conducidas con eficiencia y consistencia. Además, el uso de AHP para fundamentar el criterio de evaluación provee un resultado confiable;

P7. En esta propuesta se sugiere concentrar el esfuerzo en generar un ranking sobre cada criterio de evaluación con el fin de no sobrepasar el tiempo estimado para cada evaluación, y además, contar con una documentación sistemática de los resultados;

P8. Este método esta basado en la reutilización del conocimiento adquirido en sus distintas fases. Es por esto que utiliza un repositorio para organizar el conocimiento adquirido al final de cada proceso, con el fin de poder aplicar un feedback futuro;

	P1	P2	P3	P4	P5	P6	P7	P8
Baseline fixed	–				+			
Small increments		+						
Screening phase		+					+	
Marketing and contractual issues outside			–	–				
Concurrent phases	+	+						
Organization maturity considered				+	+			+
Alternatives conducted efficiently				–	+			
Repository					+		–	+
Intensive reuse					+		–	+
AHP & detailed Evaluation						+	–	
Assessment phase								+

Tabla. Influencia de los principios ágiles en OTSO

5.4 PECA

PECA [39] ha sido desarrollada desde el *Software Engineering Institute* (SEI) y por la *National Council Canada (NRC),* y además, hace parte del estándar ISO 14598 [52]. Esta basada en procesos de evaluación de productos software COTS adaptables a la necesidad de diferentes proyectos. La idea principal es comenzar con una planeación para evaluar un producto o productos COTS, terminando con una recomendación sobre la decisión a tomar. El objetivo de los procesos se basa en proveer la información necesaria para la toma de decisiones. El método PECA, tiene en cuenta que en los proyectos COTS, no basta únicamente con los requisitos de los *Stakeholders*, ya que existen otros factores que influencian los proyectos, tales como las aptitudes de los vendedores, su reputación, posicionamiento en el mercado, personal de soporte técnico, entre otras. En el método PECA, se establece un criterio de evaluación que debe ser derivado desde los requisitos, y se considera al criterio de evaluación como el conjunto

de hechos o estándares por el cual los productos son juzgados. En la colección de datos, el método PECA envuelve la ejecución de los planes de evaluación para determinar el funcionamiento de diferentes productos contra los criterios de evaluación que han sido desarrollados. La base en aplicar el conocimiento en PECA, radica en las iteraciones, donde los evaluadores pueden aprender de cada iteración y pueden aplicar el conocimiento en nuevos conceptos sobre los productos COTS, y nuevos criterios para la evaluación. El método PECA sugiere un análisis de los resultados, debido a que en la fase de análisis se produce un número grande de datos, hechos y listas de verificación.

5.4.1 Análisis de PECA

P1. El método considera que los requisitos pueden sufrir modificaciones durante el desarrollo del proyecto. También considera un análisis de sensibilidad y de costo para controlar y modificar los requisitos y los procesos obtenidos de diferentes procesos de evaluación;

P2. Esta propuesta, sugiere que se debe poder escoger el primer producto que se adapte a los requisitos de los usuarios, o considerar incluir múltiples filtros en una iteración, con el fin de seleccionar más de un componente por iteración;

P3. PECA no sugiere la interacción directa del vendedor dentro del equipo de trabajo sugerido en el método. Aunque propone entrevistas y obtención de información del proveedor para completar el criterio de evaluación;

P4. El factor humano en PECA es relevante. Considera como poderoso contar con un balance de trabajo dentro del equipo de trabajo. Sugiere un *"charter"* para definir los nombres de los roles de los miembros del equipo de trabajo, además, el equipo sugerido debe de contar con un tamaño pequeño similar a los equipos ágiles. Sin embargo, no se sugieren roles básicos ni responsabilidades a asignar, para conformar un equipo COTS;

P5. Aunque, el método cuenta con procesos consistentes que no son complejos de implementar y que interactúan entre si dentro de iteraciones, no cuenta con procesos guiados para el equipo COTS sugerido, por lo cual no se puede deducir si los procesos son agotadores para el equipo COTS;

P6. La propuesta considera dentro del equipo de trabajo la calidad técnica. Se sugiere que el equipo debe estar conformado por personal técnico que aporte experiencia y calidad al desarrollo del proyecto. Además, la propuesta dice que

al contar con procesos de alta calidad, obtendremos resultados óptimos en la evaluación;

P7. La simplicidad esta envuelta dentro de los procesos de desarrollo de esta propuesta. El método es conciente que si contamos con una excesiva documentación, podemos obstruir el desarrollo del proyecto. Considera dentro de sus procesos, soluciones simples para obtener resultados según el grado de rigurosidad. La forma como la simplicidad puede verse afectada en este método, depende de la forma de obtener el dossier del producto, ya que puede llegar a ser compleja y puede suponer documentación que no se utilicé por nadie;

P8. Las diferentes iteraciones sugeridas en PECA, sugieren ganar en entendimiento y comprensión, sobre la evaluación de los requisitos y la del producto. En este método se trata de recuperar los datos con el propósito de contribuir al aprendizaje del proyecto. El dossier del producto, se propone con el fin de descubrir hechos, evaluar resultados, realizar clasificaciones, entre otros. Pero el método no propone un repositorio formal de datos, el cual deba ser mantenido y actualizado.

	P1	P2	P3	P4	P5	P6	P7	P8
Flexibility to the realities	+							
Multiple filters		+			+			
Charter				+		+		
Information Vendor			+					
Small size team.				+				
Not exquisitely documented process							+	
Product dossier							–	+
Consistent evaluation process				+			–	
Rigorous evaluation							–	

Tabla. Influencia de los principios ágiles en PECA.

5.5 Applying Social-Technical Approach for COTS Selection (STACE)

El método STACE [40, 53], adopta la definición de un sistema de información dada por UKAIS, la cual adopta tres perspectivas importantes como la dimensión humana, la

organización y tecnología. Las perspectivas humanas resaltan necesidades varias, donde usan la información de la tecnología para ejecutar su trabajo. La organización es la que utiliza los sistemas de información, la cual esta compuesta de diferentes estructuras, objetivos, políticas, de una cultura dentro de la organización, la cual puede afectar la perspectiva de un sistema de información. Y la tecnología en los sistemas de información tales como hardware y software, proveen una manera efectiva y eficiente de procesar los datos, además, de transformarlos en distintos productos de información.
Dentro de las cuatro fases definidas por el Framework STACE, encontramos una fase de definición de requisitos, la cual comprende las siguientes actividades: estudio del mercado, recopilación de los documentos del sistema, conocimientos del dominio y trabajo con los *Stakeholders*. La fase de definición de requisitos sirve de base para la definición del criterio *socio-técnico*, con el cual se tienen en cuenta las especificaciones tecnológicas, los factores de calidad del producto y las características de la funcionalidad. El método STACE considera importante la participación del cliente durante el desarrollo del sistema de información, ya que puede incrementar la aceptación del usuario del componente, y además, se podrán desarrollar las capacidades reales solicitadas por el usuario.

5.5.1 Análisis de STACE

P1. El método STACE sugiere que se clasifiquen los requisitos de los usuarios, priorizando el grado de importancia. Además, dentro de su flujo de procesos, STACE contempla la verificación de los requisitos;

P2. Las iteraciones que están contempladas dentro del flujo de procesos de este método, no tienen en cuenta realizar entregas frecuentes dentro de sus procesos con el fin de observar resultados tempranos;

P3. El método contempla el almacenamiento de los datos del vendedor dentro de una base de datos de resultados. Además, sugiere establecer contacto con el vendedor al momento de realizar la evaluación del componente, pero no se tiene en cuenta la inclusión del vendedor dentro de un equipo COTS;

P4. STACE le da importancia al factor social dentro del desarrollo del proyecto COTS, ya que el método es consiente de que el mayor causa en las fallas del software son provocadas por la gente. STACE tiene en cuenta al usuario final

dentro de sus procesos. Pero el método no sugiere la conformación de un equipo de trabajo, ni la asignación de roles ni responsabilidades;

P5. Los procesos dentro de STACE sugieren procesos interrelacionados entre si, donde cada fase cuenta con sus propios procesos internos que interactúan entre si. Desafortunadamente el método no sugiere un equipo COTS, con el fin de comprobar si estos procesos son sostenibles para el equipo;

P6. STACE es un método que esta influenciado por los métodos OTSO y PORE, por lo cual adopta prácticas sugeridas en estos métodos, como AHP y técnicas de análisis de características, además, basa algunos de sus principios en el estándar ISO/IEC 9126:1991 (E);

P7. El método no sugiere documentar los procesos, y propone en aras de la experiencia, que cada organización adopte el método de recolección de datos que más se adapte a su estructura organizativa;

P8. Esta propuesta sugiere aprovechar el conocimiento generado desde pasadas experiencias, acumulándolo en una base de datos;

	P1	P2	P3	P4	P5	P6	P7	P8
Prioritization requirement	+							
Interrelated process		+			+			
Customer in evaluation team				+				
Vendor data gathered			+					
Define standard and qualify process						+	–	
AHP, outranking methods, analysis technique						+	–	
Database							–	+
Social factors considered				+				

Tabla. Influencia de los principios ágiles en STACE

5.6 Evolutionary Process for Integrating COTS-Based System (EPIC)

A diferencia de las anteriores propuestas, EPIC [41] contempla un ciclo de desarrollo completo que se basa en las fases de *Rational Unifed Process* (*RUP*) [4, 5]: "*Inception, Elaboration, Construction, Transition,*" donde se tiene en cuenta la integración y el

mantenimiento del sistema de información compuesto por COTS. El método EPIC fue desarrollado por *U.S. Air Force*, con el fin de soportar soluciones de sistemas basados en componentes COTS. EPIC esta cimentado en trabajos previos del *Software Engineering Institue* (SEI): en la línea de procesos necesarios para la administración de sistemas basados en COTS p.e., [1, 2], en técnicas de ingeniería para el diseño y el desarrollo de de sistemas basados en COTS [3], y en líneas de trabajo propias del SEI. A su vez EPIC también esta basado en el RUP y en el desarrollo del modelo en espiral propuesto por el DR. Barry Boehm [6]. EPIC basa la clave de su solución en la necesidad de definir y realizar simultáneamente la compensación entre las necesidades de los *Stakeholders*, los procesos del negocio, los segmentos del mercado donde se desarrollan los COTS, el diseño de la arquitectura y los riesgos de programación. El método tiene presente características propias de los COTS que resultan como consecuencia de las restricciones del mercado. Por esta razón propone realizar la negociación constante con los *Stakeholders* según las condiciones del mercado. Dentro de sus diversas actividades, también propone la documentación de la funcionalidad extra que provee cada componente.

5.6.1 Análisis de EPIC

P1. Se tiene en cuenta que los requisitos de los usuarios deben ser negociados de acuerdo a las restricciones del Mercado. Así mismo, EPIC propone procedimientos para manejar una negociación flexible de los requisitos dentro de iteraciones.

P2. Aunque se proponen iteraciones continuas, en las cuales los clientes van ganando en comprensión y conocimiento acerca del componente y sus necesidades, el método propone realizar el refinamiento de estas necesidades en cada iteración.

P3. EPIC tiene presente que la funcionalidad del producto depende del proveedor del componente. Se tiene en cuenta además, los aspectos legales dentro de su fase de Elaboración. No es claro si el rol de *Acquirer* propuesto en [54], aparte de realizar el dossier respectivo, trabaja conjuntamente con el vendedor o solo lo identifica y describe.

P4. Los usuarios son motivados a participar en las diferentes fases del método. Se sugiere la conformación de un equipo de trabajo, y en [54] se proponen roles

y responsabilidades. Pero el factor humano y de motivación no es tenido en cuenta

P5. La interacción de sus cuatro esferas de procesos, permite contener el desarrollo sostenible de la selección de un componente, equilibrando las necesidades del usuario con el estado actual del mercado

P6. Se busca en cada iteración ganar en excelencia y comprensión sobre el componente y las necesidades del usuario. Su conjunto de prácticas y el modelo del negocio, casi siempre basado en *UML*, enriquecen al método. En [54] se hace hincapié en resaltar roles y responsabilidades, pero no se hace exigencia alguna en el nivel técnico de estos roles.

P7. EPIC puede ver afectada su simplicidad, por medio del conjunto de actividades que se deben realizar por cada iteración, y los resultados que se deben obtener de cada fase. Además, por cada fase se propone la revisión en profundidad de actividades que han sido previamente revisadas. Por otra parte EPIC propone la construcción de un prototipo del sistema, lo cual puede llega a ser complejo para grandes sistemas de información

P8. Por medio de cada iteración se busca refinar y ajustar las necesidades y las características de los componentes. Al igual EPIC, sugiere almacenar la información de la funcionalidad extra que proveen los componentes seleccionados. No es claro si se propone la utilización de un repositorio de información, o si se busca ajustar el conocimiento adquirido al aprendizaje de los roles propuestos.

	P1	P2	P3	P4	P5	P6	P7	P8
negotiation With stakeholders	+							
Interrelated of the tour spheres		+			+			
Essential roles				+				
Vendor Dossiers			+				–	
knowledge is captured across iterations						+		
RUP and UML						+	–	
Spiral development		+						
Phase Task Over							–	+
Social factors not considered				–				

Table. Influencia de los principios ágiles en EPIC

5.7 Observaciones generales

La siguiente tabla resume el resultado del análisis propuesto. Se ha identificado los 8 principios utilizando el identificador *Pi* introducido en las secciones anteriores. Por cada método *M* y principio *Pi*, clasificamos el grado de cobertura de *M* para *Pi* usando la siguiente lógica:

- 'a' significa que *M* cubre total mente a *Pi.*
- 'b' significa que aunque *M* no tenga intención explicita sobre *Pi* maneja bien o no tiene un manejo claro o apropiado de él.
- 'c' significa que *M* no trabaja bien con *Pi*, aunque exista una presencia parcial de *Pi* en *M*.
- 'd' significa que *Pi* no tiene presencia en *M*.

	P1	P2	P3	P4	P5	P6	P7	P8
CARE	a	c	c	b	a	a	d	a
BANCKSEC	a	c	c	c	b	a	c	b
OTSO	b	a	c	d	b	b	d	b
PECA	a	a	c	b	b	a	b	a
STACE	a	c	c	b	c	a	c	b
EPIC	a	a	b	b	b	a	b	b

Table. Análisis Comparativo

De la tabla anterior, se deducen varios aspectos con respecto a los principios ágiles y su forma de influencia sobre los métodos de selección de componentes COTS analizados:

- Observamos que el principio P1, está bien cubierto por todos los métodos, probablemente porque los métodos seminales OTSO y PORE, fueron explícitos en recalcar la importancia de realizar concurrentemente la adquisición de requisitos y la evaluación de COTS, además, consideraban el cambio que pueden sufrir los requisitos durante el desarrollo del proyecto, a su vez, de la manera como se debía entrar a negociar estos cambios con los *Stakeholders*.
- Este mismo hecho es percibido en el principio P6, lo cual puede ser a causa de que estos métodos se apoyan en modelos y frameworks que provienen del mundo académico. Aunque se debe tener presente, que como hecho opuesto a estos aportes académicos, en [55] se ha reportado que las técnicas empleadas en estos métodos son rechazadas en el ámbito industrial, debido a

la presiones del tiempo, o por que los modelos no son viables y son inadecuados para ser tenidos en cuenta [23].

- El principio **P5** y **P8** están razonablemente bien cubiertos por los métodos. El desarrollo sostenible proviene de procesos bien definidos, que cuenten con una secuencia lógica de desarrollo, manteniendo una coherencia en su ejecución. Así mismo, la reflexión está soportada por los repositorios de información, los cuales pretenden almacenar conocimiento adquirido a través del proyecto, o almacenar características de los eventos y actores que pueden ser reutilizados para ajustar el comportamiento del proyecto y del equipo de trabajo.
- El resto de principios no están bien cubiertos por todos los métodos. Es difícil reconciliar la simplicidad con otros principios. Si realizamos la siguiente consideración: ¿qué tan simple puede llegar a ser que un método que cuente con herramientas técnicas, pueda llegar a necesitar un adiestramiento especial para obtener un mayor beneficio?, podremos deducir que debemos contar con gente capacitada o que conozca dichas herramientas técnicas, o tendremos que invertir en capacitación adicional. Además, tampoco es obvio que la selección de componentes COTS pueda conducir entregas frecuentes, por que la dependencia que pueden generar los diferentes componentes y la arquitectura actual del sistema de información pueden llegar a convertirse en restricciones importantes.
- Un punto clave que puede ser observado, es que los métodos actuales no consideran las motivaciones individuales (**P4)** como factor fundamental o clave de sus procesos, enfocándose más en aquellos aspectos técnicos, que en los aspectos sociales que impactan directamente sobre el individuo, considerando en muchos casos al humano como un mero *agente software*.
- Apoyados en la observación anterior, podemos deducir que la causa de que el principio P3 no esté bien cubierto por los métodos actuales, pueda ser la falta de tener en cuenta las interacciones sociales de los diferentes actores involucrados en el proyecto. Por lo cual no se considera importante el aspecto legal del proyecto, donde a menudo el resultado de la selección del componente COTS se define como un conjunto de productos, en vez de un pliego de contratos como en realidad sucede. Además, se desconoce el potencial que se puede obtener al trabajar directamente con los proveedores

conjuntamente, ya que por lo general el proveedor es visto como un adversario o como una mera fuente de información, en vez de ser un colaborador activo que puede llegar a brindar beneficios para la organización.

6 Conclusiones

Del análisis realizado en este estudio, son varias las conclusiones que se pueden extraer en aras de proveer y realizar mejoras a los procesos de selección de componentes COTS. Un aporte interesante, es el que se puede obtener del conjunto de valores y principios ágiles extraídos desde el mundo ágil, los cuales no distan demasiado de la realidad del contexto de los componentes COTS. Esto se debe, a que es posible extrapolar su significado y aprovechar sus fundamentos, para tratar de analizar la agilidad de los procesos COTS, y el ambiente en el que se desenvuelve y desarrolla la selección de los componentes candidatos. Desde este punto de vista, no es totalmente claro, que la ingeniería de requisitos utilizada dentro de los métodos de selección de componentes COTS, pueda adaptarse a la evolución constante y dinámica, de los diferentes segmentos de mercado donde se desarrollan los componentes COTS. Este conjunto de procesos de toma de requisitos, debe ser adaptable a las exigencias de los usuarios finales, así como al estado actual del mercado. Por este motivo, se debe evaluar que tan ágil resulta contar con procesos robustos y frameworks que necesiten adiestramiento especial, para la toma de requisitos, ya que estos procesos, pueden afectar al rendimiento del proyecto, como al buen ambiente de trabajo. Así mismo, se deben buscar posibles prácticas que ayuden agilizar y mejorar el entendimiento, la comprensión y la comunicación entre los actores que participan de la selección de los COTS. Por otra parte, se hace patente la necesidad de establecer un conjunto de roles y responsabilidades, que puedan ser verificables y que sean asignadas a las personas que opten por integrar el equipo de selección de los COTS, con el fin de cubrir el conjunto de nuevos procesos y responsabilidades que han emergido a raíz de los proyectos COTS. Entre este grupo de procesos que aún están por madurar, se encuentra el manejo de los proveedores, el establecimiento de contratos, la exploración del sistema, la evaluación y selección de los componentes candidatos, la integración del componente al sistema de información, entre otros. Así mismo, la búsqueda de la simplicidad dentro los procesos COTS, como el evitar documentar información que no es utilizada por

nadie, ha de ser una premisa al momento de seleccionar un componente candidato. Finalmente, los métodos y procesos de selección de componentes COTS deben prestar más atención al factor humano dentro de los proyectos, siempre buscando mejorar el ambiente de trabajo, así como al entorno donde se desarrolla el proyecto, y buscando aprovechar el conocimiento generado por cada proceso.

Referencias

[1] Meyers, B., Oberndorf, P.: "*Managing Software Acquisition: Open Systems and COTS Products*". New York, NY: Addison-Wesley, SEI Series in Software Engineering, 2001.

[2] Carney, D., Hissam, S., Plakosh, D.: "*Complex COTS-based Software Systems: Practical Steps for their Maintenance.*" Journal of Software Maintenance: Research and Practice, 12, 6 (2000): 357-376.

[3] Wallnau, K., Hissam, S., Seacord, R.: "*Building Systems from Commercial Components.*" New York, NY: Addison-Wesley, SEI Series in Software Engineering, 2002.

[4] Kruchten, P.: "*The Rational Unified Process: An Introduction*" 2nd ed. New York, NY: Addison-Wesley Object Technology Series, March 2000.

[5] Rational Unified Process, (software product) version 2000.02.10. Rational Software Corporation. http://www-306.ibm.com/software/rational/ ,last accessed in January 2006.

[6] Boehm, B.: *"A Spiral Model of Software Development and Enhancement."* IEEE Computer, 21, 2 (February 1998): 61-72.

[7] Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Kern, R., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: "*Manifesto for Agile Software Development*" http://www.agilemanifesto.org, 2001, last accessed in January 2006.

[8] Abts, C.: "*COTS Software Integration Cost Modeling Study*". University of southern California, contract F30602-94-C-1095. 29 June 1997.

[9] Basili, V., Bohem B.: "*COTS-based systems top 10 list*". Computer Volume 34, Issue 5, May 2001 Page(s): 91 – 95

[10] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: "*Agile Software Development Methods: Review and Analysis*" January 01, 2002.

[11] Alleman, G.: "*Agile Project Management Methods for ERP: How to Apply Agile Processes to Complex COTS Projects and Live to Tell About It*" In *Extreme Programming and Agile Methods: XP/Agile Universe* LNCS 2418, 2002.

[12] Chung, L., Cooper, K., Courtney, S.: "*COTS-Aware Requirements Engineering and Software Architecting*" Proceedings of the Intl' Conference on Software Engineering Research and Practice (SERP), 2004.

[13] FAA SERC. "*Lessons Learned in Developing Commercial Off-The-Shelf (COTS) Intensive Software System*" October 2, 2000.

[14] Lamsweerde, A.: "*Goal-Oriented Requirements Engineering: A Guided Tour*" Proceedings of the 5th IEEE ISRE, 2001

[15] Yu, E.: "*Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering*" Proceedings of the 3rd IEEE ISRE, 1997.

[16] Yu, E., Mylopoulos, J., Lesperance, Y.: "*AI Models for Business Process Reengineering*" IEEE Expert, August 1996, pp. 16-23.

[17] Larman, C., Basili, V.: "*Iterative and Incremental Development: A Brief History*" IEEE Computer, November 2004.

[18] Mcilory, M.: "*Mass Produced Software Components*" Proc. of NATO Conference on Software Engineering, Garmisch-Partenkirchen, NATO Science Committee, NATO, Brussels, 1969.

[19] Zeyu, J., Jacob, H., We, Y.: "*Testing and Quality Assurance for Components-Based Software*" ISBN 1-58053-480-5, 2003 ARTECH HOUSE, INC.

[20] Heineman, T., William T., Councill, S., Flynt, M., John, R., Mary, S.: "*Component-based software engineering and the issue of trust.*" 661-664

[21] Martin, C.: "*Agile Development: Principles, Patterns and Process*" Prentice Hall, 2002.

[22] IEEE at http://ieeexplore.ieee.org/ , last accessed in January 2006

[23] Wiegers, K.: "*Read my lips: no new models!*" Software, IEEE Volume 15, Issue 5, Sept.- Oct. 1998 Page(s): 10 – 13.

[24] Cockburn, A., Highsmith, J.: "*Agile Software Development: The People Factor*" IEEE Computer, December 2002.

[25] Beck, K.: "*Extreme Programming Explained: Embrace Change*" Addison Wesley, 1999.

[26] Beedle, M., Schwaber. K.: "*Agile Software Development with SCRUM*" Prentice Hall, 2001

[27] Larman, C.: "*Applying UML and Patterns* (3rd edition)" Prentice Hall, 2005

[28] Lauesen, S.: "*COTS Tenders and Integration Requirements*" Proceedings of the 12th IEEE RE, 2004.

[29] Brownsword, L., Place, P.: "*Lessons Learned Applying Commercial Off-the-Shelf Products*" Report CMU/SEI-99-TN-015, June 2000

[30] Kunda, D., Brooks, L.: "*Applying Social-Technical Approach for COTS Selection*" Proceedings of the fourth UKAIS Conference, University of York, April 1999

[31] Curtis, B., Krasner, H., Iscole, N.: "*A field study of the software design process for large system*" Communication of the ACM, 31(11):1268-1286, November 1988.

[32] Morisio, M., Reaman, C., Basili, V., Parra, A., Kraft, S., Condon, S.: "*COTS-based software development: processes and open issues*" Journal of Systems and Software 61 (2002): 189-199.

[33] Boehm, B., Abts, C.: "*COTS integration: plug and pray*?" Computer Volume 32, Issue 1, Jan. 1999 Page(s):135-138.

[34] Egyed, A., Kwan, J., Madachy, R.: "*Developing Multimedia Applications with the WinWin Spiral Model"* In Proceedings ESEC/FSE 97, November 1997

[35] Saaty, T.: "*How to make a decision: The analytic hierarchy process*" European Journal of Operations Research, no. 48, pp. 9 - 26, 1990.

[36] COCOTS at http://sunset.usc.edu/research/COCOTS/, last accessed in January 2006

[37] Maiden, N., Kim, H., Ncube, C.: "*Rethinking Process Guidance for Selecting Software Components*" Proceedings of 1st ICCBSS, LNCS 2255, 2002

[38] Kontio, J.: "*A Case Study in Applying a Systematic Method for COTS Selection*" In Proceedings 18th Intl' Conference on Software Engineering (ICSE), 1996.

[39] Dorda, C., Dean, C., Morris, E., Oberndorf, P.: "*A Process for COTS Software Product Evaluation.*" Proceedings of 1st ICCBSS, LNCS 2255, 2002

[40] Kunda, D.: "*STACE: Social Technical Approach to COTS Software Evaluation*" In Component-Based Softwa-re Quality - Methods and Techniques, LNCS 2693, 2003.

[41] Alberts, C., Brownsword, L.: "*Evolutionary Process for Integrating COTS-Based System (EPIC): An Overview*" Technical Report Cmu/SEI-2002-TR-099 ESC-TR-2002-009 July 2002

[42] Kontio, J., Chen, S., Limperos, K., Tesoriero, R., Caldiera, G., Deutsch, M.: "*A COTS Selection Method and Experiences of Its Use*" Twentieth Annual Software Engineering Workshop, November 29-30, 1995.

[43] Chung, L., Cooper, K.: "*Defining Goals in a COTS-Aware Requirements Engineering Approach"* Systems Engineering journal, Volume: 7 Number 1, 2004, pp. 61-83.

[44] Jacobson, I., Booch, G., Rumbaugh, J.: "*The Unified Software Development Process*" Addison Wesley Longman, Inc., USA, 1999.

[45] Boehm, B., Port, D., Abi-Antoun, M., Egyed, A.: "*Guidelines for the Life Cycle Objectives (LCO) and the Life Cycle Architecture (LCA) deliverables for Model-Based Architecting and Software Engineering (MBASE)*" TR USC-CSE-98-519, USC-Center for Software Engineering

[46] Ncube, C., Maiden, N.: "*Guidance for Parallel Requirements Acquisition and COTS Software Selection.*" In Proceedings of 4th ISRE 1999: 133-141

[47] Maiden, N., Croce, V., Kim, H., Sajeva, G., Topuzidou, S.: "*SCARLET: Integrated Process and Tool Support for Selecting Software Components*" Component-Based Software Quality 2003: 85-98.

[48] Maiden, N., Ncube, C.: "*Acquiring COTS Software Selection Requirements.*" IEEE Software 15(2), 1998.

[49] Ncube, C., Maiden, N.: "*PORE: Procurement Oriented Requirements Engineering Method for a Component-Based System Engineering Development Paradigm.*" In Proceedings of the 2nd International Workshop on Component-Based Software Engineering (CBSE), 1999.

[50] Fowler, M.: "*Is design dead?*" XP2000 Proceedings. http://www.martinfowler.com/articles/designDead.html, Last Significant Update: May 2004. Last accessed in January 2006

[51] Kontio, J.: "*OTSO: A Systematic Process for Reusable Software Component Selection*" University of Maryland Technical Report CS-TR-3478, College Park, MD, 1995.

[52] International Organization for Standardization (ISO): ISO/IEC 14598-1:1999 – Information Technology – Software Product Evaluation. Geneva, Switzerland, (1999).

[53] Kunda, D.: "*A Social-Technical Approach to Selecting Software Supporting COTS-Based System*" University of York Department of Computer Science Technical Report YCST-2002-01. October 2001.

[54] Peraire, C., Pannone R. "*The IBM Racional Unified Process for COTS-based projects: An introduction*" http://www-128.ibm.com/developerworks/rational/library/aug05/peraire-pannone/ . 15 Aug 2005. last accessed in January 2006

[55] Torchiano, M., Morisio, M.: ”*Overlooked Aspects of COTS-Based Development*” IEEE Software 21(2), 2004.