# SELECTION OF AGILE MODELS AND METHODOLOGIES FOR SOFTWARE PROJECTS

Sáez Martínez, Pedro Javier; Rodríguez Montequín, Vicente; Villanueva Balsera, Joaquin; Cueto Cuiñas, Marcos

Universidad de Oviedo

In IT environments, project management is increasingly important, more jobs are offered for this profile, seeking people who are capable of complying with procedures that increase the chances of success of a project.

The use of a model of development or management is based on the type of project. It is wrong to think that software project management is equal to agile model and this is equal to Scrum. The purpose of this study is to provide information on this issue to assist those who are beginning in the world of project management and who have shown interest in agile methodologies.

To this purpose, when starting a software project, the existing models are analyzed in terms of management and using a series of selection criteria, it allows us to assess which alternative would be the best. It could also be compared to each other the existing agile methodologies and techniques that can be used.

***Keywords:*** *Software; Development model; Agile methodology*

# SELECCIÓN DE MODELOS Y METODOLOGÍAS AGILES EN PROYECTOS SOFTWARE

En el entorno de la tecnología de la información la gestión de proyectos cada vez tiene más importancia, se ofertan más puestos para este perfil, buscando gente capacitada para cumplir unos procedimientos que aumenten las posibilidades de éxito de un proyecto.

La utilización de un modelo de desarrollo o gestión va en función del tipo de proyecto. Es un error pensar que la gestión del proyecto software es igual a modelo ágil y que este es igual a Scrum. El objeto de este trabajo es aportar información sobre este tema para ayudar a aquellos que se están iniciando en el mundo de la gestión de proyectos y que han mostrado interés por las metodologías ágiles.

Para ello, al comenzar un proyecto software se analizan los modelos existentes desde el punto de vista de gestión y utilizando unos criterios de selección nos permite evaluar cuál sería la mejor alternativa. También se podría comparar las metodologías y técnicas ágiles existentes que se pueden usar con cada una de ellas.

***Palabras clave:*** *Software; Modelos de desarrollo; Metodologías agiles*

Correspondencia: Vicente Rodríguez Montequín. Area de Proyectos de Ingeniería. Escuela de Minas. C/Independencia 13. C.P. 33004. Oviedo

## 1. Introducción

En la Ingeniería Informática, no es raro encontrarse proyectos que tras invertir una cantidad de dinero y tiempo considerables, dejan el producto a medias, con grandes deficiencias o simplemente que no llegan a nada.

Esto no quiere decir que los ingenieros informáticos se hayan olvidado de la importancia de una gestión adecuada de un proyecto, pero muchas veces no se le da el valor que merece, y simplemente se limitan a aprender a hacer una buena toma de requisitos y un presupuesto, centrándose más en tareas de diseño y programación.

Al no existir un único modelo exitoso a seguir, muchos equipos de desarrollo se limitan a abandonarlos, pero sin utilizar otras metodologías. Esto lleva de vuelta a un escenario donde la ausencia de una metodología dispara las probabilidades de fracaso de un proyecto.

En el mundo de la tecnología de la información (IT), la gestión de proyectos cada vez tiene más importancia, cada vez se ofertan más puestos para este perfil, buscando gente capacitada a seguir unos procedimientos que aumenten las posibilidades de éxito de un proyecto.

Este hecho está siendo escuchado por el mundo laboral, y los informáticos se han lanzado a formarse sobre el modelo de gestión del ciclo de vida de un proyecto de repunta para el mundo software, el modelo ágil. Lo cual ha llevado a muchos a pensar que gestión de proyecto software es igual a modelo ágil y que este es igual a Scrum. Esto es un error, y está basado en el desconocimiento existente en la materia.

El objetivo de este trabajo es aportar información sobre este tema para ayudar a aquellos que se están adentrando en el mundo de la gestión de proyectos y que han mostrado interés por las metodologías ágiles. Para lo cual se estudiarán los distintos modelos de ciclo de vida y en caso de encajar el modelo ágil valorar que metodología ágil es óptima para el tipo de proyecto.

Las metodologías ágiles surgieron como una posible solución a este problema ya que están orientadas a proyectos software donde la incertidumbre suele ser muy alta tanto en tiempo como en costes, los requisitos cambian constantemente y las tecnologías evolucionan a una velocidad de vértigo que hace que la mayoría de las veces no puedas usar unos proyectos como referencia de otros.

Y ante todo este panorama, cuando un director de proyectos software, o una empresa de IT tiene que elegir qué modelo de ciclo de vida quiere para sus proyectos y que metodología va a seguir, se encuentra ante una ingente cantidad de información, con pros y contras para cada modelo y cada metodología.

En este trabajo se explica que modelo de ciclo de vida es más adecuados en cada caso, y en el supuesto de que para tu proyecto u organización el ágil sea el más correcto, se da una introducción de las distintas metodologías que existen para este modelo (y no sólo Scrum), haciendo una comparativa de ellas en base a varios factores, para finalmente explicar distintas técnicas que se puedan usar con cada una de ellas. Con el fin de facilitar la selección mediante unas tablas, datos, diagramas e información centralizada.

## 2. Escenario de la situación actual

Para conocer el escenario, se estudiaran las modelos de ciclo de vida existentes y las metodologías que se pueden aplicar en un proyecto que encaje en un modelo ágil.

## 2.1. Carencias en los Proyectos Software

El estudio "The CHAOS Report" (Standish Group, 2010), cuyo objetivo es encontrar y combatir las causas de los fracasos, se ha convertido en un referente mundial sobre los factores que inciden en el éxito o fracaso de los proyectos de IT. De este estudio se extrae como resultado, que poco a poco se ha ido mejorando, viendo una evolución lenta pero constante hacía mayores ratios de éxitos, pero aun así el número de proyectos no exitosos sigue siendo muy alto.

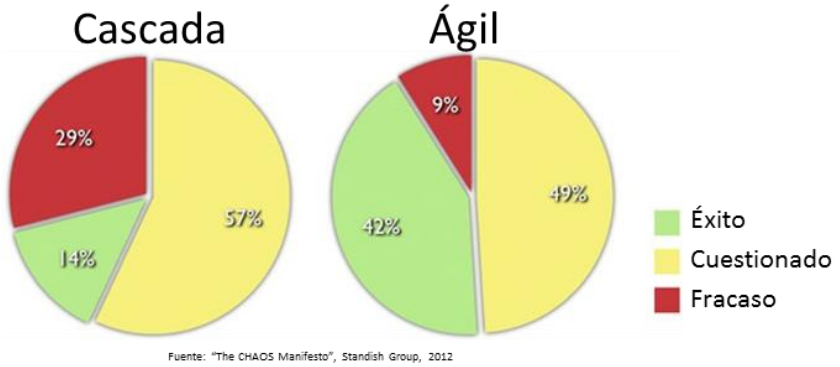
Para buscar las razones de este fracaso, Daniel Piorun (2001) realizó un estudio con aproximadamente 50 responsables de proyectos, con el objeto de analizar las causas que alimentan los fracasos. En dicho estudio se encontraron tres principales causas que afectan los proyectos de forma negativa, estas son:

- Cambios en los objetivos definidos a nivel estratégico (21 %).
- No utilización, o mala utilización de metodologías de trabajo (31 %).
- Problemas humanos, de dirección, comunicación y conflictos entre las personas (48 %).

En el informe de Standish se observa, que de las diez principales causas de los fracasos, siete están referidos a factores humanos (escasa participación de los usuarios, requerimientos y especificaciones incompletas, cambios frecuentes en los requerimientos, falta de soporte ejecutivo). Las causas que apunta el Standish Group en su gran mayoría podrían ser clasificadas en uno de los tres grupos presentados por Piorun, y podemos ver claramente que los porcentajes más altos se encuentran enmarcados en los temas relacionados con el recurso humano.

El reto de las nuevas metodologías es cubrir todos estas carencias, y quizás en ese apartado es donde los modelos ágiles estén sacando mayor ventaja a las tradiciones, y por eso están obteniendo mejores porcentajes de éxito y están ganando fama y repercusión dentro del mundo del software.

**Figure 1: Éxito de Modelo en Cascada y Ágil según Standish Group**




Fuente: "The CHAOS Manifesto", Standish Group, 2012

## 2.2. Modelos de ciclo de vida existentes

Como primer paso, se procede a estudiar las características de los modelos de ciclo de vida más relevantes y con más impacto en el desarrollo del software.

El modelo en cascada, fue introducido por Winston W. Royce en 1970 (Royce, 1970), algunas veces llamado el ciclo de vida clásico, sugiere un enfoque sistemático y secuencial hacia el desarrollo del software, que se inicia con la especificación de requisitos del cliente y que continúa con el análisis, diseño, implementación y pruebas. Sus principales etapas de

este modelo según Sommerville (2005) son: análisis y definición de requisitos, diseño del sistema y del software, implementación y prueba de unidades, integración y prueba del sistema, funcionamiento y mantenimiento.

El modelo en V,(Davis, 1993) fue diseñado por Alan Davis, y contiene las mismas etapas que el ciclo de vida en cascada puro pero a diferencia de aquel, busca hacer la actividad de pruebas más efectiva y productiva, mediante la elaboración de planes y casos de prueba a medida que se avanza en el desarrollo del proyecto. Se desarrolló para terminar con algunos de los problemas del enfoque de cascada tradicional. Los defectos estaban siendo encontrados demasiado tarde en el ciclo de vida, ya que las pruebas no se introducían hasta el final del proyecto. Las pruebas necesitan empezarse lo más pronto posible en el ciclo de vida y estas actividades deberían ser llevadas a cabo en paralelo con las actividades de desarrollo. Los probadores necesitan trabajar con los desarrolladores y analistas de tal forma que puedan realizar estas actividades y tareas y producir una serie de entregables de pruebas.

El modelo iterativo, derivado del ciclo en cascada, busca reducir el riesgo que surge entre las necesidades del usuario y el producto final por malos entendidos durante la etapa de recogida de requisitos. Consiste en la iteración de varios ciclos de vida en cascada, al final de cada iteración se le entrega al cliente una versión mejorada o con mayores funcionalidades del producto. El cliente es quien después de cada iteración evalúa el producto y lo corrige o propone mejoras. Estas iteraciones se repetirán hasta obtener un producto que satisfaga las necesidades del cliente. Este modelo se suele utilizar en proyectos en los que los requisitos no están claros por parte del usuario.

El modelo incremental,(Mills, 1999) su filosofía es construir incrementando las funcionalidades del programa. El primer incremento es a menudo un producto esencial, sólo con los requisitos básicos, pero muchas características suplementarias (algunas conocidas, otras no) no se incorporan. Este modelo se centra en la entrega de un producto operativo con cada incremento. Los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación.

El modelo en espiral,(Boehm, 1986) se basa en una serie de ciclos repetitivos para ir ganando madurez en el producto final, hasta lograr el objetivo deseado. Cada ciclo en la espiral representa una fase del proceso del software. Así el ciclo más interno puede aludir a la viabilidad del sistema, el siguiente ciclo a la definición de requerimientos, el siguiente ciclo al diseño del sistema, y así sucesivamente.

Tiene en cuenta el riesgo que aparece a la hora de desarrollar software, para ello, se comienza mirando las posibles alternativas de desarrollo, se opta por la de riesgos más asumibles y se hace un ciclo de la espiral. Si el cliente quiere seguir haciendo mejoras en el software, se vuelven a evaluar las nuevas alternativas y riesgos y se realiza otra vuelta de la espiral, así hasta que llegue un momento en el que el producto software desarrollado sea aceptado y no necesite seguir mejorándose con otro nuevo ciclo.

Toma los beneficios de los modelos incremental y por prototipos (derivado del iterativo), pero se tiene más en cuenta el concepto de riesgo que aparece debido a las incertidumbres e ignorancias de los requerimiento proporcionados al principio del proyecto o que surgirán durante el desarrollo. Este sistema es muy utilizado en proyectos grandes y complejos.

El modelo ágil, se basa en el desarrollo iterativo e incremental, teniendo presente los cambios y respondiendo a estos mediante la colaboración de un grupo de desarrolladores auto-organizados y multidisciplinares. Promueve un proceso de gestión de proyectos que fomenta el trabajo en equipo, la organización y responsabilidad propia, un conjunto de mejores prácticas de ingeniería que permiten la entrega rápida de software de alta calidad, y

un enfoque de negocio que alinea el desarrollo con las necesidades del cliente y los objetivos de la compañía. Las características básicas de los proyectos gestionados con un modelo ágil son:

- Incertidumbre: La dirección indica la necesidad estratégica que se desea cubrir, ofreciendo máxima libertad al equipo de trabajo.
- Equipos auto-organizados: No existen roles especializados
- Autonomía: Libertad para la toma de decisiones.
- Auto-superación: De forma periódica se evalúa el producto que se está desarrollando.
- Auto-enriquecimiento: Transferencia del conocimiento.
- Fases de desarrollo solapadas: Las fases no existen como tal sino que se desarrollan tareas/actividades en función de las necesidades cambiantes durante todo el proyecto.
- Control sutil: Establecimientos de puntos de control para realizar un seguimiento adecuado sin limitar la libertad y creatividad del equipo.

### 2.3. Metodologías Ágiles

A continuación se describen brevemente algunas de las metodologías ágiles más destacadas, con el fin de conocer sus características y ventajas para ser aplicadas en un proyecto que encaje en un modelo de ciclo de vida ágil.

Comenzamos con Adaptive software Development (ASD), (Highsmith, 2013) la cual fue desarrollada por Jim Highsmith y Sam Bayer a comienzos de 1990 y publicado en el año 2000. Esta metodología se adapta al cambio en lugar de luchar contra él. Se basa en la adaptación continua a circunstancias cambiantes. No hay un ciclo de planificación-diseño-construcción del software, sino un ciclo especular-colaborar-aprender:

- Especular, en esta fase las tareas se orientan a fijar objetivos del proyecto, estimar el marco temporal, determinar el número, duración, objetivos y funcionalidad de cada iteración.
- Colaborar, consiste en desarrollo concurrente del trabajo de construcción y gestión del producto. En esta fase del ciclo son revisados a fondo los requerimientos del proyecto. Se define cómo se va a trabajar de acuerdo a las habilidades de los integrantes del grupo.
- Aprender, la idea de ASD es el aprendizaje continuo en cada ciclo. Es un elemento crítico para la eficacia de los equipos. En cada iteración se revisa: calidad desde el punto de vista del cliente y los desarrolladores, rendimiento y situación del proyecto.

Agile Unified Process (AUP), es una versión simplificada de RUP (Rational Unified Process), el cual fue desarrollado por IBM. Describe una manera simple de entender el desarrollo de aplicaciones de negocio usando técnicas ágiles y conceptos heredados del RUP. El enfoque usa técnicas ágiles incluyendo desarrollo orientado a pruebas, modelado ágil, gestión de cambios ágil y refactorización de bases de datos para mejorar la productividad. Consta de las siguientes fases: Inicio (alcance), Elaboración (arquitectura), Construcción (construir incrementalmente) y Transición (validar y despliegue).

Crystal, (Cockburn, 2004) concebido por Alistair Cockburn (uno de los firmantes del manifiesto ágil), este modelo no describe una metodología cerrada, sino un conjunto de ellas, junto con los criterios para seleccionar y adecuar la más apropiada al proyecto. Cada

proyecto software puede caracterizarse según dos dimensiones, tamaño o dimensión (número de personas en el proyecto) y criticidad (consecuencia de los errores). En Crystal un determinante común a casi todas las metodologías ágiles para el éxito o fracaso, de un proyecto son las personas. Esta metodología se base en 7 propiedades que son: entregas frecuentes, mejora reflexiva, comunicación osmótica (cara a cara), seguridad personal (expresar su opinión), enfoque (periodos de no interrupción al equipo), fácil acceso al usuario experto y pruebas automatizadas.

Dynamic Systems Development Method (DSDM CONSORTiUM), definida para que fuera independiente de las herramientas y que pudiera ser utilizado en proyectos de tipo RAD (Rapid Application Development). Toma como referencia las buenas prácticas de la gestión ágil y entre sus principios se encuentran: involucración del usuario, equipo con poder de decisión, entregas frecuentes, desarrollo iterativo e incremental, los cambios son reversibles, entre otras. DSDM consta de tres fases:

- Pre-proyecto, preparado para que el proyecto comience y llegue a ser un éxito.
- Ciclo de vida del proyecto, consta de 5 etapas: estudio de viabilidad, estudio de negocio, iteración de modelo funcional, iteración de diseño y construcción e implementación.
- Post-proyecto, mantenimiento y corrección de errores.

Extreme Programming (XP) (Beck, 2000) fue ideada por Kent Beck, a partir de un conjunto de buenas prácticas y un modo de realizar el desarrollo de software, fundamentado en entregar constantemente, el mayor valor al cliente. En XP, hay cinco valores, catorce principios, trece prácticas primarias y once prácticas como corolario.

El proceso cosiste en un ciclo en que el cliente define que aporta valor se estiman esfuerzos y se decide que construir. Consta de 6 fases: exploración, planificación de la entrega, iteraciones, producción, mantenimiento y muerte del proyecto.

Feature Driven Development (FDD) ideada por Jeff De Luca y Peter Coad, combina el desarrollo dirigido por modelos con el desarrollo ágil. Se centra en el diseño de un modelo inicial, cuyo desarrollo será dividido en función a las características que debe cumplir el software e, iterativamente, se diseñará cada una de estas características.

Por tanto, cada iteración consta de dos partes, diseño e implementación de cada característica. A diferencia de otras metodologías ágiles no cubre todo el ciclo de vida sino sólo las fases de diseño y construcción. No requiere un modelo específico de proceso y se complementa con otras metodologías. Además, hace énfasis en aspectos de calidad durante todo el proceso e incluye un monitoreo permanente del avance del proyecto, por eso este tipo de metodología está dirigido al desarrollo de aplicaciones con un alto grado de criticidad.

Lean Software Development (LSD), (Poppendieck & Poppendieck, 2003) se basa en los fundamentos de la filosofía Lean para la fabricación. No es una metodología de ingeniería de software en el sentido convencional, es una síntesis de principios y una filosofía para construir sistemas de software. Si lean se considera un conjunto de principios más que prácticas, la aplicación de conceptos lean al desarrollo software y la ingeniería software puede ayudar a mejorar la calidad. Uno de sus principales principios consiste en estudiar el flujo de trabajo para eliminar la “muda” (desperdicios o procesos inútiles).

Kanban Software Development (Kanban), (Anderson, 2004) está basada en la metodología de fabricación industrial del mismo nombre. Su objetivo es gestionar de manera general como se van completando tareas, pero también se ha utilizado en la gestión de proyectos de desarrollo software. Se base en el desarrollo incremental, dividiendo el trabajo en partes.

Una de las principales aportaciones es que utiliza técnicas visuales para ver la situación de cada tarea. No existen unas fases definidas, sino que se habla de un flujo.

Scrum (Schwaber, 2009) es un proceso ágil que se puede usar para gestionar y controlar desarrollos complejos de software y productos usando prácticas iterativas e incrementales, mediante la utilización de prácticas tendientes a manejar la impredictibilidad y el riesgo a niveles aceptables.

Scrum incluye un conjunto de prácticas y roles. Los roles principales en Scrum son el "ScrumMaster" que mantiene los procesos y trabaja junto con el jefe de proyecto, el "Product Owner" que representa a las personas implicadas en el negocio y el "Team" que incluye a los desarrolladores.

Durante cada iteración (sprint), típicamente un periodo de 2 a 4 semanas (longitud decidida por el equipo), el equipo crea un incremento de software operativo. El conjunto de características que entra en una iteración viene del "product backlog", que es un conjunto priorizado de requisitos de trabajo de alto nivel que se han de hacer. Durante una iteración, nadie puede cambiar el backlog de la iteración, lo que significa que los requisitos están congelados para esa iteración. Cuando se completa una iteración, el equipo demuestra el uso del software.

Un principio clave de Scrum es el reconocimiento de que durante un proyecto los clientes pueden cambiar sus pensamientos sobre lo que quieren y necesitan, y de que los desafíos que no se pueden predecir no se pueden tratar fácilmente de una forma predictiva o planificada tradicional.

Scrumban (Kniberg & Skarin, 2010) es una metodología derivada y combinación de los métodos de desarrollo Scrum y Kanban. Sigue el flujo de trabajo continuo como lo define Kanban, pero se incluyen elementos de Scrum como, las reuniones diarias de 15 minutos y pequeñas retrospectivas con el afán de mejorar el proceso.

## 3. Método propuesto para la lección de modelo y metodología

Los pasos a realizar por un gestor comienzan por la elección del modelo de ciclo de vida a usar, y una vez elegido este, pensar que metodología es la más adecuada para el proyecto.

Para tomar estas decisiones se deberá identificar unos aspectos clave en cada proyecto, unas características genéricas que los diferencien a alto nivel para así poder clasificarlos. Los criterios a considerar serían la complejidad del problema, la criticidad del proyecto, el tiempo que disponemos para hacer la entrega final, si el usuario o cliente desea entregas parciales, la disponibilidad del cliente, la comunicación que existe entre el equipo de desarrollo y el usuario, el nivel del equipo, la certeza (o incertidumbre) que se tiene de que los requerimientos datos por el usuario son correctos y completos.

Estos criterios se han organizado en grupos relacionados con el proyecto, el equipo y el cliente.

### 3.1. Selección de modelo

El modelo en cascada es adecuado cuando se disponen de todos los requisitos desde el principio o en productos maduros donde los requisitos es difícil que cambien o en equipos débiles. En base a unos requisitos estables, podamos hacer unas estimaciones correctas y unos diseños adecuados que se mantengan a lo largo del tiempo. Al estar todo bien definido, de forma línea, y ser fácilmente entendible, este modelo podrá ser aplicado por casi cualquier equipo, tenga la experiencia que tenga y no necesitará del cliente más que en la fase de toma de requisitos, y en la entrega.

En el caso de modelo en V se aplica a proyectos pequeños donde los requisitos son entendidos fácilmente, en aplicaciones que si bien son simples, necesitan una confiabilidad muy alta.

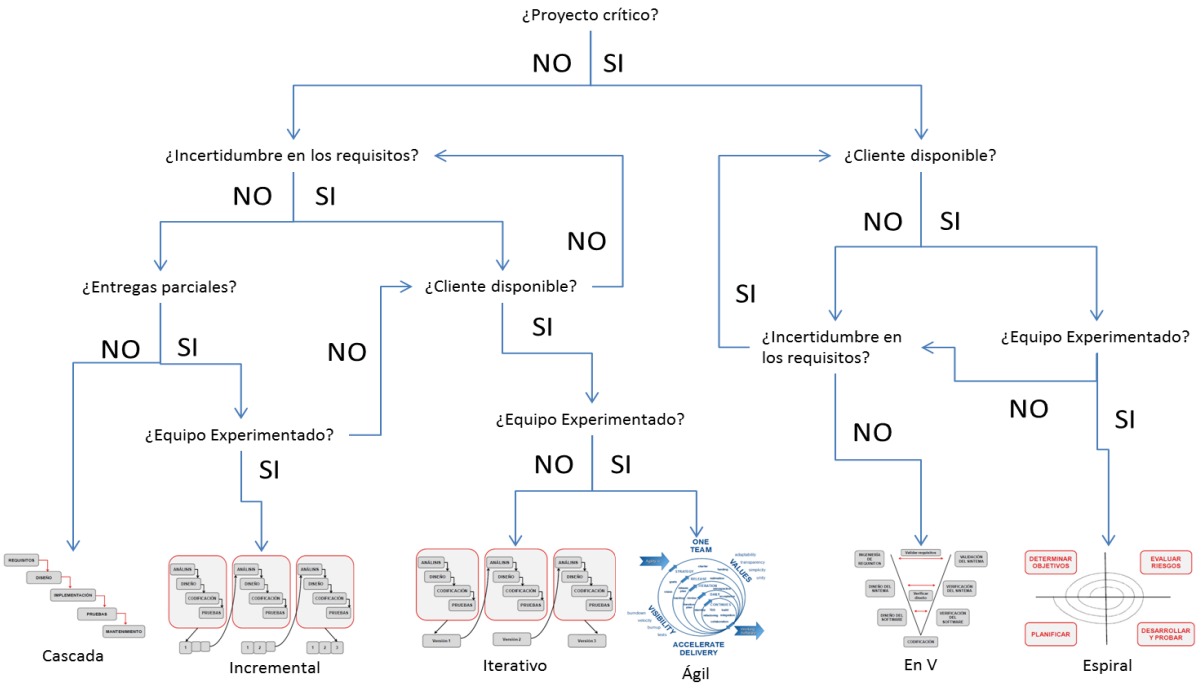
El modelo iterativo suele utilizar en proyectos en los que los requerimientos no están claros de parte del usuario, por lo que se hace necesaria la creación de distintos prototipos para presentarlos y conseguir la conformidad del cliente. Se puede adoptar el modelo mencionado en aplicaciones medianas a grandes, en las que el usuario o cliente final no necesita todas las funcionalidades desde el principio del proyecto y al que no le importa estar involucrado de forma activa en el proyecto. Quizás una empresa que debe migrar sus aplicaciones hacia otra arquitectura, desea hacerlo paulatinamente, es un candidato ideal para este tipo de modelo de ciclo de vida.

El modelo incremental es útil cuando nos encontramos ante un proyecto con unos requisitos conocidos pero donde el personal necesario para una implementación completa no está disponible. Los primeros incrementos se pueden implementar con menos gente. Además, los incrementos se pueden planear para manejar los riesgos técnicos. Este modelo de ciclo de vida no está pensado para cierto tipo de aplicaciones, sino que está orientado a cierto tipo de usuario o cliente y es útil cuando el usuario necesite entregas rápidas, aunque sean parciales, de un producto operativo.

Si en el modelo anterior, se parte de que no hay incertidumbre en los requisitos iniciales, en el modelo en espiral, es consciente de que se comienza con un alto grado de incertidumbre. Este último es adecuado para proyectos largos, caros, complicados y con incertidumbre en los requisitos. Para ello requiere de un equipo con la suficiente experiencia como para detectar y evaluar riesgos. Este modelo no llega a ser una alternativa al resto de modelos, sino que se podría situar por encima de ellos como un marco de trabajo.

Por último, el modelo ágil está pensado para proyectos con requisitos muy cambiantes, pensados especialmente para equipos pequeños y con experiencia, con plazos reducidos, requisitos volátiles y nuevas tecnologías

**Figura 1: Árbol de decisión para selección de modelo**



Como se puede apreciar en el arbol de decisión propuesto, en algunos casos se puede llegar a una situación de bucles infinitos, ante lo cual la medida a adoptar será cambiar una de nuetras respuestas (con las implicaciones que ello conlleve en el proyecto) para así poder avanzar en el árbol.

### 3.2. Selección de metodología ágil

El estudio realizado por (Iacovelli & Souveyet, 2008) clasifica las metodologías a través de cuatro puntos de vista, cada uno representando un aspecto de las metodologías, y cada punto de vista a su vez se caracteriza por un conjunto de atributos.

- Uso, refleja el por qué utilizar metodologías ágiles. Los atributos de esta vista tratan de evaluar todos los beneficios que el equipo de desarrollo y el cliente obtienen utilizando este tipo de metodologías: incremento de la productividad, calidad y satisfacción.
- Capacidad de agilidad, representa cuál es la parte ágil de la metodología. Los atributos de esta vista representan todos los aspectos del concepto de agilidad y su evaluación refleja que aspectos están incluidos en una metodología. Apuntar que el atributo colaboración se refiere a la relación que se establece entre los clientes y el equipo de desarrollo.
- Aplicabilidad, el objetivo de esta vista es mostrar el impacto de los aspectos ambientales en el método. Representa cuando el entorno es favorable para la aplicación de metodologías ágiles. Este aspecto se describe por los siguientes atributos, cada uno correspondiente a una característica del entorno.
- Procesos y productos, la vista de los procesos y productos representa cómo están caracterizados los procesos de la metodología y cuáles son los productos de sus actividades. Los procesos se componen de dos dimensiones. La primera representa el nivel de abstracción de sus directrices y reglas. La segunda dimensión son las actividades de desarrollo de software cubiertas por las metodologías ágiles. Por otro lado se listan los posibles productos de las actividades.

Para este trabajo se parte del estudio (Iacovelli & Souveyet, 2008) que identifica cuatro dimensiones y se opta por dar un valor de un enumerado para así dejar la tabla con un conjunto de atributos que sólo admiten dos posibles valores, lo cual, servirá para hacer nuestra elección de la metodología.

**Tabla 1: Valoración de "Uso" de las metodologías**

| | Pregunta | ASD | AUP | Crystal | DSDM | XP | FDD | LSD | Kanban | Scrum | Scrumban |
|---|---|---|---|---|---|---|---|---|---|---|---|
| USO ¿Por qué utilizar una metodología ágil? | Respeto de las fechas de entrega | V | V | V | V | F | V | F | F | V | F |
| | Cumplimiento de los requisitos | V | V | V | V | V | V | V | V | V | V |
| | Respeto de un nivel de calidad | V | V | F | F | F | V | V | F | F | F |
| | Satisfacción del usuario | F | F | F | V | V | F | V | F | V | V |
| | Adaptación a entornos turbulentos | V | F | F | V | V | F | F | V | V | V |
| | Favorable al off shoring | V | V | V | V | F | F | V | V | F | V |
| | Aumento de la productividad | F | F | F | V | V | F | V | V | V | V |

**Tabla 2: Valoración de "Capacidad de Agilidad" de las metodologías**

| | Pregunta | ASD | AUP | Crystal | DSDM | XP | FDD | LSD | Kanban | Scrum | Scrumban |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AD DE AGILIDA parte de agilida d incluid | Iteraciones cortas | F | F | V | V | V | V | V | F | V | F |
| | Colaboración | V | V | V | V | V | F | V | V | V | V |
| | Centrado en las personas | V | V | V | V | V | F | V | V | V | V |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Política de refactoring | F | V | F | V | V | F | V | F | F | F |
| Política de pruebas | V | V | V | V | V | V | V | F | V | V |
| Integración de los cambios | V | V | F | V | V | F | V | V | V | V |
| De peso ligero | F | F | F | F | V | V | V | V | V | V |
| Requisitos funcionales pueden cambiar | V | V | F | V | V | F | V | V | V | V |
| Requisito no funcional puede cambiar | V | F | F | F | F | F | F | V | F | V |
| Plan de trabajo se puede cambiar | F | F | F | F | V | F | V | V | F | V |
| Recursos humanos pueden cambiar | V | F | V | F | V | F | F | V | F | V |
| Se pueden cambiar indicadores | F | V | F | F | V | F | V | F | F | F |
| Reactividad | H | I | H | I | I | IP | I | H | I | H |
| Intercambio de conocimientos | A | B | A | B | A | B | A | A | A | A |

**Tabla 3: Valoración de "Aplicabilidad" de las metodologías**

| | | Pregunta | ASD | AUP | Crystal | DSDM | XP | FDD | LSD | Kanban | Scrum | Scrumban |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| APLICABILIDAD | ¿Cuándo un ambiente es favorable para usar este método? | Tamaño del proyecto | P | G/P | G/P | G/P | P | G | G/P | P | G/P | G/P |
| | | Complejidad del proyecto | A | A | A | A | B | A | A | B | A | A |
| | | Riesgos del proyecto | A | A | A | A | B | A | A | B | A | A |
| | | Tamaño del equipo | P | G/P | G/P | G/P | P | G | G/P | P | P | P |
| | | Grado de interacción con el cliente | A | B | B | A | A | B | A | A | A | A |
| | | Grado de interacción con los usuarios finales | B | B | B | A | B | B | B | B | A | B |
| | | Grado de interacción entre los miembros del equipo | A | B | A | A | A | B | A | B | A | A |
| | | Grado de innovación | A | A | B | A | A | B | A | B | A | A |
| | | Organización del equipo | J | J | AU | J | AU | J | AU | AU | AU | AU |

**Tabla 4: Valoración de "Procesos y productos" de las metodologías**

| | | Pregunta | ASD | AUP | Crystal | DSDM | XP | FDD | LSD | Kanban | Scrum | Scrumban |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Nivel de abstracción de las normas y directrices | | | | | | | | | | |
| PROCESOS Y PRODUCTOS | ¿Cómo están caracterizados los procesos de la metodología y cuáles son los productos de sus actividades? | Gestión de proyectos | V | V | V | V | F | V | F | F | V | F |
| | | Descripción de procesos | V | V | V | V | V | V | F | F | V | V |
| | | Normas y orientaciones concretas sobre las actividades y productos | F | V | F | F | V | V | F | F | V | V |
| | | Actividades cubiertas por la metodología | | | | | | | | | | |
| | | Puesta en marcha del proyecto | V | V | F | V | V | V | F | F | V | F |
| | | Definición de requisitos | V | V | F | V | V | V | V | F | V | F |
| | | Modelado | V | V | V | V | V | V | F | F | V | F |
| | | Código | V | V | V | V | V | V | V | V | V | V |
| | | Pruebas unitarias | V | V | V | V | V | V | V | V | V | V |
| | | Pruebas de integración | V | V | V | V | V | V | V | V | V | V |
| | | Prueba del sistema | V | V | V | V | V | V | V | V | V | V |
| | | Prueba de aceptación | V | V | V | V | V | V | V | V | V | V |
| | | Control de calidad | V | V | F | F | F | V | V | F | F | F |
| | | Uso del sistema | F | V | F | V | F | F | F | F | F | F |
| | | Productos de las actividades de la metodología | | | | | | | | | | |
| | | Diseño del modelo | F | V | F | V | F | V | F | F | V | F |
| | | Código fuente comentado | V | V | V | V | V | V | V | F | V | V |
| | | Ejecutable | V | V | V | V | V | V | V | V | V | V |
| | | Pruebas unitarias | V | V | V | V | V | V | V | V | V | V |
| | | Pruebas de integración | V | V | V | V | V | V | V | V | V | V |
| | | Pruebas de sistema | V | V | V | V | V | V | V | V | V | V |
| | | Pruebas de aceptación | V | V | V | V | V | V | V | V | V | V |
| | | Informes de calidad | V | V | F | F | F | V | V | F | F | F |
| | | Documentación de usuario | F | V | F | V | F | V | V | F | V | F |

Los acrónimos utilizados en las tablas son los siguientes:

- V = Verdadero, F = Falso
- A = Alto, B = Bajo
- IP = Inicio del Proyecto, H = Hito, I = Iteración
- G = Grande, P = Pequeño
- J = Jerárquico, AU = Auto-Organizado

Tomando como base las tablas presentadas para las metodologías, se realiza un formulario al jefe de proyecto con columna preguntas de las tablas anteriores. Comparando los resultados introducidos en los formularios con las tablas, se identificará la metodología que mejor se adapta a la forma de trabajo al proyecto. La metodología adecuada será la que mayor número de coincidencias tenga con el cuestionario anterior.

**Tabla 5: Verificación de los criterios para "Uso" a un caso particular**

| | | Pregunta | | ASD | AUP | Crystal | DSDM | XP | FDD | LSD | Kanban | Scrum | Scrumban |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| USO | ¿Por qué utilizar una metodología ágil? | Respeto de las fechas de entrega | V | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| | | Cumplimiento de los requisitos | V | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | Respeto de un nivel de calidad | V | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| | | Satisfacción del usuario | V | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| | | Adaptación a entornos turbulentos | V | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| | | Favorable al off shoring | F | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| | | Aumento de la productividad | F | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | | …… | … | …. | …. | …. | …. | … | …. | …. | …. | …. | …. |
| | | Total | | 42 | 40 | 32 | 38 | 27 | 32 | 37 | 26 | 41 | 31 |

En este supuesto las metodologías candidatas para el proyecto serian ASD, Scrum y AUP. Como resultado no se tiene una "única metodología válida" para este proyecto o la organización, sino que "la más adecuada" es la forma de interpretarlo. Es decir, el método da una forma de ordenar las metodologías ágiles disponibles en orden de "más adecuada" a "menos adecuada".

## 4. Conclusiones

Tras documentar en este trabajo el porqué del fracaso de los proyectos software, analizar distintos modelos de ciclo de vida y distintas metodologías ágiles, se ha llegado a varias conclusiones:

- Los modelos de ciclo de vida y las metodologías, son complementarios entre sí. En muchos casos, los paradigmas pueden y deben combinarse de forma que puedan utilizarse las ventajas de cada uno en un único proyecto.
- No se debe establecer un modelo y una metodología por defecto para todos los proyectos. Cada proyecto es un mundo que depende además de otros importantes factores como son el equipo que lo va a desarrollar o el cliente que lo solicita. En este trabajo se clasifica cual es aquel modelo y metodología que mejor se adapta a las características del proyecto.
- La elección de un modelo y una metodología a seguir es un paso crítico en cualquier proyecto software, y así lo deben entender tanto los jefes de proyecto, como el equipo que lo forman, como la organización para la que trabajan.

Este trabajo propone unas pautas en el inicio del camino de un proyecto software. El paso inicial propone un árbol de decisión para la elección del modelo de ciclo de vida del proyecto, en función de unos criterios resultado de recopilación de la literatura. En el caso

de que el modelo que mejor se ajuste al proyecto sea el modelo ágil se propone un formulario que dará una aproximación sobre la metodología a usar.

## 5. Referencias bibliográficas

Anderson, D. J. (2004). *Agile management for software engineering : applying the theory of constraints for business results*. PRENTICE-HALL INTERNATIONAL EDITION.

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN 0-201-61641-6.

Boehm, B. (1986). A Spiral Model of Software Development and Enhancement. *SIGSOFT Softw. Eng. Notes* 11(4), 14–24.

Cockburn (2004). *Crystal Clear: A Human-Powered Methodology for Small TeamsCockburn.* HARLOW: Addison Wesley.

Davis, A. M. (1993). *Software requirements: objects, functions, and states*. Englewood Cliffs, N.J.: PTR Prentice Hall. ISBN 013805763X 9780138057633.

DSDM CONSORTiUM. *DSDM*. [online]. Available from: http://www.dsdm.org/. [Accessed 2014-03-17].

Highsmith, J. (2013). *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Addison-Wesley.

Iacovelli, A. & Souveyet, C. (2008). *Framework for Agile Methods Classification*. Universite Paris.

Kniberg, H. & Skarin, M. (2010). *Kanban and Scrum: making the most of both*. [S.l.]: C4Media, Inc. ISBN 9780557138326 0557138329.

Mills, H. D. (1999). The management of software engineering, Part I: Principles of software engineering. *IBM Systems Journal* (2.3), 289 – 295.

Piorun, D. (2001). *Liderando Proyectos*. Macchi, Ediciones.

Poppendieck, M. & Poppendieck, T. D. (2003). *Lean software development: an agile toolkit*. Boston, Mass.: Addison-Wesley.

Royce, W. W. (1970). Managing the Development of Large Software Systems: Concepts and Techniques., 1970.

Schwaber, K. (2009). *Agile Project Management with Scrum*. Microsoft Press. ISBN 9780735637900.

Sommerville, I. (2005). *Ingeniería del software*. Pearson Educación.

Standish Group (2010). *CHAOS Report 2009*. Standish Group.