Máster Universitario en Software Libre

Proyecto Fin de Máster

Curso Académico 2012/2013

Metodologías, Modelos de Desarrollo y ALM

Autor: Ricardo García Fernández

Tutor:

Índice general

Capítulo 1

Introducción

SidelabCode Stack es una forja de desarrollo de Software para su uso como herramienta **ALM**. Es una herramienta FLOSS (Free Libre Open Source Software) con Licencia **TBC**.

El desarrollo de este proyecto se basa en el diseño e implementación del proceso de desarrollo acorde con las metodologías ágiles a través de la forja *SidelabCode Stack*. Unificando herramientas a través de las distintas APIs basadas en la interoperabilidad, facilitando la instalación, la replicación y la recuperación de los datos mediante el uso de Software Libre para construir Software de calidad.

El uso de metodologías ágiles se encuentra intrínsecamente relacionado a lo largo del proceso de desarrollo e implementación de la forja. Por otra parte podemos afirmar que no

es una herramienta intrusiva para la aplicación de las distintas metodologías ligadas a un proceso de desarrollo.

Presentando las distintas metodologías de desarrollo de software definidas analizando sus características tanto las buenas como las malas y como éstas se aplican al diseño de la herramienta encaminado un proceso de desarrollo fluido y ágil para un desarrollador o un grupo de desarrolladores.

El proceso de desarrollo de software a implementar como base para la herramienta SidelabCode Stack, es el **iterativo e incrementa**.

Proceso ágil que se expande sin desviar su orientación al objetivo fijado. Tiene la capacidad de variar el contenido en el camino, tanto a nuevas inclusiones en como a la restricción de otras, mientas que su objetivo final, se mantiene fijo. Es un proceso que se adapta a los cambios, asumiendo en cada iteración nuevas características para transformarlas en resultados.

Un punto a tener en cuenta es la ergonomía con la que cuenta la herramienta para el día a día y las soluciones que aporta con respecto a otras herramientas que cohabitan en el mismo campo.

Todo con un mismo fin; producir software de calidad evaluable, es decir, no es necesario crear un producto perfecto sino se sabe mejorarlo y adaptarlo a nuevas necesidades. Para eso utilizamos las herramientas ALM en los proyectos ya que nos ayudan a tener una visión diaria y a posteriori otorgando una perspectiva para poder medir y evaluar las mejoras entre dos puntos de tiempo. Por lo tanto *la evolución del proyecto y la adaptabilidad a los cambios* como mayor valor.

1.1. Etimología

Sidelab es, un laboratorio de software. Partiendo de esta base, encontramos una definición exacta para SidelabCode [1]:

[1] `http://code.sidelab.es/projects/sidelab/wiki/Sidelab`

> *Sidelab es el "laboratorio de software y entornos de desarrollo integrados"(Software and Integrated Development Environments Laboratory). Es un grupo de entusiastas de la programación con interés en prácticamente todos los aspectos del desarollo, desde los lenguajes de programación y los algoritmos avanzados, hasta la ingeniería del software y la seguridad informática. Nuestros principales intereses se centran en el desarrollo software y la mejora y personalización de los entornos de desarrollo integrados (IDEs) y herramientas relacionadas.*

En el otro extremo del nombre se encuentra *Code* se refiere al código en sí hacia donde se orienta esta herramienta. Por ultimo pero no menos importante tenemos *Stack*, es una pila de servicios para la gestión y el desarrollo de proyectos software.

Como resultado tenemos **SidelabCode Stack** una Forja de desarrollo de aplicaciones orientada a un proceso de desarrollo.

1.2. Trabajo en TSCompany

TSCompany (trabajo desempeñado a grandes rasgos) ->aquí se puede justificar muy bien el trabajo por la necesidad de implantar en un entorno real una forja funcional.

Explicación del trabajo efectuado en TSCompany durante el desarrollo de la implementación de la Forja.

El prácticum del Máster me dio la oportunidad de entrar a trabajar en la empresa TSCompany para cubrir la plaza de becario. Desde el principio he estado interesado en la producción de software de calidad, en las herramientas e control de versiones, el desarrollo orientado a tests, la documentación y de como unificar las herramientas para obtener un rendimiento óptimo a la hora de desarrollar un proyecto, es decir, un desarrollador que desarrollar al 100 %.

Esta colaboración se basó en la aportación al proyecto de *Software Libre* SidelabCode Stack.

Profesionalmente siempre he tenido ejemplos cercanos acerca del desarrollo de software de calidad y me ha han inculcado el desarrollo orientado a tests *TDD*. Siempre he perseguido la simplicidad y de esta forma la replicación de un entorno, pruebas, instalaciones y la estandarización de la programación para una comprensión sencilla.

El objetivo inicial era pulir el funcionamiento de la herramienta SidelabCode Stack unificando el proceso de desarrollo en la instalación ya que habían módulos que todavía no estaban conectados, es decir, centralizados, por lo que la funcionalidad quedaba dispersa para el usuario final. Por ejemplo, el usuario tenía que repetir la ejecución de una serie pasos para poder incluir un proyecto en un repositorio distribuido Git sin que éste tuviera relación con el gestor de tareas, había herramientas que estaban desacopladas.

Había que dar forma al modelo de desarrollo y promocionarlo dentro de la estructura de la forja para que resultase fácil al usuario empezar con el desarrollo de un proyecto dentro del entorno SidelabCode.

Después de la formalización de las distintas versiones del proyecto, se invirtió un tiempo para la implantación de la forja en la empresa TSCompany para su uso diario. Partiendo de la migración de la información de todos los proyectos y usuarios de la empresa para así poder gestionarlos a través de un interfaz común, SidelabCode Stack. Debido al uso de herramientas de Software Libre la migración fue fluida y nada agresiva para los usuarios. El entorno de trabajo seguía siendo el mismo pero en este caso aportando un extra de calidad y métricas para la evaluación de los proyectos desarrollados.

Capítulo 2

Motivación

En el mismo punto, damos la bienvenida a *SidelabCode Stack*, la herramienta ALM para el proceso de desarrollo.

Empezaremos ubicando la forja de desarrollo SidelabCode Stack. ¿ Cual es la motivación que nos lleva a implementar un nuevo ALM ?

Debido a la necesidad de tener un esqueleto para el desarrollo de un proyecto a través del uso de herramientas de Software Libre actuales generando una nueva herramienta de Software Libre.

Partiendo del principio DRY (*Don't Repeat Yourself*) y de la mano de *No reinventar la rueda* el proyecto surgió con la necesidad de crear una forja de desarrollo Libre y usable en distintos entornos.

Después de analizar Las posibles soluciones existentes, se optó por la unificación de las herramientas evaluadas para la generación de una nueva forja.

Después de analizar estas soluciones ALM existentes se optó por crear una nueva Forja a partir de las herramientas necesarias para el proceso de desarrollo de Software de Calidad, siguiendo las metodologías ágiles existentes.

La necesidad de crear una herramienta que aglutinase la gestión de las tareas, el código fuente y la orientación del desarrollo a los Tests. Todo el proceso de desarrollo unificado para facilitar el trabajo.

La integración del proceso de desarrollo en la implementación de una Forja ALM es el punto clave de SidelabCode Stack. Para crear un entorno adecuado se tuvieron en cuenta distintas características en el proceso de creación:

- Ser efectiva: la modificación de una capacidad específica no requerirá el cambio de la plataforma completa, ni un cambio en el despliegue de la solución.
- Ser mantenible y que sus funcionalidades sean fácilmente extensibles y reutilizables.
- Permitir la fácil portabilidad de los datos de los proyectos entre distintas forjas para la migración.

Capítulo 3

Objetivos

Constancia, metodología, facilidad de uso basado en herramientas al alcance de cualquier desarrollador dentro de un proceso de desarrollo para crear software de calidad. Estos son los principios en los que se basa la implementación de SidelabCode Stack como forja de desarrollo teniendo como objetivos tangibles la unificación del uso de las herramientas necesarias para cumplir con éstos como objetivo.

Integrar el desarrollo completo de un proyecto a partir de estos objetivos convertidos en los siguientes requerimientos.

3.1. Proyecto, usuarios, permisos y roles

La gestión del proyecto a partir de la definición de los usuarios, permisos y roles definidos para el mismo. Centralizar esta gestión para así obtener un control fluido y centralizado de los *"poderes"* otorgados según el rango del usuario a través de una herramienta.

3.2. Carpetas compartidas

Generar un espacio de carpetas públicas/privadas para cada uno de los proyectos a partir de los permisos de cada usuario. Asegurando el acceso público/privado a los recursos que

se publiquen.

3.3. ITS

Empezando por el Análisis de Requerimientos en un proyecto es necesaria una herramienta que ayude en el proceso de desarrollo, en esta caso un *ITS Issue Tracking System* para la gestión y el mantenimiento de las tareas de cada proyecto. La piedra angular de todo proyecto de software. El ITS otorga un control total del flujo de trabajo y responsabilidades a los usuarios del proyecto, una base de documentación y gestión del tiempo prioritaria en todo proyecto de software.

Utilizar el ITS para gestionar las tareas y los posibles errores dentro de cada proyecto para planificar la corrección o la implementación de éstas, dividiendo las entregas del proyecto en versiones durante el proceso continuo de desarrollo.

3.4. Repositorio de código

El Repositorio de código asociado a cada proyecto. Sin repositorio de código no hay proyecto. De esta forma la cooperación entre usuarios de un proyecto se agiliza bajo el manto de un sistema de control de versiones de código fuente. Esta herramienta produce la información necesaria para generar un histórico de cambios y poder evaluar la evolución del proyecto.

Además se ha de dotar de una herramienta para la Gestión del repositorio. Es decir, la gestión de parches, actualizaciones y desarrollo por ramas dentro de un entorno controlado.

3.5. Gestión de librerías

La gestión de librerías centralizada para los proyectos dentro de la forja, ofrece la posibilidad de interconectar distintas librerías dentro de todo el entorno. Publicando las librerías clasificadas a través de las distintas versiones en un repositorio de librerías común. La reutilización del código y la posibilidad de mejora entre distintos proyectos que compartan librerías mejora la calidad del mismo.

3.6. TDD Test Driven Development

El desarrollo dirigido por Tests *TDD* ayuda a la generación de código de calidad, legible y reutilizable. Siguiendo este proceso, el tiempo dedicado a la replicación de errores disminuye por lo que se puede dedicar más tiempo a la corrección del error en si.

Otorga datos válidos para el análisis del código y controlar la evolución de la cobertura de Tests para poder evaluar la deuda técnica. Genera los datos suficientes para facilitar la corrección del a misma.

3.7. CI - Integración Continua

El TDD, el desarrollo por ramas y los despliegues en diferentes entornos completan un proceso de desarrollo que culmina en la integración continua del proyecto. Este es el control al más alto nivel dentro de la parte técnica en donde el control de la calidad del código se toma como estandarte. Prevé los errores previos entre distintas versiones del código al unificar las ramas, fallos en los tests y validación de la calidad del código.

Crear despliegues en distintos entornos o replicar entornos similares al entorno final desde esta herramienta para minimizar los errores o poder replicar fácilmente los mismos cuando surjan.

3.8. Interoperabilidad

La interoperabilidad entre todas las herramientas para gestionar la configuración a través de una herramientas centralizada. Este apartado es donde se muestra el potencial de la forja de desarrollo ya que la configuración se expande o replica a las demás herramientas dando vida a la comunicación entre ellas para crear la forja.

Se transforma la configuración inicial y se establecen los protocolos de comunicación entre cada una de las herramientas dentro del proceso de desarrollo habiendo definido los pasos a seguir a partir de las *API*s existentes.

Capítulo 4

Procesos de desarrollo

Un proceso de desarrollo en el mundo del software se define como:

> *El proceso de transformación de unos requerimientos en una solución. Un conjunto de pautas a seguir para completar el ciclo de vida de la solución de una manera organizada, sistemática y que ayude a las personas a completar los objetivos fijados*

En SidelabCode Stack se optó por implementar el proceso de desarrollo de software *iterativo e incremental* para crear el diseño de la forja SidelabCode Stack a través de metodologías ágiles.

El proceso de desarrollo infiere directamente en la calidad del software que se construye. Es la parte más importante en el software ya que un proceso de desarrollo óptimo para una solución otorga las herramientas necesarias para una mejor evolución del mismo. Cada proceso de desarrollo ha de aplicarse a la solución según los requisitos de la misma, no todos los procesos de desarrollo son válidos para todos los proyectos.

4.1. Software de Calidad

Desarrollar Software de Calidad, para ellos nos encontramos con la palabra *Çalidad"* tan subjetiva en muchos ámbitos, pero que en el desarrollo puede ser bastante objetiva ya

que se trata de Software, una ciencia evaluable. La Calidad del Software es el conjunto de cualidades que lo caracterizan y determinan su viabilidad y utilidad; Mantenibilidad, Fiable, Eficiencia y Seguridad.

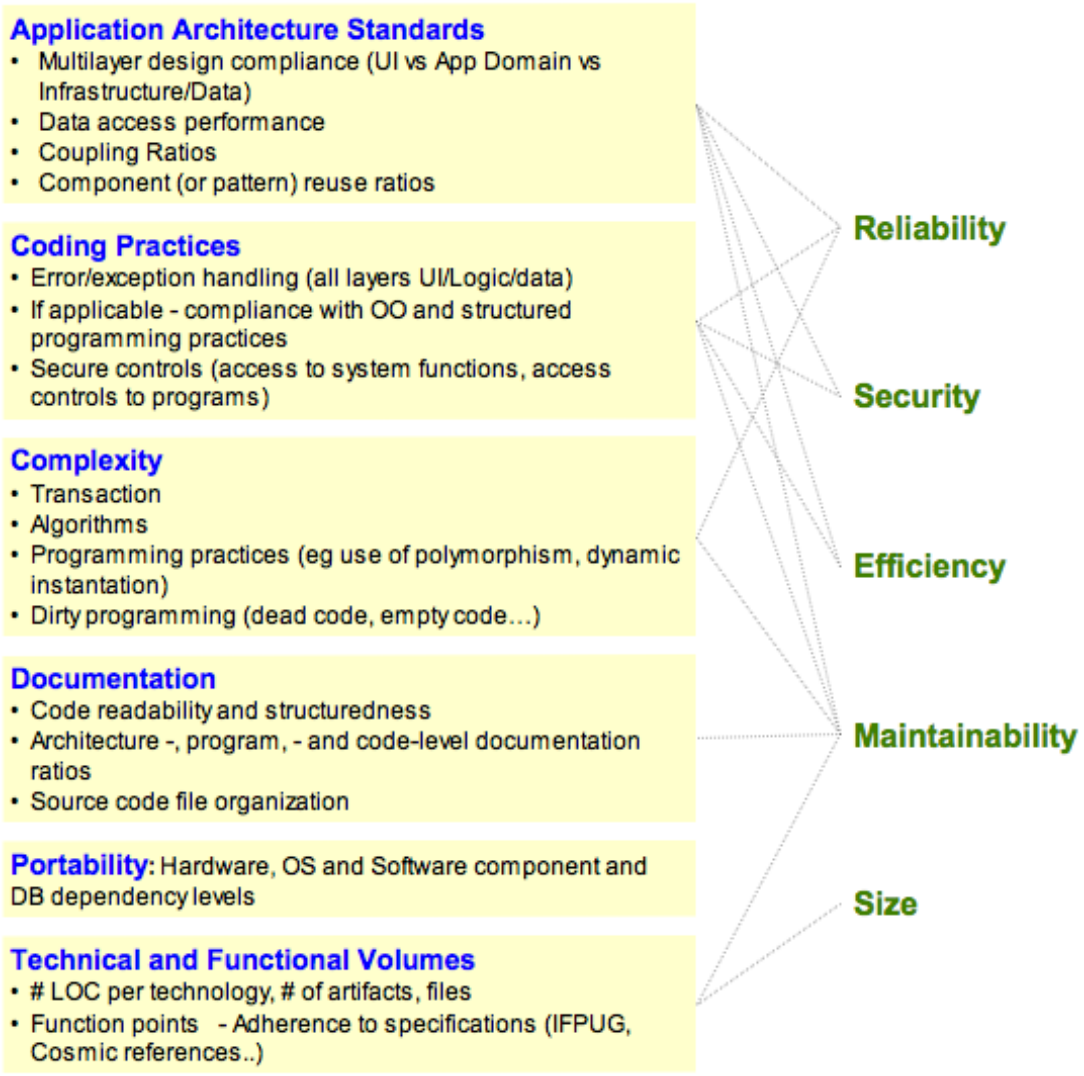

Figura 4.1: Software de Calidad

Un software hecho para ejecutarse una sola vez no requiere el mismo nivel de calidad mientras que un software para ser explotado durante un largo necesita ser fiable, seguro, mantenible y flexible para disminuir los costes.

- *Mantenibilidad*: El software debe ser diseñado de tal manera, que permita ajustarlo a los cambios en los requerimientos. Esta característica es crucial, debido al inevitable cambio del contexto en el que se desempeña un software.

- *Fiabilidad*: Incluye varias características además de la fiabilidad, como la aplicación de estándares, complejidad, tratamiento de errores.

- *Eficiencia*: Tiene que ver con el uso eficiente de los recursos que necesita un sistema para su funcionamiento.

- *Seguridad*: La evaluación de la seguridad requiere un control sobre la arquitectura, el diseño y las buenas prácticas.

4.2. Proceso iterativo

¿ Que es el proceso iterativo ?

> *La primera versión debe contener todos los requerimientos del usuario y lo que se va a hacer en las siguientes versiones es ir mejorando aspectos como la funcionalidad o el tiempo de respuesta.*[1]

Se centra más en la inmediatez de la primera versión y en las mejoras posteriores que se van creando enfocadas a la solución final. En el proceso también juega una parte fundamental la comunicación con el cliente a través de la visualización de los resultados por iteraciones. De esta forma se consigue una buena coordinación entre el cliente y el equipo de desarrollo para la consecución de los objetivos.

[1] Procesos Iterativos e Incrementales - `http://esalas334.blogspot.es/1193761920/`

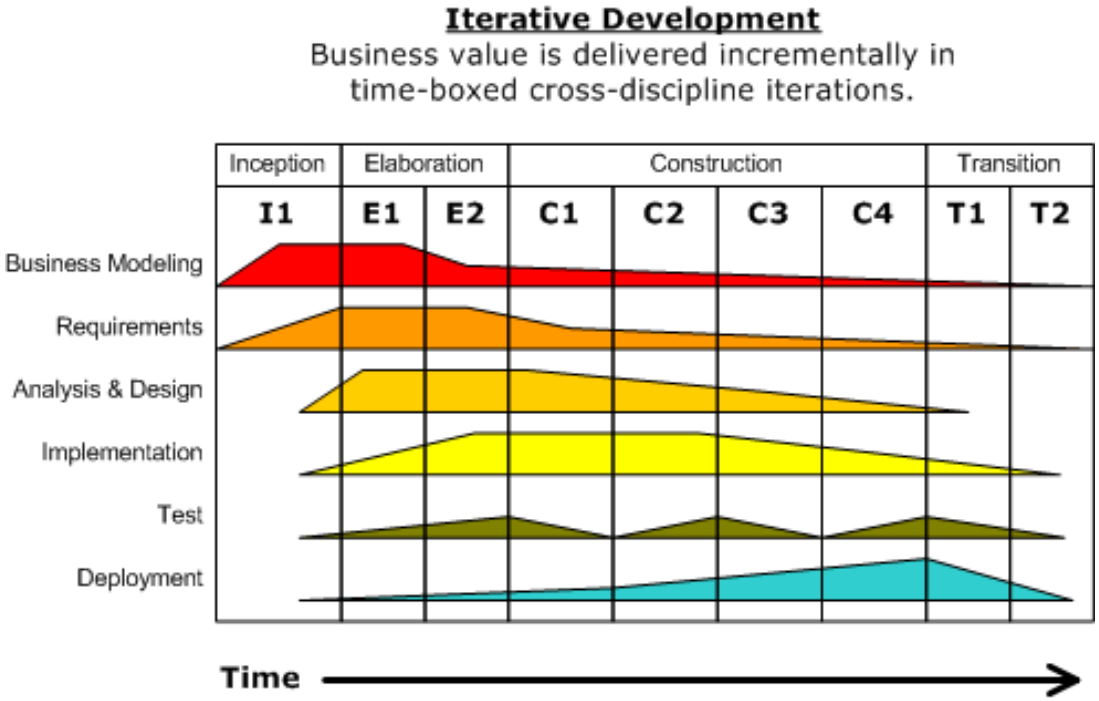

Figura 4.2: Desarrollo Iterativo

Se han de tener en cuenta posibles cambios entre iteraciones pero nunca del resultado completo, para que de esta forma se pueda controlar a tiempo la *desviación* que pueda existir en el proceso de la creación de producto.

> *Como la idea que representa la palabra iterativo, un proceso de desarrollo de software iterativo es aquel al que se lo piensa, como una serie de tareas agrupadas en pequeñas etapas repetitivas. Estas "pequeñas etapas repetitivas"son las iteraciones.*[2]

La base el proceso de desarrollo Iterativo provee un conjunto de pasos para el desarrollo de la solución que se repiten iteración tras iteración para la creación de mejoras tangibles y/o evaluables.

[2]Proceso de Desarrollo Iterativo - Fernando Soriano - `http://fernandosoriano.com.ar/?p=13`

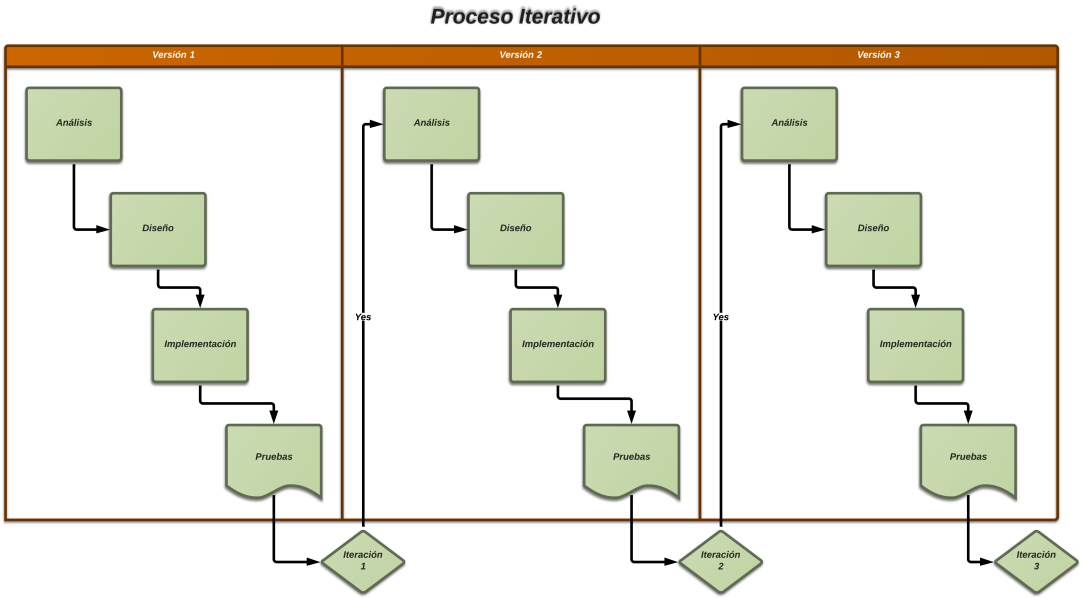

Figura 4.3: Proceso Iterativo

En cada iteración se construye una pieza funcional del producto final, completa, testeada, documentada e integrada en la solución final. La visión completa de este proceso muestra una línea de iteraciones separadas funcionalmente unas de otras que en conjunto, forman la solución final. Iteraciones independientes unas de otras a través de un desarrollo lineal agrupando pequeños ciclos de desarrollo.

- *Duración fija*, quiere decir que una vez establecidos los tiempos o planificación de la iteración, la iteración termina en la fecha exacta establecida. Si el equipo no pudo cumplir lo planificado, el desarrollo pendiente pasa a otra iteración.
- Estimación de tiempos cortos, las *"buenas prácticas"* hablan de que una iteración debiera durar entre 2 y 6 semanas.
- Es como un ciclo de desarrollo completo, ya que en una iteración se realizan actividades de análisis, diseño, implementación, pruebas, etc.

4.3. Proceso incremental

El Proceso Incremental fue propuesto por *Harlan D. Mills* en 1980.

Sugirió el enfoque incremental de desarrollo como una forma de reducir la repetición del trabajo en el proceso de desarrollo y dar oportunidad de retrasar la toma de decisiones en los requisitos hasta adquirir experiencia con el sistema.

El modelo incremental combina elementos del modelo lineal secuencial (aplicados repetidamente) con la filosofía interactiva de construcción de prototipos. El modelo incremental aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario.

Cada secuencia lineal produce un *ïncremento"* en el desarrollo de la solución. Por ejemplo, en relación a la forja SidelabCode Stack; en la primera versión estaba accesible el módulo de Jenkins, en el siguiente incremento la configuración de Jenkins se ligaba automáticamente a la configuración del los usuarios por proyecto, el siguiente incremento se publicaban las instrucciones para gestionar Jenkins a partir de una cuenta y facilitar la configuración para los distintos entornos.

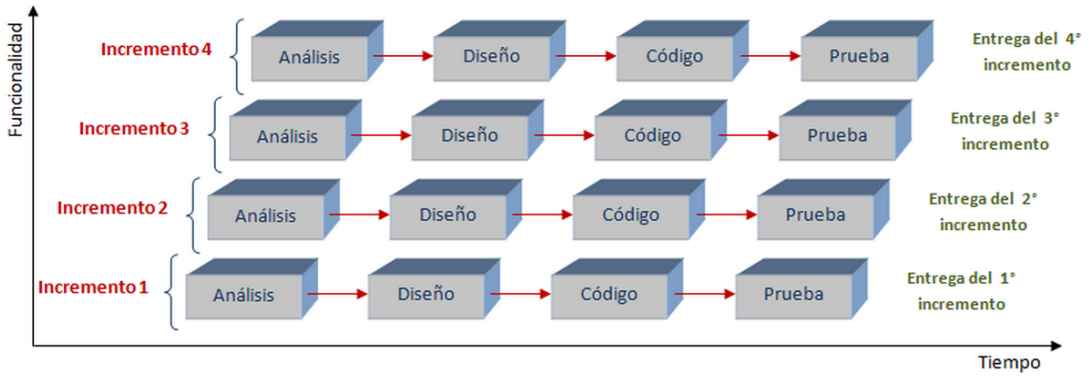

Figura 4.4: Modelo Incremental

Al iniciar el desarrollo, los clientes o los usuarios, identifican a grandes rasgos las funcionalidades que proporcionará el sistema. Se define un bosquejo de requisitos funcionales y será el cliente quien se encarga de priorizar que funcionalidades son más importantes. Con las prioridades definidas, se puede confeccionar el plan de incrementos, en donde cada incremento se compone de un subconjunto de funcionalidades a desarrollar.

4.4. Iterativo e Incremental

Desarrollo iterativo e incremental. La conjunción de estos dos tipos de desarrollo aúnan las mejores cualidades de ambos para gestión de un equipo de trabajo en la construcción de una solución.

El proceso iterativo e incremental se basa en incrementos por cada una de las iteraciones en el proceso de desarrollo. La idea básica de este proceso es desarrollar una solución a través de las iteraciones de ciclos a partir de los incrementos en la funcionalidad para que los desarrolladores mejoren su productividad en torno al proyecto a partir de pequeños hitos que completan versiones usables de la solución desde la primera implementación.

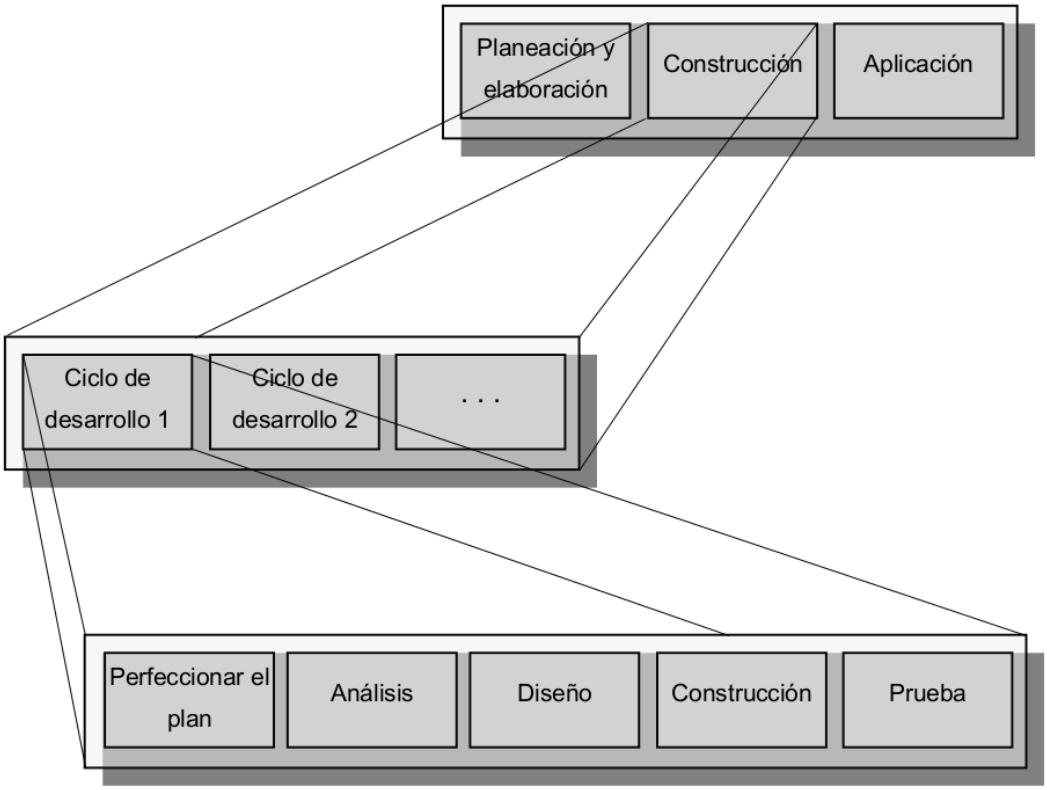

Figura 4.5: Proceso iterativo e Incremental por Larman

La evolución de la solución se basa en las iteraciones pasadas añadiendo los nuevos requisitos/objetivos (pueden ser mejoras o nuevas funcionalidades). Se basa en incrementar

el valor del trabajo hecho para tener un control del proceso más exhaustivo priorizando los objetivos. Por cada iteración existen modificaciones en el diseño y en las funcionalidades.

La comunicación y la implicación en el desarrollo del proyecto con el usuario/cliente desde el inicio del proyecto es crucial ya que el proceso parte desde una solución inicial para incluir versiones usables con el usuario/cliente. De esta forma todas las partes aportan sus distintos puntos de vista de una manera continua implicándose en el proceso y midiendo el crecimiento de la solución paso a paso, incremento a incremento.

Este proceso de desarrollo cercano a las *Metodologías Ágiles*, como ellas tiene el mismo fin, la implicación, desarrollo, fiabilidad, confianza, aprendizaje, versatilidad, responsabilidad, comunicación con la solución desarrollada y el usuario/cliente durante el proceso.

Una de las claves en este proceso es la retroalimentación y el aprendizaje del grupo de trabajo a partir de las iteraciones. De esta forma el trabajo hecho repercute en las iteraciones futuras aportando nuevos conocimientos sobre el proceso y el desarrollo. Creando mejores iteraciones y evoluciones de la solución, *profesionalizando el trabajo*.

Las Iteraciones han de ser de una duración corta de *2 a 6 semanas* para que la comunicación a todos los niveles del proyecto siga siendo fluida y para que si en algún caso se haya de desechar una Iteración no se pierda mucho trabajo desarrollado en ella. Esto no ocurre muy a menudo pero se contempla por diferentes causas:

- Abandono del proyecto.
- Cambio de Cliente/Usuario.
- Falta de recursos.

Las Iteraciones *çortas"* otorgan al modelo un alto nivel de versatilidad a la hora de evolucionar y evaluar el recorrido del trabajo hecho en el proyecto, para así poder predecir la organización de las futuras iteraciones.

4.4.1. Fases del proceso

El proceso de desarrollo Iterativo e Incremental está basado en tres fases:

- Iniciación.
- Iteración.
- Lista del Control del Proyecto.

El objetivo de la *Iniciación* es la implementación inicial para crear un producto con el cual el usuario/cliente pueda interactuar y tener las primeras impresiones. El equipo de desarrollo y el usuario/cliente toman como punto base esta fase de *Iniciación*.

Esta primera implementación ha de servir de guía para la evolución del desarrollo en cada una de las iteraciones, la base.

La *Lista de Control del Proyecto* es el lugar donde se definen las tareas que han de cumplimentarse durante el proceso de desarrollo. Todos los aspectos relacionados con la implementación de la solución se encuentran definidos en la lista, funcionalidades, diseño, errores, mejoras. La Lista de Control está en constante evolución, no es un muro estático, ya que se van adjuntando las funcionalidades y/o mejoras y posibles nuevas funcionalidades. De esta forma se evalúan las prioridades en la fase de análisis por cada iteración y se decide que tareas han de implementarse y cuales son desechadas para la siguiente iteración.

La *Iteración* es un conjunto modular de acciones a llevar a cabo para cumplir con las tareas que se definen para evolucionar la solución por incrementos. Debe estar sujeta a cambios en el diseño, nuevas tareas añadidas a la lista de control y sobretodo, ser simple.

4.4.2. Desarrollo Iteración

El desarrollo del proyecto viene medido por las Iteraciones que se conectan una a otra secuencialmente.

La iteración comienza a partir del análisis basado en la retroalimentación de los usuarios

y los servicios de análisis disponibles. Los elementos a tener en cuenta en el análisis son:

- Estructura.
- Modularidad.
- Ergonomía.
- Eficiencia.
- Objetivos logrados.

Se han tener en cuenta los posibles riesgos que puedan surgir en la Iteración para evaluarlos y eliminarlos en el momento de definir la línea base de la arquitectura. Además el equipo de desarrollo ha de dominar y estar al tanto del lenguaje empleado en los requisitos, el problema que se va a abordar y de esta manera ser capaces de asumir los posibles riesgos o imprevistos que puedan surgir.

Los resultados del análisis se reflejan en la Lista de Control del proyecto para añadir, modificar y ordenar por prioridades para la siguiente Iteración.

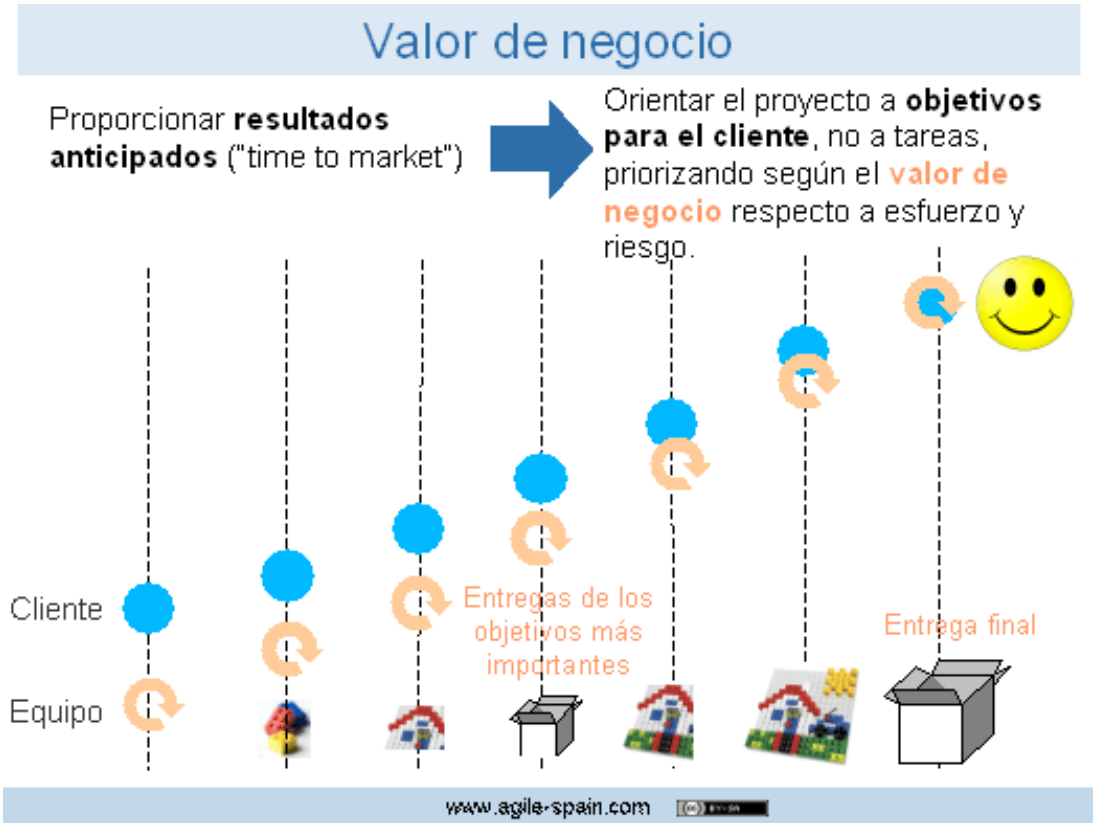

Figura 4.6: Valor de negocio Iterativo e Incremental

En las Iteraciones posteriores ha de aumentar la capacidad de reducir los riesgos, desarrollar los componentes e ir evolucionando incremento a incremento hacia la versión final para el usuario/cliente.

Cada Iteración se reduce a un *miniproyecto* (Se les llama miniproyectos porque no es algo que el usuario haya pedido) que consta del proceso de requisitos, análisis, diseño, implementación y prueba.

El proceso de desarrollo en una Iteración se reduce a una serie de guías o pasos a seguir en torno a las modificaciones que surgen a la medida que se avanza en el desarrollo. El proceso Iterativo e Incremental se basa en la flexibilidad por lo tanto las modificaciones de la Iteración han de ser otra herramienta más para lograr los objetivos y como tales:

- Cualquier dificultad encontrada en el diseño, desarrollo y prueba de una modificación puede alertar de la necesidad de cambiar el diseño o la implementación. Se han de desarrollar las modificaciones con una estructura modular y aislada (en su medida)

para poder trabajar en un problema o solución concreta y que no afecte al resto de la implementación.

- Las modificaciones han de ser sencillas de implementar, sino se ha de rediseñar el la solución.

- Las modificaciones han de ser más sencillas conforme se van completando iteraciones. Si esto no ocurre existe un problema de diseño y puede incurrir en el exceso de soluciones *ad-hoc*, parches.

- Los parches se contemplan como soluciones temporales en distintas iteraciones para que no sea necesario un cambio en el diseño, pero sólo para casos excepcionales. Si los parches proliferan se ha de replantear el diseño.

- La implementación existente ha de ser analizada constantemente para certificar que sigue el camino marcado por los objetivos a corto y largo plazo del proyecto. De esta forma se controlan las posibles desviaciones de tiempo y el trabajo hecho por el grupo de trabajo.

- Los herramientas de análisis se han de utilizar para validad los análisis y/o funcionalidades de las implementaciones parciales de la solución.

- La participación del usuario/cliente en el proceso ha de ser solicitada y analizada para contemplar posibles deficiencias o errores en la implementación actual. De esta forma es como se ha de crear una canal de comunicación para interactuar con el grupo de trabajo.

4.5. Gestión de tareas

Gestión de tareas: ¿qué hay que hacer? ¿quién tiene que hacerlo? ->Sistemas de gestión de tickets

4.6. Código versionado

Código versionado ->repositorios de código

4.7. TDD y CI

TDD ->sistemas de CI para asegurar que los tests se pasan regularmente

4.8. Desarrollo por canales

4.9. Herramientas

Capítulo 5

ALM Tools

* ALM Tools - Forjas de desarrollo * Qué es una forja * Objetivos * Componentes * Estudio del arte de forjas * Problemática ->administración, costes... * Tablas comparativas * Algunos ejemplos y sus limitaciones * Conclusiones del estudio de forjas

ALM Tools significa: *Application Lifecycle Management Tools*. La gestión del ciclo de vida de una aplicación, el conjunto de herramientas encargadas de guíar al desarrollador a través de un camino basado en metodologías para la creación de un Software hacia su estado del arte.

Integrar el proceso de desarrollo de Software a través de las ALM Tools como vehículo, es decir no existe en si mismo una herramienta ALM, sino que la herramienta ALM gobierna a las herramientas incluidas en el proceso de desarrollo.

5.1. Historia

En todas las empresas o comunidades que desarrollan Software siempre se aplica un proceso de desarrollo a la creación del producto. Cada una utiliza distintas herramientas para la gestión de los proyectos de Software, gestor de correo, gestor de incidencias, repositorio de código, integración continua. Pero en la mayoría de los casos de una forma dispar y sin seguir ninguna convención.

A veces el no conocimiento de otras herramientas o la no inclusión de nuevas puede hacer que el desarrollo del proyecto no mejore, partiendo de la base de que el desarrollo puede ser óptimo para las herramientas utilizadas, carece de perspectivas de mejora a corto plazo.

Si se opta por la integración de una nueva herramienta en el proceso de desarrollo el coste de integración se habría de evaluar ya que se debería dedicar un esfuerzo a la integración ad-hoc de la nueva herramienta para el uso en este mismo entorno con el coste que conlleva, evaluación, test, integración, interoperabilidad, es decir un nuevo proyecto dentro del mismo proyecto.

Después de esta integración en el proceso de desarrollo en la empresa habría que hacer un esfuerzo para salvaguardar la información a través de las distintas herramientas por separado.

El proceso de unificación y reutilización de herramientas a los desarrolladores nos puede parecer familiar si lo comparamos con el uso de los Frameworks a principios de los años 2000. Muchas empresas o comunidades empezaron a adecuar e implementar sus desarrollos en base a un Framework creado por ellos mismos. Estos Frameworks se adecuan a sus requerimientos pero su uso era interno en la empresa y por lo tanto lejano a los estándares. Uno de los más famosos es sin duda el caso de Spring, proyecto de más de 10 años de edad que goza de buena salud y aceptación, incluso equivoca a algunos entre Java y Spring. En este pequeño paralelismo podemos encontrar el estado de las forjas de desarrollo ALM Tools, cuando el proyecto requiere de herramientas para facilitar su ciclo de vida y se van ensamblando una tras otra, que perfectamente las podemos llamar librerías, en una integración **ad-hoc** y siguiendo unos pasos repetitivos en cada nuevo proyecto, en los que humanamente todos nos podemos equivocar debido a que depende de cada uno seguir cada uno de los pasos. Estas herramientas tienden a convertirse en pequeños estándares dentro de cada grupo de desarrolladores y a repetirse en futuros proyectos, pero debido a los desarrollos **ad-hoc** carecen de escalabilidad e integración con nuevas soluciones de una forma ágil, es un escollo actualizar y por otro lado replicar un estándar para la implantación de la forja, no se tiende a dejar puertas abiertas para que más adelante el herramienta mejore. Se podría definir como uno de los casos de inanición en el desarrollo

de Software o muerte por éxito.

En este punto es donde entra la famosa interoperabilidad entre las herramientas, la necesidad de interoperabilidad entre la herramientas a través de una comunicación estándar. Aquí encontramos el punto clave de las ALM Tools, la **integración de herramientas** dentro de un proceso de desarrollo.

En post de evitar la falta de replicación, además de la importancia de la interoperabilidad se ha de tener en cuenta la replicación del contenido o la gestión de la instalación de un ALM. Siempre se ha de pensar mirando hacia adelante, no es necesario implementar las mejoras pero sí, dejar un hueco para que casen bien. Un ejemplo que puede ilustrar esta frase es la programación basada en Interfaces mediante Java, ya que las Intefaces ofrecen soluciones para implementar a medida y si se actualiza la Interfaz (en este caso es el esqueleto de la clase) para añadir una nueva funcionalidad con un método, los métodos anteriores mantienen su comportamiento dentro de cada clase que la implementa y adquieren la posibilidad de aumentar la funcionalidad implementando la nueva solución, adecuada a su entorno pero nueva, de esta forma la interoperabilidad entre las clases que utilicen esta Interfaz también se mantiene.

5.2. Qué es una forja

5.3. Objetivos

5.4. Componentes

Este conjunto de herramientas se puede dividir en varios grupos:

- Gestión de Requisitos.
- Arquitectura.
- Desarrollo.

- Test.
- Issue tracking system.
- Continuous Integration.
- Release Management.

Hoy en día las forjas ALM abundan, además de gozar de una gran popularidad entre los proyectos de Software, como podemos ver en los casos de SourceForge, Googlecode y Github (más adelante discutiremos cada proyecto). En este caso ALM Tools as a Service, debido al servicio que ofrecen, pero sólo las que son FLOSS permiten replicar ese mismo entorno en tu propia máquina, un dato muy importante a tener en cuenta, porque siempre se ha de mirar hacia adelante.

5.5. Estado del arte de forjas

- ClinkerHQ `http://clinkerhq.com/` - Privado.
- Github - `http://github.com/` - Privado.
- SourceForge con Allura - `http://sourceforge.net/projects/allura/` - Software Libre pero incompleta (integrated Wiki, Tracker, SCM (svn, git and hg), Discussion, and Blog tools).
- Cloudbees DEV@Cloud `http://www.cloudbees.com/dev.cb` - Privado.
- CollabNet con CloudForge `http://www.cloudforge.com/` - Privado.
- Plan.io - `http://plan.io/en/` - Privado.
- Bitnami - `http://bitnami.com/` - Privado
- GForge
- Collab.net
- Google Code
- Bitbucket - https://bitbucket.org/mswlmanage2013/mswl-bitbucket-alm-tools/wiki/Home

5.5.1. Problemas en algunas forjas

5.5.2. Tablas comparativas

5.6. Conclusiones del estudio de forjas

Capítulo 6

SCStack

* SCStack

6.1. Arquitectura

6.1.1. Diseño e Implementación

Análisis técnico de las Herramientas utilizadas.

6.2. Provisionamiento (puppet)

6.3. Componentes

6.4. Interoperabilidad

Interoperabilidad entre las herramientas que componen la forja. API Rest.

6.5. Pruebas y Validación

Pruebas a través de virtualización de sistemas operativos mediante herramientas libres; Vagrant, kvm, puppet.

Capítulo 7

Comunidades FLOSS

El punto diferenciador en el proyecto haciendo hincapié en la interacción con las comunidades de software de cada una de las herramientas.

Capítulo 8

Desarrollo de un proyecto

Cómo llevar a cabo el desarrollo de un proyecto con SCStack. Desarrollo en paralelo de un proyecto mediante github + travis-ci vs gerrit + jenkins.

8.1. Estructura de un proyecto

8.1.1. subsection name

8.1.2. Tareas

8.1.3. Binarios

Binarios y/o fuentes empaquetados en lenguajes que no generan binarios.

8.2. Crear un proyecto

8.2.1. Proyecto en redmine

8.2.2. Repositorio git

8.2.3. Jobs en Jenkins

8.2.4. Repositorios Archiva

repositorios de archiva para binarios.

8.3. Alta usuarios

Dar de alta desarrolladores/Project Owners (usuarios de la forja).

8.4. Proceso de desarrollo basado en ramas

Proceso de desarrollo basado en ramas.

Capítulo 9

Epílogo

9.1. Conclusiones

El uso de estas herramientas y su incremento de la calidad en el desarrollo, por encima de todo siendo FLOSS debido a eso la versatilidad que otorga en el momento de unificarlas en una herramienta nueva; SidelabCode Stack.

9.2. Lecciones aprendidas

Colaboración entre distintos proyectos y comunidades, interoperabilidad entre herramientas, Forjas de desarrollo y los elementos más comunes de las mismas.

9.3. Trabajo Futuro

Impulso de la comunidad a través de los canales habituales.

Integración y gestión de nuevas herramientas comunes para los desarrolladores.

Centralización de la instalación.

Apéndice A

Apéndice 1

Bibliografía

[1] Timothy Budd. *Introducción a la programación orientada a objetos*. Addison-Wesley Iberoamericana, 1994.

[2] Mark Lutz. *Programming Python*. O'Reilly, 2001.

[3] Fredrik Lundh. *Python standard library*. O'Reilly, 2001.

[4] George Reese. *Managing and using MySQL*. O'Reilly, 2002.