Proceso de desarrollo SCRUM

Ken Schwaber
Métodos de desarrollo avanzados
131 Middlesex Turnpike Burlington, MA 01803
correo electrónico virman@aol.com Fax: (617) 272-0555

RESUMEN. *La filosofía declarada y aceptada para el desarrollo de sistemas es que el proceso de desarrollo es un enfoque bien entendido que puede ser planificado, estimado y completado con éxito. En la práctica se ha demostrado que esto es incorrecto. SCRUM asume que el proceso de desarrollo de sistemas es un proceso impredecible y complicado que sólo puede describirse a grandes rasgos como una progresión general. SCRUM define el proceso de desarrollo de sistemas como un conjunto flexible de actividades que combina herramientas y técnicas conocidas y viables con lo mejor que un equipo de desarrollo puede idear para construir sistemas. Dado que estas actividades son flexibles, se utilizan controles para gestionar el proceso y el riesgo inherente. SCRUM es una mejora del ciclo de desarrollo orientado a objetos iterativo/incremental que se utiliza habitualmente.*

PALABRAS CLAVE: *SCRUM SEI Capacidad-Madurez-Modelo Proceso Empírico*

1. Introducción

En este artículo presentamos un proceso de desarrollo, SCRUM, que trata las principales partes del desarrollo de sistemas como una caja negra controlada. Lo relacionamos con la teoría de la complejidad para mostrar por qué este enfoque aumenta la flexibilidad y produce un sistema que responde tanto a los requisitos iniciales como a los adicionales descubiertos durante el desarrollo en curso.

Se han probado numerosos enfoques para mejorar el proceso de desarrollo de sistemas. Cada uno de ellos ha sido promocionado como una "mejora significativa de la productividad". Todos han fracasado a la hora de producir mejoras espectaculares.[1] Como señaló Grady Booch, "a menudo llamamos a esta condición la crisis del software, pero francamente, un mal que se ha prolongado tanto tiempo debe llamarse normal".[2]

En este documento se aplican conceptos del control de procesos industriales al campo del desarrollo de sistemas. El control de procesos industriales define los procesos como "teóricos" (totalmente definidos) o "empíricos" (caja negra). Cuando un proceso de caja negra se trata como un proceso totalmente definido, se producen resultados imprevisibles.

[1] 1 Brooks, F.P. “No silver bullet—essence and accidents of software engineering.” Computer 20:4:10-19, April 1987.
[2] Object Oriented Analysis and Design with Applications, p. 8, Grady Booch, The Benjamin/Cummings Publishing Company, Inc., 1994

Un número importante de procesos de desarrollo de sistemas no están completamente definidos, pero se tratan como si lo estuvieran. El resultado es la imprevisibilidad sin control. El enfoque SCRUM trata estos procesos de desarrollo de sistemas como una caja negra controlada.

Takeuchi y Nonaka[3] observaron por primera vez variantes del enfoque SCRUM para el desarrollo de nuevos productos con equipos pequeños de alto rendimiento en Fuji-Xerox, Canon, Honda, NEC, Epson, Brother, 3M, Xerox y Hewlett-Packard. Coplien[4] observó que un enfoque similar aplicado al desarrollo de software en Borland era el proyecto de desarrollo de C++ de mayor productividad jamás documentado. Más recientemente, Sutherland[5] ha aplicado un enfoque perfeccionado del proceso SCRUM al desarrollo de Smalltalk y Schwaber[6] al desarrollo de Delphi.

El enfoque SCRUM se utiliza en empresas de software de vanguardia con un éxito considerable. Los analistas del sector creen que SCRUM puede ser apropiado para que otras organizaciones de desarrollo de software obtengan los beneficios esperados de las técnicas y herramientas orientadas a objetos.[7]

2. Visión general

Nuestro nuevo enfoque del desarrollo de sistemas se basa en la gestión de procesos definidos y de caja negra. Llamamos a este enfoque la metodología SCRUM (véase Takeuchi y Nonaka, 1986), por el SCRUM del rugby: una formación cerrada de delanteros que se unen en posiciones específicas cuando se convoca un scrumdown.[8]

Como se verá más adelante, SCRUM es una mejora del enfoque iterativo e incremental para la entrega de software orientado a objetos, documentado inicialmente por Pittman[9] y ampliado posteriormente por Booch.[10] Puede utilizar los mismos roles para el personal del proyecto, como los esbozados por Graham[11], por ejemplo, pero organiza y gestiona el proceso del equipo de una nueva manera.

SCRUM es una metodología de gestión, mejora y mantenimiento de un sistema existente o un prototipo de producción. Parte de la base de que el diseño y el código ya existen, lo que

---

[3] Takeuchi, Hirotaka and Nonaka, Ikujiro. January-February 1986. “The New New Product Development Game.” Harvard Business Review

[4] Coplien, J. “Borland Software Craftsmanship: A New Look at Process, Quality and Productivity.” Proceedings of the 5th Annual Borland International Conference, June 5, 1994. Orlando, Florida.

[5] Sutherland, Jeff. ScrumWeb Home Page: A Guide to the SCRUM Development Process. Jeff Sutherland’s Object Technology Web Page, 1996

[6] Schwaber, Ken. “Controlled Chaos: Living on the Edge.” American Programmer, April 1996.

[7] Aberdeen Group. Upgrading To ISV Methodology For Enterprise Application Development. Product Viewpoint 8:17, December 7, 1995.

[8] Gartner, Lisa. The Rookie Primer. Radcliffe Rugby Football Club, 1996 <http://vail.al.arizona.edu/rugby/rad/rookie_primer.html>

[9] Pittman, Matthew. Lessons Learned in Managing Object-Oriented Development. IEEE Software, January, 1993, pp. 43-53.

[10] Booch, Grady. Object Solutions: Managing the Object-Oriented Project. Addison-Wesley, 1995.

[11] Graham, Ian. Migrating to Object Technology. Addison-Wesley, 1994.

prácticamente siempre ocurre en el desarrollo orientado a objetos debido a la presencia de bibliotecas de clases. SCRUM abordará los esfuerzos de desarrollo de sistemas heredados, totalmente nuevos o rediseñados más adelante.

Los lanzamientos de productos de software se planifican en función de las siguientes variables:

- Requisitos del cliente: cómo debe mejorarse el sistema actual.
- Presión de tiempo: qué plazo se necesita para obtener una ventaja competitiva.
- Competencia: qué hace la competencia y qué se necesita para superarla.
- Calidad: cuál es la calidad requerida, dadas las variables anteriores.
- Visión: qué cambios son necesarios en esta fase para cumplir la visión del sistema.
- Recursos: de qué personal y financiación se dispone.

Estas variables conforman el plan inicial de un proyecto de mejora de software. Sin embargo, estas variables también cambian durante el proyecto. Una metodología de desarrollo exitosa debe tener en cuenta estas variables y su naturaleza evolutiva.

3. Situación actual del desarrollo

Los sistemas se desarrollan en un entorno muy complejo. La complejidad está tanto en el entorno de desarrollo como en el entorno de destino. Por ejemplo, cuando se inició el desarrollo del sistema de control del tráfico aéreo, no había que tener en cuenta los sistemas cliente-servidor de tres niveles ni la desregulación de las aerolíneas. Sin embargo, estos cambios ambientales y técnicos se produjeron durante el proyecto y se tuvieron que tomar en cuenta dentro del sistema que se estaba construyendo.

Entre las variables del entorno se encuentran:

- Disponibilidad de profesionales cualificados: cuanto más nueva sea la tecnología, las herramientas, los métodos y el dominio, menor será el grupo de profesionales cualificados.
- Estabilidad de la tecnología de implementación: cuanto más nueva sea la tecnología, menor será la estabilidad y mayor la necesidad de equilibrar la tecnología con otras tecnologías y procedimientos manuales.
- Estabilidad y potencia de las herramientas: cuanto más nueva y potente sea la herramienta de desarrollo, menor será el grupo de profesionales cualificados y más inestable será la funcionalidad de la herramienta.
- Eficacia de los métodos: qué métodos de modelado, pruebas, control de versiones y diseño se van a utilizar, y cuán eficaces, eficientes y probados son.
- Experiencia en el dominio: ¿hay profesionales cualificados disponibles en los distintos dominios, incluidos el empresarial y el tecnológico?
- Nuevas funciones: qué funciones totalmente nuevas se van a añadir y hasta qué punto se ajustarán a la funcionalidad actual.

- Metodología: ¿el enfoque general del desarrollo de sistemas y la utilización de los métodos seleccionados promueve la flexibilidad, o se trata de un enfoque rígido y detallado que restringe la flexibilidad?
- Competencia: ¿qué hará la competencia durante el proyecto? ¿Qué nuevas funcionalidades se anunciarán o lanzarán?
- Tiempo/Financiamiento: ¿de cuánto tiempo se dispone inicialmente y a medida que avanza el proyecto? Cuántos fondos de desarrollo hay disponibles.
- Otras variables: cualquier otro factor al que haya que responder durante el proyecto para garantizar el éxito del sistema resultante y entregado, como las reorganizaciones.

La complejidad global es una función de estas variables:

complejidad = f(variables del entorno de desarrollo + variables del entorno de destino)

donde estas variables pueden cambiar, y de hecho lo hacen, durante el transcurso del proyecto.

A medida que aumenta la complejidad del proyecto, mayor es la necesidad de controles, en particular la evaluación continua y la respuesta al riesgo.

Los intentos de modelar este proceso de desarrollo han encontrado los siguientes problemas:

- Muchos de los procesos de desarrollo no están controlados. Las entradas y salidas se desconocen o están poco definidas, el proceso de transformación carece de la precisión necesaria y el control de calidad no está definido. Los procesos de testing son un ejemplo.
- Hay un número indeterminado de procesos de desarrollo que sirven de puente entre procesos conocidos pero no controlados. Los procesos detallados para garantizar que un modelo lógico tiene el contenido adecuado para dar lugar a un modelo físico satisfactorio son uno de esos procesos.
- Las entradas del entorno (requisitos) sólo pueden tenerse en cuenta al principio del proceso. A partir de ese momento se requieren complejos procedimientos de gestión del cambio.

Los intentos de imponer un modelo metodológico micro, o detallado, al proceso de desarrollo no han funcionado porque el proceso de desarrollo aún no está completamente definido. Actuar como si el proceso de desarrollo estuviera definido y fuera predecible hace que no estemos preparados para los resultados impredecibles.

A pesar de que el proceso de desarrollo está definido de manera incompleta y es dinámico, numerosas organizaciones han desarrollado metodologías de desarrollo detalladas que incluyen los métodos de desarrollo actuales (estructurados, OO, etc.). La metodología Cascada fue uno de los primeros procesos de desarrollo de sistemas definidos. En la figura 1 se muestra una imagen de la metodología Cascada.

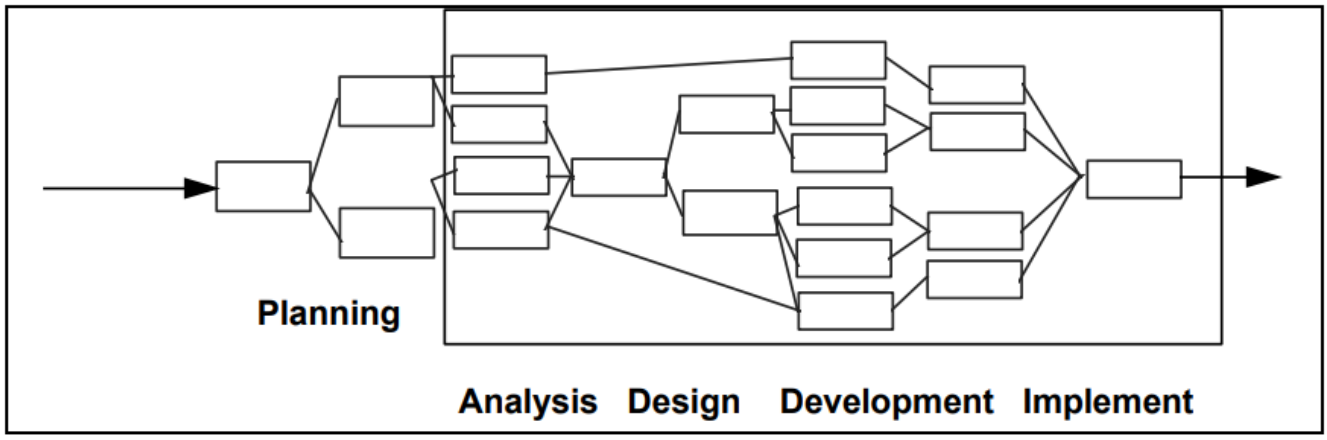

Figura 1: Metodología Cascada

Aunque el enfoque en cascada impone el uso de procesos no definidos, su naturaleza lineal ha sido su mayor problema. El proceso no define cómo responder a los resultados inesperados de cualquiera de los procesos intermedios.

Barry Boehm[12] introdujo la metodología Espiral para resolver este problema. Cada una de las fases en Cascada termina con una actividad de evaluación de riesgos y creación de prototipos. La metodología Espiral se muestra en la figura 2.

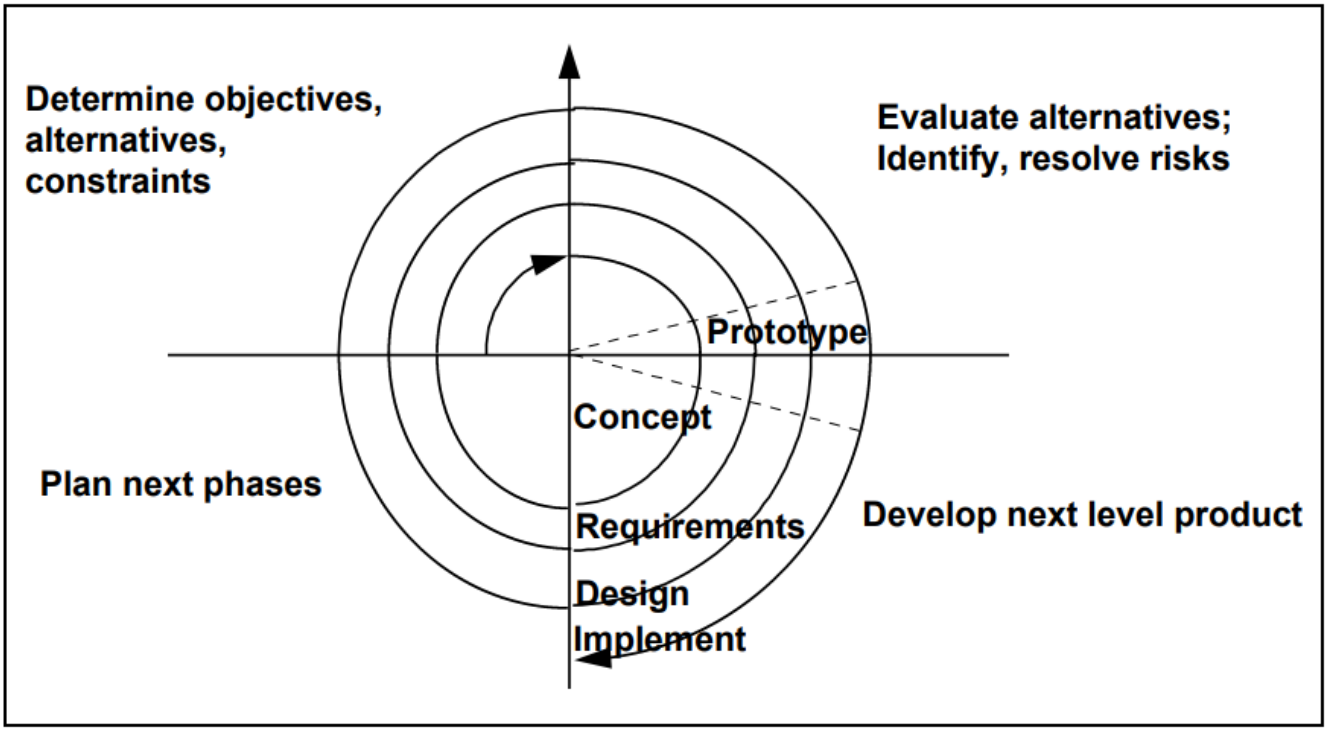

Figura 2: Metodología Espiral

[12] Boehm, B.W. 1985. "A Spiral Model of Software Development and Enhancement," from Proceedings of an International Workshop on Software Process and Software Environments, Coto de Caza, Trabuco Canyon, California, March 27-29, 1985.

La metodología Espiral "pela la cebolla", avanzando a través de las "capas" del proceso de desarrollo. Un prototipo permite a los usuarios determinar si el proyecto va por buen camino, si debe devolverse a fases anteriores o si debe terminarse. Sin embargo, las fases y los procesos de las fases siguen siendo lineales. El trabajo de requisitos se sigue realizando en la fase de requisitos, el trabajo de diseño en la fase de diseño, y así sucesivamente, con cada una de las fases consistente en procesos lineales y explícitamente definidos.

La metodología Iterativa mejora la metodología Espiral. Cada iteración consiste en todas las fases estándar de Cascada, pero cada iteración sólo aborda un conjunto de funcionalidades analizadas. El resultado global del proyecto se ha dividido en subsistemas priorizados, cada uno con interfaces limpias. Con este enfoque, se puede probar la viabilidad de un subsistema y una tecnología en las iteraciones iniciales. Las iteraciones posteriores pueden añadir recursos al proyecto y aumentar la velocidad de entrega. Este enfoque mejora el control de costes, garantiza la entrega de sistemas (aunque sean subsistemas) y mejora la flexibilidad general. Sin embargo, el enfoque iterativo sigue esperando que los procesos de desarrollo subyacentes estén definidos y sean lineales. Véase la figura 3.

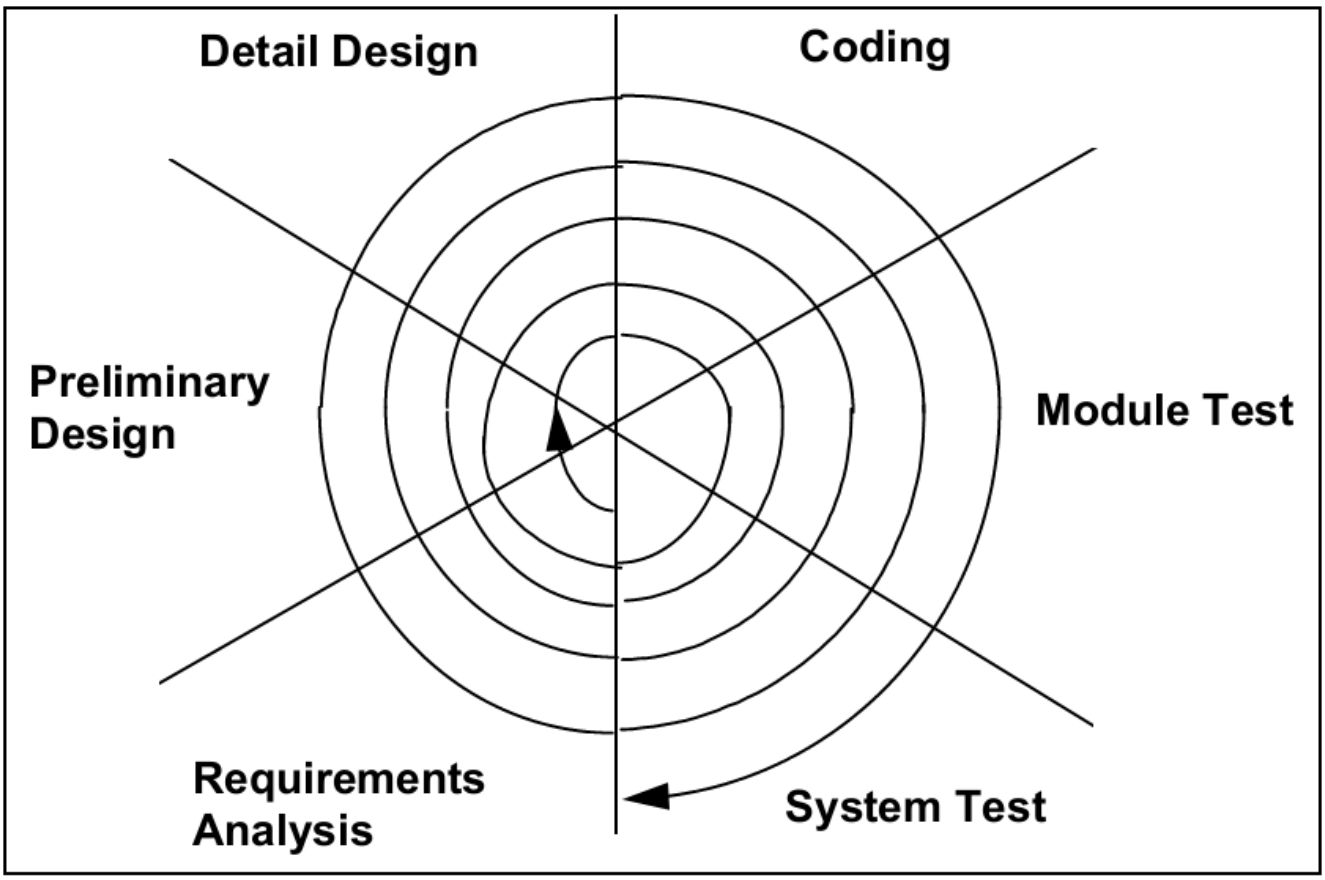

Figura 3: Metodología Iterativa

Dada la complejidad del entorno y la mayor dependencia de los nuevos sistemas "de última generación", el riesgo que soportan los proyectos de desarrollo de sistemas ha aumentado y se ha intensificado la búsqueda de mecanismos para gestionar este riesgo.

Se puede argumentar que las metodologías actuales son mejores que no tener nada. Cada una mejora a la otra. Los enfoques Espiral e Iterativo implantan mecanismos formales de control del riesgo para hacer frente a resultados imprevisibles. Se proporciona un marco de desarrollo.

Sin embargo, cada uno se basa en la falacia de que los procesos de desarrollo son procesos definidos y predecibles. Pero los resultados imprevisibles se producen a lo largo de los proyectos. El rigor implícito en los procesos de desarrollo ahoga la flexibilidad necesaria para hacer frente a los resultados imprevisibles y responder a un entorno complejo.

A pesar de su amplia presencia en la comunidad de desarrollo, nuestra experiencia en el sector demuestra que la gente no utiliza las metodologías salvo como un macro mapa de procesos, o por sus descripciones detalladas de métodos.

El siguiente gráfico muestra el entorno de desarrollo actual, utilizando cualquiera de los procesos de Cascada, Espiral o Iterativo. A medida que la complejidad de las variables aumenta, incluso a un nivel moderado, la probabilidad de un proyecto "exitoso" disminuye rápidamente (un proyecto exitoso se define como un sistema que es útil cuando se entrega). Véase la figura 4.

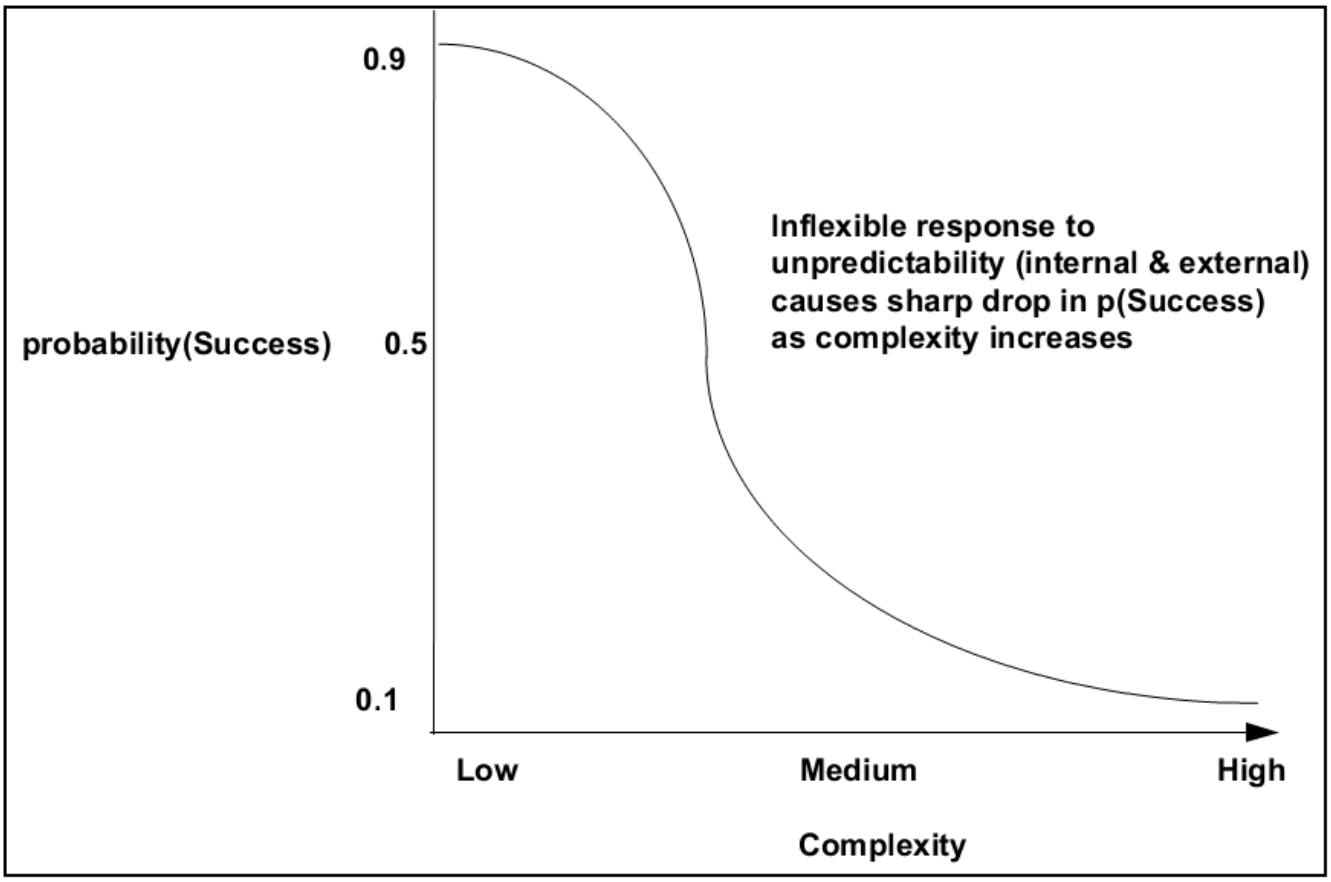

Figura 4: Riesgo/complejidad del proceso definido

4. Metodología SCRUM

El proceso de desarrollo del sistema es complicado y complejo. Por lo tanto, se requiere la máxima flexibilidad y un control adecuado. La evolución favorece a los que operan con la máxima exposición al cambio del entorno y han optimizado la adaptación flexible al cambio. La evolución deselecciona a quienes se han aislado del cambio ambiental y han minimizado el caos y la complejidad de su entorno.

Se necesita un enfoque que permita a los equipos de desarrollo operar de forma adaptativa dentro de un entorno complejo utilizando procesos imprecisos. El desarrollo de sistemas complejos se produce en circunstancias que cambian rápidamente. Producir sistemas ordenados en circunstancias caóticas requiere la máxima flexibilidad. Cuanto más se acerque el equipo de desarrollo al borde del caos, sin dejar de mantener el orden, más competitivo y útil será el sistema resultante. Langton ha modelado este efecto en simulaciones por ordenador[13] y su trabajo lo ha convertido en un teorema fundamental de la teoría de la complejidad.
La metodología puede ser el factor más importante para determinar la probabilidad de éxito. Las metodologías que fomentan y apoyan la flexibilidad tienen un alto grado de tolerancia a los cambios en otras variables. Con estas metodologías, el proceso de desarrollo se considera imprevisible desde el principio, y se establecen mecanismos de control para gestionar la imprevisibilidad.

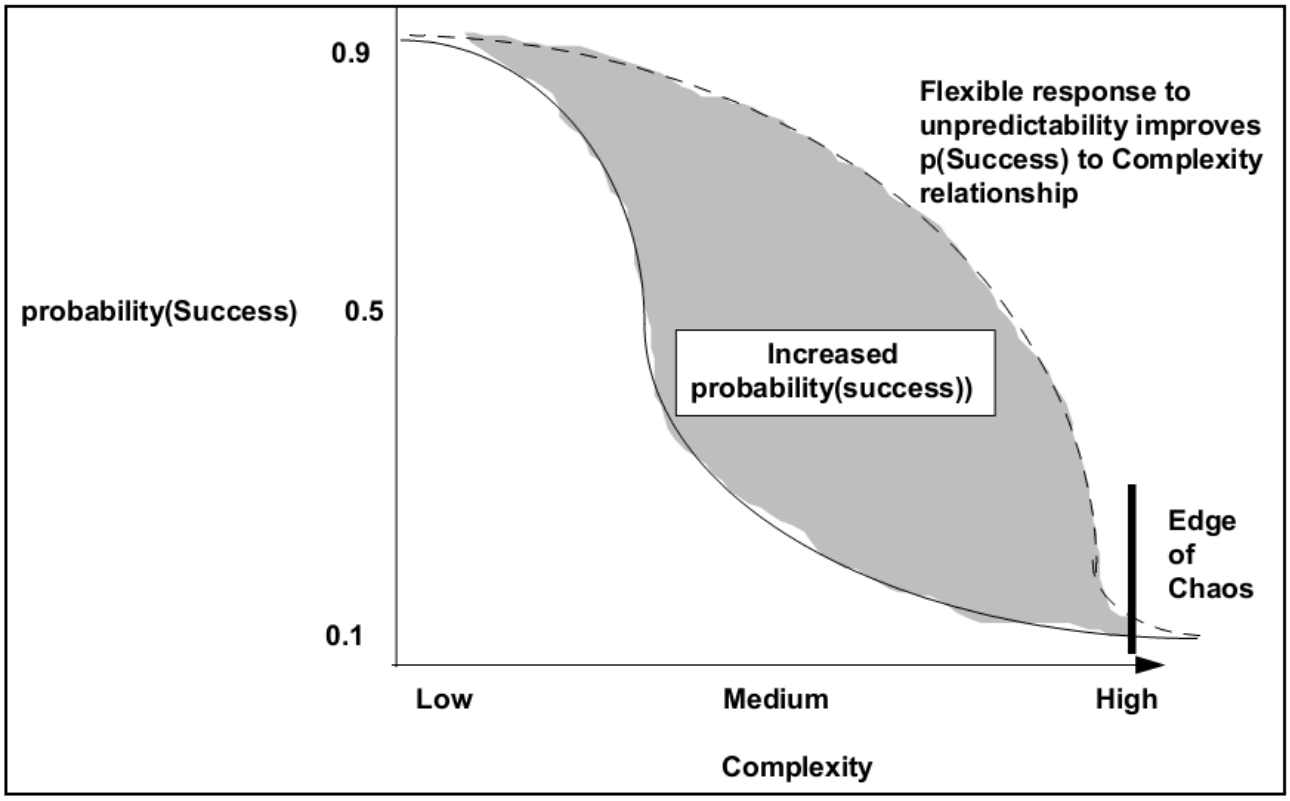

Figura 5: Gráfico de comparación de riesgo/complejidad

[13] Langton, Christopher. Artificial Life. In Artificial Life , Volume VI: SFI Studies in the Sciences of Complexity (Ed. C. Langton) Addison-Wesley, 1988.

Si graficamos la relación entre la complejidad del entorno y la probabilidad de éxito con una metodología flexible que incorpora controles y gestión de riesgos, la tolerancia al cambio es más duradera. Véase la figura 5.

Las figuras 4 y 5 reflejan las experiencias de desarrollo de software de ADM, Easel, VMARK, Borland y prácticamente todos los demás desarrolladores de software "empaquetado". Estas organizaciones han asumido el riesgo y la complejidad del entorno durante los proyectos de desarrollo. Se experimentó un mayor impacto del producto, proyectos exitosos y ganancias de productividad. Se construye el mejor software posible.

Las metodologías en Cascada y Espiral establecen el contexto y la definición de los entregables al inicio de un proyecto. Las metodologías SCRUM e Iterativas planifican inicialmente el contexto y la definición general de los entregables, y luego los hacen evolucionar durante el proyecto en función del entorno. SCRUM reconoce que los procesos de desarrollo subyacentes están definidos de forma incompleta y utiliza mecanismos de control para mejorar la flexibilidad.

La principal diferencia entre el enfoque definido (Cascada, Espiral e Iterativo) y el empírico (SCRUM) es que el enfoque SCRUM asume que los procesos de análisis, diseño y desarrollo en la fase Sprint son imprevisibles. Se utiliza un mecanismo de control para gestionar la imprevisibilidad y controlar el riesgo. Los resultados son la flexibilidad, la capacidad de respuesta y la fiabilidad. Véase la figura 6.

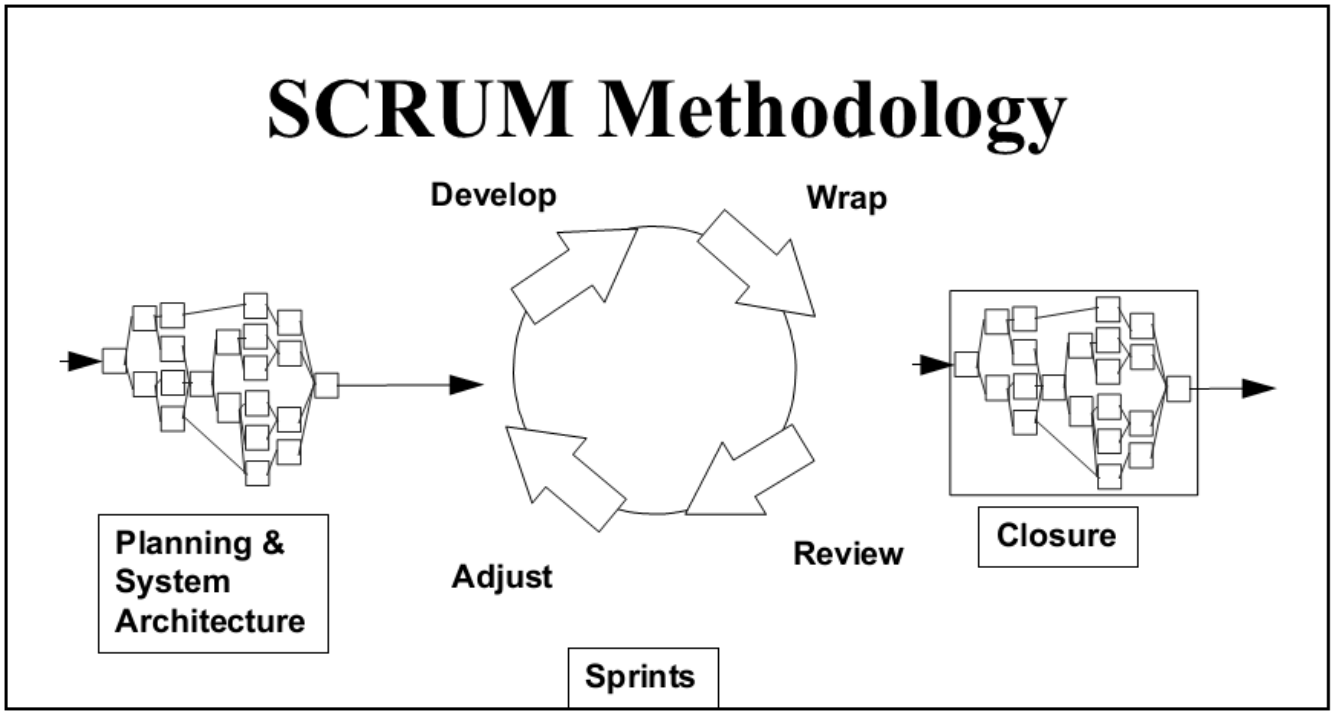

Figura 6: Metodología SCRUM

Las características de la metodología SCRUM son:

- La primera y la última fase (Planificación y Cierre) consisten en procesos definidos, donde todos los procesos, entradas y salidas están bien definidos. El conocimiento de cómo realizar estos procesos es explícito. El flujo es lineal, con algunas iteraciones en la fase de planificación.
- La fase de Sprint es un proceso empírico. Muchos de los procesos de la fase de Sprint no están identificados o no están controlados. Se trata como una caja negra que requiere controles externos. En consecuencia, los controles, incluida la gestión de riesgos, se ponen en cada iteración de la fase de Sprint para evitar el caos y maximizar la flexibilidad.
- Los sprints son no-lineales y flexibles. Cuando se dispone de ellos, se utiliza el conocimiento explícito del proceso; de lo contrario, se recurre al conocimiento tácito y a la prueba y error para construir el conocimiento del proceso. Los sprints se utilizan para hacer evolucionar el producto final.
- El proyecto está abierto al entorno hasta la fase de Cierre. El producto final puede modificarse en cualquier momento durante las fases de Planificación y Sprint del proyecto. El proyecto permanece abierto a la complejidad del entorno, incluidas las presiones competitivas, de tiempo, de calidad y financieras, a lo largo de estas fases.
- El entregable se determina durante el proyecto en función del entorno.

La figura 7 compara las principales características de SCRUM con las de otras metodologías.

	Waterfall	Spiral	Iterative	SCRUM
Defined processes	Required	Required	Required	Planning & Closure only
Final product	Determined during planning	Determined during planning	Set during project	Set during project
Project cost	Determined during planning	Partially variable	Set during project	Set during project
Completion date	Determined during planning	Partially variable	Set during project	Set during project
Responsiveness to environment	Planning only	Planning primarily	At end of each iteration	**Throughout**
Team flexibility, creativity	Limited - cookbook approach	Limited - cookbook approach	Limited - cookbook approach	**Unlimited during iterations**
Knowledge transfer	Training prior to project	Training prior to project	Training prior to project	**Teamwork during project**
Probability of success	Low	Medium low	Medium	**High**

Figura 7: Comparación de las metodologías

4.1 Fases de SCRUM

SCRUM tiene los siguientes grupos de fases:

4.1.1. Previo al juego

- Planificación: Definición de un nuevo lanzamiento basado en el backlog conocido actualmente, junto con una estimación de su calendario y coste. Si se está desarrollando un nuevo sistema, esta fase consiste en la conceptualización y el análisis. Si se trata de la mejora de un sistema existente, esta fase consiste en un análisis limitado.
- Arquitectura: Diseñar cómo se implementarán los ítems del backlog. Esta fase incluye la modificación de la arquitectura del sistema y el diseño de alto nivel.

4.1.2. El juego

- Sprints de desarrollo: Desarrollo de la funcionalidad del nuevo lanzamiento, con respeto constante a las variables de tiempo, requisitos, calidad, coste y competencia. La interacción con estas variables define el final de esta fase. Hay múltiples sprints de desarrollo iterativos, o ciclos, que se utilizan para evolucionar el sistema.

4.1.3. Después del juego

- Cierre: Preparación para el lanzamiento, incluyendo la documentación final, las pruebas por etapas previas al lanzamiento y el lanzamiento.

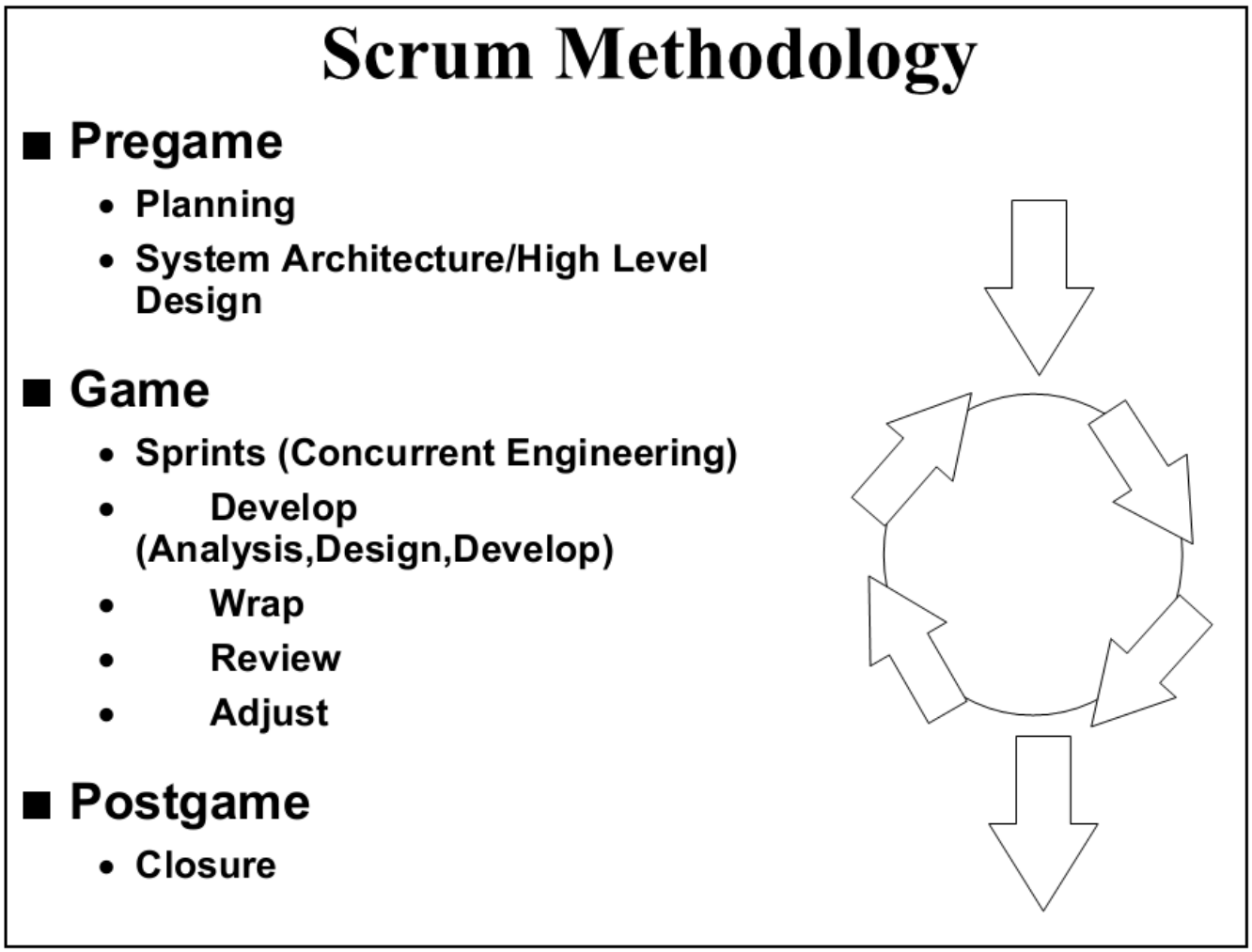

Figura 8: Fases de SCRUM

4.2 Pasos de las fases

Cada una de las fases tiene los siguientes pasos:

4.2.1. Planificación

- Elaboración de un exhaustivo backlog.
- Definición de la fecha de entrega y de la funcionalidad de una o más lanzamientos.
- Selección del lanzamiento más adecuado para el desarrollo inmediato.
- Mapeo de paquetes de productos (objetos) para los ítems del backlog en la versión seleccionada.
- Definición del equipo de proyecto para la creación del nuevo lanzamiento.
- Evaluación del riesgo y de los controles de riesgo adecuados.
- Revisión y posible ajuste de los ítems y paquetes del backlog.
- Validación o re-selección de las herramientas e infraestructuras de desarrollo.
- Estimación del coste de lanzamiento, incluido el desarrollo, el material colateral, el marketing, la formación y la puesta en marcha.
- Verificación de la aprobación y el financiamiento del management.

4.2.2. Arquitectura/Diseño de alto nivel

- Revisar los ítems asignados del backlog.
- Identificar los cambios necesarios para implementar los ítems del backlog.
- Realizar un análisis de dominio en la medida necesaria para construir, mejorar o actualizar los modelos de dominio para reflejar el nuevo contexto y los requisitos del sistema.
- Perfeccionar la arquitectura del sistema para que sea compatible con el nuevo contexto y los nuevos requisitos.
- Identificar cualquier problema o issues en el desarrollo o en la implementación de los cambios.
- Reunión de revisión del diseño, cada equipo presenta el enfoque y los cambios para implementar cada ítem del backlog. Reasignar los cambios según sea necesario.

4.2.3. Desarrollo (Sprint)

La fase de Desarrollo es un ciclo iterativo de trabajo de desarrollo. El management determina que se cumple el tiempo, la competencia, la calidad o la funcionalidad, se completan las iteraciones y se produce la fase de Cierre. Este enfoque también se conoce como Ingeniería Concurrente. El desarrollo consta de los siguientes macroprocesos:

- Reunión con los equipos para revisar los planes de lanzamiento.
- Distribución, revisión y ajuste de las normas a las que se ajustará el producto.
- Sprints iterativos, hasta que el producto se considera listo para su distribución.

Un Sprint es un conjunto de actividades de desarrollo realizadas durante un periodo predefinido, normalmente de una a cuatro semanas. El intervalo se basa en la complejidad del producto, la evaluación del riesgo y el grado de supervisión deseado. La velocidad y la intensidad del Sprint dependen de la duración seleccionada del Sprint. El riesgo se evalúa continuamente y se ponen

en marcha controles de riesgo y respuestas adecuadas. y respuestas adecuadas al riesgo. Cada Sprint consta de uno o más equipos que realizan lo siguiente:

- Desarrollar: definir los cambios necesarios para la implementación de los requisitos del backlog en paquetes, abrir los paquetes, realizar análisis de dominio, diseñar, desarrollar, implementar, probar y documentar los cambios. El desarrollo consiste en el microproceso de descubrimiento, invención e implementación.
- Envoltura: cerrar los paquetes, crear una versión ejecutable de los cambios y cómo se implementan los requisitos del backlog.
- Revisión: Todos los equipos se reúnen para presentar el trabajo y revisar el progreso, planteando y resolviendo cuestiones y problemas, agregando nuevos ítems del backlog. Se revisa el riesgo y se definen las respuestas adecuadas.
- Ajustar: consolidar la información recopilada de la reunión de revisión en los paquetes afectados, que incluyen un aspecto diferente y nuevas propiedades.

Cada Sprint va seguido de una revisión, cuyas características son :

- Todo el equipo y el management del producto están presentes y participan.
- La revisión puede incluir clientes, ventas, marketing y otros.
- La revisión cubre los sistemas ejecutables funcionales que abarcan los objetos asignados a ese equipo e incluyen los cambios realizados para implementar los ítems del backlog.
- La forma en que se implementan los ítems del backlog mediante cambios que pueden cambiar según la revisión.
- Se pueden introducir y asignar nuevos ítems del backlog a los equipos como parte de la revisión, cambiando el contenido y la dirección de los entregables.
- El momento de la próxima revisión se determina en función del progreso y la complejidad. Los Sprints suelen tener una duración de 1 a 4 semanas.

4.2.4. Cierre

Cuando el equipo de gestión considera que las variables de tiempo, competencia, requisitos coste y calidad coinciden para que se produzca un nuevo lanzamiento, declaran el lanzamiento "cerrado" y entran en esta fase. En esta fase se prepara el producto desarrollado para su lanzamiento general.

La integración, la prueba del sistema, la documentación del usuario, la preparación del material de capacitación y la preparación del material de marketing se encuentran entre las tareas de cierre.

4.3. Controles SCRUM

Operar al borde del caos (imprevisibilidad y complejidad) requiere controles de gestión para evitar caer en el caos. La metodología SCRUM incorpora estos controles generales y sueltos, utilizando técnicas OO para la construcción real de los entregables.

El riesgo es el principal control. La evaluación del riesgo conduce a cambios en otros controles y respuestas por parte del equipo.

Los controles de la metodología SCRUM son:

- Backlog: Requisitos de funcionalidad del producto que no se abordan adecuadamente en el lanzamiento actual del producto. Los errores, los defectos, las mejoras solicitadas por los clientes, las funcionalidades de los productos de la competencia, las funcionalidades de la competencia y las actualizaciones tecnológicas son ítems del backlog.
- Lanzamiento/mejoras: ítems del backlog que en un momento dado representan un lanzamiento viable basado en las variables de requisitos, tiempo, calidad y competencia.
- Paquetes: Componentes u objetos del producto que deben cambiarse para implementar un ítem del backlog en una nueva versión.
- Cambios: Cambios que deben producirse en un paquete para implementar un ítem del backlog.
- Problemas: Problemas técnicos que ocurren y deben ser resueltos para implementar un cambio.
- Riesgos: los riesgos que afectan al éxito del proyecto se evalúan continuamente y se planifican respuestas. Otros controles se ven afectados como resultado de la evaluación de los riesgos.
- Soluciones: soluciones a los problemas y riesgos, que a menudo dan lugar a cambios.
- Issues: Cuestiones generales del proyecto y del proyecto que no están definidas en términos de paquetes, cambios y problemas.

Estos controles se utilizan en las distintas fases de SCRUM. El management utiliza estos controles para gestionar el backlog. Los equipos utilizan estos controles para gestionar los cambios y los problemas. Tanto el management como los equipos gestionan conjuntamente los problemas, los riesgos y las soluciones. Estos controles se revisan, modifican y reconcilian en cada reunión de revisión del Sprint.

4.4 Resultados de SCRUM

El producto entregado es flexible. Su contenido viene determinado por variables del entorno, como el tiempo, la competencia, el coste o la funcionalidad. Los determinantes de los entregables son la inteligencia de mercado, el contacto con el cliente y la habilidad de los desarrolladores. Durante el proyecto se realizan frecuentes ajustes del contenido del producto final en respuesta al entorno. El entregable puede determinarse en cualquier momento del proyecto.

4.5 Equipo de proyecto SCRUM

El equipo que trabaja en la nueva versión incluye a los desarrolladores a tiempo completo y a las partes externas que se verán afectadas por la nueva versión, como marketing, ventas y clientes. En los procesos tradicionales de lanzamiento, estos últimos grupos se mantienen alejados de los equipos de desarrollo por miedo a complicar demasiado el proceso y a provocar interferencias

"innecesarias". El enfoque SCRUM, sin embargo, acoge y facilita su participación controlada a intervalos determinados, ya que esto aumenta la probabilidad de que el contenido y el calendario de la publicación sean adecuados, útiles y comercializables.

Para cada nueva versión se forman los siguientes equipos:

- Management: Liderado por el Product Manager, define el contenido inicial y el calendario del lanzamiento, y luego gestiona su evolución a medida que el proyecto avanza y las variables cambian. El management se ocupa del backlog, los riesgos y el contenido de la versión.
- Development teams: Los equipos de desarrollo son pequeños, y cada uno de ellos contiene desarrolladores, documentadores y personal de control de calidad. Se utilizan uno o varios equipos de entre tres y seis personas cada uno. A cada uno se le asigna un conjunto de paquetes (u objetos), incluyendo todos los ítems del backlog relacionados con cada paquete. El equipo define los cambios necesarios para implementar los ítems del backlog en los paquetes y gestiona todos los problemas relacionados con los cambios. Los equipos pueden ser funcionales (se les asignan los paquetes que abordan conjuntos específicos de la funcionalidad del producto) o del sistema (se les asignan capas únicas del sistema). Los miembros de cada equipo se seleccionan en función de sus conocimientos y experiencia en relación con los conjuntos de paquetes, o experiencia en el dominio.

4.6 Características de SCRUM

La metodología SCRUM es una metáfora del juego del rugby. El rugby evolucionó a partir del fútbol inglés (soccer) bajo la intensa presión del juego:

> El estudiante de rugby William Webb Ellis, de 17 años, inaugura un nuevo juego cuyas reglas serán codificadas en 1839. Jugando al fútbol para el colegio de 256 años de antigüedad en East Warwickshire, Ellis ve que el reloj se está acabando con su equipo en desventaja, así que coge el balón y corre con él desafiando las reglas.
>
> **The People's Chronology**, Henry Holt and Company, Inc. Copyright © 1992.

Los proyectos SCRUM tienen las siguientes características:

- Entregables flexibles: el contenido de los entregables viene dictado por el entorno.
- Calendario flexible: el producto final puede ser necesario antes o después de lo previsto inicialmente.
- Equipos pequeños: cada equipo no tiene más de 6 miembros. Puede haber varios equipos dentro de un proyecto.

- Revisiones frecuentes: el progreso del equipo se revisa con la frecuencia que dicta la complejidad del entorno y el riesgo (normalmente ciclos de 1 a 4 semanas). Cada equipo debe preparar un ejecutable funcional para cada revisión.
- Colaboración: se espera que haya colaboración interna y externa durante el proyecto.
- Orientado a objetos: cada equipo abordará un conjunto de objetos relacionados, con interfaces y comportamientos claros.

La metodología SCRUM comparte muchas características con el deporte del Rugby:

- El contexto está establecido por el campo de juego (entorno) y las reglas del rugby (controles).
- El ciclo principal es mover el balón hacia adelante.
- El rugby evolucionó a partir de la ruptura de las reglas del fútbol, adaptándose al entorno.
- El juego no termina hasta que el entorno lo dicta (necesidad del negocio, competencia, funcionalidad, calendario).

5. Ventajas de la metodología SCRUM

Las metodologías de desarrollo tradicionales están diseñadas únicamente para responder a la imprevisibilidad de los entornos externos y de desarrollo al inicio de un ciclo de mejora. Los enfoques más recientes, como la metodología Espiral de Boehm y sus variantes, siguen siendo limitados en su capacidad de responder a los cambios de requisitos una vez que el proyecto ha comenzado.

La metodología SCRUM, en cambio, está diseñada para ser bastante flexible en todo momento. Ofrece mecanismos de control para planificar el lanzamiento de un producto y luego gestionar las variables a medida que avanza el proyecto. Esto permite a las organizaciones cambiar el proyecto y los entregables en cualquier momento, y ofrecer la versión más adecuada.

La metodología SCRUM da libertad a los desarrolladores para idear las soluciones más ingeniosas a lo largo del proyecto, a medida que se produce el aprendizaje y cambia el entorno.

Los equipos pequeños de desarrolladores que colaboran entre sí son capaces de compartir sus conocimientos tácitos sobre los procesos de desarrollo. Se proporciona un excelente entorno de formación para todas las partes.

La tecnología orientada a objetos es la base de la metodología SCRUM. Los objetos, o características del producto, ofrecen un entorno discreto y manejable. El código procedimental, con sus muchas y entrelazadas interfaces, es inapropiado para la metodología SCRUM. SCRUM puede aplicarse selectivamente a sistemas procedimentales con interfaces limpias y estar fuertemente orientado a los datos.

6. Estimación de proyectos SCRUM

Los proyectos SCRUM pueden estimarse utilizando la estimación estándar de puntos de función. Sin embargo, es aconsejable estimar la productividad en aproximadamente el doble de la métrica actual. Sin embargo, esta estimación sólo sirve para empezar, ya que el calendario y el coste globales se determinan de forma dinámica en respuesta a los factores del entorno.

Nuestras observaciones nos han llevado a concluir que los proyectos SCRUM tienen tanto velocidad como aceleración. En cuanto a las funciones entregadas, o los ítems del backlog completados:

- la velocidad y la aceleración iniciales son bajas, ya que la infraestructura se construye/modifica
- a medida que la funcionalidad básica se incorpora a los objetos, la aceleración aumenta
- la aceleración disminuye y la velocidad se mantiene alta

Es necesario seguir desarrollando métricas para los procesos empíricos.

7. Metodologías de desarrollo de sistemas: Definidas o empíricas

El desarrollo de sistemas es el acto de crear una construcción lógica que se implementa como lógica y datos en los ordenadores. La construcción lógica consta de entradas, procesos y salidas, tanto macro (construcción completa) como micro (pasos intermedios dentro de la construcción completa). El conjunto se conoce como sistema implementado.

Durante la construcción del sistema se crean muchos artefactos. Los artefactos pueden utilizarse para guiar el pensamiento, comprobar la integridad y crear una pista de auditoría. Los artefactos consisten en documentos, modelos, programas, casos de prueba y otros entregables creados antes de crear el sistema implementado. Cuando está disponible, un *metamodelo* define el contenido semántico de los artefactos del modelo. La *notación* describe las convenciones de graficación y documentación que se utilizan para construir los modelos.

El enfoque utilizado para desarrollar un sistema se conoce como *método*. Un *método* describe las actividades implicadas en la definición, construcción e implementación de un sistema; un *método* es un marco de trabajo. Dado que un *método* es un proceso lógico para la construcción de sistemas (proceso), se conoce como *metaproceso* (un proceso para modelar procesos).

Un *método* tiene microcomponentes y macrocomponentes. Los macrocomponentes definen el flujo global y el marco temporal para realizar el trabajo. Los microcomponentes incluyen *reglas generales de diseño*, *patrones* y *reglas generales*. Las *reglas generales de diseño* indican las propiedades que deben alcanzarse o evitarse en el diseño o los enfoques generales que deben adoptarse al construir un sistema. construir un sistema. Los *patrones* son soluciones que pueden aplicarse a un tipo de actividad de desarrollo; son soluciones que esperan los problemas que se

producen durante una actividad en un método. Las *reglas general*es consisten en un conjunto de sugerencias y consejos.

La figura 9 visualiza la relación entre el Método, los Artefactos y el Sistema.

Aplicando los conceptos del control de procesos industriales al campo del desarrollo de sistemas, los *métodos* pueden clasificarse como "teóricos" (totalmente definidos) o "empíricos" (caja negra).

Es fundamental categorizar correctamente los métodos de desarrollo de sistemas. La estructura apropiada de un *método* para construir un tipo particular de sistema depende de si el método es teórico o empírico.

Los modelos de *procesos teóricos* se derivan de los *primeros principios*, utilizando los balances de materia y energía y las leyes fundamentales para determinar el modelo. Para que un *método* de desarrollo de sistemas se considere teórico, debe ajustarse a esta definición.

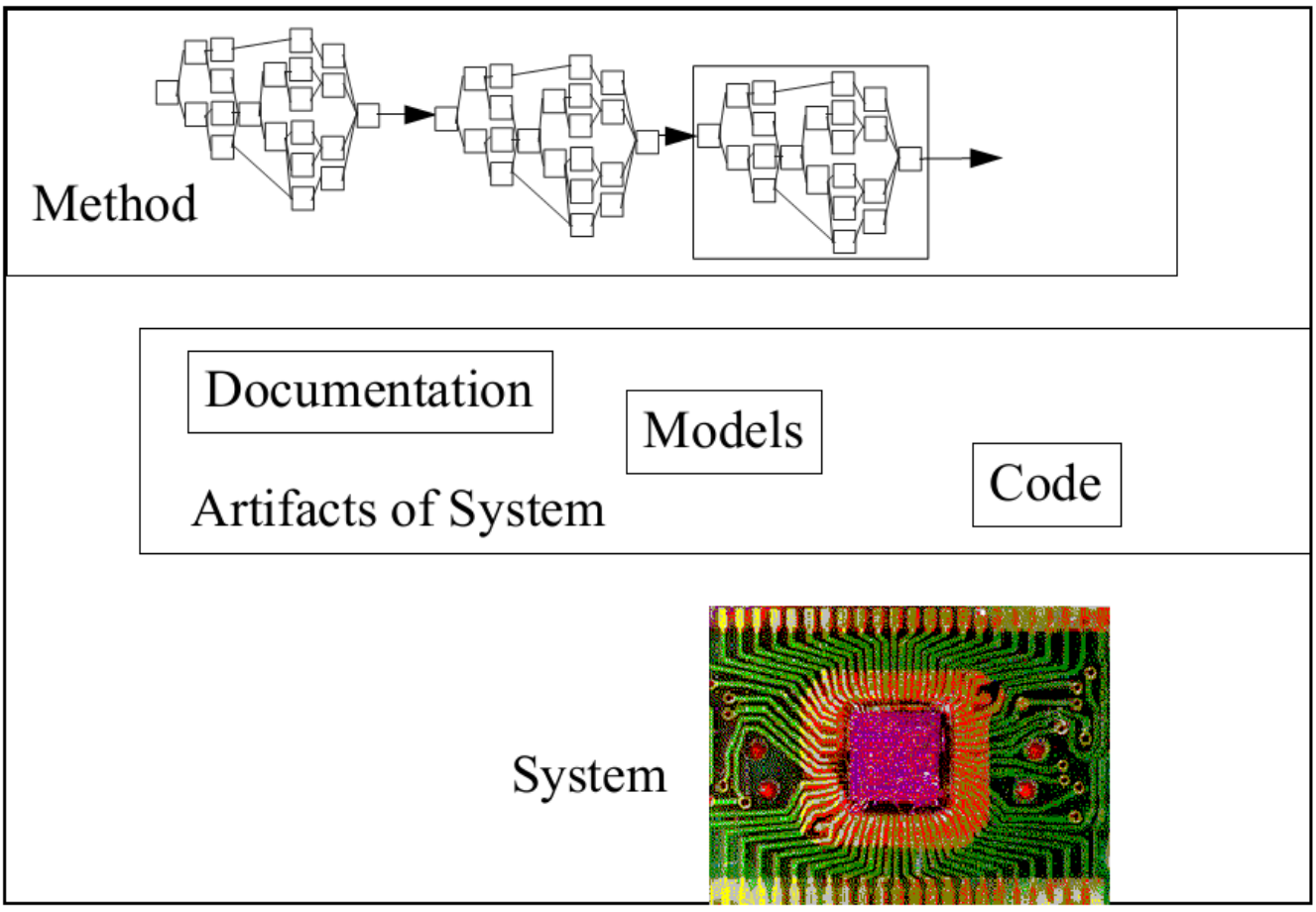

Figura 9: Método, Artefacto y Sistema

Los modelos de ***procesos empíricos*** se derivan de la categorización de las entradas y salidas observadas, y de la definición de los controles que hacen que se produzcan dentro de los límites

prescritos. El modelado de procesos empíricos implica la construcción de un modelo de proceso estrictamente a partir de datos de entrada/salida obtenidos experimentalmente, sin recurrir a ninguna ley relativa a la naturaleza y propiedades fundamentales del sistema. No es necesario un conocimiento *previo* del proceso (aunque puede ser útil); el sistema se trata como una caja negra.

Las características principales de los modelos teóricos y empíricos se detallan en la Tabla 1.

Tras la inspección, afirmamos que el proceso de desarrollo de sistemas es empírico:

1. Los primeros principios aplicables no están presentes
2. El proceso sólo está empezando a comprenderse
3. El proceso es complejo
4. El proceso está cambiando

Theoretical Modeling	Empirical Modeling
1. Usually involves fewer measurements; requires experimentation only for the estimation of unknown model parameters.	Requires extensive measurements, because it relies entirely on experimentation for the model development.
2. Provides information about the internal state of the process.	Provides information only about that portion of the process which can be influenced by control action.
3. Promotes fundamental understanding of the internal workings of the process.	Treats the process like a “black box.”
4. Requires fairly accurate and complete process knowledge.	Requires no such detailed knowledge; only that output data be obtainable in response to input changes.
5. Not particularly useful for poorly understood and/or complex processes.	Quite often proves to be the only alternative for modeling the behavior of poorly understood and/or complex processes.
6. Naturally produces both linear and nonlinear process models.	Requires special methods to produce nonlinear models.

Tabla 1: Modelo teórico y empírico

La mayoría de los metodólogos están de acuerdo con esta afirmación; "...no se puede esperar que un *método* te diga todo lo que tienes que hacer. Escribir software es un proceso creativo, como la pintura, la escritura o la arquitectura… (un *método*) proporciona un marco que dice cómo hacerlo e identifica los lugares donde se necesita la creatividad. Pero aún hay que aportar la creatividad...."[14]

[14] James Rumbaugh, October 1995, “What Is a Method”., Object Journal

Clasificar los métodos de desarrollo de sistemas como empíricos es fundamental para la gestión eficaz del proceso de desarrollo de sistemas.

Si los *métodos* de desarrollo de sistemas se clasifican como *empíricos*, se requieren medidas y controles porque se entiende que el funcionamiento interno del *método* está tan poco definido que no se puede contar con que funcione de forma predecible.

En el pasado, los *métodos* se han proporcionado y aplicado como si fueran teóricos. Como consecuencia, no se confiaba en las mediciones y no se utilizaban los controles que dependían de ellas.

Muchos de los problemas en el desarrollo de sistemas se han producido debido a esta categorización incorrecta. Cuando un proceso de caja negra se trata como un proceso totalmente definido, se producen resultados imprevisibles. Además, no existen los controles necesarios para medir y responder a la imprevisibilidad.

En la práctica, en múltiples empresas y en múltiples proyectos dentro de algunas empresas, se ha observado que SCRUM proporciona una solución viable a estos problemas. Los proyectos se entregan a tiempo y a menudo superan las expectativas tanto de los usuarios como del management. Aunque el trabajo en un equipo de desarrollo SCRUM es intenso, los desarrolladores se ven recompensados por el alto espíritu de equipo, un profundo sentido de logro y la sensación de que el desarrollo puede ser una experiencia agradable y satisfactoria.

References

1. Aberdeen Group. Upgrading To ISV Methodology For Enterprise Application Development. Product Viewpoint 8:17, December 7, 1995.
2. Bach, James. “Process Evolution in a Mad World.”Borland International, Scotts Valley, CA.
3. Bach, James. “The Challenge of “Good Enough”Software”, American Programmer, October 1995.
4. Boehm, B.W. 1985. “A Spiral Model of Software Development and Enhancement,” from Proceedings of an International Workshop on Software Process and Software Environments, Coto de Caza, Trabuco Canyon, California, March 27-29, 1985.
5. Booch, Grady. Object Oriented Analysis and Design with Applications. The Benjamin/Cummings Publishing Company, Inc., 1994, p. 8
6. Booch, Grady. Object Solutions: Managing the Object-Oriented Project. Addison-Wesley, 1995.
7. Brooks, F.P. “No silver bullet—essence and accidents of software engineering.” Computer 20:4:10-19, April 1987.
8. Coplien, J. “Borland Software Craftsmanship: A New Look at Process, Quality and Productivity.”Proceedings of the 5th Annual Borland International Conference, June 5, 1994. Orlando, Florida.
9. DeGrace, P. and Hulet Stahl, L. 1990. Wicked Problems, Righteous Solutions. Yourdon Press

10. Gartner, Lisa. The Rookie Primer. Radcliffe Rugby Football Club, 1996 <http://vail.al.arizona.edu/rugby/rad/rookie_primer.html>
11. Gleick, J. 1987. Chaos, Making A New Science. Penguin Books.
12. Graham, Ian. Migrating to Object Technology. Addison-Wesley, 1994.
13. Kahn, D. and Sutherland, J. March-April 1994. “Object Insider: Let’s start under-promising and over-delivering on OT.” Object Magazine.
14. Langton, Christopher. Artificial Life. In Artificial Life, Volume VI: SFI Studies in the Sciences of Complexity (Ed. C. Langton) Addison-Wesley, 1988.
15. Nonaka, Ikujiro and Takeuchi, Hirotaka. 1995. The Knowledge Creating Company: How Japanese Companies Create the Dynamics of Innovation,Oxford University Press.
16. Ogunnaike, B. 1994. Process Dynamics, Modeling, and Control. Oxford University Press.
17. Pittman, Matthew. Lessons Learned in Managing Object-Oriented Development. IEEE Software, January, 1993, pp. 43-53.
18. Rumbaugh, October 1995, “What Is a Method”. Journal of Object Oriented Programming.
19. Schwaber, Ken. “Controlled Chaos: Living on the Edge.” American Programmer, April 1996.
20. Sutherland, Jeff. ScrumWeb Home Page: A Guide to the SCRUM Development Process. Jeff Sutherland’s Object Technology Web Page, 1996 <http://www.tiac.net/users/jsuth/scrum/index.html>
21. Takeuchi, Hirotaka and Nonaka, Ikujiro. January-February 1986. “The New New Product Development Game.” Harvard Business Review.