

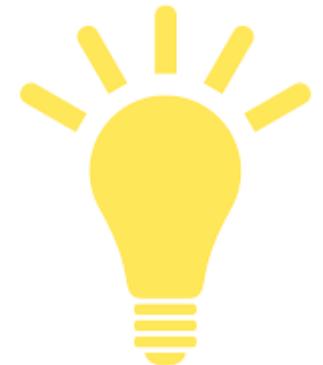
# Computación Ubicua

## Sesión 4 – Metodologías de Desarrollo

Ana Castillo Martínez  
Javier Albert Segui

# Fábula del columpio

- Pedro quería **construir un columpio** en su jardín para su hija como regalo de cumpleaños, así que llamó a su hermana María, con quien tenía confianza, para que se encargarse de todo y él no tener que preocuparse por nada. Sin embargo, Pedro quería estar informado en todo momento, así que María le transmitiría toda la información sobre la evolución del proyecto



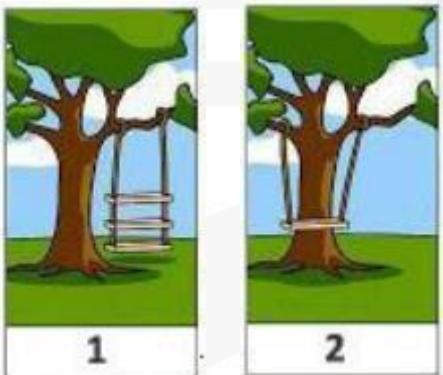
# Fábula del columpio

- **1º Como Pedro lo explicó:** Pedro no poseía un lenguaje técnico ni sabía cómo llevar a cabo su proyecto de columpio porque nunca había construido uno, pero tenía una imagen clara de cómo quería que quedase y sabía cuál era el dinero que estaba dispuesto a gastarse, así que se lo contó a María como mejor supo



# Fábula del columpio

- **2º Como María lo entendió:** María escuchó a Pedro, pero se quedó sólo con los elementos de entrada y no profundizó más, por lo que no llegó a comprender a la perfección la idea de Pedro



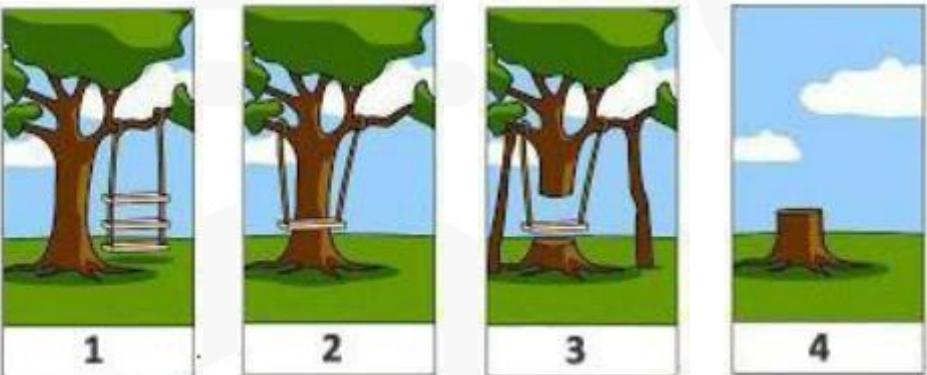
# Fábula del columpio

- **3º Como Juan lo entendió:** María contrató a Juan, que es diseñador de columpios y tiene una empresa de diseño. Él ya había diseñado muchos columpios anteriormente por lo que tenía una idea preconcebida de lo que Pedro necesitaba, y en vez de preguntar a María para entender mejor lo que quería, creó una solución muy vistosa y fascinante, aunque no sabía con exactitud si era posible llevarla a cabo por coste y plazo



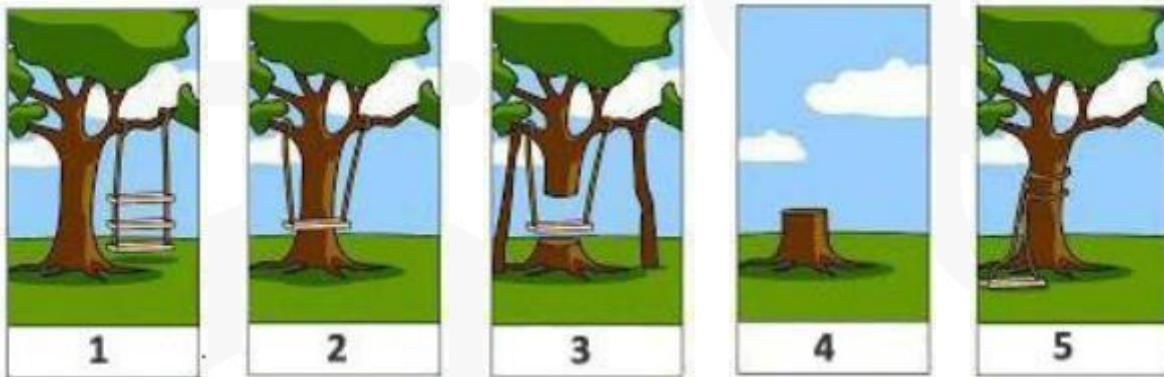
# Fábula del columpio

- **4º Como Juan pensó en el mantenimiento y en su vida útil:** A la hora de diseñarlo, Juan no pensó en los posibles gastos derivados de su funcionamiento a lo largo de toda su vida útil. Si en un futuro la cuerda del columpio se rompiera y no resultase rentable para Pedro cambiarla, pues que cortase el árbol y construyese otro columpio nuevo



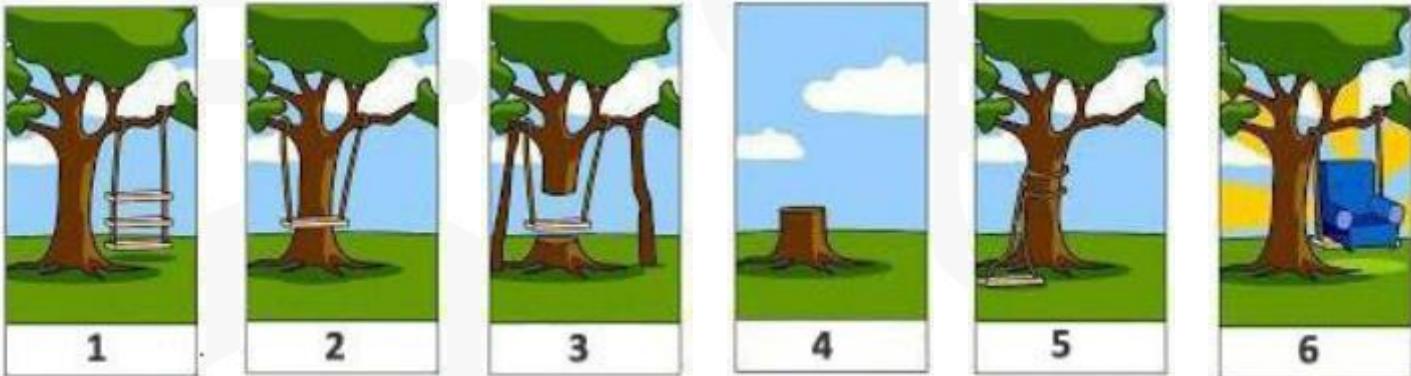
# Fábula del columpio

- **5º Como José lo definió:** José trabaja para Juan en el departamento de costes y planificación, sin embargo, las ideas de Juan chocan a menudo con los elementos disponibles para llevarlas a cabo, así que modificó la idea original para ajustarla en coste y plazo



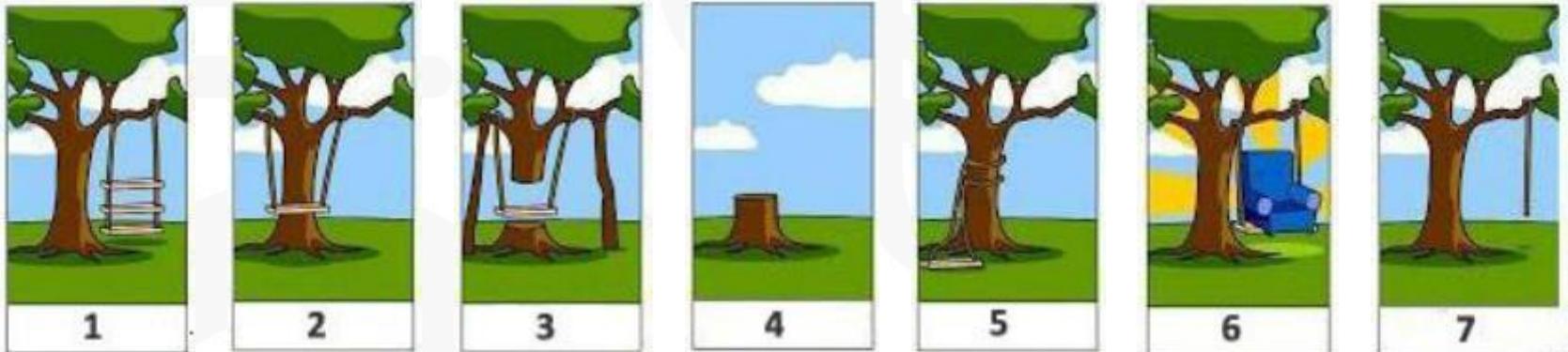
# Fábula del columpio

- **6º Como Jorge lo describió:** Jorge trabaja para Juan en el departamento de ventas, y explicó el diseño a María, pero como el papel lo aguanta todo y él trabaja a comisión, no le importó adornar la idea original para realizar la venta



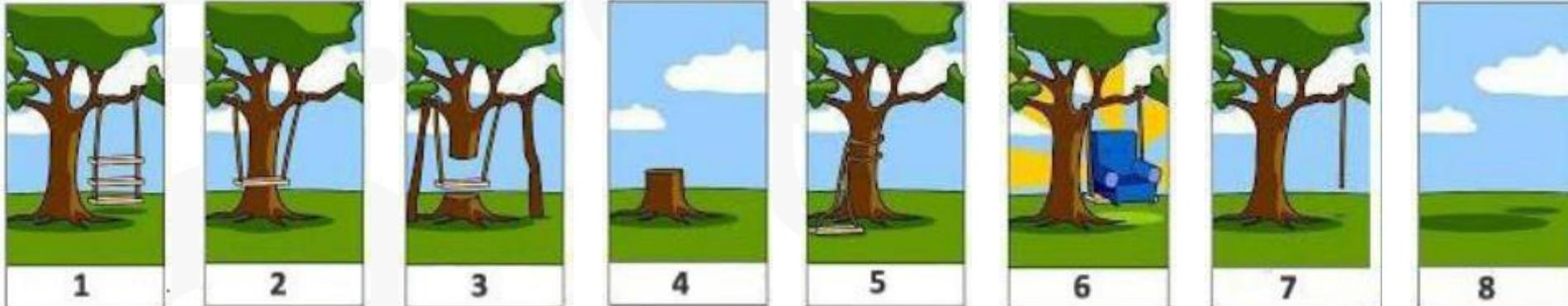
# Fábula del columpio

**7º Como Manolo lo ejecutó:** Manolo siempre se ha dedicado a construir columpios. A él le llegó la idea inicial de Juan, el proyecto modificado por José e incluso la documentación de venta de Jorge, entre tanta contradicción decidió hacer lo que pudo, sobre todo porque el cumpleaños de la hija de Pedro se acercaba y tenía que terminarlo



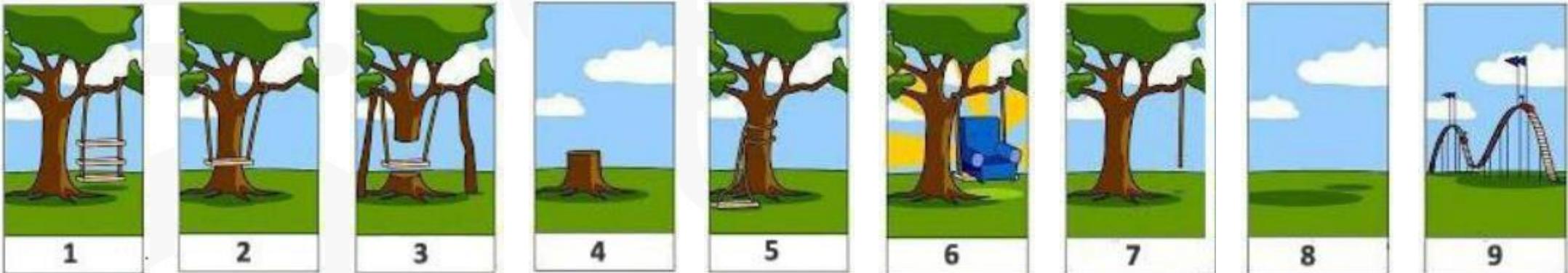
# Fábula del columpio

**8º Como fue la comunicación entre Manolo y José con María:** José no quiso contarle a María que había tenido que modificar el proyecto que le presentó Jorge y que tanto le gustó. Manolo tampoco informó de las múltiples incoherencias del proyecto que había encontrado por miedo a que eligiesen a otro constructor para llevarlo a cabo



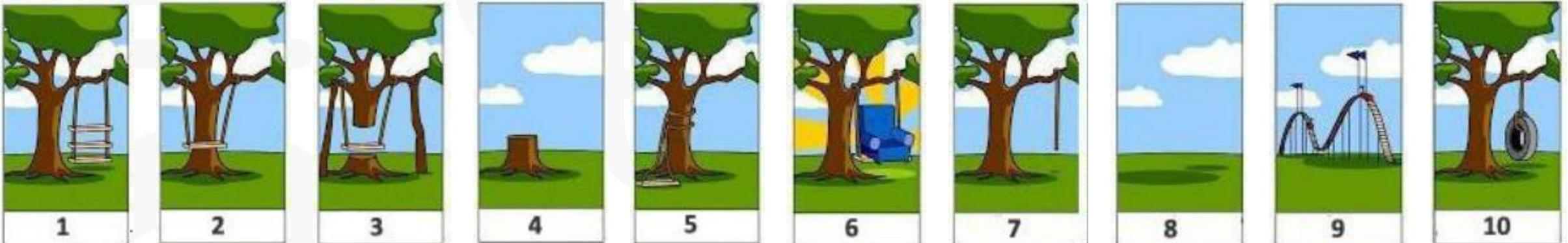
# Fábula del columpio

**9º Como le fue facturado a Pedro:** Por los múltiples cambios que sufrió el proyecto y por los múltiples beneficios que sacaron los intervenientes de él, al final, facturaron el columpio a Pedro a precio de atracción de feria

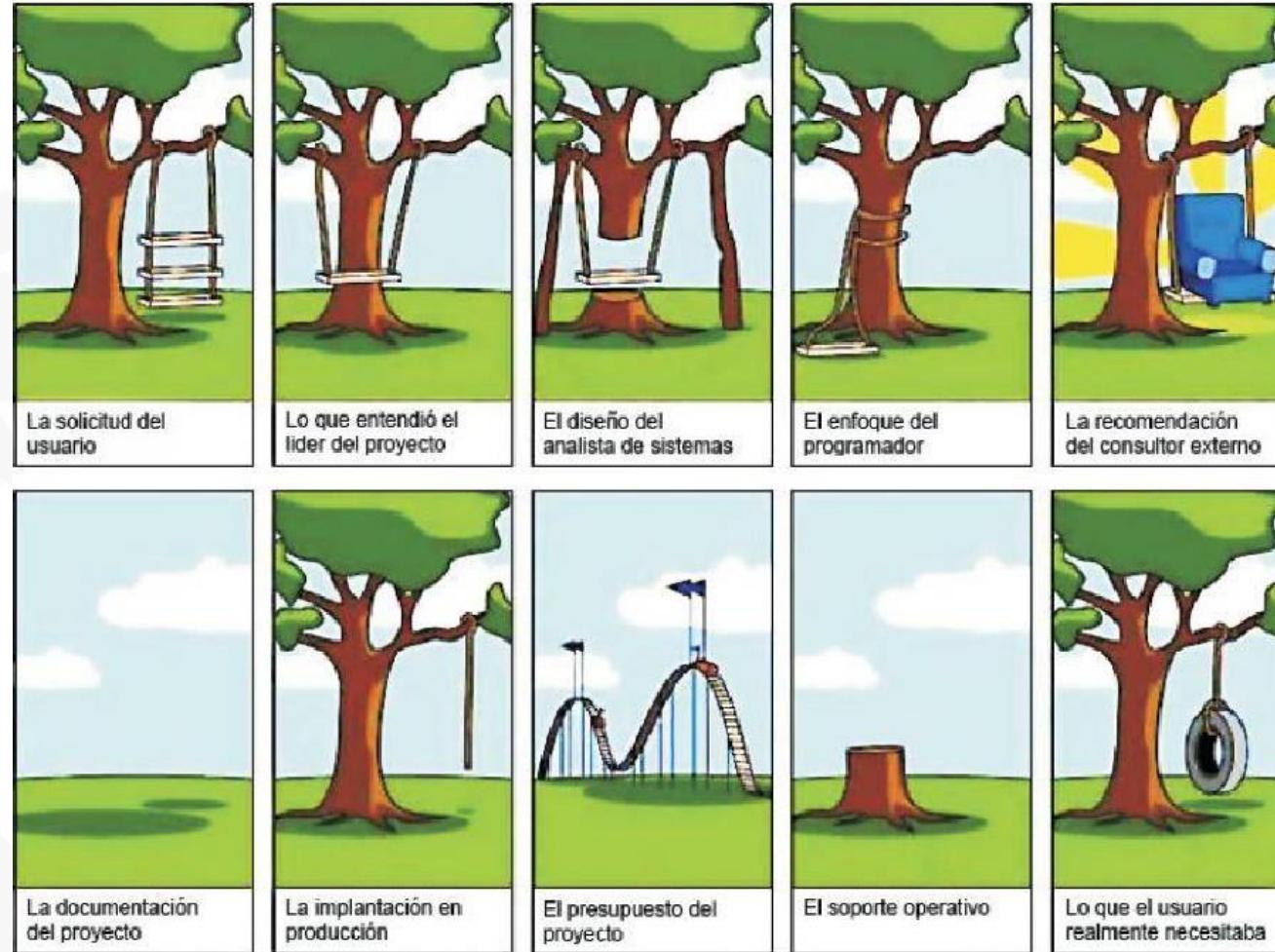


# Fábula del columpio

**10º Lo que Pedro realmente necesitaba:** "Pero, ¿era tan complicada mi idea?" Pensó Pedro, quien se sintió estafado y engañado, y no volvió a confiar en ningún constructor ni diseñador de columpios



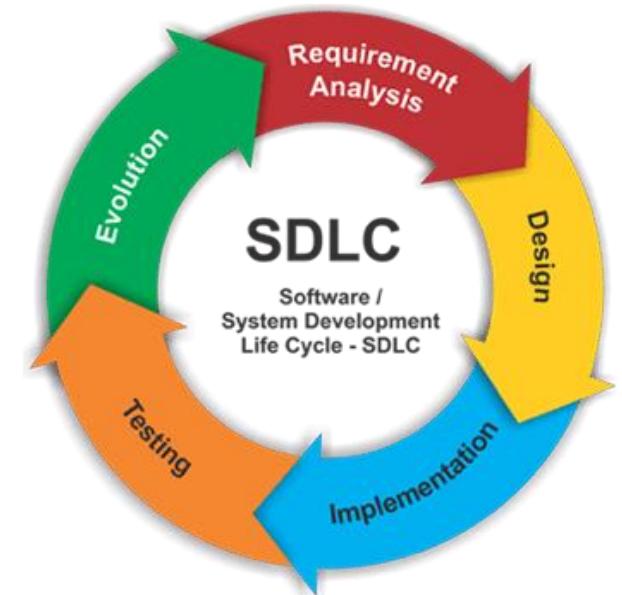
# Fábula del columpio



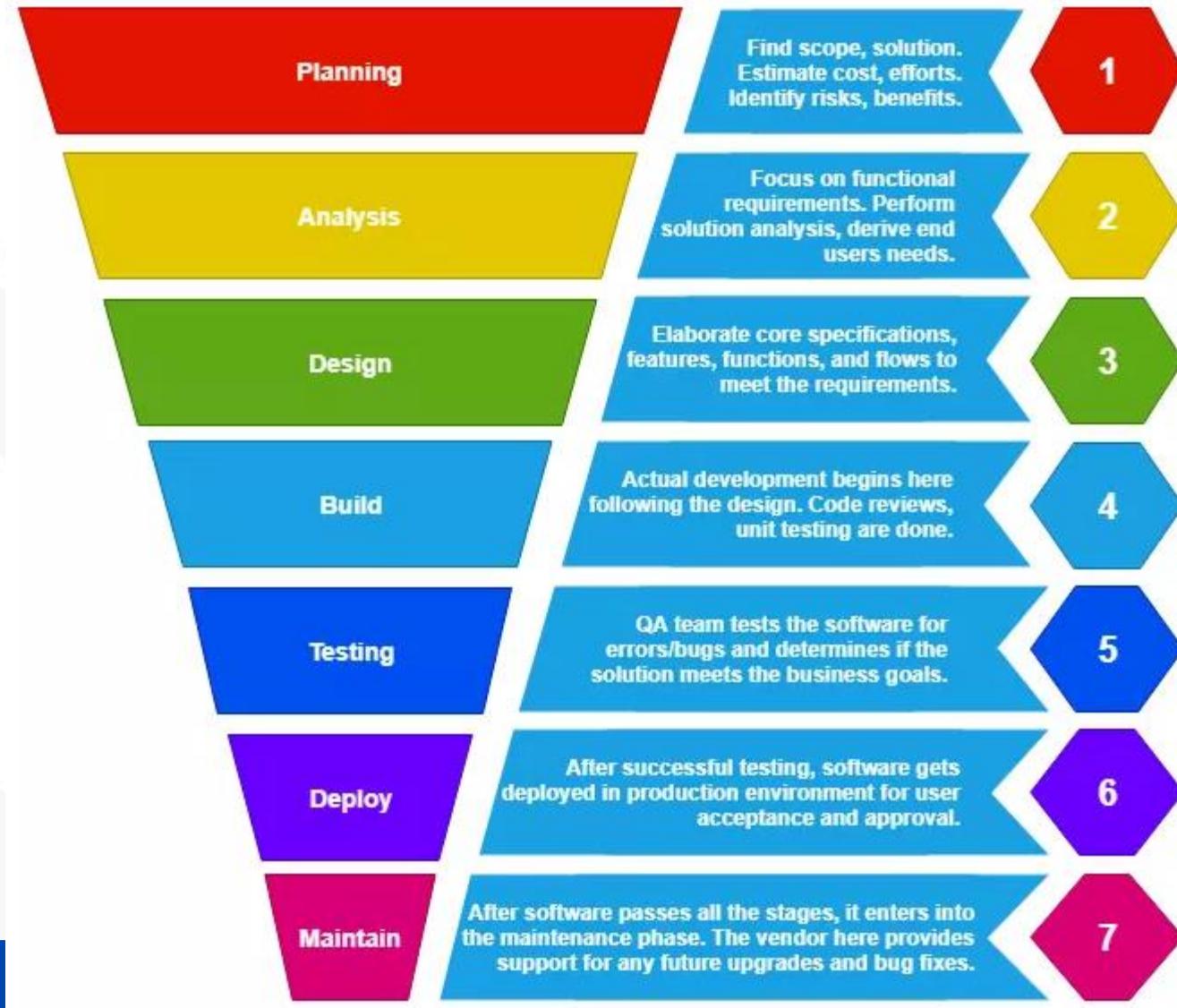
# Definiciones

# Ciclo de Vida de Desarrollo de Sistemas (SDLC)

- El ciclo de vida de desarrollo de sistemas (SDLC) es el **proceso de creación o modificación** de los sistemas, modelos y metodologías que la gente usa para desarrollar estos sistemas de software
- En ingeniería de software el concepto de SDLC sostiene muchos tipos de metodologías de desarrollo de software. Estas metodologías constituyen el marco para la planificación y el control de la creación de una información en el proceso de desarrollo de software



# Fases del SDCL



# Metodologías de desarrollo

- La metodología de desarrollo en ingeniería de software es un **marco de trabajo usado para estructurar, planificar y controlar** el proceso de desarrollo en sistemas de información
  - Una filosofía de desarrollo de programas de computación con el enfoque del proceso de desarrollo de software
  - Herramientas, modelos y métodos para asistir al proceso de desarrollo de software



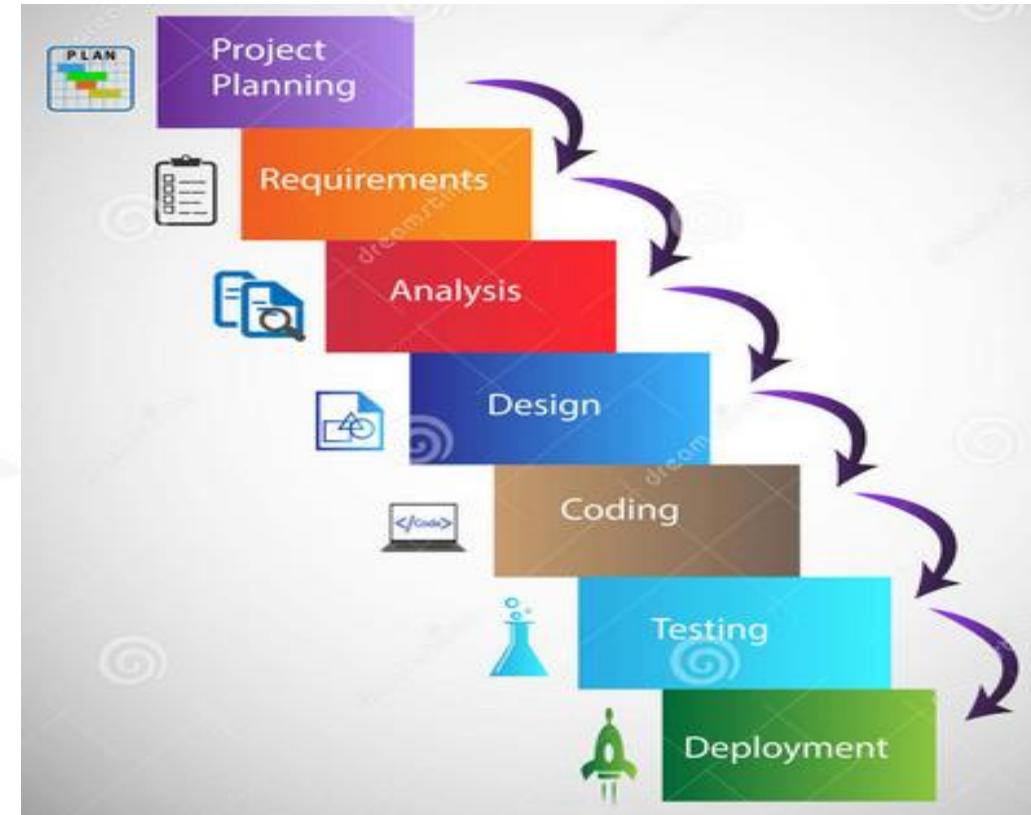
# Tipos de metodologías

Metodologías Tradicionales	Metodologías Agiles
Basadas en normas provenientes de <b>estándares</b> seguidos por el entorno de desarrollo	Basadas en heurísticas provenientes de <b>prácticas de producción</b> de código
Cierta <b>resistencia a los cambios</b>	<b>Preparados para cambios</b> durante el proyecto
Proceso mucho <b>más controlado</b> , con numerosas políticas/normas	Proceso <b>menos controlado</b> , con pocos principios
El <b>cliente interactúa con el equipo</b> de desarrollo mediante reuniones	El <b>cliente es parte del equipo</b> de desarrollo
<b>Grupos grandes</b> y posiblemente distribuidos	<b>Grupos pequeños</b> y trabajando en el mismo sitio
La <b>arquitectura del software es esencial</b> y se expresa mediante modelos	<b>Menos énfasis</b> en la arquitectura del software

# Metodologías tradicionales

# Modelo en cascada

- Modelo con un enfoque secuencial propuesto en 1970
- Las fases del ciclo de vida se ordenan de forma que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior
- Cualquier fallo en fases anteriores será arreglado en la fase actual

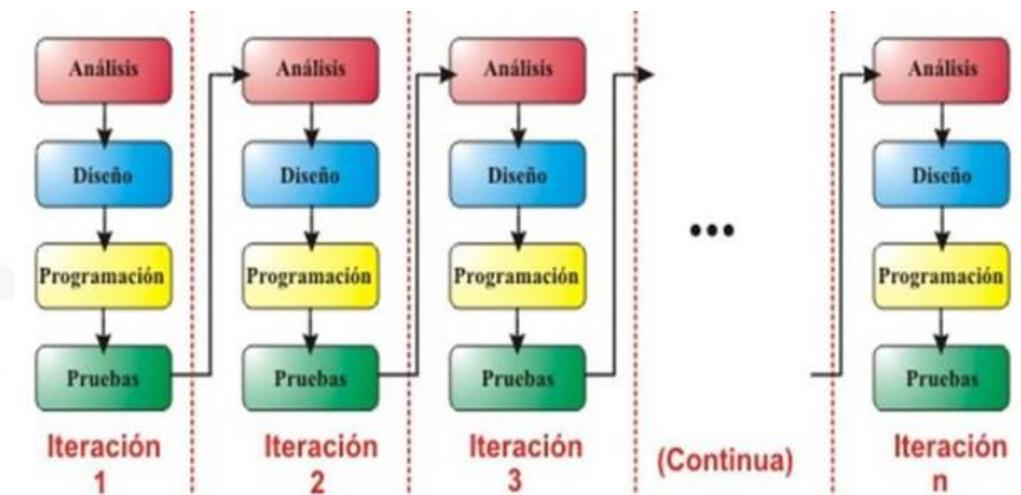


# Modelo en cascada

Ventajas	Inconvenientes
<ul style="list-style-type: none"><li>• Perfecto para proyectos con <b>requisitos muy definidos</b></li><li>• Método muy estructurado que es muy útil para <b>gente con poca experiencia</b></li><li>• Permite <b>comenzar</b> el desarrollo <b>con rapidez</b></li><li>• <b>Precisión</b> de calendarios y presupuestos</li><li>• Útil para <b>proyectos grandes</b></li></ul>	<ul style="list-style-type: none"><li>• <b>Poca flexibilidad</b></li><li>• En la realidad <b>pocos proyectos</b> siguen un flujo secuencial</li><li>• Necesidad de reunir todos los <b>requisitos desde inicio</b></li><li>• Es <b>difícil para el cliente</b> mostrar todos los requerimientos del proyecto desde el inicio</li><li>• Las <b>revisiones de proyectos</b> de gran complejidad son <b>difíciles</b></li><li>• Los usuarios tienen una <b>participación limitada</b></li></ul>

# Modelo Incremental

- Modelo iterativo y creciente
- Surge como solución a los problemas del modelo en cascada
- Se van haciendo mini cascadas en cada iteración, de forma que pasa por todas sus fases
- Una vez acabada una mini cascada, comienza la siguiente iteración, y así sucesivamente



# Modelo Incremental

## Ventajas

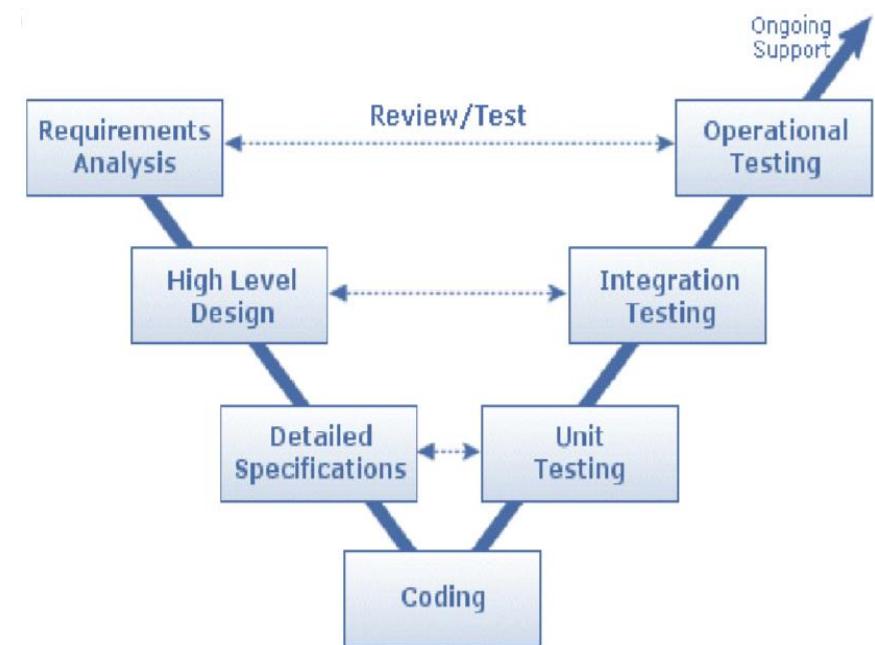
- Generación de **software rápido** y en etapas tempranas
- Es un modelo **flexible** a los cambios
- **Facilidad** para probar y depurar en iteraciones pequeñas
- Fácil **gestión de los riesgos**
- Cada iteración es un **hito**

## Inconvenientes

- Cada fase de una **iteración es rígida** y no se superpone con otras
- Posibles **problemas de arquitectura** en el caso de no definir correctamente los requisitos

# Modelo en V

- Proceso que representa la secuencia de pasos en el desarrollo del ciclo de vida de un proyecto. La V significa «Verificación y vendeta»
- Es muy similar al modelo de cascada clásico ya que es muy rígido y contiene una gran cantidad de iteraciones
  - El lado izquierdo representa la descomposición de las necesidades, y la creación de las especificaciones del sistema
  - El lado derecho representa la integración de las piezas y su verificación.



# Modelo en V

## Ventajas

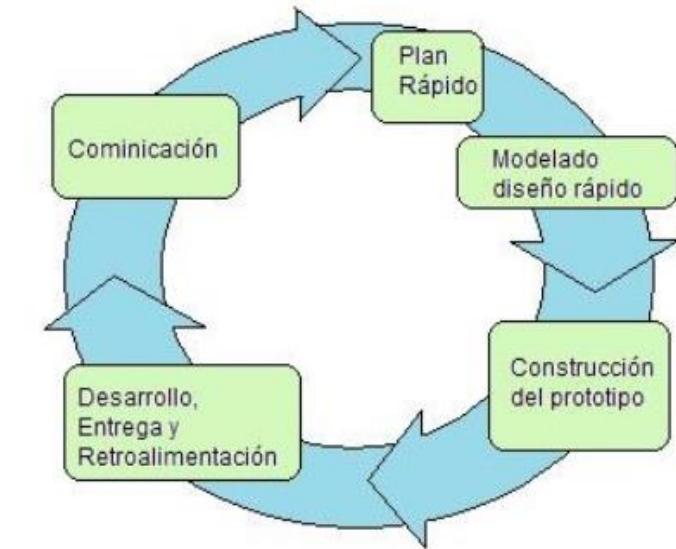
- Especifica bien los **roles de las pruebas** a realizar
- Implica **chequeos** de cada etapa
- Sencillo y de **fácil aprendizaje**
- **Involucra al usuario** en las pruebas

## Inconvenientes

- **No permite retomar etapas** inmediatamente anteriores
- Si un proceso es mal diseñado debe ser revisado, ocasionando **sobrecostes**
- Las **pruebas** pueden ser **caras** y a veces **no efectivas**

# Modelo basado en prototipado

- Modelo con un enfoque iterativo
- Se basa en realizar pequeños prototipos finales de la aplicación de forma que sus funcionalidades se construyen encima de la versión anterior, hasta llegar al producto definitivo y su entrega al cliente



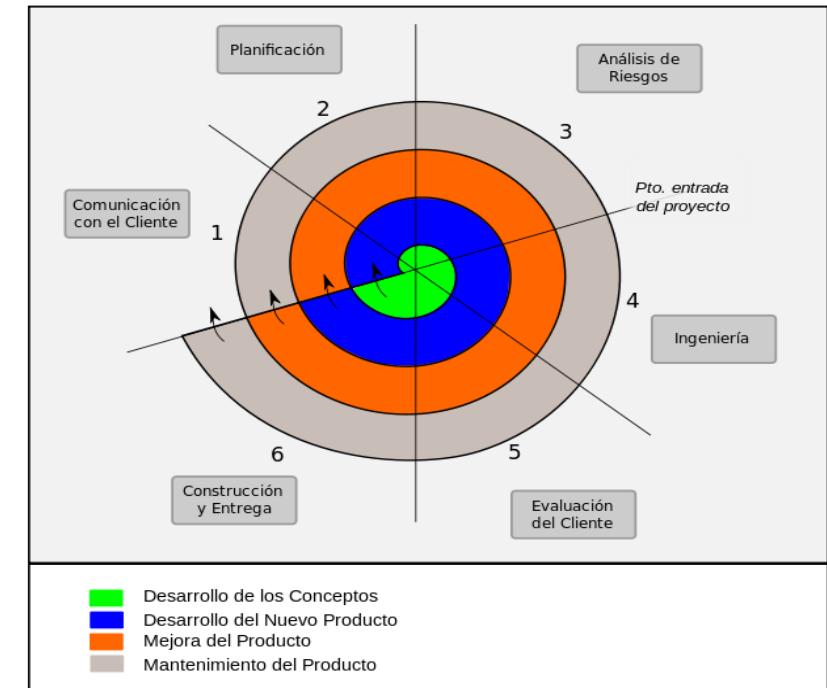
MODELO DE CONSTRUCCIÓN DE PROTOTIPOS

# Modelo basado en prototipado

Ventajas	Inconvenientes
<ul style="list-style-type: none"><li>• Ofrece <b>visibilidad del producto</b> desde el inicio</li><li>• Es <b>flexible</b> con respecto a los cambios</li><li>• Permite la <b>retroalimentación del cliente</b></li><li>• El prototipo es un <b>documento vivo</b> del producto final</li><li>• El <b>cliente reacciona mucho mejor</b> ante el prototipo</li><li>• <b>Reduce el riesgo</b> de producir algo que no cumpla con las necesidades del cliente</li></ul>	<ul style="list-style-type: none"><li>• Es un <b>desarrollo lento</b></li><li>• Necesidad de realizar una <b>fuerte inversión</b> en un producto que se descartará</li><li>• Puede <b>aumentar el coste</b> del desarrollo</li></ul>

# Modelo en espiral

- Permite un enfoque evolutivo
- Las actividades de este modelo se conforman en una espiral, en la que cada bucle o iteración representa un conjunto de actividades.
- Toma en consideración la evaluación de los riesgos
- Utiliza la creación de prototipos como mecanismo de reducción de riesgos



# Modelo en espiral

Ventajas	Inconvenientes
<ul style="list-style-type: none"><li>• Se adapta a lo largo de la vida del software</li><li>• Mejor reacción ante riesgos en cada nivel</li><li>• Permite aplicar prototipos</li><li>• Permite incluir otros métodos de desarrollo en las iteraciones</li><li>• Se tienen puntos de control en cada iteración</li></ul>	<ul style="list-style-type: none"><li>• Dificultad para convencer a clientes sobre un enfoque evolutivo</li><li>• Requiere mucha administración</li><li>• Dificultad para definir los objetivos, metas que indiquen que podemos avanzar al siguiente ciclo</li><li>• Complejo para sistemas pequeños</li><li>• Puede caer en un desarrollo de nunca acabar</li></ul>

# Metodologías agiles

# Metodologías Agiles

- Las metodologías ágiles son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno



# Manifiesto Ágil

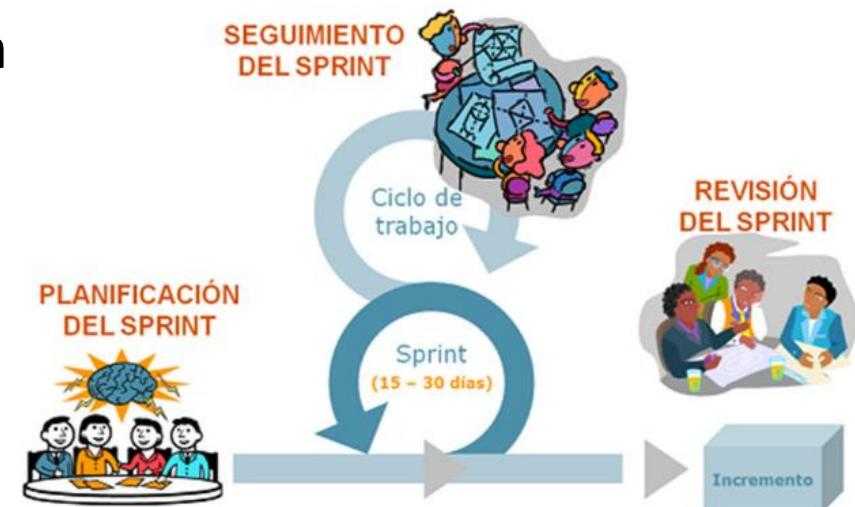
- Principios comunes a las metodologías ágiles de desarrollo
  1. La prioridad es **satisfacer al cliente** mediante la entrega temprana y continua de software
  2. **Los cambios son bienvenidos** aunque llegan tarde al desarrollo
  3. **Entregas frecuentes** de software que funcione, en periodos breves
  4. El **cliente y los desarrolladores deben trabajar juntos** de forma cotidiana a través del proyecto
  5. Construcción de proyectos en torno a **individuos motivados**, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea
  6. La forma más eficiente y efectiva de intercambiar información es mediante el **cara a cara**

# Manifiesto Ágil

7. El **software que funciona** es la principal medida del progreso
8. Promueven un **desarrollo sostenido**, donde se mantenga un ritmo constante
9. La **atención continua** a la excelencia técnica enaltece la agilidad
10. La  **simplicidad** como arte de maximizar la cantidad de trabajo que no se hace, es esencial
11. Las mejores arquitecturas, requisitos y diseños emergen de **equipos que se auto-organizan**
12. El equipo debe reflexionar sobre cómo **ser más efectivo** y ajustar su conducta

# Metodología SCRUM

- Metodología basada en la experiencia y la toma de decisiones
- Scrum se centra en la división del trabajo en distintos bloques que pueden ser abordados en periodos cortos de tiempo (1-4 semanas) denominados Sprints
- Se compone de 4 eventos que componen cada entrega:
  - Reunión de planificación del Sprint (Sprint Plan)
  - Scrum diario (Daily Scrum)
  - Revisión del Sprint (Sprint Review)
  - Retrospectiva del Sprint (Sprint Retrosctive)



# Metodología SCRUM

- Equipo de desarrollo:
  - **Dueño del producto.** Persona responsable de transmitir al equipo de desarrollo la visión del producto que se desea crear
  - **Scrum Master.** Persona encargada de ayudar al equipo y los clientes a comprender las iteraciones
  - **Equipo de desarrollo.** Equipo de 5-9 personas responsable de desarrollar y entregar el producto

# Metodología SCRUM



## CORAJE

Los miembros del equipo tiene el coraje de hacer lo correcto y resolver problemas difíciles



## FOCO

Todos se enfocan en el trabajo del Sprint y los objetivos del Equipo



## COMPROMISO

Las personas se comprometen a alcanzar los objetivos del equipo



## RESPETO

Los miembros del equipo se respetan mutuamente para ser personas capaces e independientes



## FRANQUEZA

El equipo y sus partes interesadas acuerdan ser abiertos sobre todo el trabajo y los desafíos para realizar el trabajo



SCRUM VALUES

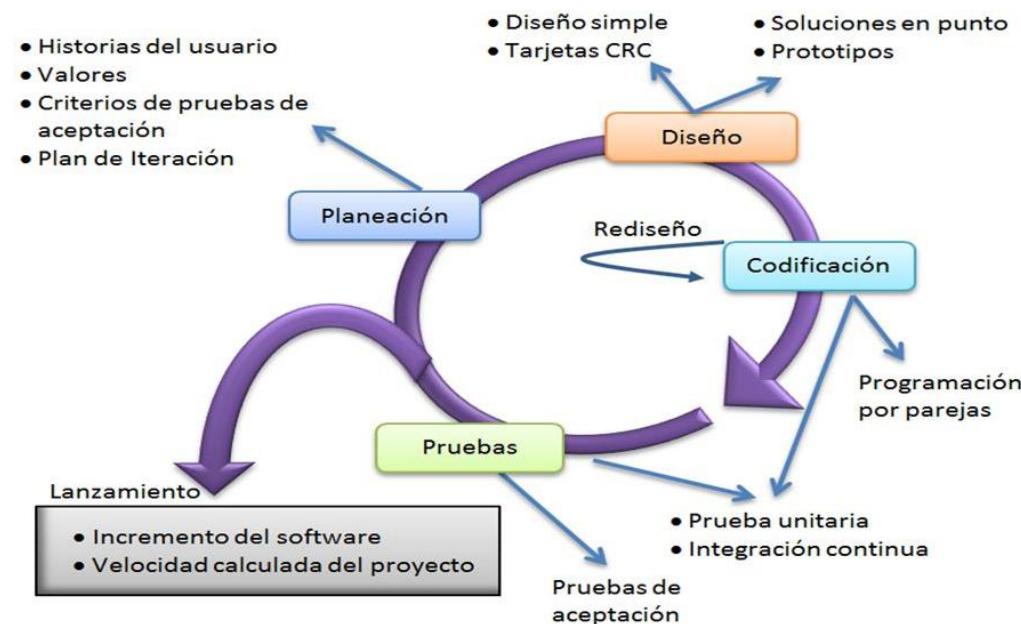
Scrum Values © 2017 Scrum.org

# Metodología SCRUM

Ventajas	Inconvenientes
<ul style="list-style-type: none"><li>• Gestión de las expectativas del cliente</li><li>• Generación de resultados anticipados</li><li>• Flexibilidad y adaptación a los contextos</li><li>• Gestión sistemática de riesgos</li><li>• Cada persona sabe qué tiene que hacer</li><li>• Equipo mas motivado</li><li>• Programación organizada</li></ul>	<ul style="list-style-type: none"><li>• Difícil implantación en equipos grandes</li><li>• Requiere de una exhaustiva definición de las tareas y sus plazos</li><li>• Exige una alta cualificación o formación</li><li>• Exceso de reuniones con pocos avances</li><li>• Problemas si el desarrollo está restringido por un plazo o precio de entrega</li><li>• Requiere de un equipo con experiencia</li></ul>

# Metodología XP (Extreme Programming)

- Proporciona un desarrollo iterativo e incremental
- Metodología basada en reglas y principios que se han utilizado a lo largo de la historia del desarrollo software
- Esta metodología pone el énfasis en la retroalimentación continua entre cliente y el equipo de desarrollo y es idónea para proyectos con requisitos imprecisos y muy cambiantes



# Metodología XP (Extreme Programming)

- Roles de trabajo:
  - **Programador.** Escribe las pruebas unitarias y produce el código del sistema
  - **Cliente.** Escribe las historias de usuario y las pruebas funcionales para validar su implementación
  - **Encargado de pruebas.** Ayuda al cliente a escribir las pruebas funcionales. Ejecuta pruebas regularmente
  - **Encargado de seguimiento.** Proporciona realimentación al equipo. Debe verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado
  - **Entrenador.** Es el responsable del proceso global, y de que se apliquen correctamente las prácticas XP
  - **Consultor.** Miembro externo que guía al equipo para resolver un problema
  - **Jefe del proyecto.** Construye el plantel del equipo, obtiene los recursos necesarios y maneja los problemas que se generan

# Metodología XP (Extreme Programming)

- Valores establecidos como fundamento para el trabajo realizado:

- Eficaz entre los ingenieros de software y otros participantes, XP pone el énfasis en la colaboración estrecha pero informal (verbal) entre los clientes y los desarrolladores.

Comunicación



- XP restringe a los desarrolladores para que diseñen sólo para las necesidades inmediatas, en lugar de considerar las del futuro.

Simplicidad

$$1+1=2$$

- Se obtiene de tres fuentes: el software implementado, el cliente y otros miembros del equipo de software.

Retroalimentación



- Un término más apropiado sería disciplina. Por ejemplo, es frecuente que haya mucha presión para diseñar requerimientos futuros.

Valentía



- Entre sus miembros, entre otros participantes y los integrantes del equipo, e indirectamente para el software en sí mismo.

Respeto

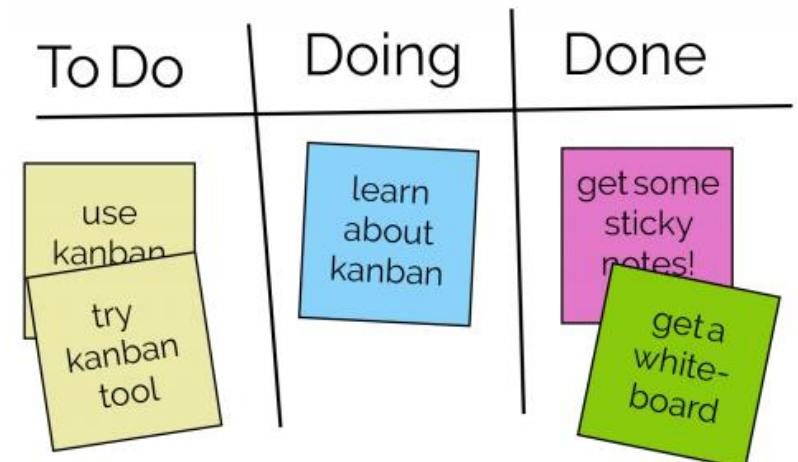


# Metodología XP (Extreme Programming)

Ventajas	Inconvenientes
<ul style="list-style-type: none"><li>• Programación organizada</li><li>• Mayor eficiencia en la planificación y pruebas</li><li>• Menor tasa de errores</li><li>• Propicia la satisfacción del programador</li><li>• Fomenta la comunicación entre clientes y desarrolladores</li><li>• Fácil adaptación ante cambios o problemas</li><li>• Realización de pruebas continuas durante el proyecto</li><li>• El cliente tiene el control sobre las prioridades</li></ul>	<ul style="list-style-type: none"><li>• Es recomendable emplearlo sólo en proyectos a corto plazo</li><li>• Altas comisiones en caso de fallar</li><li>• Requiere de un rígido ajuste al inicio del proyecto</li><li>• Demasiado costoso e innecesario</li></ul>

# Metodología Kanban

- Es una metodología complementaria a SCRUM
- Se visualiza el flujo de trabajo y este se tiene que dividir en tareas e incluirlas en el tablero
- El tablero se organiza en 3:
  - Tareas que hay que hacer
  - Tareas en curso
  - Tareas terminadas



# Metodología Kanban

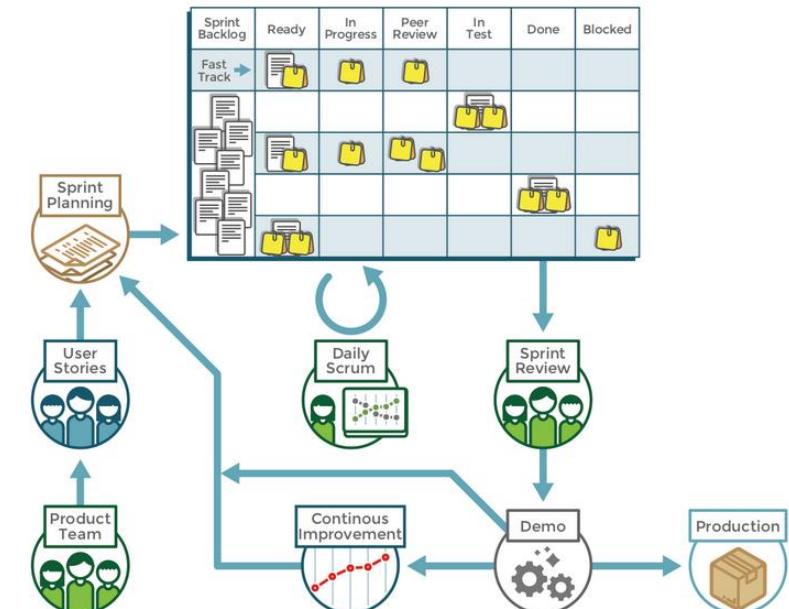
- Principios de Kanban:
  - **Visualizar el flujo de trabajo:** esquematizar el flujo de trabajo para que de manera visual se puedan ver las oportunidades de mejora fácilmente
  - **Limitar el trabajo en progreso:** limitar la cantidad de trabajo para que se pueda realizar y administrar de forma razonable
  - **Respetar los roles** y mantener un liderazgo en todos los niveles
  - **Seguimiento**, monitoreo y análisis constantes para buscar formas de mejoras allanando el terreno para futuras actualizaciones

# Metodología Kanban

Ventajas	Inconvenientes
<ul style="list-style-type: none"><li>• <b>Estimula el rendimiento</b></li><li>• Mayor <b>organización</b> y colaboración</li><li>• <b>Comodidad</b> para distribuir el trabajo</li><li>• <b>Fácil</b> de aplicar</li><li>• Gestión de tareas <b>muy visual</b></li><li>• Promueve el <b>trabajo en equipo</b></li><li>• <b>Control</b> del flujo de trabajo</li><li>• <b>Reducción del tiempo</b> perdido</li><li>• <b>Facilidad</b> para añadir mejoras</li></ul>	<ul style="list-style-type: none"><li>• No es una técnica específica del desarrollo software</li><li>• Dificultad para anticiparse a los problemas</li><li>• Difícil implantación en proyectos grandes</li><li>• Falta de reglas</li></ul>

# Metodología Scrumban

- Metodología derivada de los métodos de desarrollo Scrum y Kanban
- El flujo de trabajo es el establecido por Kanban, pero con elementos de Scrum
  - Reuniones diarias
  - Análisis retrospectivos para incorporar mejoras



# Metodología Scrumban

- Kanbanizar Scrum
  - Visualizar el trabajo previsto en el Sprint
  - Gestionar el flujo de forma activa por medio de la autoorganización
  - Mejorar en base a la experiencia y a la retrospectiva
  - Añadir límites WIP
- Scrumizar Kanban
  - Introducir roles de Scrum
  - Introducir los eventos

# Metodología Scrumban

Ventajas	Inconvenientes
<ul style="list-style-type: none"><li>• Permite conocer el estado del proceso de ejecución del proyecto</li><li>• Introduce soluciones ante errores</li><li>• Permite un mayor análisis de las tareas realizadas</li><li>• Mejora la interacción de los miembros del grupo</li><li>• Aumenta la productividad de proyectos complejos</li><li>• Favorece la adaptabilidad de las herramientas al proyecto</li></ul>	<ul style="list-style-type: none"><li>• Difícil implantación en equipos grandes</li><li>• Requiere de una exhaustiva definición de las tareas y sus plazos</li><li>• Exceso de reuniones con pocos avances</li><li>• Problemas si el desarrollo está restringido por un plazo o precio de entrega</li><li>• Requiere de un equipo con experiencia</li><li>• Inflexible a los cambios</li></ul>

# Metodología Lean

- Creación de equipos altamente preparados y que puedan llevar tareas en poco tiempo
- El equipo es lo mas importante. Cuando el equipo más haya aprendido y más unidos se encuentren el tiempo cada vez será menor
- Su principal ventaja es tener un equipo sólido, con programadores capaces de analizar la situación, tomar decisiones y llevarlas a cabo
- Su objetivo es buscar una mayor satisfacción de los clientes con el menor número de recursos posible
- Es perfecta para proyectos a medio plazo



Created by Håkan Forss @hakanforss http://hakanforss.wordpress.com

# Metodología Lean

- Se puede aplicar siguiendo 7 principios:

1. **Eliminar el desperdicio.** Eliminar lo que no aporte valor (Muda). Código basura, mala toma de requisitos, documentación excesiva...
2. **Ampliar aprendizaje.** Todos los miembros del equipo deben tener mentalidad de aprendizaje continuo
3. **Tomar decisiones tardías.** No tomar la toma de requisitos inicial al pie de la letra, ir construyéndola con los requisitos del cliente
4. **Entregar rápido.** Cada entrega incluirá funcionalidades requeridas por los usuarios lo antes posible
5. **Potenciar el equipo.** Permitir que los desarrolladores participen en la toma de decisiones
6. **Crear integridad.** Contar con un buen sistema de integración continua que incluya pruebas automatizadas, de usabilidad, etc
7. **Visualizar todo el conjunto.** *"Pensar en grande, actuar en pequeño, equivocarse rápido y aprender con rapidez"*

# Metodología Lean

## Ventajas

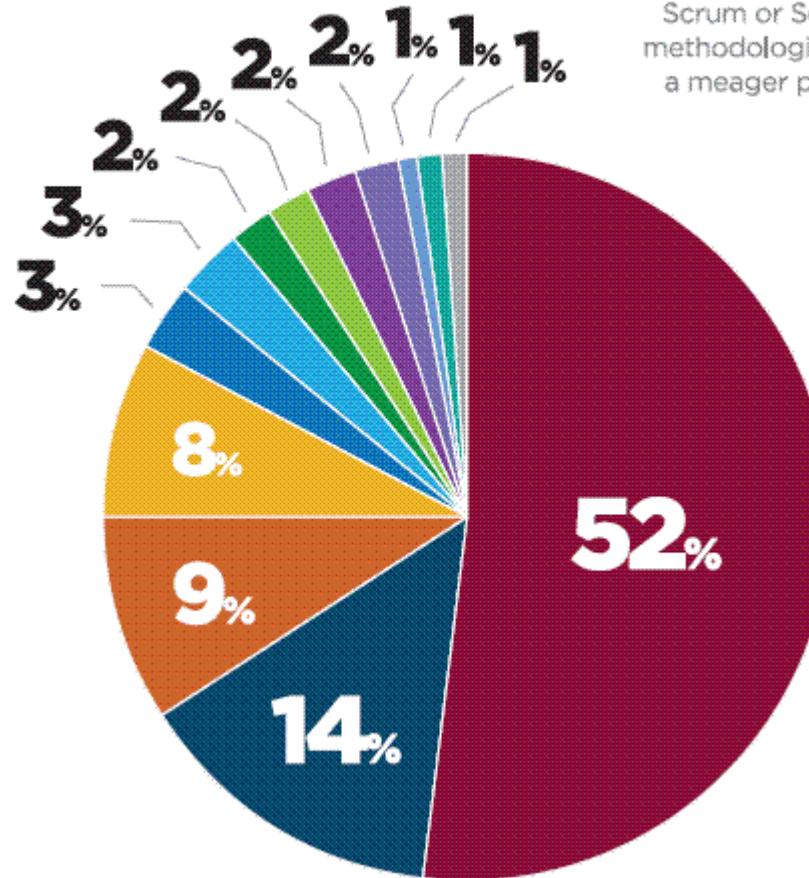
- La optimización de procesos permite **reducir costes**
- El emprendedor asume **menos riesgos**
- **Reducción del plazo** de ejecución y las actividades sin valor
- Se fomenta el **trabajo en equipo**

## Inconvenientes

- **Dependencia** de la cohesión del equipo
- **Fuerte inversión** inicial
- Necesidad de un **equipo con altas cualidades**
- **Toma de decisiones tardía** puede acarrear problemas

# Uso real de metodologías ágiles

# Uso de metodologías ágiles

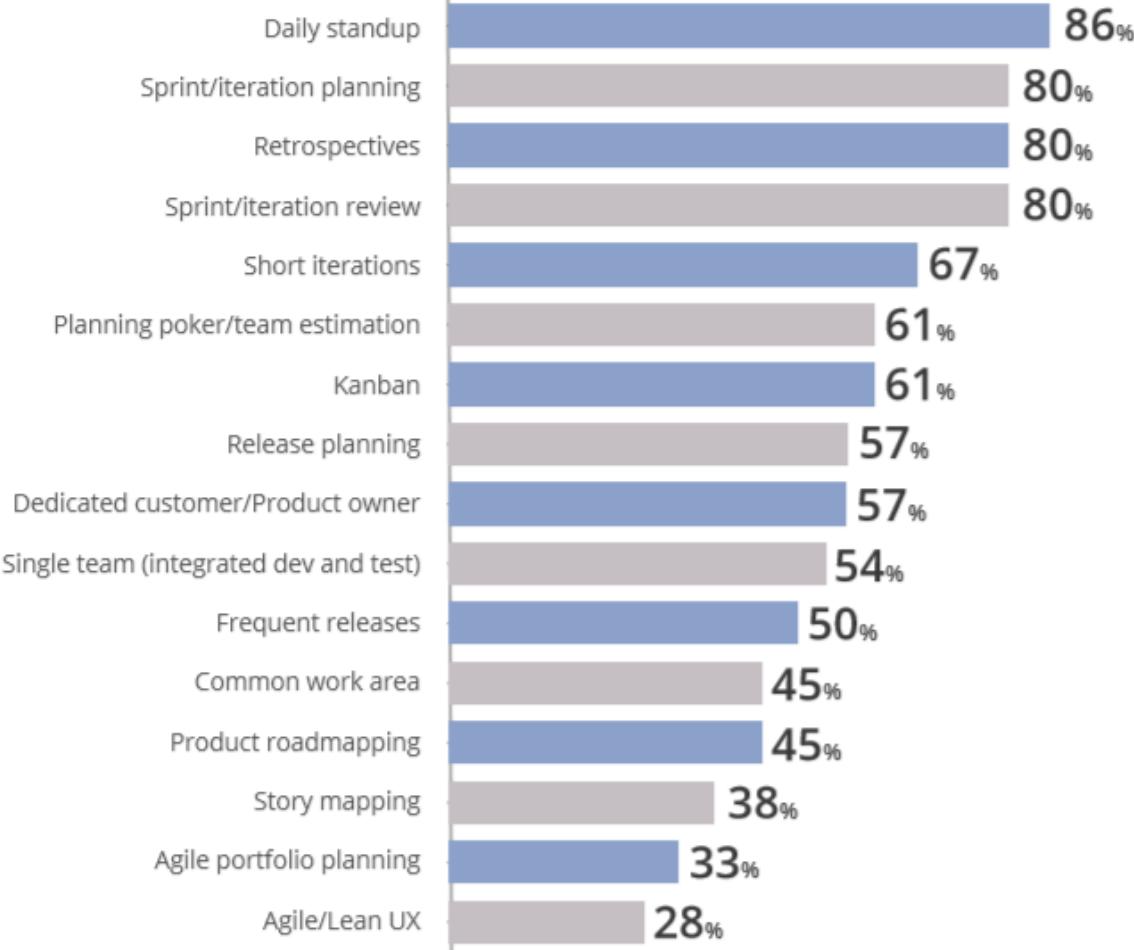
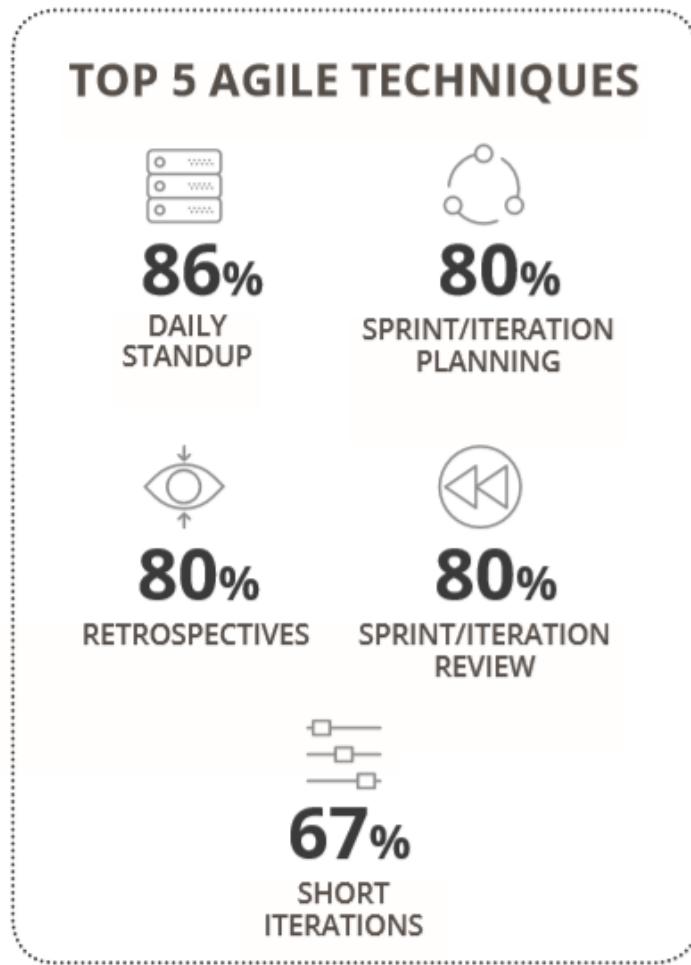


## AGILE METHODOLOGY USED

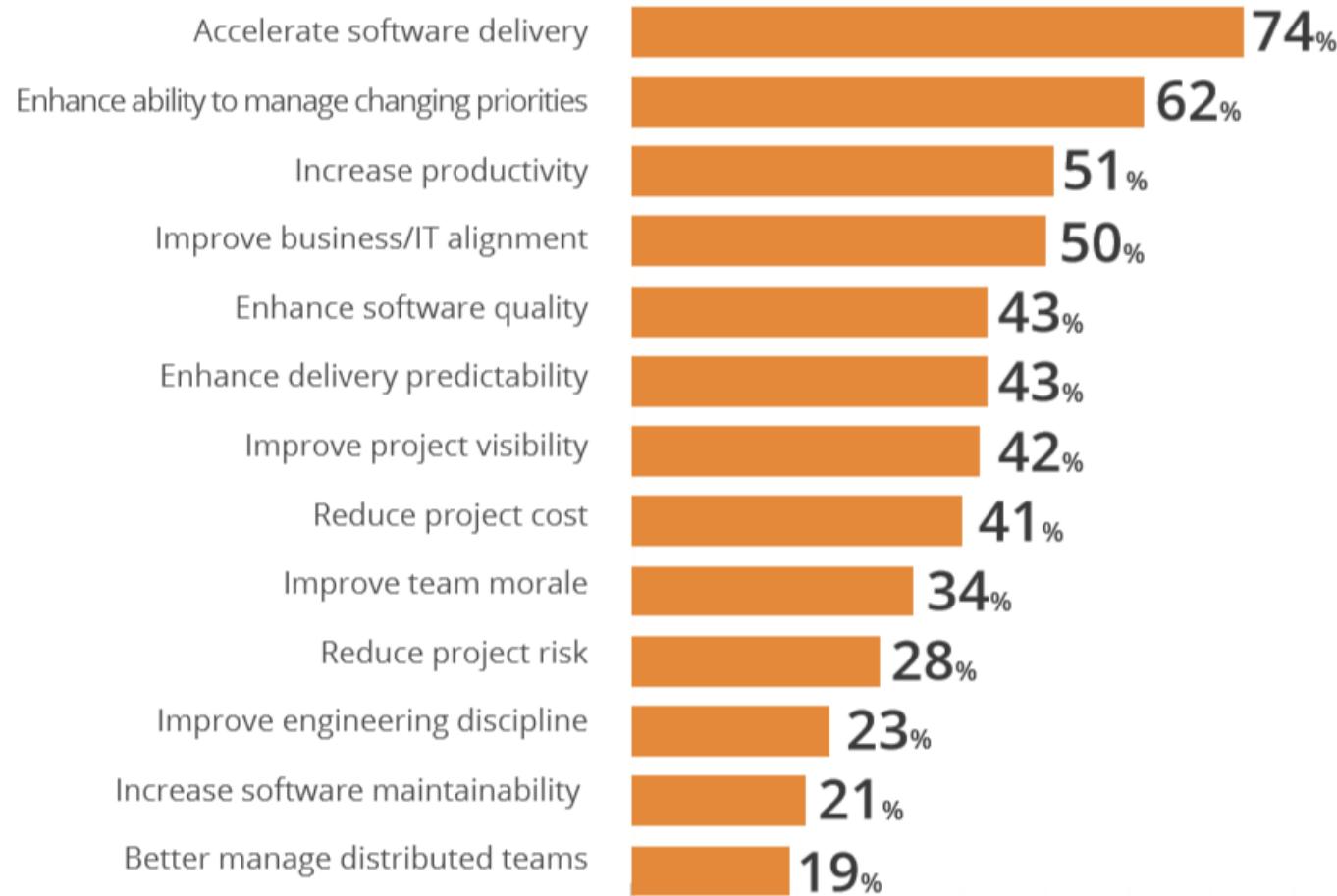
Scrum or Scrum variants continue to make up more than two-thirds of the methodologies being used, while Kanban has entered the scene this year as a meager player. The only category that saw growth this year was Custom Hybrids (9% up from 5%).

- Scrum
- Scrum/XP Hybrid
- Custom Hybrid
- Don't Know
- Kanban
- Scrumban
- Feature-Driven Development
- Extreme Programming XP
- Lean
- Other
- Agile Unified Process (AgileUP)
- Agile Modeling
- Dynamic Systems Development Method

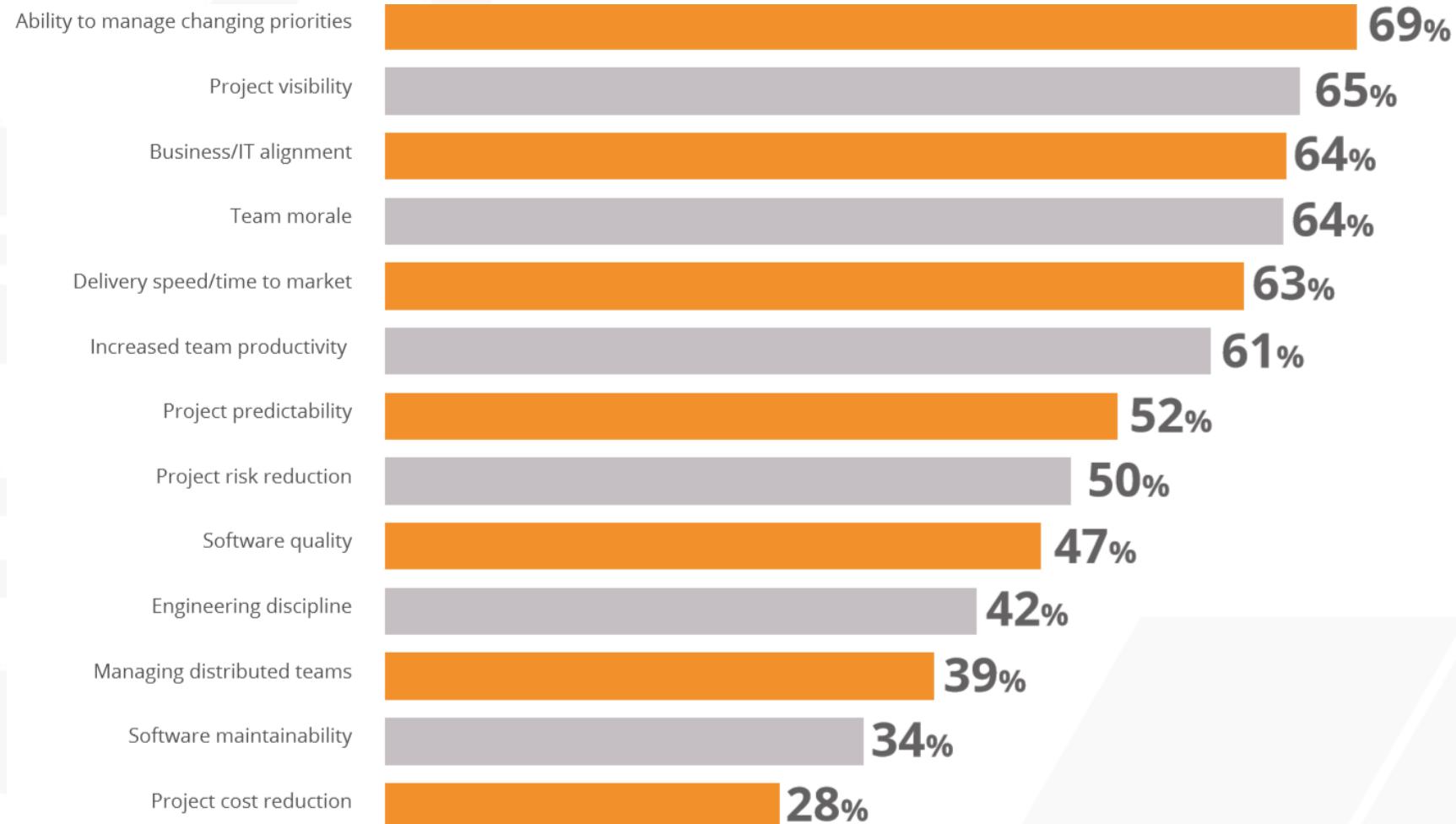
# Técnicas ágiles más usadas



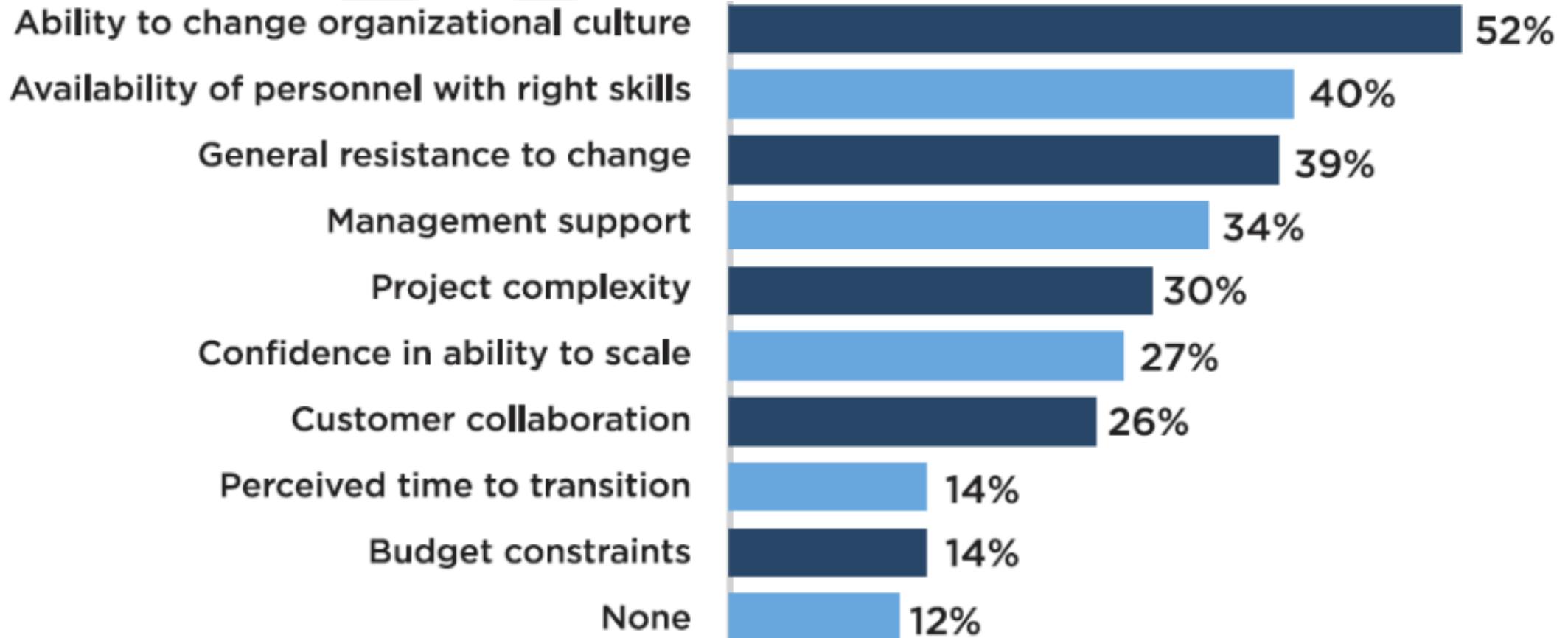
# Razones para utilizar metodologías ágiles



# Beneficios de adoptar Metodologías Ágiles



# Barreras para adoptar metodologías ágiles



# Elección de una metodología

# Elección de una metodología

- Se debería tener en cuenta la capacidad de cada metodología para afrontar proyectos:
  - Con **requerimientos** claros y concretos o cambiantes
  - Con un **tiempo** disponible amplio o reducido
  - Con unas **funcionalidades** sencillas o complejas
- También hay que:
  - Evaluar el **nivel de conocimiento** de la tecnología a usar
  - Determinar la **complejidad del sistema** a crear
  - El nivel de claridad y concreción de los **requerimientos**
  - Establecer la **fiabilidad** esperada del sistema
  - Determinar los **plazos**
  - Establecer los **límites** para cada fase del proyecto