



---

# Tema 1 – Desarrollando Software





# ¿Qué es el software?

---

- ✗ El **software** es un transformador de información, para ello: adquiere, gestiona, modifica, produce o trasmite información.

Software = Programa de computadora

- ✗ No es sólo código y datos, también forma parte del software la **documentación** producida durante su desarrollo.



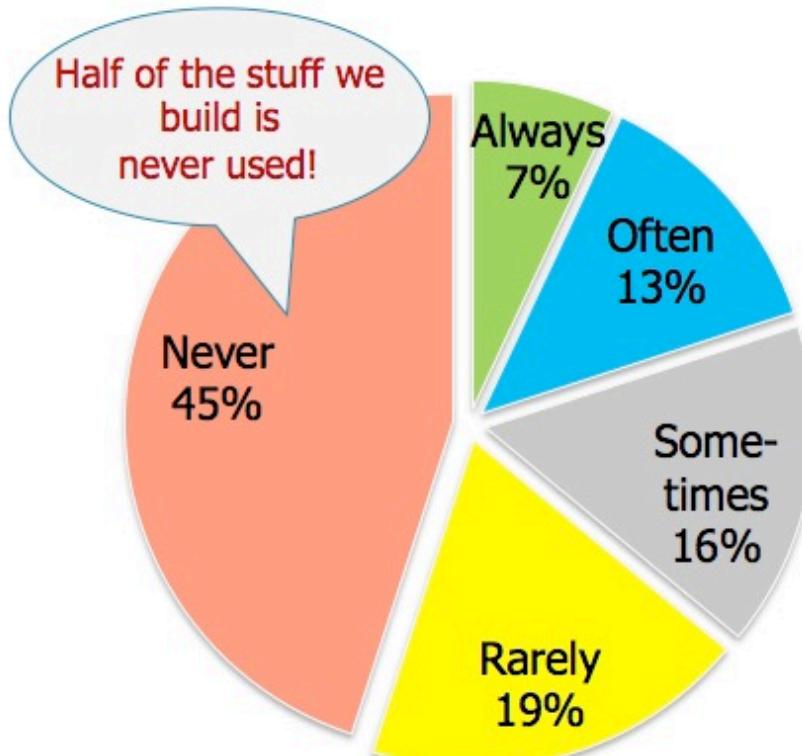
# Problemas del software

---

- ✗ Mal funcionamiento (Calidad).
- ✗ Hay que mantener el volumen de software existente.
- ✗ Mantenimiento.
- ✗ Hay una demanda creciente de nuevo software.
- ✗ Adaptación a las nuevas tecnologías.
- ✗ Incremento de la complejidad.
- ✗ Requisitos incorrectos.
- ✗ ¿Cómo desarrollamos software?



# Tendemos a construir lo que no necesitamos

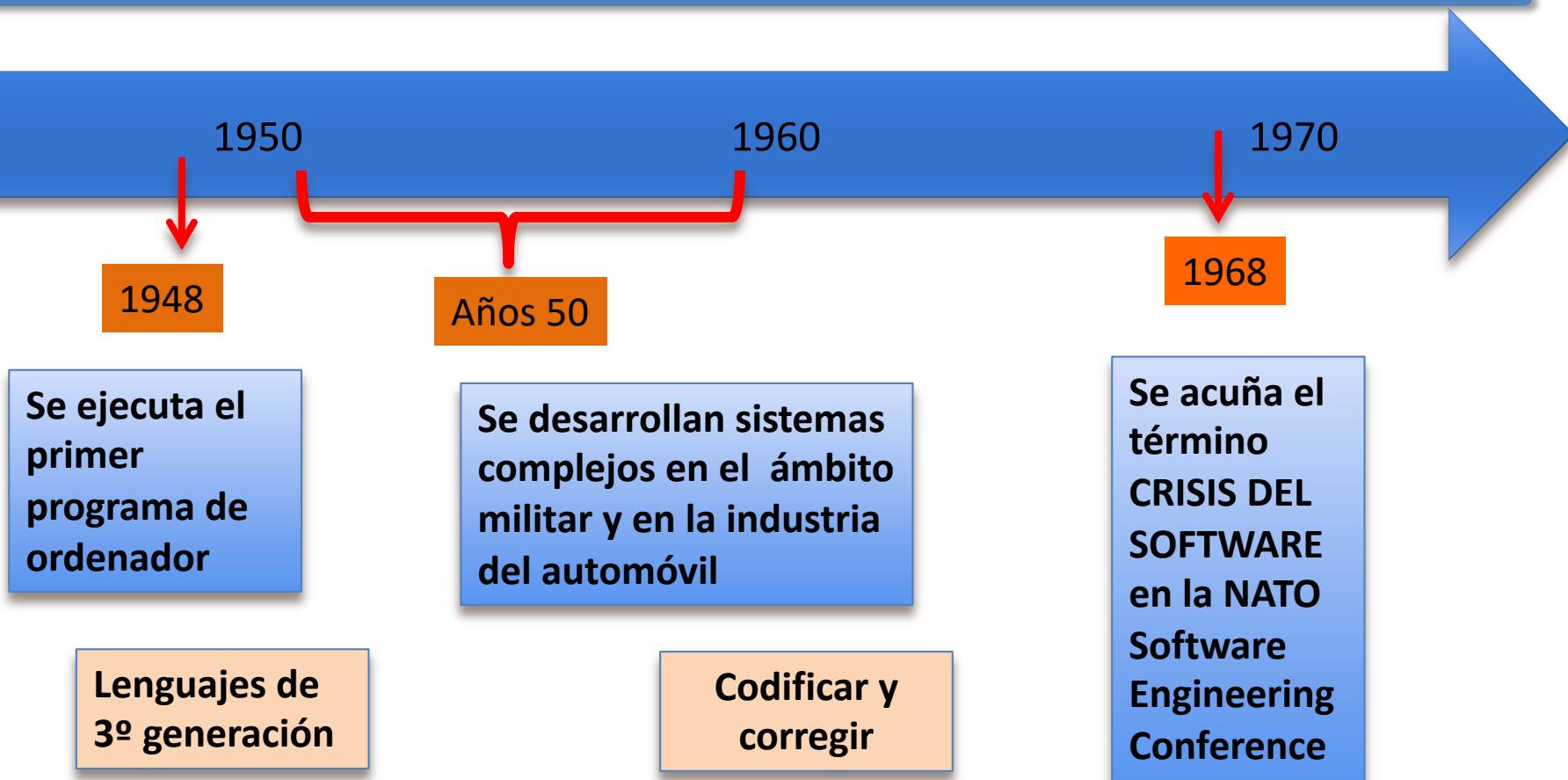


## Sources:

Standish group study reported at XP2002 by Jim Johnson, Chairman



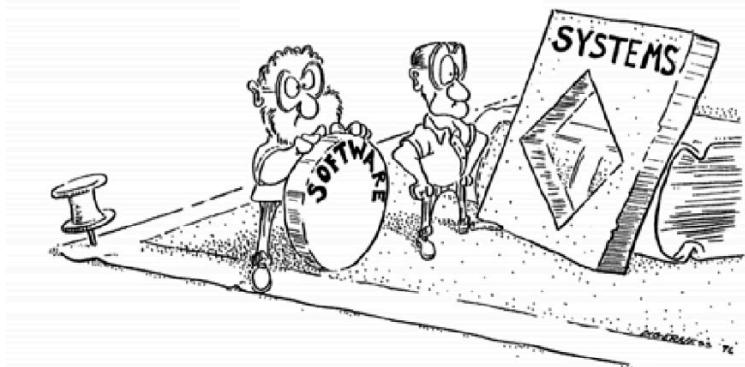
# Historia del desarrollo del software





# La crisis del software

- ✗ El software es ineficiente.
- ✗ El software no satisface los requisitos.
- ✗ Los proyectos sobrepasan las estimaciones de tiempo y recursos.
- ✗ Los proyectos se vuelven inmanejables y el software imposible de mantener.

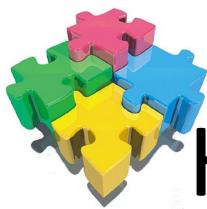




# La crisis del software: Soluciones

- ✗ Gestión Predictiva de proyectos.
- ✗ Producción basada en procesos.
- ✗ Ingeniería del Software.





# Historia del desarrollo del software

1970

1980

1990

2000

2010





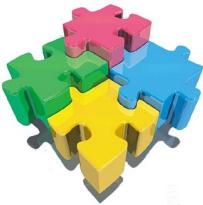
# Gestión Predictiva de Proyectos

---

- Se considera que un proyecto de desarrollo de Software ha sido un éxito si se consigue la **finalidad prevista**, con el **presupuesto** y en la **fechas** que previamente se habían estimado.

Gestión de proyectos “**predictiva**”.

- Partir de la descripción detallada de lo que hay que hacer.
- Realizar un plan detallado del proyecto (estimación).
- Supervisar y coordinar la ejecución para evitar desviaciones del plan (seguimiento y medición).



# Procesos de desarrollo de software

- × **Proceso de desarrollo:** Define el marco de trabajo para la construcción racional del software (métodos, heurísticas, principios metodológicos y herramientas).





# El proceso de desarrollo de software

---

- ✖ Los modelos de ciclos de vida del software dividen dicho desarrollo en unas etapas.
- ✖ Distintos modelos: Etapas consideradas, posición relativa de las mismas y las tareas a realizar en cada una.
  - ✖ **Secuenciales**: Hasta que no se acaba totalmente una etapa no comienza la siguiente.
  - ✖ **No secuenciales**: El desarrollo se considera un proceso evolutivo en el que se parte de un software con funcionalidad mínima y se desarrolla progresivamente.



# Etapas principales

- × **Planificación:** Estimar el tiempo y los costes de desarrollo del software.
- × **Especificación de requisitos:** Análisis del problema a resolver. Documento en el que se dice qué debe hacer el sistema software.
- × **Diseño:** Búsqueda de la solución. Descripción de los componentes, sus relaciones y funciones que dan solución al problema.
- × **Implementación:** Traducción del diseño a un lenguaje de programación entendible por una máquina.
- × **Control de la calidad:** Debe realizarse durante todo el proceso de desarrollo. Revisiones de todo lo que se va obteniendo junto con la prueba del código.
- × **Mantenimiento o evolución:** Reparar fallos en el sistema cuando sean descubiertos o adaptar el software a los nuevos entornos.



# La Ingeniería del Software

---

- ✖ Se necesita establecer y usar principios de ingeniería orientados a obtener software de manera económica, que sea fiable y funcione eficientemente sobre máquinas reales.
- ✖ La aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software; es decir, la aplicación de ingeniería al software. *IEEE Standard Glossary of Software Engineering Terminology*
- ✖ Proceso de aseguramiento de la calidad del proceso y del producto.

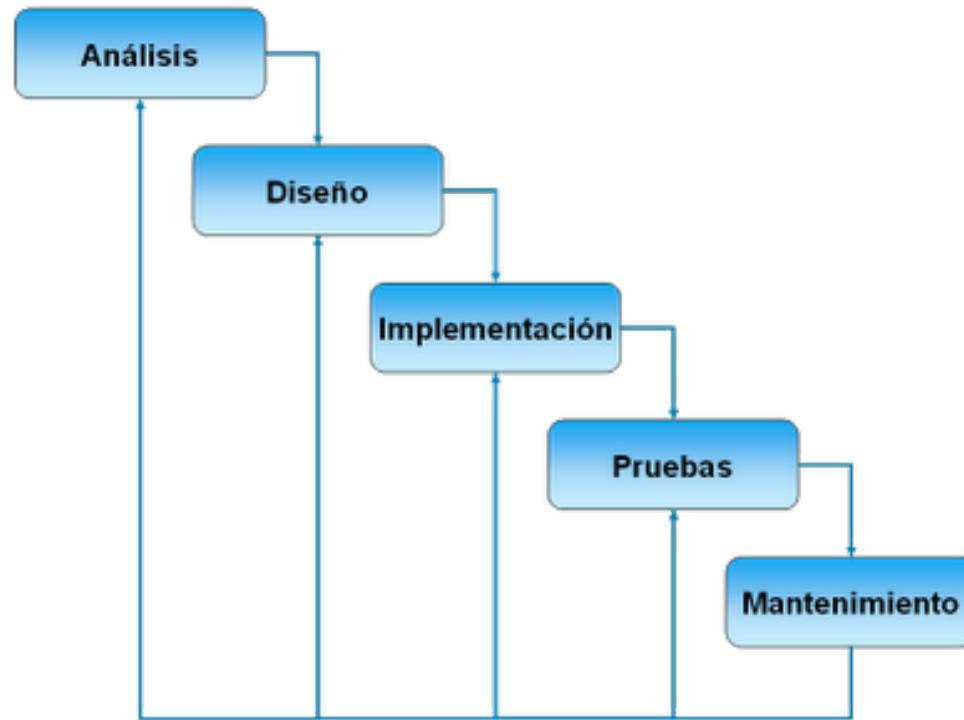


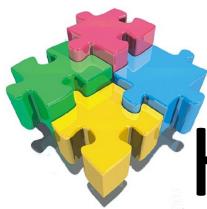
# Historia del desarrollo del software

1970 1980 1984 1986 1990 1991 1992 1994 1995 1998 2001 2005



**Modelo en cascada**

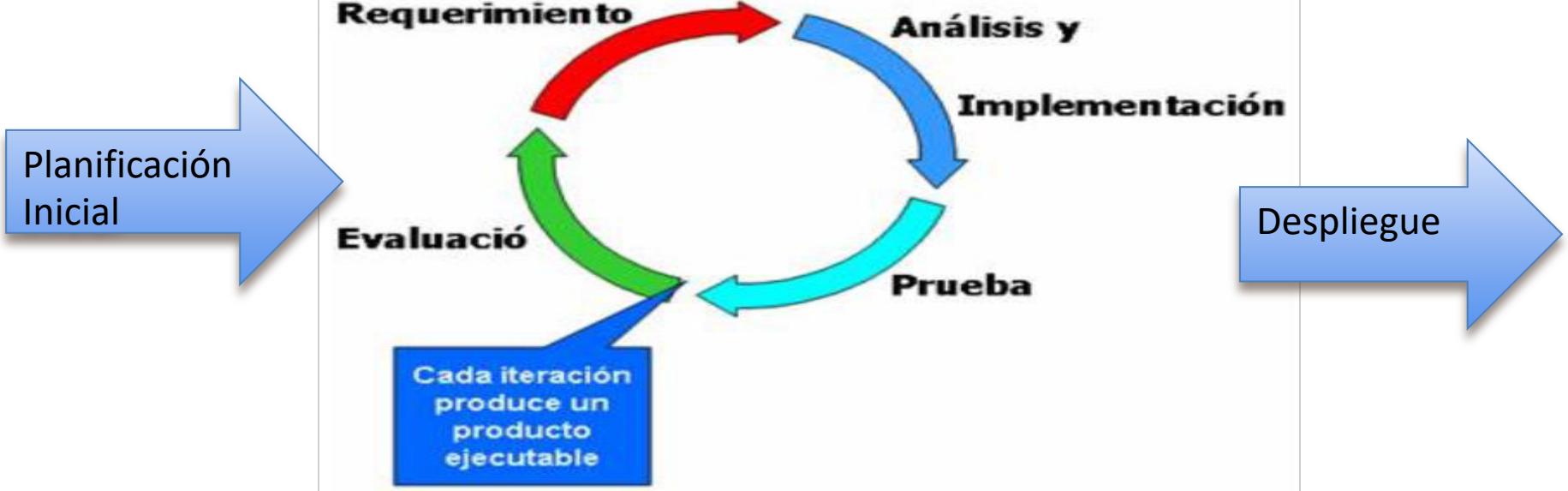




# Historia del desarrollo del software

1970 1980 1984 1986 1990 1991 1992 1994 1995 1998 2001 2005

Modelos iterativos e incrementales



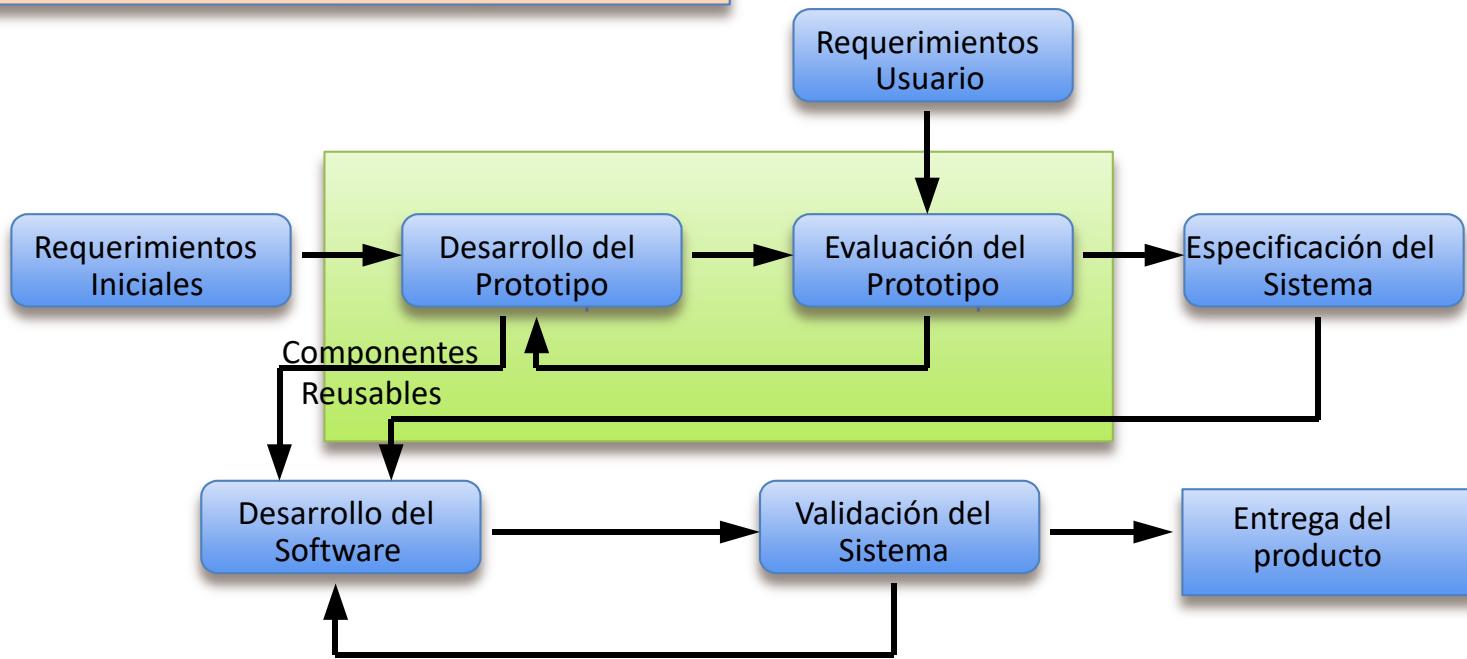


# Historia del desarrollo del software

1970 1980 1984 1986 1990 1991 1992 1994 1995 1998 2001 2005



## Modelo basado en prototipos



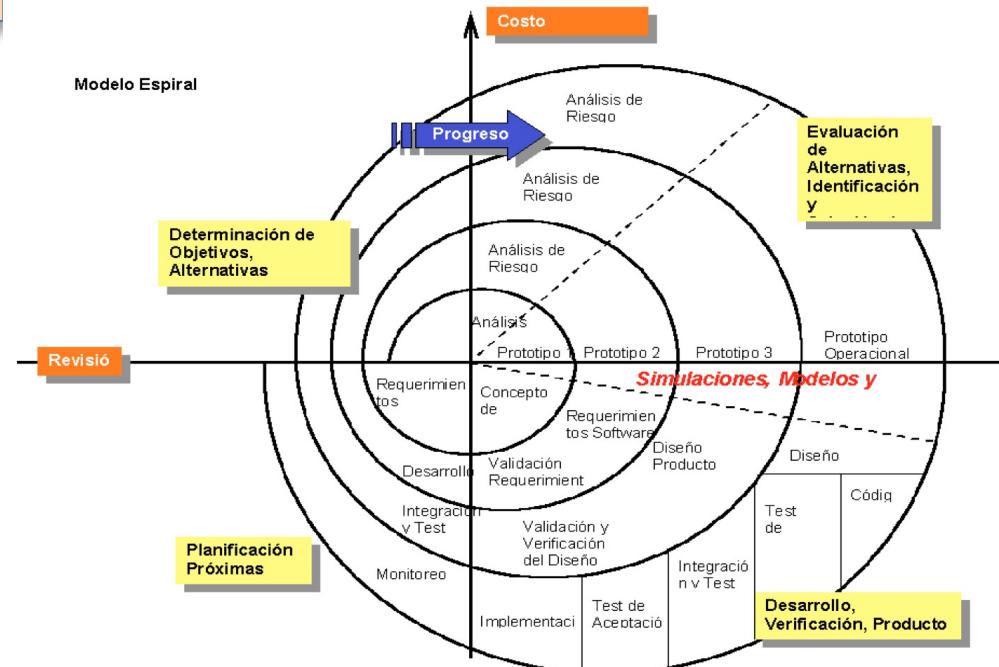


# Historia del desarrollo del software

1970 1980 1984 1986 1990 1991 1992 1994 1995 1998 2001 2005



## Modelo en espiral de Boehm



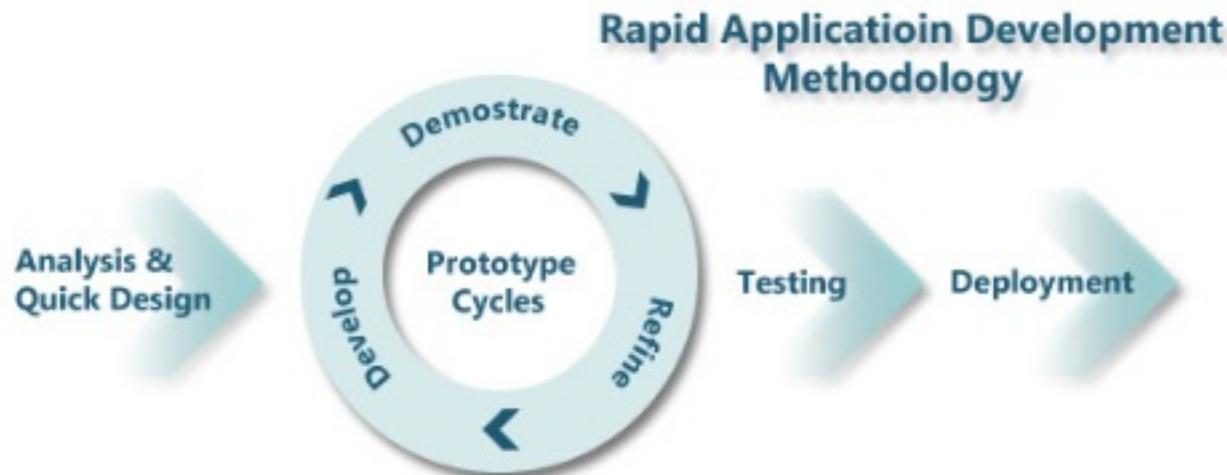


# Historia del desarrollo del software

1970 1980 1984 1986 1990 1991 1992 1994 1995 1998 2001 2005



**Desarrollo rápido de aplicaciones (RAD)**



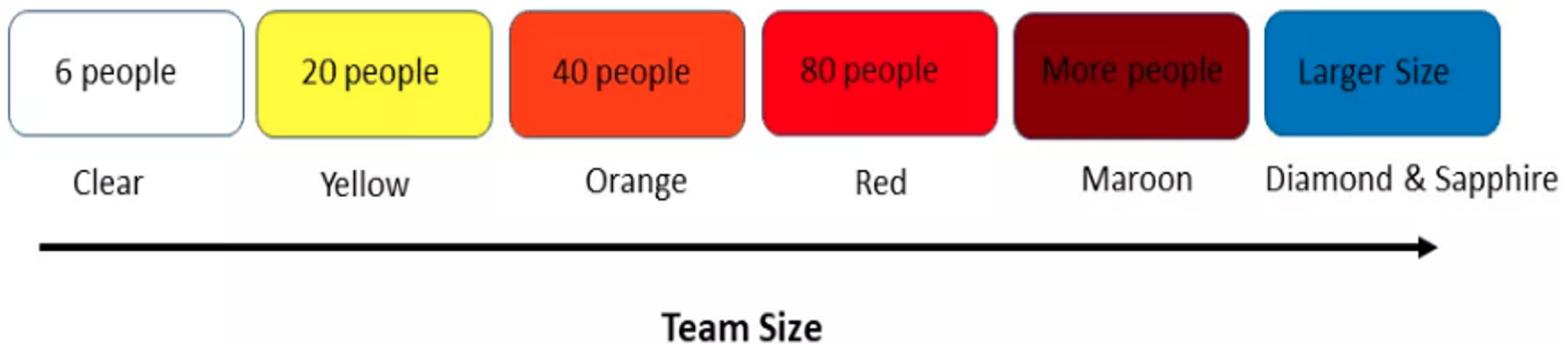


# Historia del desarrollo del software

1970 1980 1984 1986 1990 1991 1992 1994 1995 1998 2001 2005



**Crystal**



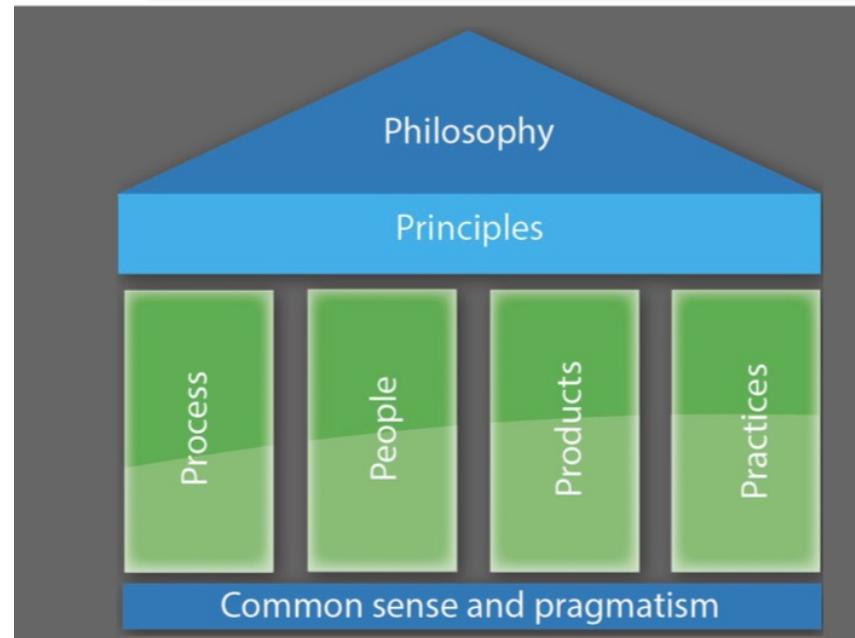


# Historia del desarrollo del software

1970 1980 1984 1986 1990 1991 1992 1994 1995 1998 2001 2005



**Dynamic Systems Development Method (DSDM)**



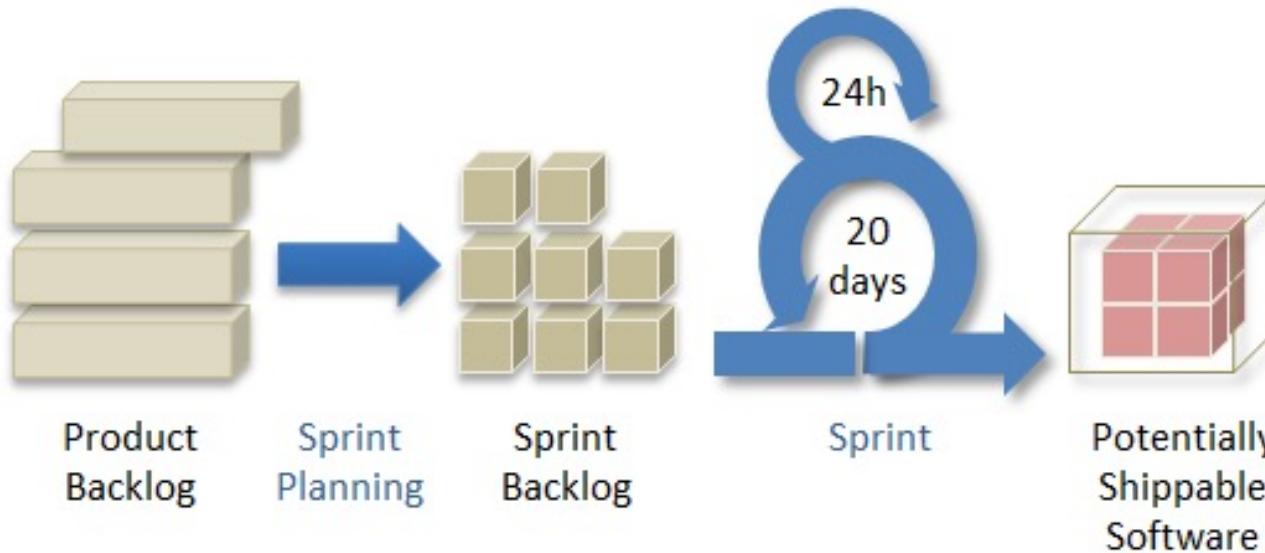


# Historia del desarrollo del software

1970 1980 1984 1986 1990 1991 1992 1994 1995 1998 2001 2005



Scrum



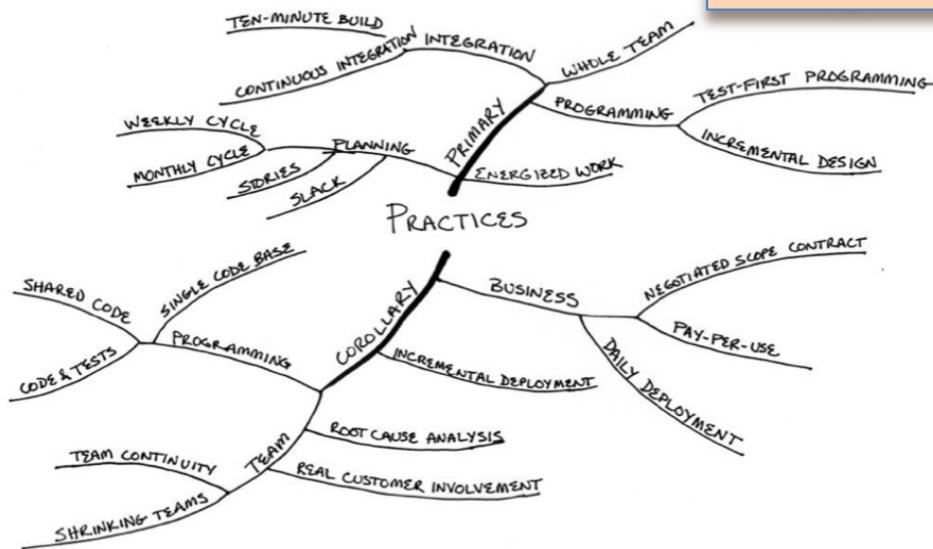


# Historia del desarrollo del software

1970 1980 1984 1986 1990 1991 1992 1994 1995 1998 2001 2005



## Programación Extrema (XP)



Kent Beck, *Extreme Programming Explained*, Segunda edición, 2005

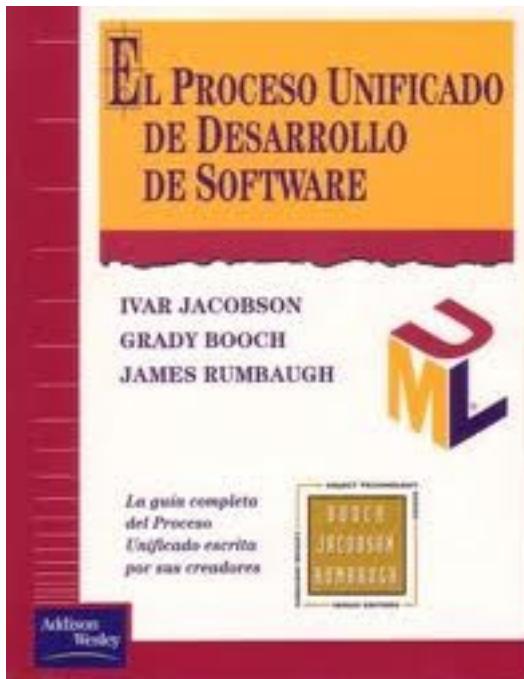


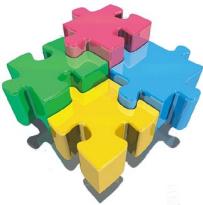
# Historia del desarrollo del software

1970 1980 1984 1986 1990 1991 1992 1994 1995 1998 2001 2005



Proceso unificado (RUP)





# Proceso Unificado. RUP

- ✖ Publicado en 1999 por Ivar Jacobson, Grady Booch y James Rumbaugh.
- ✖ Marco de desarrollo de software adaptable caracterizado por:
  - Iterativo e incremental.
  - Dirigido por los casos de uso.
  - Centrado en la arquitectura.
  - Orientado a los riesgos.
- ✖ Uso intensivo de UML.



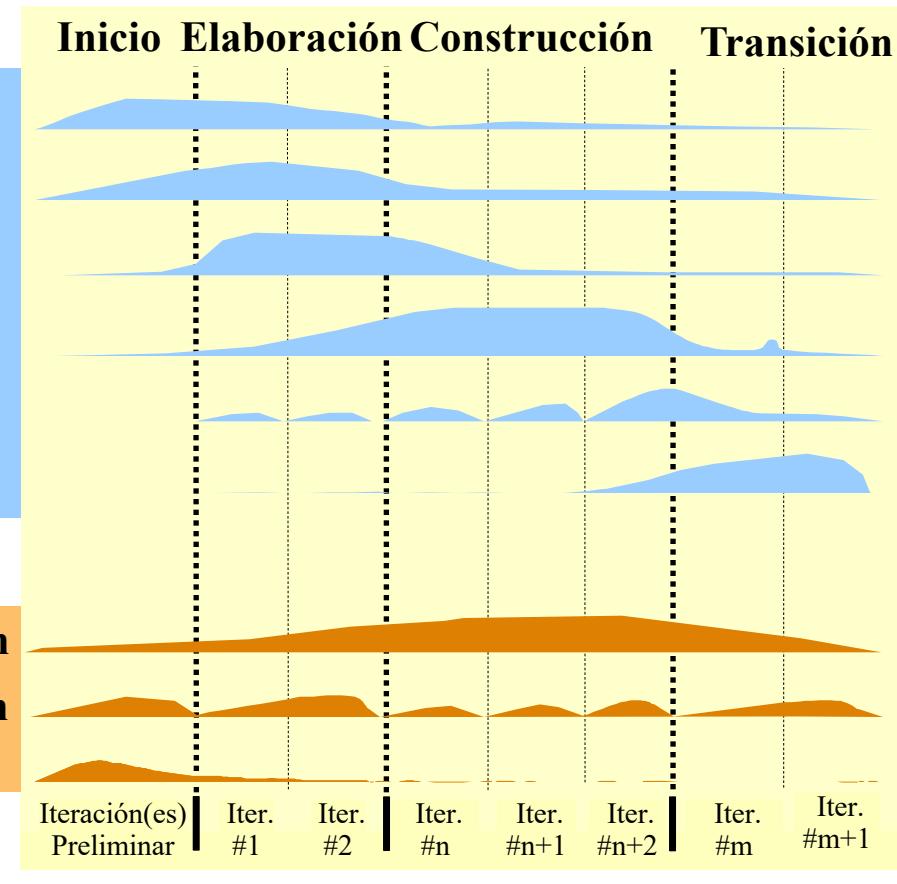
# PU. Iteraciones y Fases

## Flujos de Trabajo de Procesos

**Modelado de Negocios**  
**Requisitos**  
**Análisis y Diseño**  
**Implementación**  
**Prueba**  
**Desarrollo**

## Flujos de Trabajo de Gestión

**Admin. Configuración**  
**Administración**  
**Ambiente**



## Iteraciones



# Historia del desarrollo del software

1970 1980 1984 1986 1990 1991 1992 1994 1995 1998 2001 2005



**Manifiesto ágil**



# Principios del Manifiesto Ágil

**Manifesto Ágil**

Estamos comprometidos a promover el desarrollo de software que ayude a las personas.

**Individuos y Colaboración**

Responde a:

Esto es, a:

Kent Beck, Mike Bealeman, Arie van Vliet, Alistair Cockburn, Ward Cunningham, Martin Fowler

**Principios del Manifiesto Ágil**

*Seguimos estos principios:*

Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.

Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.

Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.

Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.

Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.

El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus

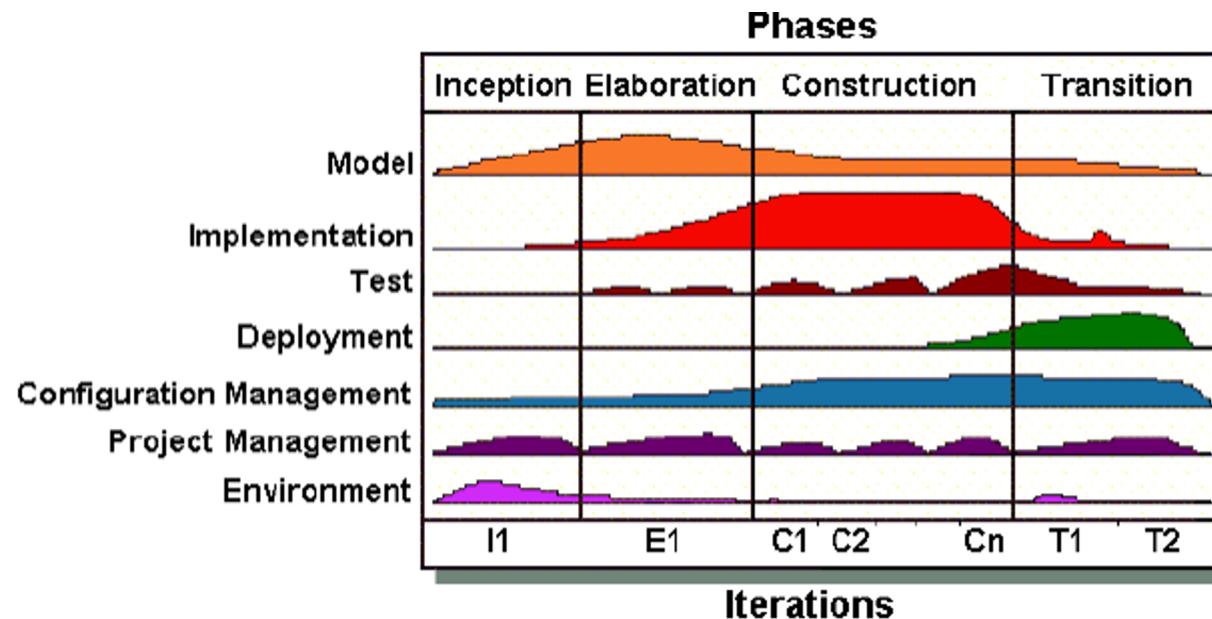
<https://agilemanifesto.org/iso/es/manifesto.html>



# Historia del desarrollo del software

1970 1980 1984 1986 1990 1991 1992 1994 1995 1998 2001 2005

Proceso Unificado Ágil (AUP)





# Calidad en el software

---

- ✖ Objetivo: Desarrollar software de calidad, mejorando procesos de desarrollo usados y el resultado final.
- ✖ Software que no funciona:
  - ✖ Solo cumple parcialmente lo que necesita el usuario, con errores y omisiones.
  - ✖ El usuario necesita un fuerte esfuerzo de adaptación al software.
  - ✖ Los usuarios no están satisfechos con el uso del software.
  - ✖ El software es lento y poco eficiente.
  - ✖ El software es poco modificable, poco entendible y con problemas técnicos.

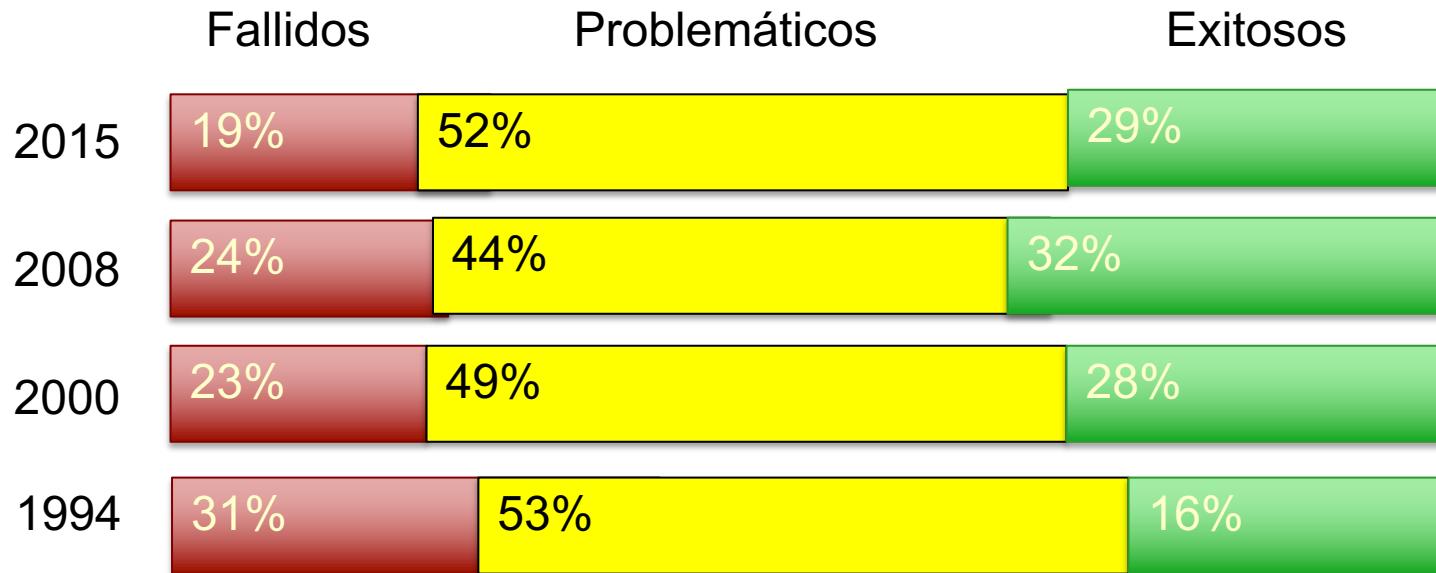


# ISO 25000. Calidad del software





# El éxito es “raro”



Se pasan en costos: 45%  
Se pasan en tiempo: 63%  
No llegan a la funcionalidad: 67%

The Standish Group, “Extreme Chaos”, 1994-2015

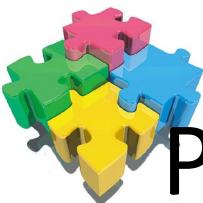


# El éxito es “raro”

	2011	2012	2013	2014	2015	2017
<b>FALLIDOS</b>	22 %	17 %	19 %	17 %	19 %	19%
<b>PROBLEMÁTICOS</b>	49 %	56 %	50 %	55 %	52 %	48%
<b>EXITOSOS</b>	29 %	27 %	31 %	28 %	29 %	33%

The Standish Group, “Extreme Chaos”, 2011 a 2017

¿Qué indican estos datos?



# Project Management Institute (PMI)

---

- Causas del fracaso de un proyecto:
  - Problemas en el alcance.
  - Comunicación pobre.
  - Falta de implicación de los implicados (stakeholders).
  - Apoyo inadecuado del patrocinador del proyecto.

La inadecuada gestión de los requisitos es la culpable del fracaso de la mitad de los proyectos. (Según el 47 % de los encuestados)



# Project Management Institute (PMI)

---

- **Puntos de mejora:**

- **Personas:** Asignar los recursos necesarios y suficientes, así como reconocer y desarrollar las habilidades necesarias para desarrollar sus funciones.
- **Procesos:** Estandarizar y formalizar los procesos.
- **Cultura:** Crear un sentimiento en todos los niveles de la utilización de las prácticas elegidas.



# Tamaño del proyecto



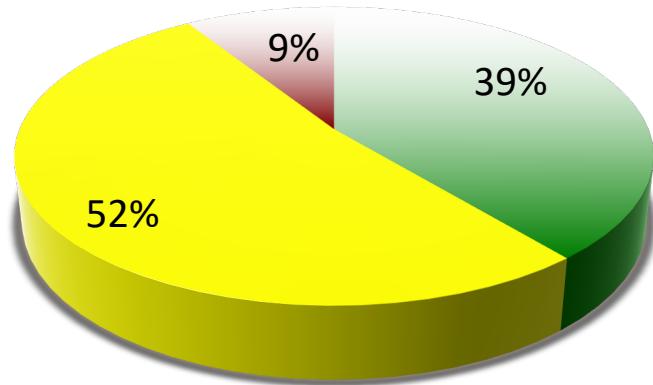
The Standish Group, "Extreme Chaos", Comparativa del éxito de la proyectos según el tamaño, de 2011 a 2015



# Metodología seguida para el desarrollo

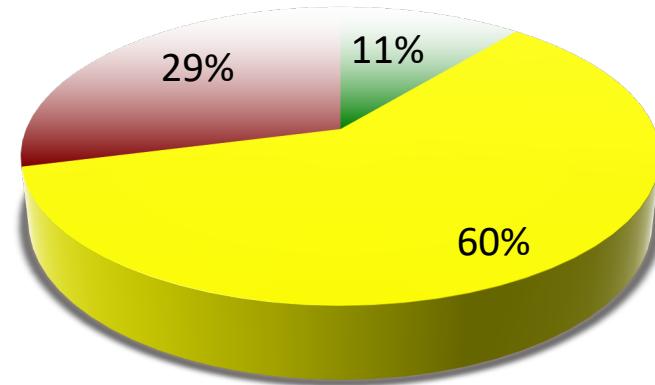
## Ágil

■ Exitosos ■ Problemáticos ■ Fallidos



## Cascada

■ Exitosos ■ Problemáticos ■ Fallidos



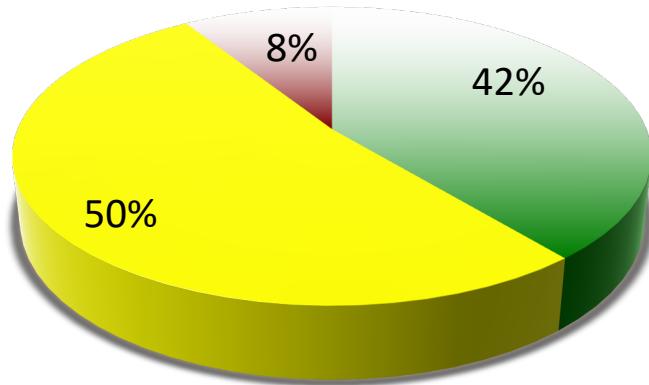
The Standish Group, "Extreme Chaos", **Comparativa del éxito de los proyectos según la metodología seguida, del 2011 a 2015**



# Metodología seguida para el desarrollo

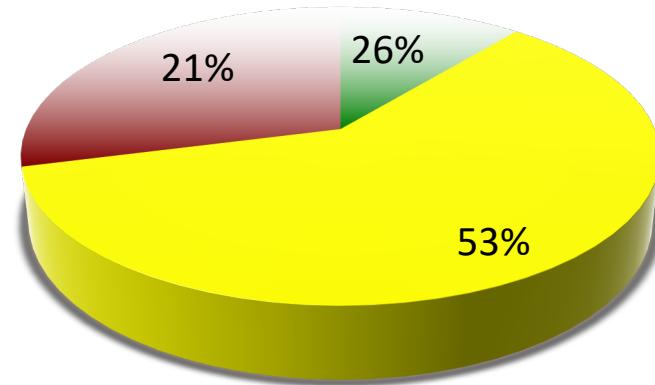
## Ágil

■ Exitosos ■ Problemáticos ■ Fallidos



## Cascada

■ Exitosos ■ Problemáticos ■ Fallidos

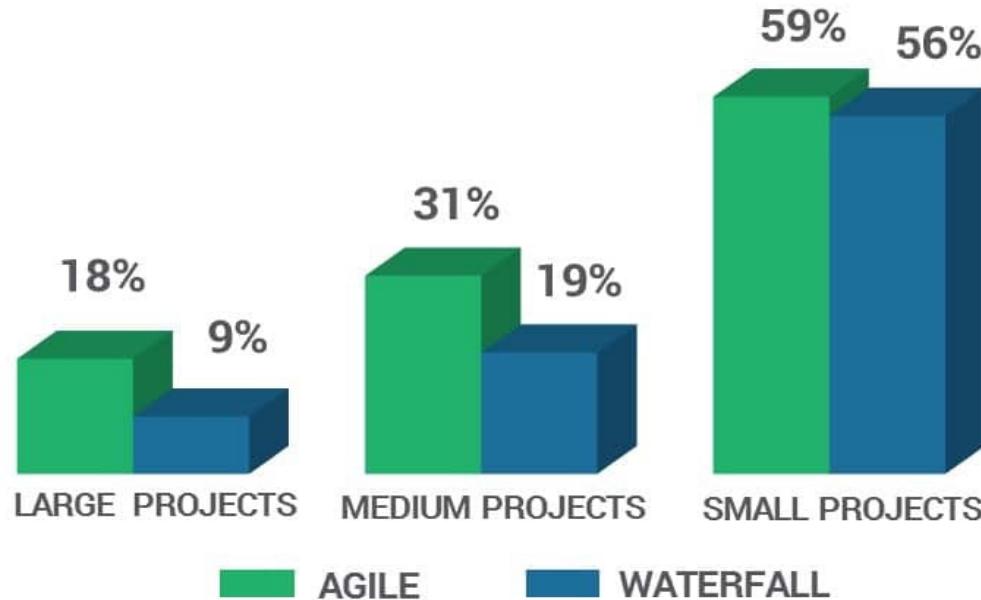


The Standish Group, "Extreme Chaos", **Comparativa del éxito de los proyectos según la metodología seguida, del 2013 a 2017**



# Metodología seguida para el desarrollo

## PROJECT SUCCESS RATES BY PROJECT SIZE **AGILE VS WATERFALL** FOR LARGE PROJECTS, AGILE APPROACHES ARE 2X MORE LIKELY TO SUCCEED



Source: Standish Group, Chaos Studies 2013-2017

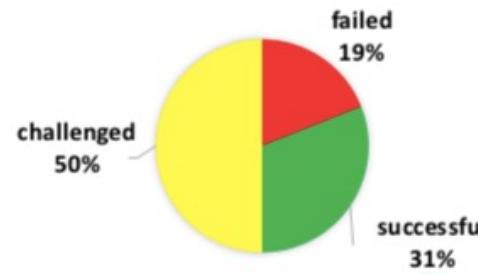
[WWW.VITALITYCHICAGO.COM](http://WWW.VITALITYCHICAGO.COM)



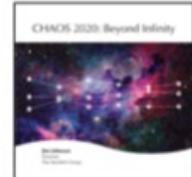
# Informe CHAOS 2020

## Project Success Quick Reference Card

Based on CHAOS 2020: Beyond Infinity Overview. January'2021, QRC by Henny Portman



Modern measurement  
(software projects)



Good Sponsor, Good Team, and Good Place are the only things we need to improve and build on to improve project performance.



The **Good Place** is where the sponsor and team work to create the product. It's made up of the people who support both sponsor and team. These people can be helpful or destructive. It's imperative that the organization work to improve their skills if a project is to succeed. This area is the hardest to mitigate, since each project is touched by so many people. Principles for a Good Place are:

- The Decision Latency Principle
- The Emotional Maturity Principle
- The Communication Principle
- The User Involvement Principle
- The Five Deadly Sins Principle
- The Negotiation Principle
- The Competency Principle
- The Optimization Principle
- The Rapid Execution Principle
- The Enterprise Architecture Principle



The **Good Team** is the project's workhorse. They do the heavy lifting. The sponsor breathes life into the project, but the team takes that breath and uses it to create a viable product that the organization can use and from which it derives value. Since we recommend small teams, this is the second easiest area to improve. Principles for a Good Team are:

- The Influential Principle
- The Mindfulness Principle
- The Five Deadly Sins Principle
- The Problem-Solver Principle
- The Communication Principle
- The Acceptance Principle
- The Respectfulness Principle
- The Confrontationist Principle
- The Civility Principle
- The Driven Principle



The **Good Sponsor** is the soul of the project. The sponsor breathes life into a project, and without the sponsor there is no project. Improving the skills of the project sponsor is the number-one factor of success – and also the easiest to improve upon, since each project has only one. Principles for a Good Sponsor are:

- The Decision Latency principle
- The Vision Principle
- The Work Smart Principle
- The Daydream Principle
- The Influence Principle
- The Passionate Principle
- The People Principle
- The Tension Principle
- The Torque Principle
- The Progress Principle



Successful project Resolution by Good Place Maturity Level:

highly mature	50%
mature	34%
moderately mature	23%
not mature	23%

Successful project Resolution by Good Team Maturity Level:

highly mature	66%
mature	46%
moderately mature	21%
not mature	1%

Successful project Resolution by Good Sponsor Maturity Level:

highly mature	67%
mature	33%
moderately mature	21%
not mature	18%



# Problemas del desarrollo de software

---

- Metodologías “predictivas”.
  - Requisitos **previos** al desarrollo.
  - ¿Qué desea el cliente?. **Contrato**.
  - Hay que **evitar cambios** en los requisitos.
  - Los cambios son **responsabilidad del cliente**.
  - **Planificamos** alcance + coste + tiempos.
  - **Gestionamos** el proyecto.



# Problemas del desarrollo de software

---

- Detección de errores en etapas finales del proyecto.
- El cliente toma decisiones importantes en fases iniciales del proyecto.
- Proyectos difíciles de gestionar.
- Reducción de calidad.
- Comunicación pobre.
- Insatisfacción del cliente, proyectos inflexibles.
- Dificultad para implantar metodologías de desarrollo.



# Entorno de desarrollo actual

---

- Requisitos cambiantes.
- Tiempo de desarrollo corto.
- Vida de las aplicaciones corta.
- Clientes y usuarios más exigentes.
- Medidas de la calidad y la productividad.



# Entorno de desarrollo actual

---

- Es necesario desarrollar y construir el producto a la vez que se investiga y descubren los requisitos y hacerlo con una capacidad de adaptarse a los cambios dictados por el entorno (**gestión de proyectos adaptable**).
- El cliente conoce la “**visión** de su producto, pero por la novedad, el valor de la innovación, la velocidad de la tecnología y del negocio no puede detallar como será el **producto final**.

No queremos desarrollar el producto perfecto.