Introducción a la Ingeniería del Software

Ingeniería del Proceso Software

Mª José Suárez-Cabal
Javier Tuya (http://www.di.uniovi.es/~tuya/)
Grupo de Investigación en Ingeniería del Software
http://giis.uniovi.es

Curso 2022-2023

Contenido

- Efectos de problemas causados por software
- Problemática general y Características del Software
- Ingeniería del Software y Definiciones
- Procesos, Técnicas y Metodologías
- Resumen

Aeropuerto de Asturias
2007-08-28

Costes económicos
Nike Disaster

- $400 millones de inversión, sistema para automatizar la gestión, producción y venta. Cadena de suministro. (2001)
  - Objetivo: tiempo de planificación y producción de calzado de 1 mes a 1 semana
  - Resultado: $100 millones de caída en ventas. Caída de acciones del 20%.
  - Problema: El sistema enviaba órdenes de producción duplicadas. Exceso de stock en calzado de baja rentabilidad y escasez en el de alta. Para entregar en tiempo se debieron realizar envíos por avión ($8 por par) en vez de por barco (75 cents por par).
- Lecciones
  - Software sólido, pero en otros negocios. Necesidad de adaptación, alto riesgo
  - No gestionar/realizar un proyecto piloto y/o paralelo
  - No se verificaron suficientemente los resultados
- Referencias:
  - http://www.computerworld.com.au/article/32990/swoosh_stumbles
  - http://www.cio.com/article/32334/Nike_Rebounds_How_and_Why_Nike_Recovered_from_Its_Supply_Chain_Disaster

Seguridad: London Ambulance Dispatching System

- 1.2 millones (libras). Gestión de llamadas y despacho de ambulancias automatizado (90's)
  - Objetivo: Eliminación de radio y teléfono a las estaciones
  - Resultado: Estimación de 20 muertos debido a los retrasos en la llegada de ambulancias
  - Problema: Las llamadas se ponían en espera a veces hasta 30 minutos. A veces llamadas perdidas y duplicadas. La sala de control estaba sobrecargada, problemas de uso del nuevo sistema
- Lecciones:
  - Implantación y puesta en marcha apresurada.
  - Inexperiencia de proveedor con un sistema tan grande.
  - No había un sistema de backup. Ausencia de paralelo.
  - Interfaz de usuario muy moderno y llamativo, pero inadecuado cuando hay alto volumen de llamadas.
- Referencias
  - http://catless.ncl.ac.uk/Risks/13.88.html#subj1.1
  - http://catless.ncl.ac.uk/Risks/14.37.html#subj11.1

Error programación/prueba: Ariane 5 (June 1996)

- 10 años, $7 billones. Agencia Espacial Europea (ESA)
  - Objetivo: capacidad de poner dos satélites de tres toneladas en órbita. Evolución del Ariane 4
  - Resultado: Junio 1996, a los 39 segundos del lanzamiento se desintegra, perdiendo dos caros satélites (sin seguro)
  - Problema: A los 36,7 segundos el software de guiado inercial produce overflow al convertir 64 bit (float) a 16 bit (int). Caída del sistema de backup, a los 0.05 segundos caída del sistema principal. El sistema de control principal recibe datos de diagnóstico que interpreta como datos de vuelo (ha cambiado la posición del cohete). Fuerte reacción para corregir la trayectoria. Desintegración por la fuerza aerodinámica. Autodestrucción.
- Causas:
  - El sistema de guiado inercial es el mismo que en Arianne 4, no se prevé protección contra este overflow porque estas comprobaciones se hacen en otro lugar y no se había producido nunca.
  - La velocidad horizontal de Ariane 5 es cinco veces superior y causa el overflow.
  - Las especificaciones y pruebas realizadas no tuvieron en cuenta esto.
- Referencias
  - http://catless.ncl.ac.uk/Risks/18.29.html#subj16.1
  - http://www.di.unito.it/~damiani/ariane5rep.html

California sues SAP over failed payroll software project (2013)

- The project dates back to 2005 and has cost taxpayers more than $250 million so far.
- "After three years, and paying SAP approximately $50 million to integrate its own software into a new payroll and benefits system for the state of California, all the [state controller's office] has to show for its investment is a system that could not get the payroll right even once over an eight-month period for a pilot group of only 1,500 employees," the lawsuit states.
- SAP is reviewing the state's complaint: "We will say -- as we have said consistently over the course of this engagement -- that SAP software is not the culprit here, nor was SAP's performance in implementing the software," Kendzie said. "Our software works exactly as it is designed and we have successfully implemented the software with other clients.

IT World (22/11/2013)

De proximidad…

18/1/2013

- "Los alcaldes achacan a un error informático que 39 concejos no presentasen sus balances"

1/9/2015

- "El colapso informático en Educación deja sin destino a 2.900 profesores interinos"

29/10/2015

- "Un fallo informático del Ministerio impide el pago de 29 millones a ganaderos"

31/01/2022

- Un fallo informático impide notificar órdenes de protección en Oviedo

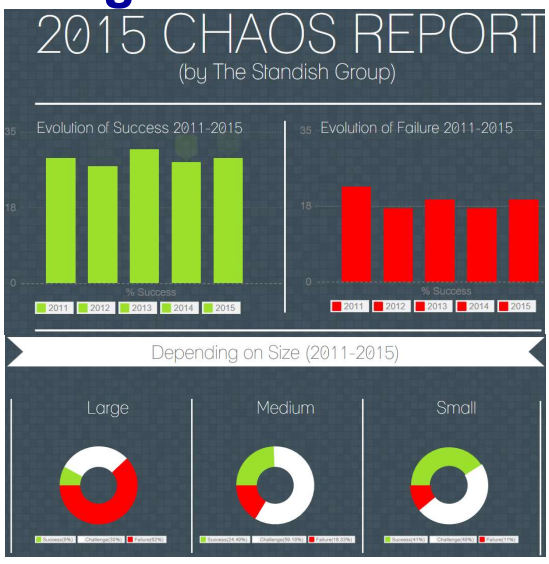
15/05/2022

- "Un fallo informático bloquea durante horas la red sanitaria, centros educativos y juzgados"

2/07/2022

- "Un fallo informático impide durante horas a las farmacias dispensar recetas electrónicas"

Algunas cifras



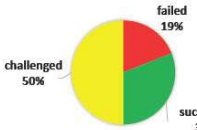
"Large projects are 10 times more likely to fail outright"

2015 Chaos Report, by the Standish Group

Algunas cifras

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:

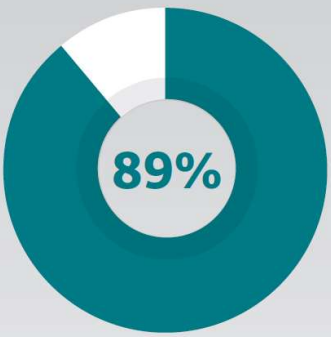
- The Decision Latency principle
- The Vision Principle
- The Work Smart Principle
- The Daydream Principle
- The Influence Principle
- The Passionate Principle
- The People Principle
- The Tension Principle
- The Torque Principle
- The Progress Principle

Algunas cifras

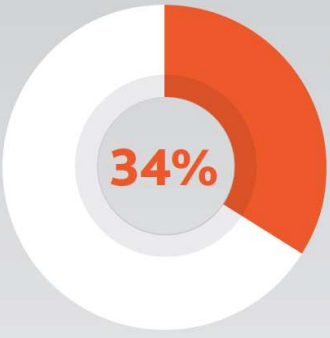
"More critical is the money that continues to be wasted when projects aren't managed well. We see US$122 million wasted for every US$1 billion invested due to poor project performance, a 12 percent increase over last year"

ON TARGET

Having proven project, program and portfolio management practices in place makes a dramatic difference in project performance.



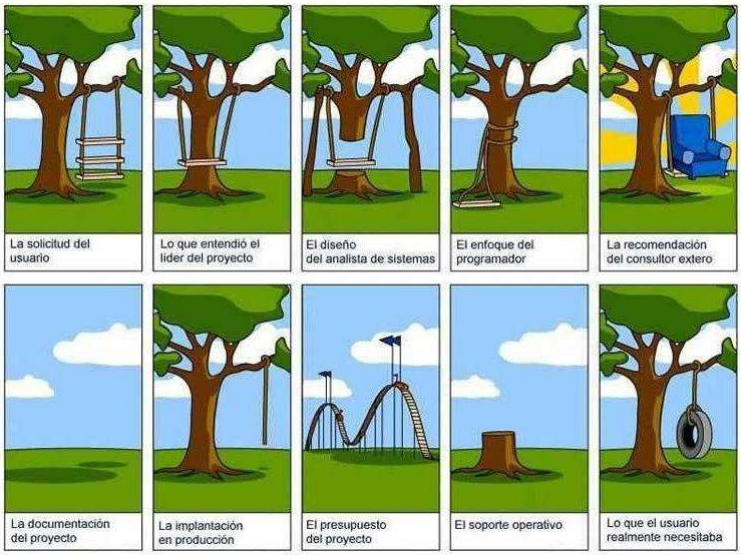
of projects at high-performing organizations meet original goals and business intent. (High performers complete 80 percent or more of projects on time, on budget and meeting original goals.)



of projects at low-performing organizations meet original goals and business intent. (Low performers complete 60% or fewer of projects on time, on budget and meeting original goals.)

PMI PULSE of the Profession & 8th Global Project Management Survey - Infographic. The Project Management Institute, 2016

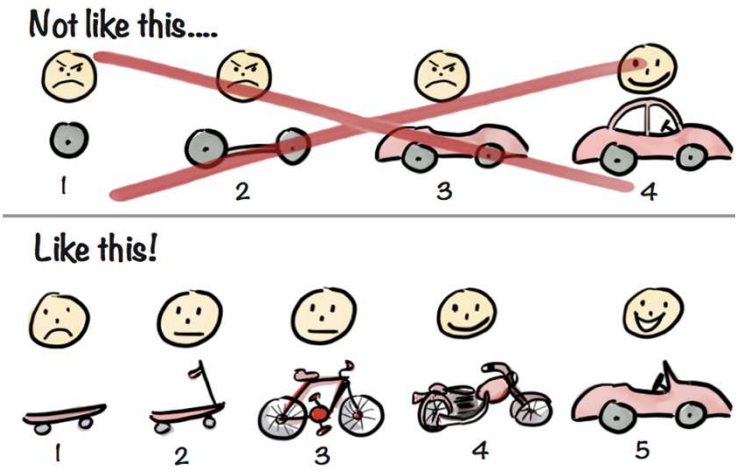
Problemática General



Naturaleza

- Software:
  - Intangible. Difícil de entender el esfuerzo de desarrollo
  - Fácil de reproducir y flexibilidad. Coste centrado en la ingeniería
  - Acumulación de deuda técnica: coste implícito del retrabajo adicional causado por “no hacerlo bien en su momento”
  - Altamente dependiente de la tecnología subyacente (muy cambiante)
- Proceso. Dificultades:
  - Definir los requisitos (cliente). Complejidad oculta al usuario
  - Probar: Problemas de calidad difíciles de detectar
  - Modificar: Comprender realmente el cambio y su impacto
  - Gestionar: Dependencia de las personas y equipos. Difícil de automatizar
- Conclusiones:
  - Hay que aprender a producirlo (“engineer software”)
  - El producto es para mejorar la vida/solucionar problemas del usuario/ cliente: Entender el problema y comunicarse de forma efectiva
  - Ser flexible y asegurar calidad, costes y plazos razonables

Desarrollo Incremental



Henrik Kniberg

Ingeniería del Software

- Término acuñado en 1968. Conf. Ing. Software de la OTAN
  - Definición: The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software (IEEE STD 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, IEEE Computer Society, 1990).
  - Diferencia con Computer Science: CS es teoría y fundamentos, SE se encarga de los aspectos prácticos del desarrollo e implantación.
- Otras definiciones de autores:
  - I. Sommerville. “Software engineering is concerned with the theories, methods and tools for developing, managing and evolving software products”
  - B.W. Boehm. “The practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate and maintain them’”
  - F.L. Bauer. “The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines”

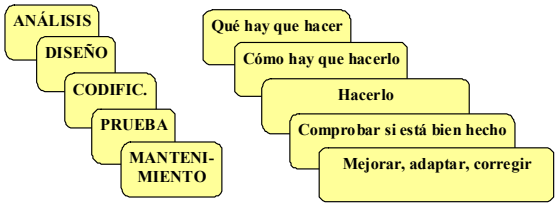
Ingeniería del Software

- Definiciones ISO (International Organization for Standardization)
  - The systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software (ISO/IEC 2382-1:1993 Information technology - Vocabulary - Part 1).
  - The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software (ISO/IEC 24765:2009 Systems and software engineering vocabulary) (=IEEE)
- Diccionario de Términos:
  - ISO/IEEE SEVOCAB: Software and Systems Engineering Vocabulary: http://pascal.computer.org/sev_display/

Procesos, técnicas y herramientas

- Proceso de Software: Conjunto de actividades y productos obtenidos durante el desarrollo de un sistema software, independientemente de su tamaño o complejidad
- Técnica: Un procedimiento sistemático empleado por un recurso humano para realizar una actividad que produce un producto, resultado o servicio, y que puede emplear una o más herramientas (PMBOK)
- Herramienta: Producto empleado para automatizar procesos o métodos
- Método/Práctica: Un determinado enfoque (“approach”) para solucionar un determinado problema. Incluye:
  - Notación (gráfica y/o textual)
  - Técnicas que han de ser utilizadas para solucionar el problema
- Metodología: Conjunto de prácticas, técnicas, procedimientos y reglas usados por quienes trabajan en una determinada disciplina (PMBOK).
  - Define la secuencia en la que se aplican los métodos, productos o documentos requeridos, controles de calidad, etc (SEI)

Procesos genéricos



- Ciclo de vida: Cómo se ordenan y coordinan los diferentes procesos
  - A framework of processes and activities concerned with the life cycle that may be organized into stages, which also acts as a common reference for communication and understanding (ISO/IEC 12207:2008 Systems and software engineering - Software life cycle processes)
  - Fases (etapas) que se secuencian, solapan o iteran (según un plan)
  - Diferentes enfoques:
    - Secuencial o cascada (waterfall). Variantes: modular, incremental
    - Iterativo/Incremental: Prototipado, Espiral, Ágiles

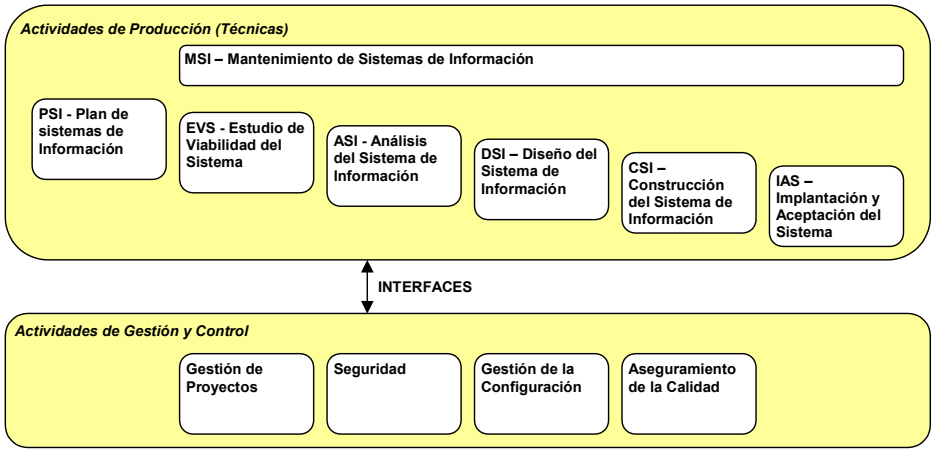
Procesos principales (ISO/IEC 12207:2008)

- Procesos a nivel de Sistema
  - Stakeholder Requirements Definition
  - System Requirements Analysis
  - System Architectural Design
  - Implementation
  - System Integration
  - System Qualification Testing
  - Software Installation
  - Software Acceptance
  - Software Operation
  - Software Maintenance
  - Software Disposal
- Procesos de Implementación de Software
  - Requirements Analysis
  - Software Architectural Design
  - Software Detailed Design
  - Software Construction
  - Software Integration
  - Software Qualification Testing

Metodologías Típicas

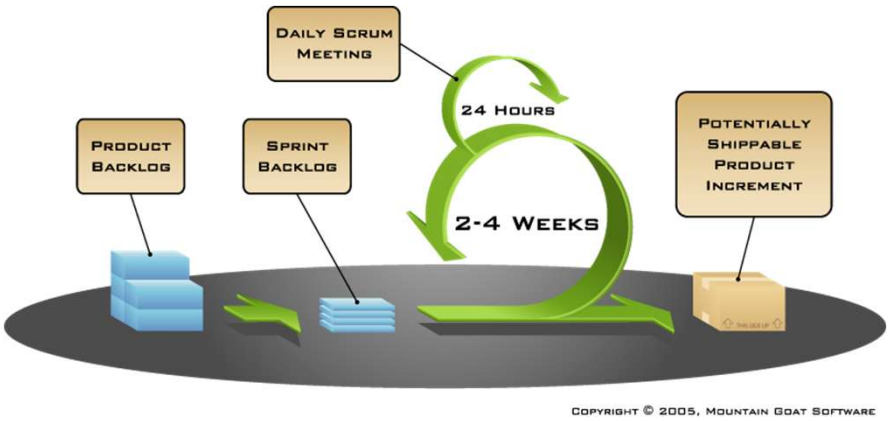
- **Métrica Versión 3** – Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información
- Proceso Unificado (Rational Unified Process)
- MEDEPA – Metodología de Desarrollo de Proyectos del Principado de Asturias
- Ágiles:
  - XP – Extreme Programming
  - **SCRUM**

Métrica Versión 3



http://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3#.UiBjQjZShiA

SCRUM



http://www.scrum.org/

Orden de presentación

- Incremental, Metodologías + Técnicas
  - Metodología + Técnicas básicas de análisis:
    1. SCRUM (Historias de Usuario)
    3. Métrica V3 (Casos de Uso)
  - Otras técnicas y proceso global de:
    2. SCRUM
    4. Métrica V3
    5. Proceso Unificado
  1. Otras metodologías, técnicas y estándares

Resumen

- El software es un producto de ingeniería. Debe construirse como tal
  - Existen particularidades que hacen más difícil el proceso
  - Suele infravalorarse la dificultad de desarrollo, implantación y mantenimiento
  - Debe responder a las necesidades de los usuarios
  - Es preciso entender el problema, comunicar e involucrar al usuario
  - No siempre es desarrollo
  - No olvidar los procesos de gestión (Calidad, Planificación)
- Conocer la problemática y los procesos de Ingeniería del Software es importante también para:
  - El proveedor (Técnico, gestor y comercial)
  - El cliente final

Bibliografía de consulta general

- Sommerville I (2011). Ingeniería de Software (9ª edición). Pearson Addison Wesley 2011.
- Pressman RS (2010). Ingeniería del Software: Un enfoque práctico (7ª edición). Mc Graw Hill 2010.
- Pfleeger SL (2002). Ingeniería del Software: Teoría y Práctica. Pearson Prentice Hall 2002.
- Deemer P, Benefield G, Larman C, Vodde B (2012). The Scrum Primer (Version 2, Traducción de Ángel Medinilla). https://scrumprimer.org (accedido Septiembre 2022).
- Piattini MG, Calvo-Manzano JA, Cervera J, Fernández-Sanz L (2003). Análisis y diseño de aplicaciones informáticas de gestión: Una perspectiva de la ingeniería del software. Ra-Ma 2003.
- The Risks Digest: Forum On Risks To The Public In Computers And Related Systems (http://catless.ncl.ac.uk/Risks, accedido Septiembre 2022)

Bibliografía específica

- Kennet S. Rubin. Essential Scrum. Addison-Wesley Signature Series, 2013
- Chris Sims, Hillary Louise Johnson. The Elements of Scrum. Dymaxicon. Version 1.0.1
- MAP (2001). Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información: Guía de Referencia. Ministerio para las Administraciones Públicas. http://administracionelectronica.gob.es (accedido Septiembre 2022).

- Cockburn A (2001). Writing Effective Use Cases. Addison-Wesley 2001.
- Larman C (2003). UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado (2ª edición). Prentice Hall 2003.
- Myers GJ, Sandler C, Badgett T (2011). The Art of Software Testing (3rd Edition). Wiley 2011.