Ingeniería de Software: Introducción

Edgar Sarmiento Calisaya
Universidad Nacional de San Agustín de Arequipa – Perú
Departamento Académico de Ingeniería de Sistemas – DAISI
Escuela Profesional de Ciencia de la Computación – EPCC
esarmientoca@unsa.edu.pe

Breve Biografía

- **Formación**:
  - Graduación en Ingeniería de Sistemas, UNSA – Arequipa
  - Maestría en Informática, UFRJ – Brasil
  - Doctorado en Informática, PUC-Rio - Brasil

- **Profesional**:
  - Ingeniero de software
  - Arquitecto de Software
  - Consultor de Software

- **Academico**:
  - Docente Auxiliar - Investigador, UNSA - Arequipa, 2018 - …

Ingeniería de Software

Desarrollo Basado en Plataformas: Web & Mobile

Lenguajes de Programación

Base de Datos

Ingeniería de Software I:

1. Ingeniería de Requisitos
2. Diseño de Software
3. Construcción de Software

Ingeniería de Software II:

1. Herramientas y Entornos de IS
2. V&V
3. Evolución

Ingeniería de Software III:

1. Gestión de Proyectos
2. Gestión de Calidad
3. Gestión de Pruebas

Desarrollo de Software Empresarial: ERP, CRM, BPM, SOA, EBS, BI

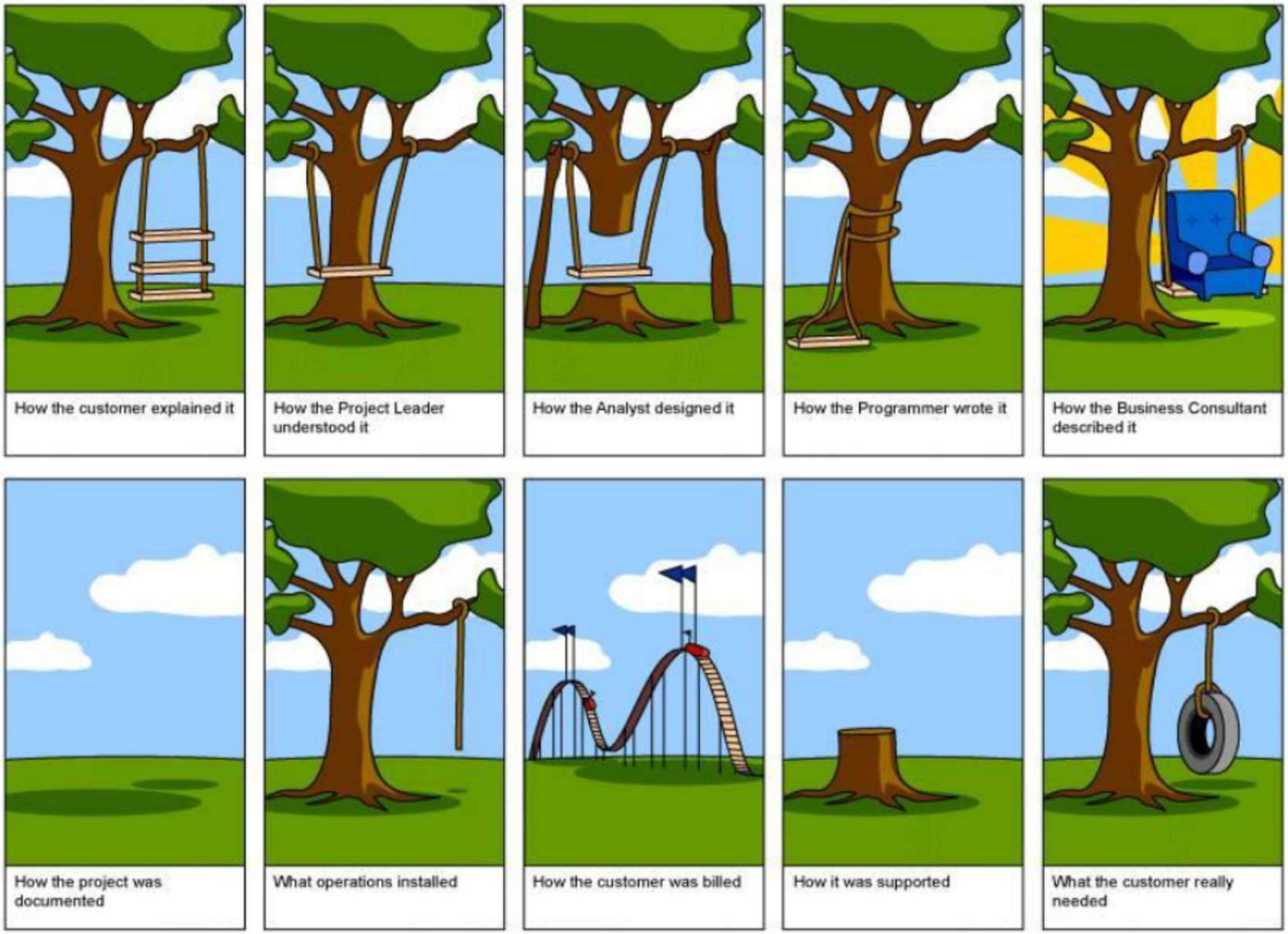
Tópicos en Ingeniería de Software

Áreas de Interés

Agenda

Temas:

1. Realidad del Software
2. Ingeniería de Software
3. Proceso de Software
4. Conclusión

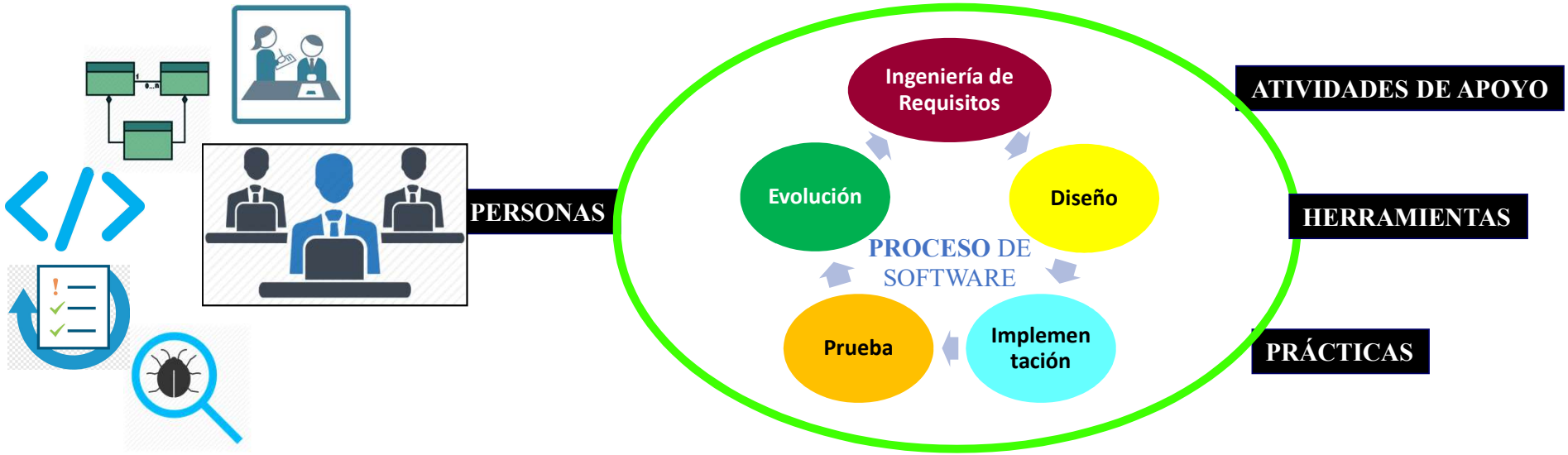

How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Realidad del Software

- **Desarrollar software es una actividad compleja** por naturaleza.
- Una de las razones de esta afirmación es que **no existe una única solución para cada escenario de desarrollo**.
- Además, nos ocupamos **todo el tiempo con personas**, lo que hace que el éxito del proyecto esté bastante relacionado con:
  - la competencia del equipo y la forma en que trabajan, y
  - para dificultar aún más, **muchas veces no hacemos uso de un proceso bien definido** para apoyar las actividades del proyecto.

Realidad del Software

- Se entiende por **proceso**, en este contexto, como un **conjunto de actividades bien definidas con los respectivos responsables de ejecución, herramientas de apoyo y artefactos producidos**. Es decir, se define cómo el equipo deberá trabajar para alcanzar el objetivo: desarrollar software con calidad dentro de plazos, costos y requisitos definidos.



Realidad del Software

- Standish CHAOS Summary Report – 2015:
  - 71% de los proyectos no alcanzan sus objetivos, y tienen como origen **Requisitos** de software imprecisos entre sus causas.

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

Realidad del Software

CHAOS RESOLUTION BY PROJECT SIZE

	SUCCESSFUL	CHALLENGED	FAILED
Grand	2%	7%	17%
Large	6%	17%	24%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%
TOTAL	**100%**	**100%**	**100%**

The resolution of all software projects by size from FY2011–2015 within the new CHAOS database.

.

57% de proyectos de tamaño medio fallaron

Standish CHAOS Summary Report – 2015

Realidad del Software

CHAOS RESOLUTION BY AGILE VERSUS WATERFALL

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

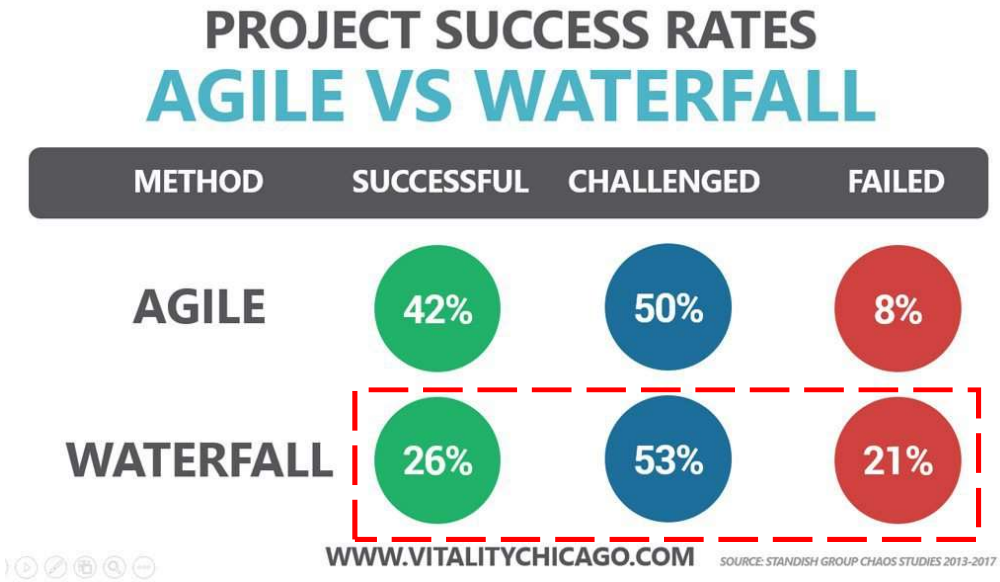
The resolution of all software projects from FY2011-2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000

Agile projects are statistically **3X** more likely to succeed, and 1/3 less likely to fail than waterfall projects

2015

Standish CHAOS Summary Report – 2015

Realidad del Software



https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/

Agile projects are statistically **2X** more likely to succeed, and 1/3 less likely to fail than waterfall projects

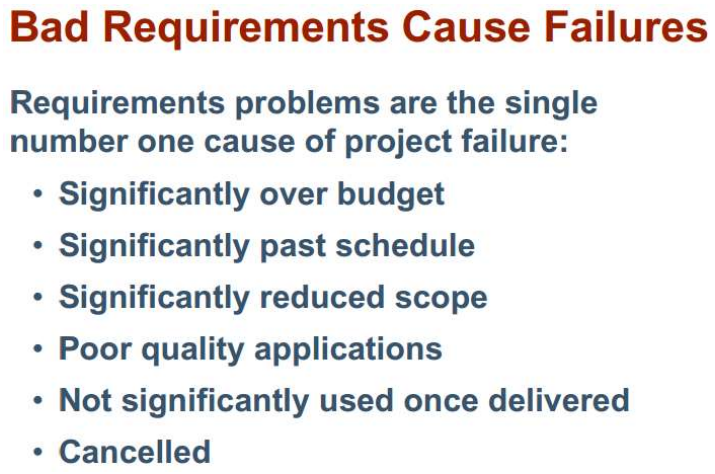
2018

Standish CHAOS Summary Report – 2018

Realidad del Software

- Project Management Institute – 2014:
  - 47% de los proyectos de software que fallaron tiene como origen Requisitos de Software imprecisos como su **causa principal**.

Firesmith Donald, 2003



Realidad del Software

Impact of Requirements Defects

Organization/Project	Overruns Attributed to Requirements Problems
NASA over two decades (Werner Gruhl)	70% of overrun amount
U.S. Census Bureau project 2009	80% cost overrun locked in solely due to poor requirements
Marine One Helicopter Program	83% cost overrun attributed by Lockheed to requirements problems
Schwaber, 2006; Weinberg, 1997; Nelson et al, 1999	"Requirements errors are the single greatest source of defects and quality problems"
Hofmann and Lehner, 2001	"Deficient requirements are the single biggest cause of software project failure."
Standish Group, The Chaos Report on 8300 IT projects	60.9% of an average 89% cost overrun

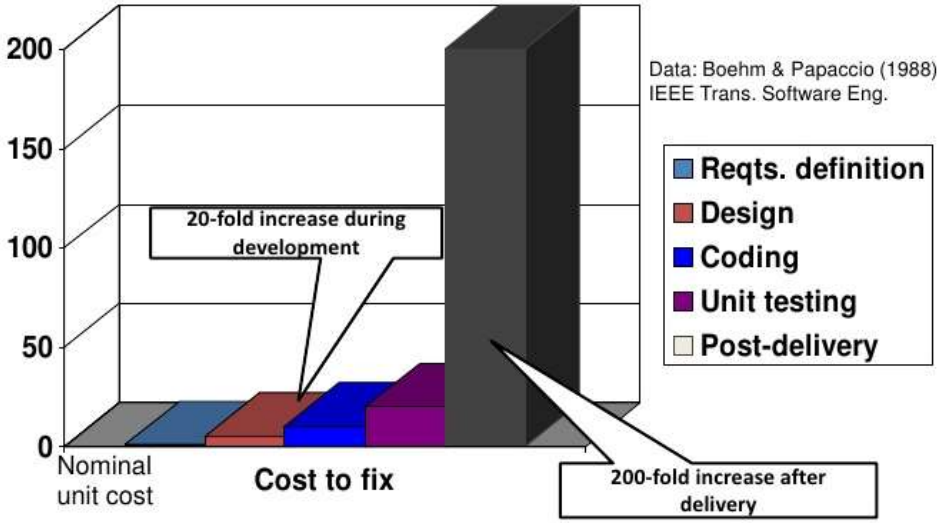
Halligan Robert, 2014 - Project Performance International (PPI)

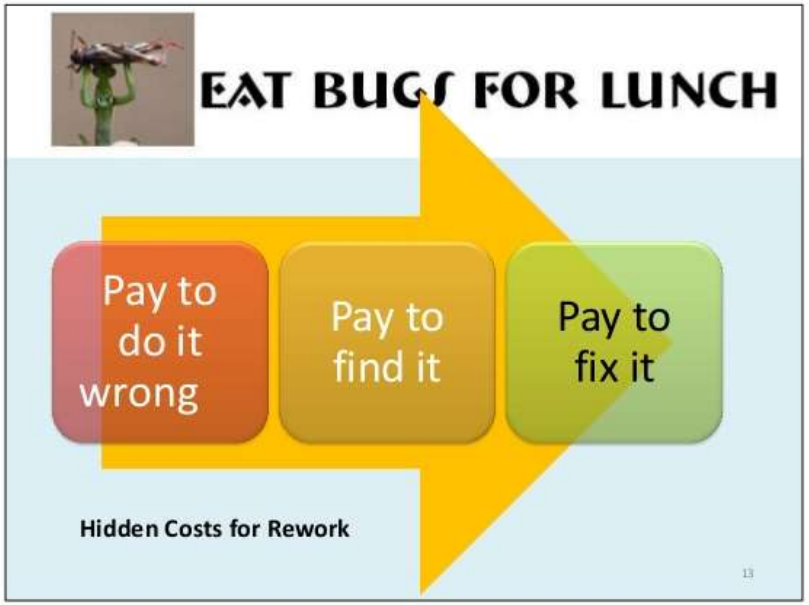
Realidad del Software

- ***HOY*** dedicamos nuestros esfuerzos a corregir lo que hicimos mal ***AYER – REWORK***

Arnd Von Staa, PUC-Rio.

Cost of Delay in Fixing Requirements Errors





Rebecca Staton-Reinstein, Irv Browntein, 2012

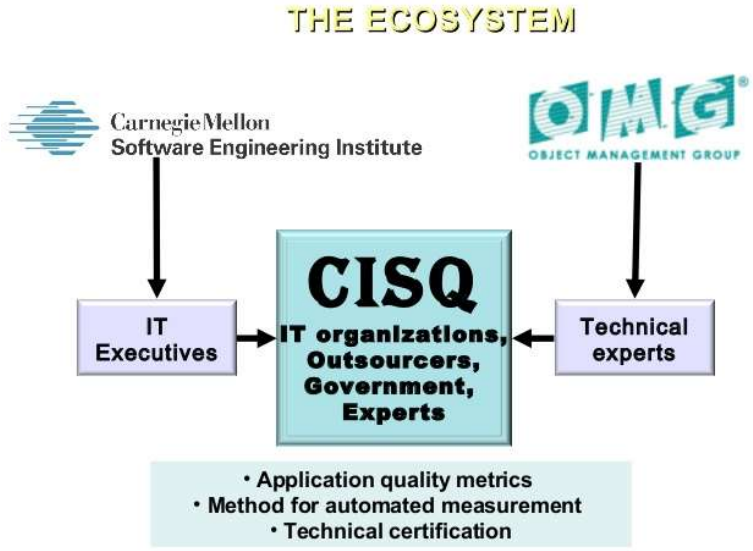
¿Qué hacer?

Artesanía → **Ingeniería**

Cambio cultural de los interesados!

Ingeniería de Software

- La **buena noticia** es que muchas empresas se están moviendo en el sentido de **definir detalladamente sus procesos** para apoyar sus actividades de **desarrollo**.
  - Productos y servicios de mejor calidad;
  - Internacionalización



Ingeniería de Software

- Sin embargo, el **escenario de desarrollo de software actual y el escenario ideal** de la ingeniería de software todavía están **distantes**.
- Varios factores contribuyen a ello, podemos citar dos:
  - El no uso de los fundamentos de la ingeniería de software para apoyar las actividades del desarrollo;
  - El mal uso de los fundamentos de la ingeniería de software para apoyar las actividades del desarrollo.

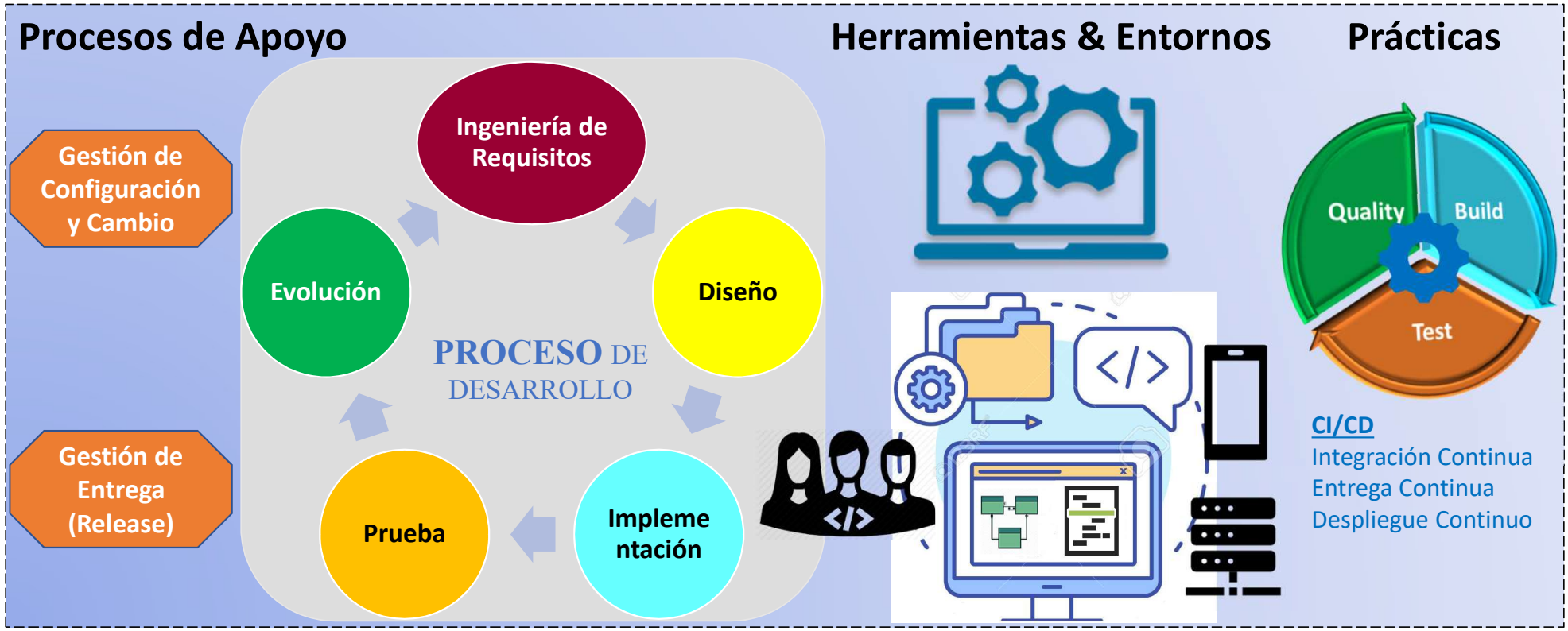
Ingeniería de Software

Ingeniería de Software (IS): La aplicación de un enfoque **sistemático**, **disciplinado** y **cuantificable** en el desarrollo, operación y mantenimiento de software [IEEE].

Ingeniería de Software

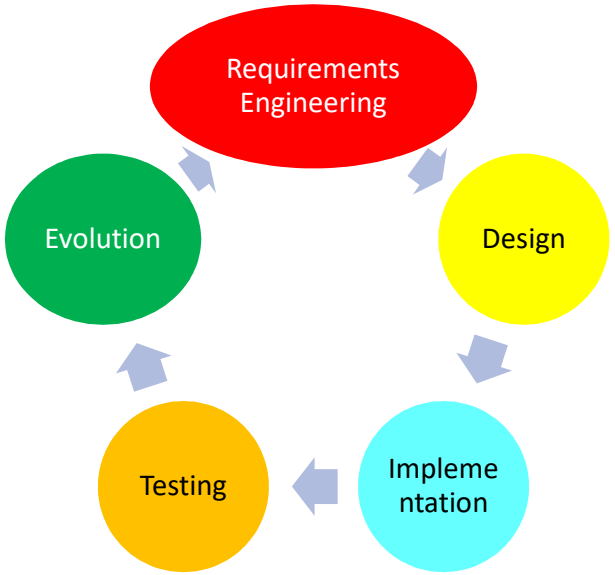
- **Ingeniería de software**: La aplicación de un enfoque sistemático, disciplinado y cuantificable en el desarrollo, operación y mantenimiento de software [IEEE].
  - **Sistemática** porque parte del principio de que existe un proceso de desarrollo definiendo las actividades que deberán ser ejecutadas.
  - **Disciplinado** porque parte del principio de que los procesos definidos serán seguidos.
  - **Cuantificable** porque se debe definir un conjunto de medidas a ser extraídas del proceso durante el desarrollo de forma que las tomas de decisión relacionadas al desarrollo del software (por ejemplo, mejora de proceso) se basen en datos reales, y no en “creencias" .
- Algunos de sus principales **objetivos** son:
  - **Calidad** de software;
  - **Productividad** en el desarrollo, operación y mantenimiento de software;
  - Permitir que los profesionales tengan **control** sobre el desarrollo de software dentro de **costos**, **plazos** y **niveles de calidad** deseados.

Ingeniería de Software



Proceso de Software

- Un **proceso de software** se puede definir como un conjunto de **actividades**, métodos, prácticas y **transformaciones** que las **personas** usan para desarrollar y mantener software y productos asociados (por ejemplo, planes de proyecto, documentos, código, casos de prueba y manuales de software). usuario) [IEEE-STD-610].



Modelos de Proceso de Software

- **Métodos**
  - **Procesos**
- **Técnicas**
- **Herramientas**

Eng. Software

debem incorporar una estratégia de desarrollo

- Las organizaciones dividen el desarrollo de proyectos de software en fases con el fin de asegurar un mejor control y vinculación con las operaciones actuales de la empresa;

- El conjunto de fases de un proyecto de software se conoce como:

Modelos de Proceso de Software

- **Métodos**
  - **Procesos**
- **Técnicas**
- **Herramientas**

Eng. Software

debem incorporar una estratégia de desarrollo

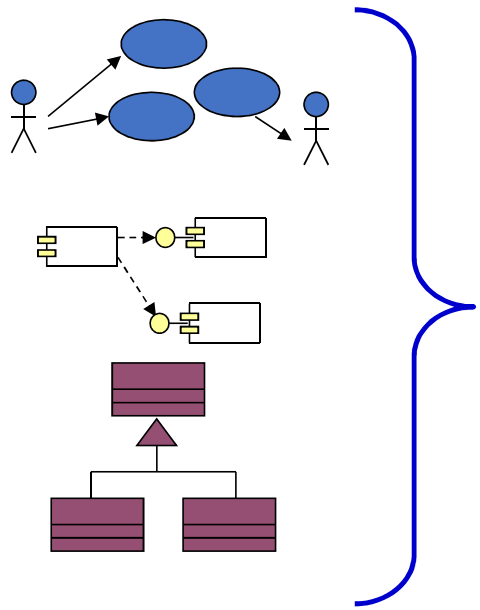
Modelo de Proceso
Paradigma de Ingeniería de Software
Ciclo de Vida
Modelo de Ciclo de Vida
Metodología

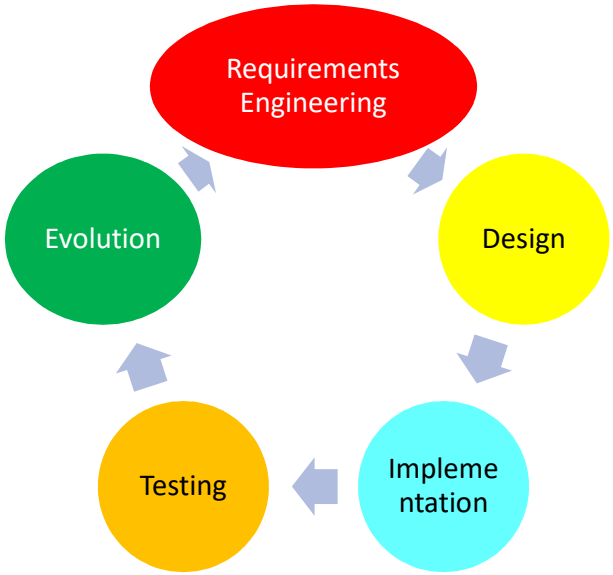
Modelos de Proceso de Software

- Es escogido con base en:
  - la naturaleza del diseño y la aplicación
  - en los métodos y herramientas a utilizar
  - en los controles y productos que necesitan ser entregados

Modelos de Proceso de Software

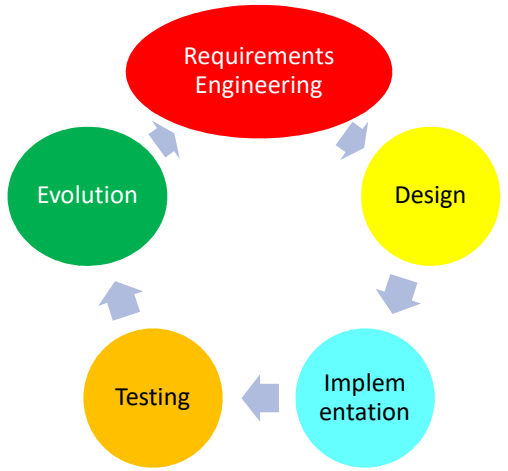
- Todo desarrollo de software puede ser caracterizado como un ciclo de **solución de problema**
- El ciclo de solución puede ser usado en diferentes niveles de resolución





Modelos de Proceso de Software

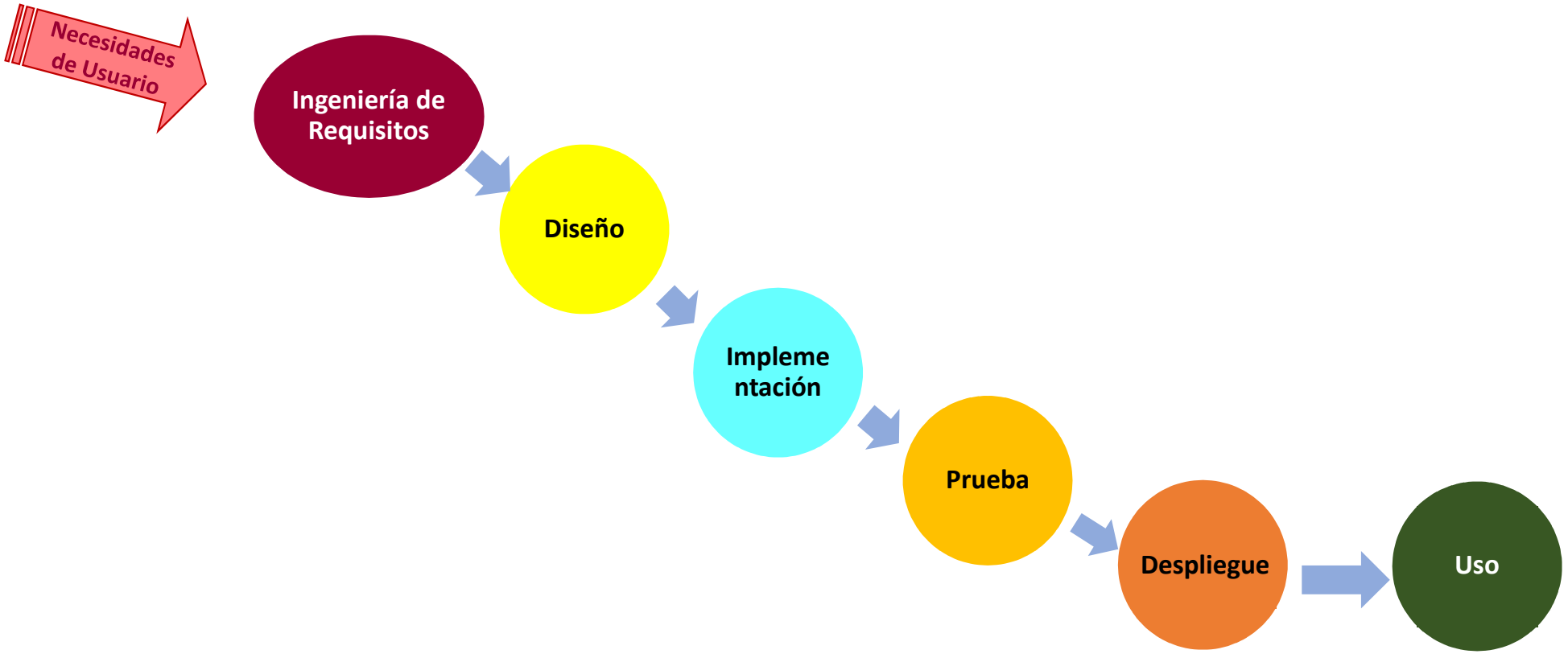
- Todas las demás actividades dependen del Proceso de desarrollo.
- Proyectos de software modernos utilizan algún tipo de metodología basada en **iteraciones** o **espiral** sobre uno en cascada.
- **Alguns Modelos de Processo:**
  - Waterfall
  - Iterative
  - Spiral
  - **Agile**
  - **Lean**
  - **DevOps**



https://www.roberthalf.com/blog/salaries-and-skills/6-basic-sdlc-methodologies-which-one-is-best

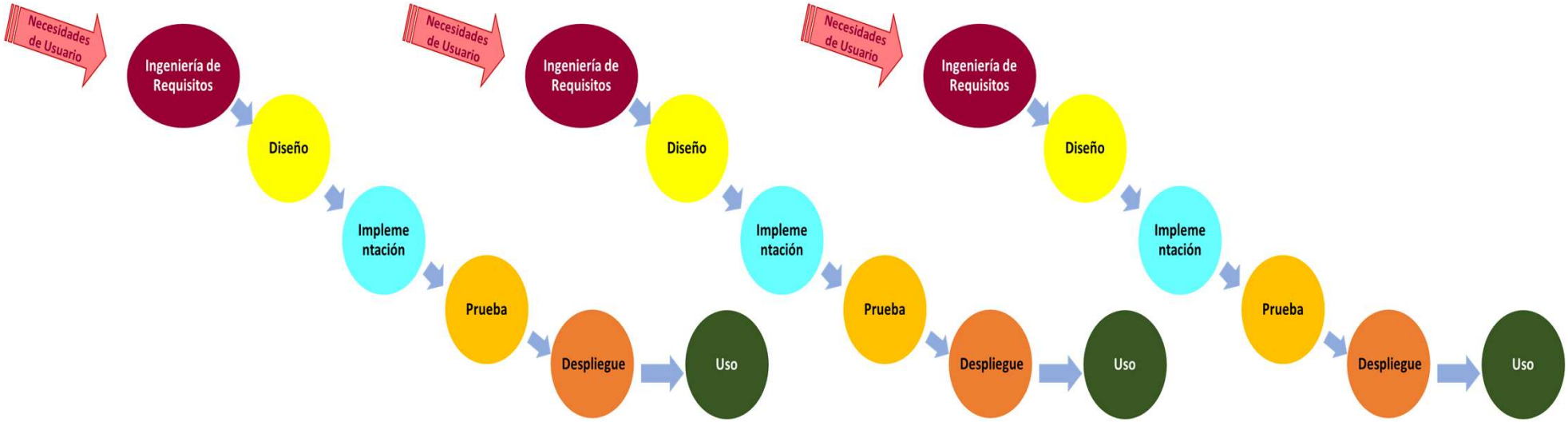
Modelos de Proceso de Software

Metodologías: cascada, iterativo, espiral, ágil, devops…



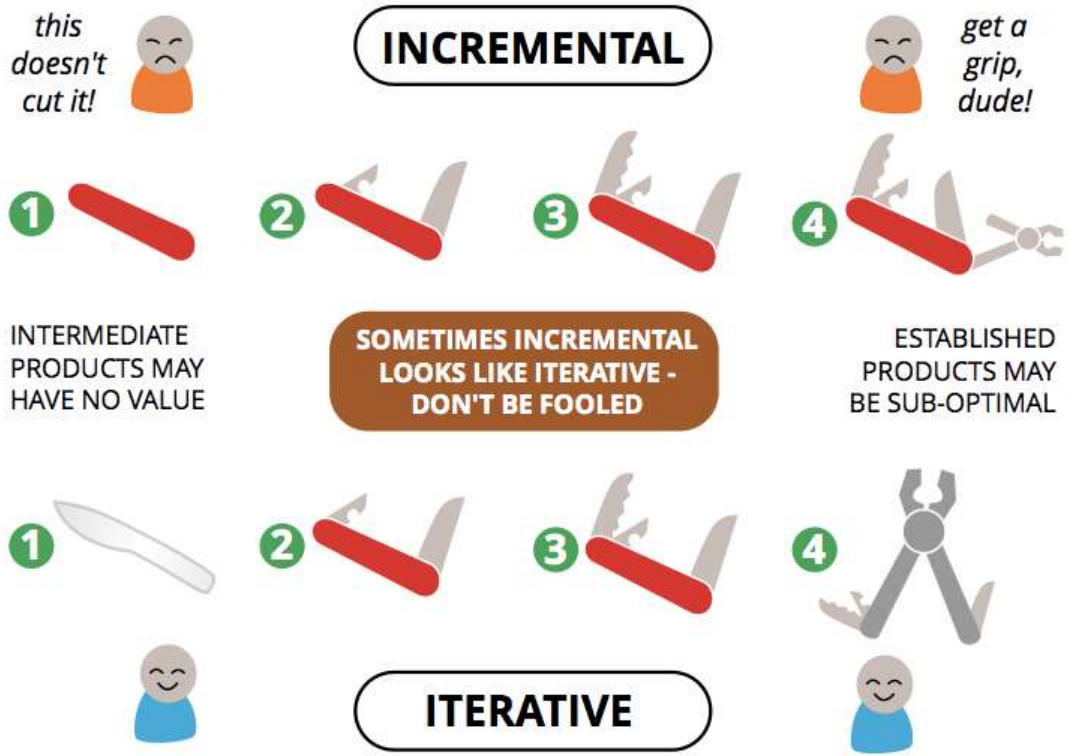
Modelos de Proceso de Software

Metodologías: cascada, **iterativo, incremental**, espiral, ágil, devops…



Modelos de Proceso de Software

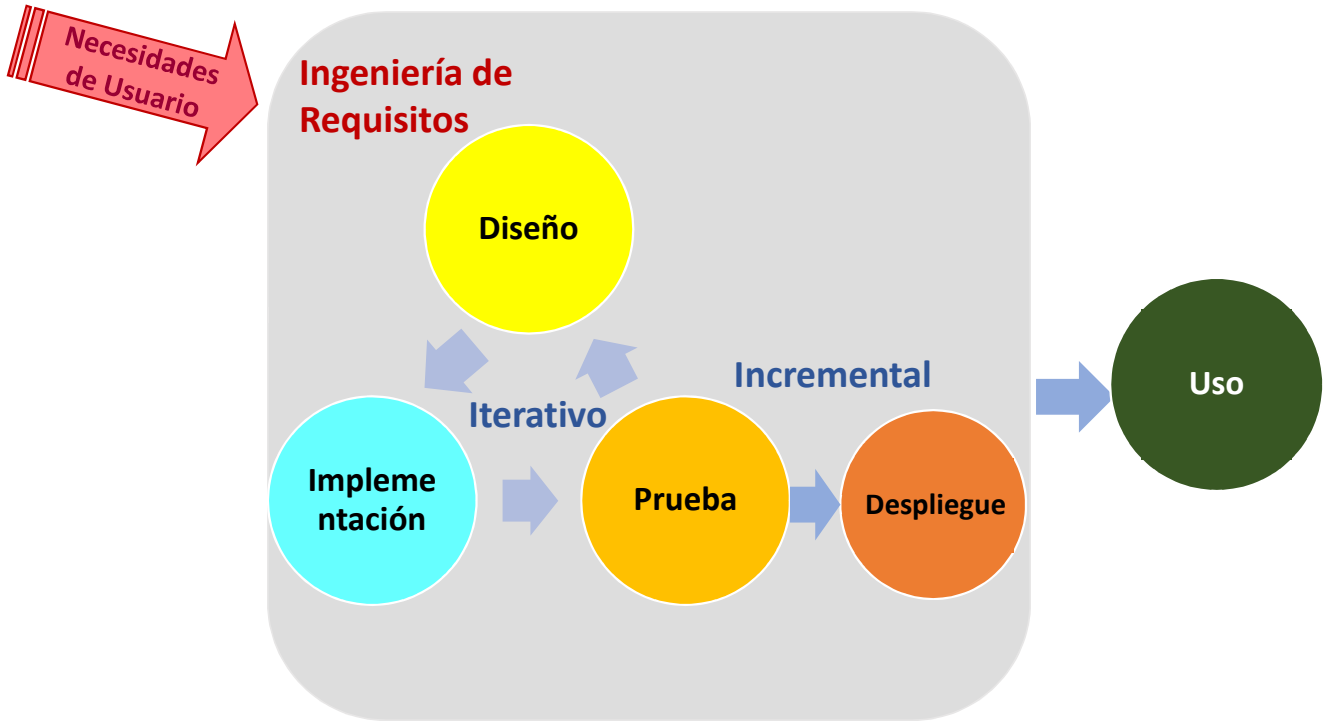
Metodologías: cascada, iterativo, incremental, espiral, ágil, devops…



Iterative vs Incremental Flashcard – safety dave

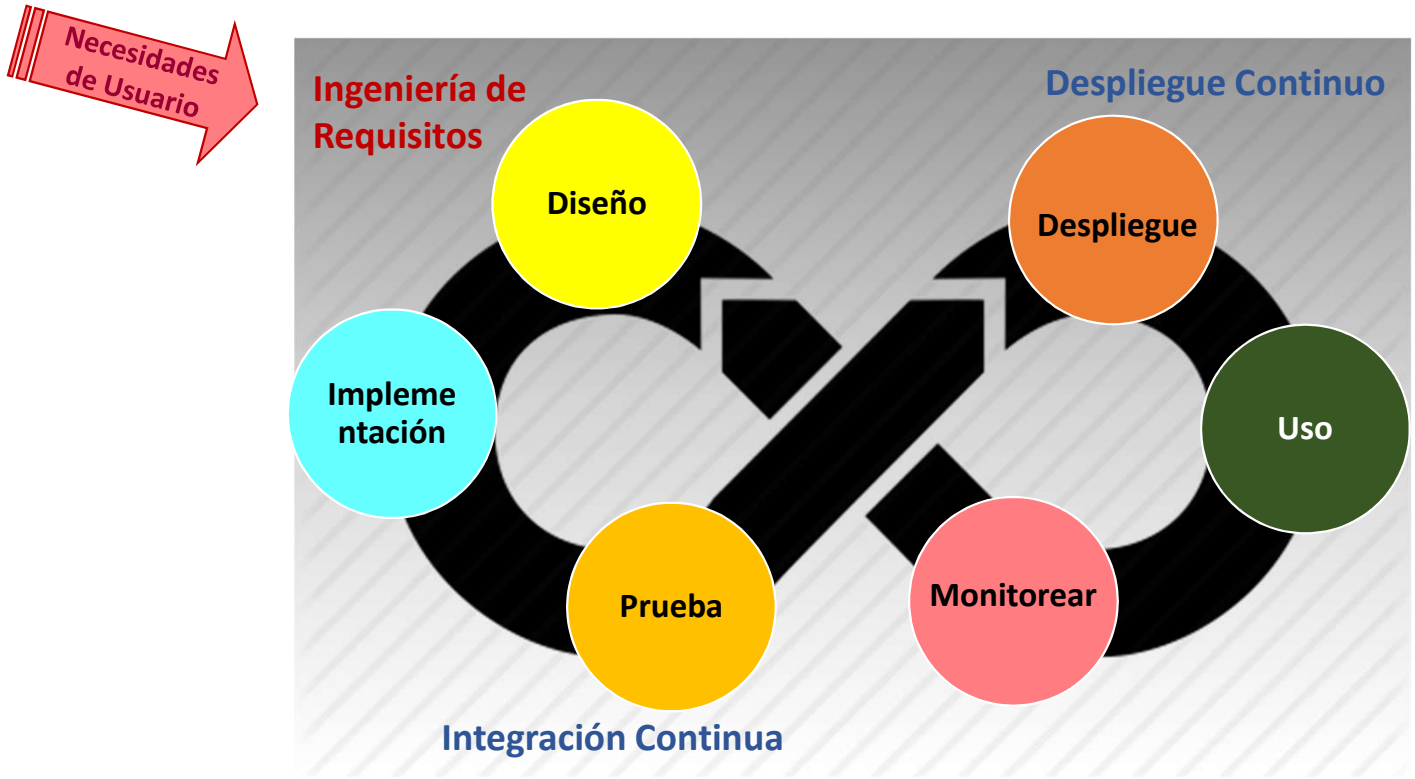
Modelos de Proceso de Software

Metodologías: cascada, iterativo, espiral, ágil, devops…



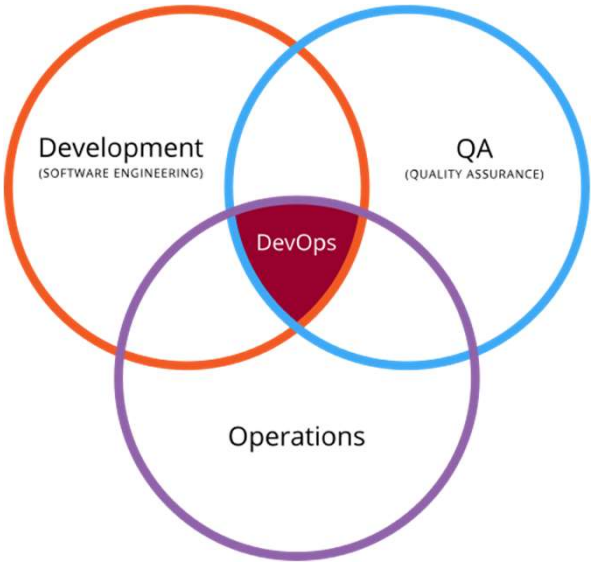
Modelos de Proceso de Software

Metodologías: cascada, iterativo, espiral, ágil, devops…



Modelos de Proceso de Software

Metodologías: cascada, iterativo, espiral, ágil, **devops**…



DevOps: Es la combinación de **filosofías** culturales, prácticas y herramientas que aumenta la capacidad de una organización para entregar aplicaciones y servicios a alta velocidad: evolucionando y mejorando productos a un ritmo más rápido que las organizaciones que utilizan procesos tradicionales de desarrollo de software y gestión de infraestructura (AWS).

Trazabilidad de Software

- Trazabilidad de Requisitos en Diseño, Código, Pruebas, Cambios, …

The subject of Use-Case 2.0 is the requirements, the system to be developed to meet the requirements, and the tests used to demonstrate that the system meets the requirements. At the heart of Use-Case 2.0 are the use case, the story and the use-case slice. These capture the requirements and drive the development of the system. Figure 5 shows how these concepts are related to each other. It also shows how changes and defects impact on the use of Use-Case 2.0.

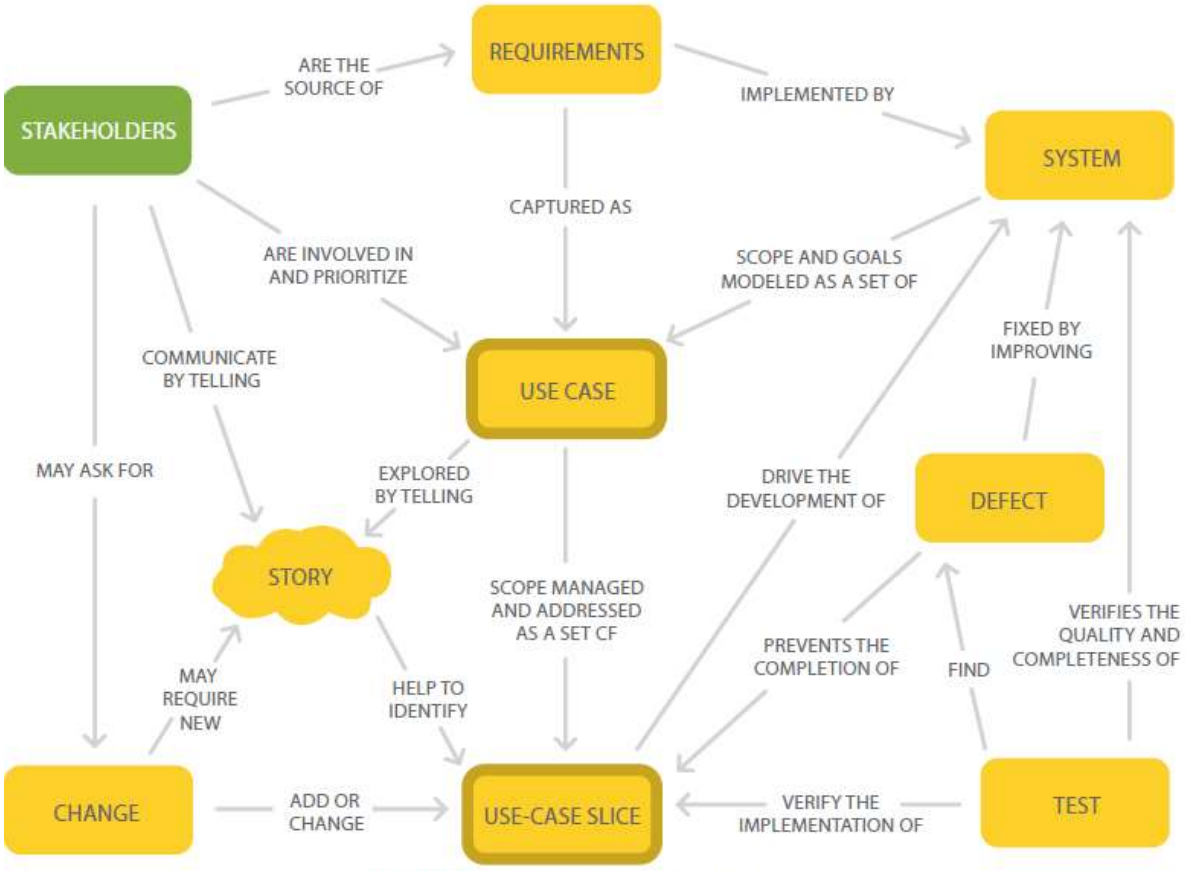


FIGURE 5: USE-CASE 2.0 CONCEPT MAP.

Jacobson, I., Spence, I., & Kerr, B. (2016). Use-Case 2.0. *Queue*, *14*(1), 94-123.

Resultados

- Más proyectos concluidos con éxito;
- Gestión adecuada de grandes equipos de software;
- Calidad del software mejorada;
- Disminución de los costos de desarrollo.

Resumen

- Todas las demás actividades dependen del proceso de desarrollo.
- Proyectos de software modernos utilizan algún tipo de metodología basada en iteraciones o espiral sobre uno en cascada.
- Um proceso gerenciado (establecido, comprendido y controlado) puede ayudar a alcanzar la calidad del producto.

Bibliografia Principal

- SOMMERVILLE, I. Engenharia de Software. Addison-Wesley, 8ª edição, 2006.
- **PRESSMAN, R. Engenharia de Software. Rio de Janeiro: McGraw Hill, 10ª edição, 2017.**
- ROCHA, A.R.; MALDONADO, J.C. Qualidade de software: teoria e prática. São Paulo: Prentice Hall, 2001.

Bibliografia Complementar

- CMMI. Disponível em: http://www.sei.cmu.edu/cmmi
- Sanches, R. Notas de Aula. Qualidade de Software. ICMC-USP-Brasil.
- Falbo, R. Notas de Aula. Tópicos Especiais – Qualidade de Software. UFES-Brasil.
- E.A.Schmitz. Fundamentos de Engenharia de SW. Ciclo de Vida de Projetos de SW. NCE, PPGI, UFRJ – Brasil
- Inês A.G.Boaventura. Engenhar'ia de Software. O produto e o Processo. UNESP – Brasil.
- Robert Half. 6 Basic SDLC Methodologies: Which One is Best?, 2017. Disponible en: https://www.roberthalf.com/blog/salaries-and-skills/6-basic-sdlc-methodologies-which-one-is-best
- Altexsoft – Software R&D Engineering. Agile Project Management: Best Practices and Methodologies, 2018. Disponible em: https://www.altexsoft.com/whitepapers/agile-project-management-best-practices-and-methodologies/
- Instinctools. Advantages of Lean Software Development. Disponible en: https://www.instinctools.com/blog/advantages-of-lean-software-development