



Universidad  
Andrés Bello®  
Conectar • Innovar • Liderar

# ICF232 Ingeniería de Software I

Primer Semestre 2019

A word cloud centered around the word "welcome" in a large, bold, orange font. Surrounding it are numerous translations of "welcome" in various languages, including Spanish, Dutch, German, French, Italian, and others. The words are arranged in a circular pattern around the central word, with some words appearing in different colors like blue, yellow, and red. The background is a solid orange color.

Words included in the word cloud:

- welcome
- bienvenido
- verwelkom
- benvenuto
- greet
- istien
- komna
- hozta
- acoger
- hospedar
- moguah
- la
- albergar
- hozott
- menerima
- veikomme
- dant
- zi
- accollire
- il
- toivottaa
- hei
- ospit
- croesawu
- aloxar
- mpina
- uvitani
- goscic
- souhaiter
- dare
- tervetulleeksi
- heten
- verwelkomen
- accogliere
- welkom
- ontvangen
- accueillir
- acollir
- roguah
- dv
- accogliere
- willkommen
- begr
- accipere
- acolher
- degemer
- etmek
- acoller
- en
- recoger
- mat
- kabul

# Presentación

Pablo Schwarzenberg, [pablo.schwarzenberg@unab.cl](mailto:pablo.schwarzenberg@unab.cl), Director de Carrera

**Acerca de mí:** Soy Doctor en Ciencias de la Ingeniería y Magister en Gestión con 22 años de experiencia en la industria de desarrollo de software, 10 de ellos en cargos directivos.



# Presentación

## Mis Expectativas

- Enseñarles a aplicar la ingeniería de software usando criterios basados en la experiencia.
- Ser capaz de apoyarlos en forma efectiva durante su proceso de aprendizaje.
- Que tengan una buena percepción de la ingeniería de software, se sientan satisfechos de lo aprendido en el curso, y se sientan motivados para seguir aprendiendo..

# Presentación

**Qué espero de ustedes**

**Esfuerzo**  
**Respeto**  
**Participación**

# Ahora ustedes (10 minutos)



¿Qué esperan del curso de Ingeniería de software?

¿Qué relación hay entre ingeniería de software y programación?

Expectativas e Ideas preconcebidas



# Acerca del Curso: Descripción

Esta asignatura pertenece al eje curricular de Aplicaciones Informáticas declarada en el perfil de egreso. En ella, se comprenderá la evolución a lo largo del tiempo y situación actual de la industria del software a nivel nacional e internacional, se analiza el ciclo de vida del software, se conocen distintas metodologías de desarrollo, y se aplican los conceptos básicos de gestión de proyectos. Todos estos elementos son materializados y puestos en práctica en un proyecto ingenieril de desarrollo de software (D.U.N 1973/2012).

Al finalizar la asignatura el estudiante dispondrá de herramientas, métodos y técnicas para que los proyectos de desarrollo de software sean efectivos y eficientes, es decir, para obtener productos de software de mayor calidad a un menor costo.





# Acerca del Curso: Aprendizajes Esperados

- Describir los principales problemas que afectan el desarrollo de sistemas de software complejos y poder aplicar criterios de evaluación y técnicas para identificar y manejar estos problemas en proyectos de software concretos.
- Describir las diferentes metodologías de desarrollo software indicando los factores que determinan su aplicación para maximizar sus beneficios en la realización de proyectos de software.
- Aplicar diferentes herramientas para manejar proyectos de software que involucren la coordinación de factores técnicos y sociales derivados del trabajo en equipo.
- Utilizar diferentes técnicas y tecnologías para definir artefactos que guíen las distintas etapas de desarrollo de un proyecto de software.

# Acerca del Curso: Aprendizajes Esperados

- Experimentar los principios asociados a la Ingeniería de Requisitos, e Ingeniería de Software Dirigida por Modelos.
- Gestionar la calidad de proceso y calidad de producto.
- Comparar las principales técnicas asociadas a la verificación y validación de productos de software de acuerdo a las necesidades del proyecto.
- Utilizar los mecanismos para gestionar la calidad de proceso y producto en proyectos de software concretos.
- Utilizar las nuevas técnicas destinadas a medir el tamaño funcional del software y estimar esfuerzos asociados a la correcta planificación del desarrollo sistemas y su valorización.

# Evaluación del Curso

**2 Solemnes**

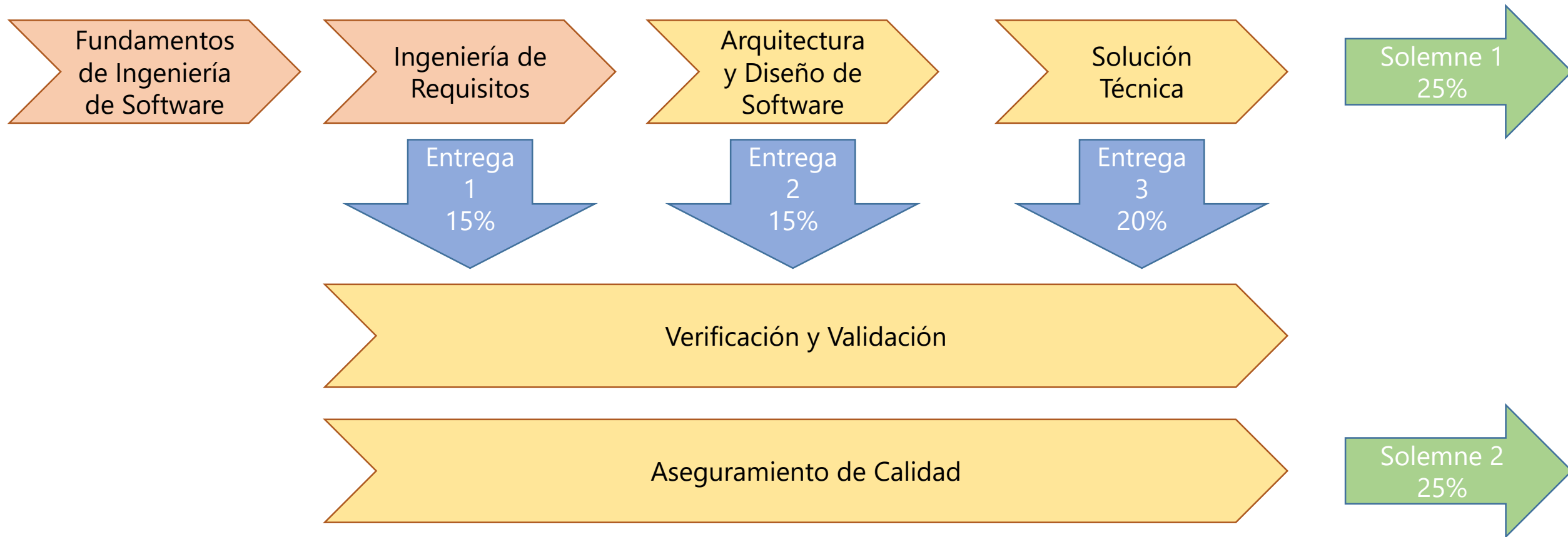
**50%**

**1 Proyecto**

**50%**

**Asistencia Laboratorio 100%**

# Acercas del Curso: Contenidos y Evaluación Formal



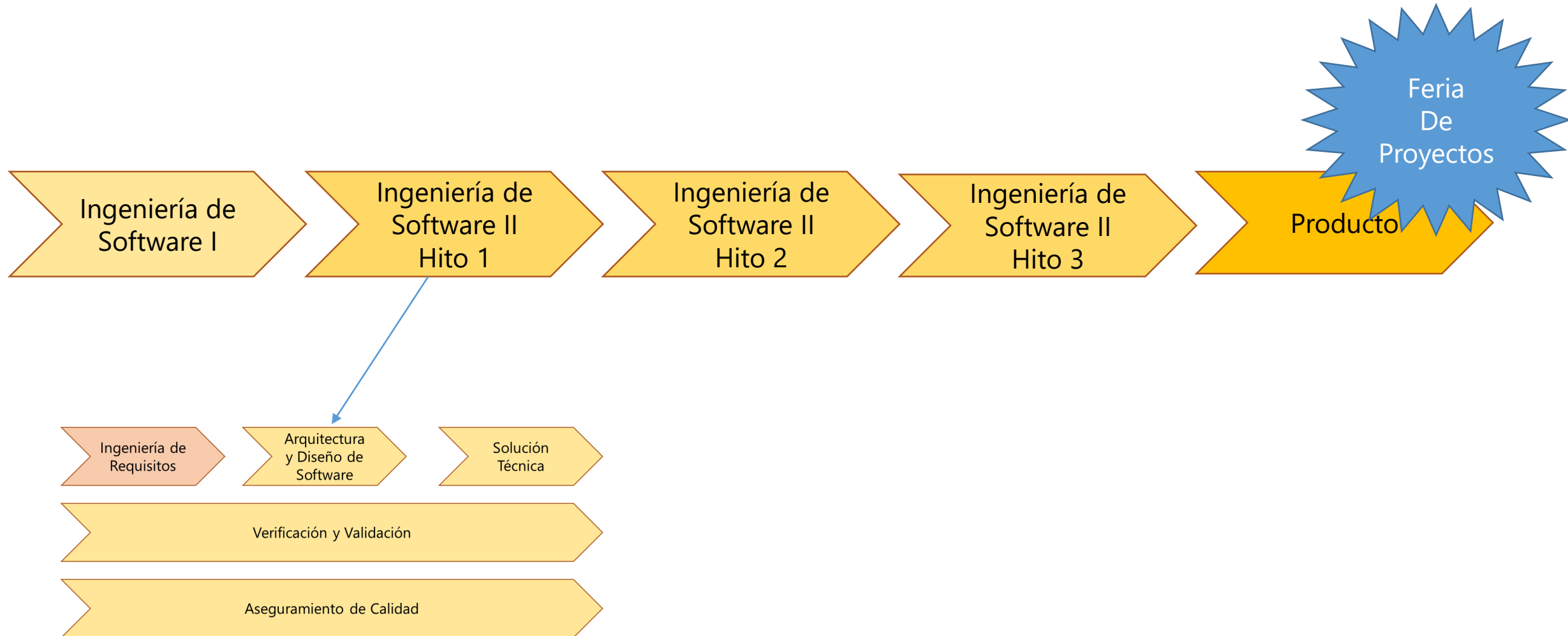
# Acerca del Curso: Proyecto

- El proyecto se desarrollará en grupos de hasta 5 estudiantes.
- Pueden proponer un tema durante las primeras dos semanas del curso (Hasta el día 17 de marzo).
  - Puede ser una idea que quieran desarrollar como emprendimiento.
  - Debe tener un cliente o destinatario identificable.
- Si no logran definir un tema propio podrán escoger uno a partir de una cartera de propuestas de proyecto.

# Acerca del Curso: Progreso del proyecto (20% de cada hito: 1 no cumple, 7 cumple)

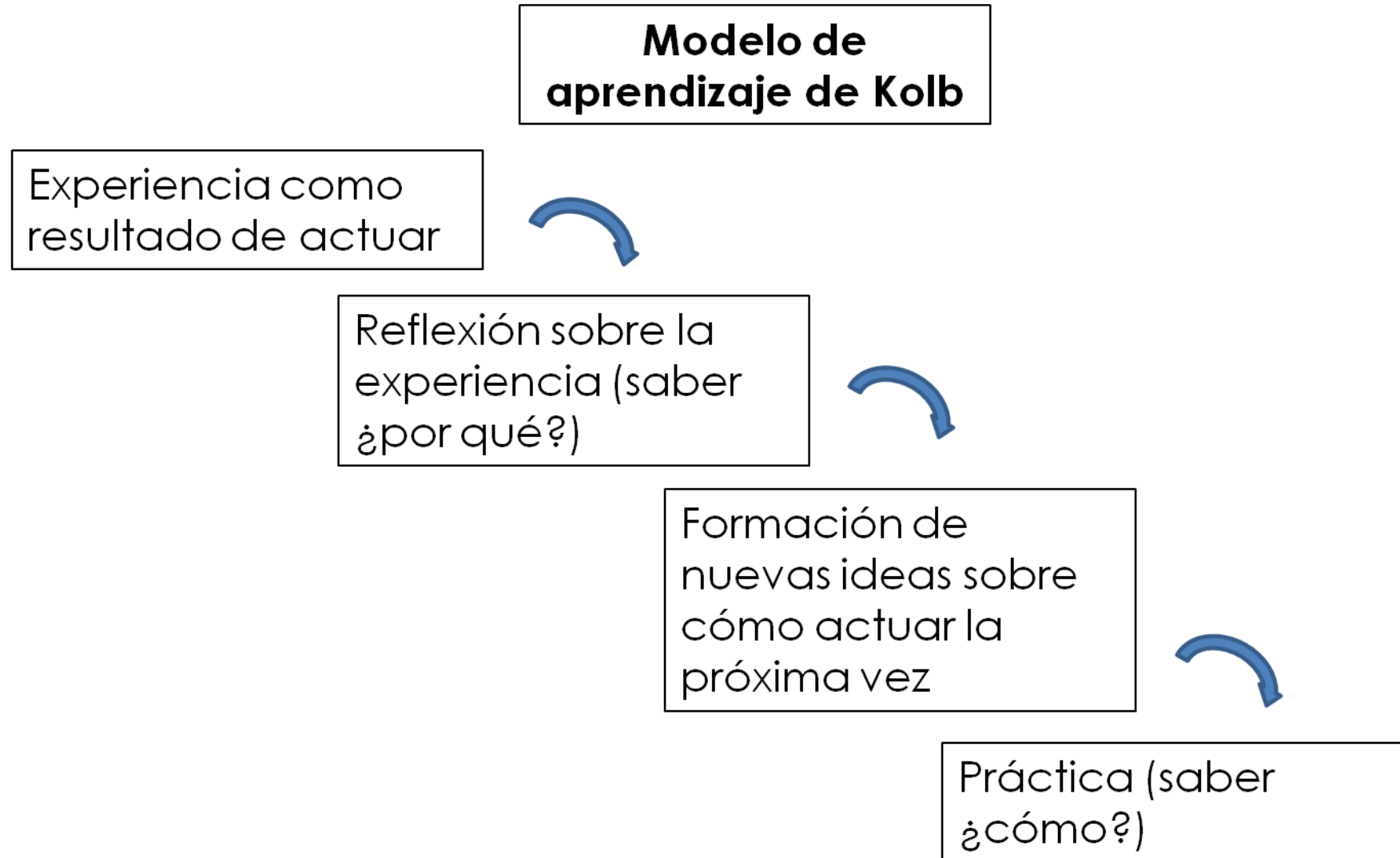
- Cada grupo debe tener un jefe de proyecto (puede rotar, a elección del grupo):
  - Debe mantener actualizada la planificación
  - Debe subir informes de progreso semanales, indicando logros (tareas completadas), problemas encontrados (tareas atrasadas) y las acciones decididas para mitigar los problemas encontrados.
  - Los informes deben estar disponibles a más tardar el día Domingo de cada semana.
- Cada integrante del grupo tendrá un rol y tareas asignadas de las que debe ser responsable en forma única.
- Cada semana cada grupo deberá presentar los problemas encontrados (tareas atrasadas) de su informe de progreso (5 minutos)

# Continuación del Proyecto en Ingeniería de Software II





# El curso combina teoría con práctica



# Fundamentos de Ingeniería de Software

# ¿Qué es ingeniería de software?

La aplicación de una aproximación **sistemática**, **disciplinada** y **cuantificable** al **desarrollo**, **operación** y **mantenimiento** del software: la aplicación de ingeniería al software (IEEE, 1993)

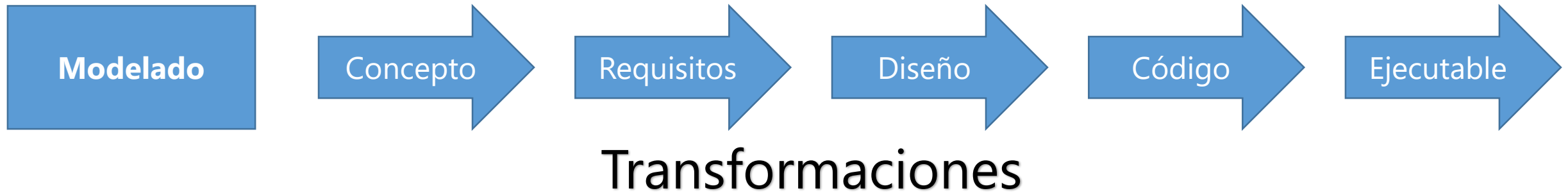
Es el **proceso** de construcción de **sistemas** intensivos en software satisfaciendo un conjunto de **restricciones** (Erdogmus, Medvidovic & Paulish, 2018).

**Social**  
Persona  
Equipo



**Técnico**  
Ingeniería  
Ciencia

# La esencia de la ingeniería de software (Navon, 2017)



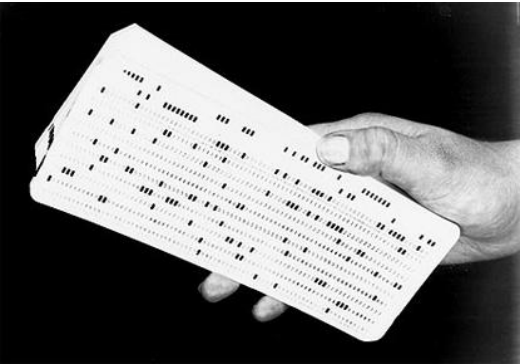
## Diseño

Encontrar la mejor solución que responda a **requisitos** sujeta a **restricciones**

## Optimización

Encontrar las transformaciones de mejor **calidad** y más **económicas**

# Historia de la Ingeniería de Software: Origen



```
n_factorial = 1  
DO i = 1, n  
  n_factorial = n_factorial * i  
END DO
```

7 de Octubre  
de 1968  
Primera  
Conferencia



1957

1963

1961 - 1972

21 de junio  
de 1948  
Primer  
Programa

Fortran  
IBM



PDP 1



Margaret Hamilton



# Historia de la Ingeniería de Software: Crisis del software



1976  
Apple I

1965

1985

Fuera de Costo  
Fuera de Plazo  
Errores de Alto Impacto  
Alto Costo de Mantención

La Complejidad

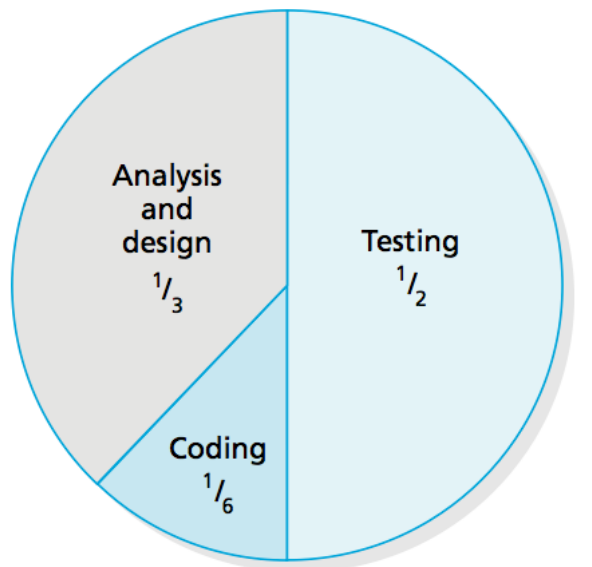
La Compatibilidad

El Cambio

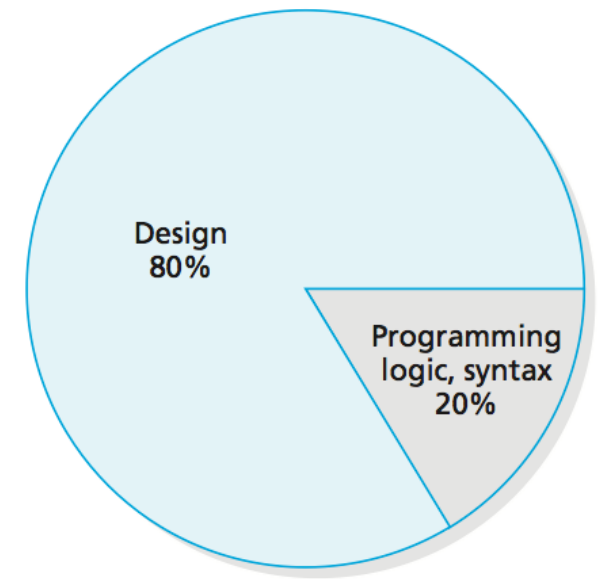
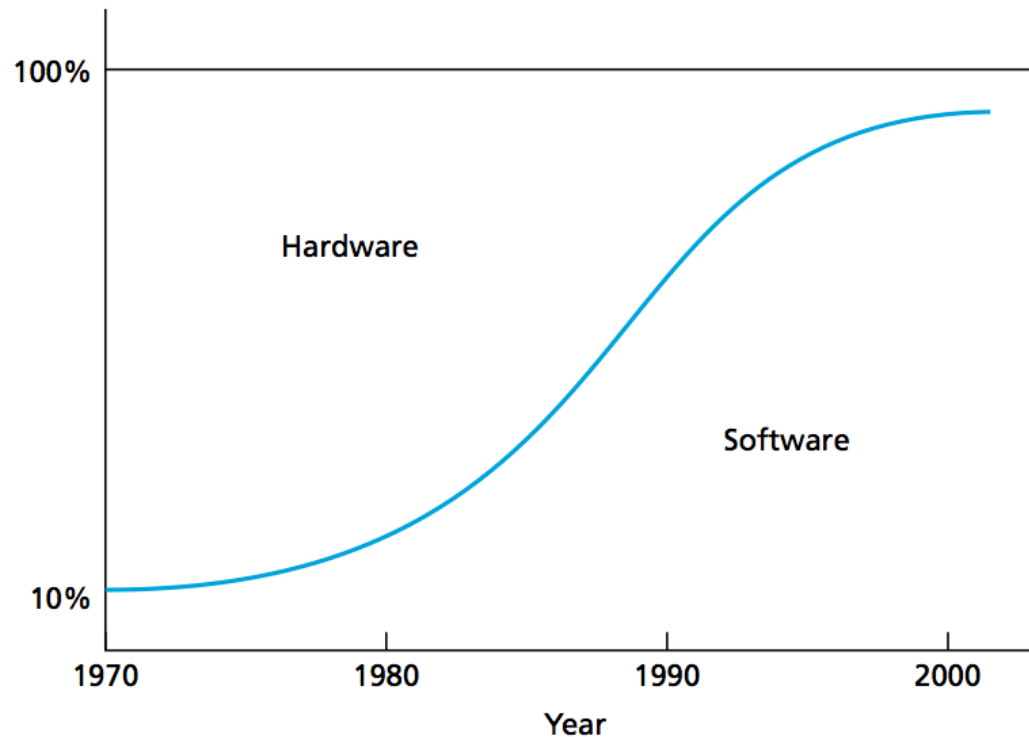
La Invisibilidad

# Historia de la Ingeniería de Software: Crisis del software

## Costo: USD 1000 Billones / Año (2017)



Costos por Etapa



Costos de Reparar Fallas (no Defectos)



# Historia de la Ingeniería de Software: Crisis del software

## Fuera de Costo y Plazo

Difícil estimar Esfuerzo y  
por lo tanto el Plazo

Difícil recuperar tiempo  
cuando hay atrasos

Estimaciones considerando  
"días ideales" 160 H/H  
8 horas, 5 días, 4 semanas

Presión para entregar  
pronto  
Ej. Ventaja competitiva



# Historia de la Ingeniería de Software: Crisis del software

	Defect Potentials	Removal Efficiency	Delivered Defects	Total Defects Delivered	High Severity Defects
CMM 5 + Six-Sigma	4.80	98.00%	0.10	960	259
TSP/PSP + Scrum	4.90	97.00%	0.15	1,470	397
TSP/PSP	5.00	96.00%	0.20	2,000	540
CMM Level 5	5.50	96.00%	0.22	2,200	594
Six-Sigma for software	5.25	94.00%	0.32	3,150	851
CMMI	6.10	94.00%	0.37	3,660	988
CMM Level 4	6.00	93.00%	0.42	4,200	1,134
CMM 3 + OO	6.10	92.00%	0.49	4,880	1,318
CMM Level 3	6.25	92.00%	0.50	5,000	1,350
Waterfall + inspections	6.50	92.00%	0.52	5,200	1,404
Agile/Scrum + OO	5.30	90.00%	0.53	5,300	1,431
TickIT	6.10	88.00%	0.73	7,320	1,976
Extreme XP	6.25	88.00%	0.75	7,500	2,025
Agile/Scrum	6.00	87.00%	0.78	7,800	2,106
Iterative	6.25	86.00%	0.88	8,750	2,363
Spiral	6.50	85.00%	0.98	9,750	2,633
Object oriented (OO)	6.00	83.00%	1.02	10,200	2,754
RUP	6.75	84.00%	1.08	10,800	2,916
SOA	2.50	55.00%	1.13	11,250	3,038
CMM Level 2	7.00	80.00%	1.40	14,000	3,780
Waterfall	7.25	80.00%	1.45	14,500	3,915
RAD	7.25	77.00%	1.67	16,675	4,502
CMM Level 1	7.50	70.00%	2.25	22,500	6,075
<b>Average</b>	<b>5.96</b>	<b>86.83%</b>	<b>0.78</b>	<b>7,785</b>	<b>2,102</b>

	1,000 Function Points	10,000 Function Points	100,000 Function Points
SOA	8.00	26.00	20.00
TSP/PSP + Scrum	15.75	10.00	9.00
CMM 5 + Six-Sigma	13.00	9.75	9.25
TSP/PSP	14.25	9.50	8.00
CMM Level 5	12.50	9.25	7.75
Six-Sigma for software	9.00	9.00	7.20
CMM 3 + OO	16.00	8.75	6.80
CMMI	11.25	8.25	7.50
Object oriented (OO)	15.50	8.25	6.75
Agile/Scrum + OO	24.00	7.75	7.00
RUP	14.00	7.75	4.20
CMM Level 4	11.00	7.50	5.50
CMM Level 3	9.50	7.25	4.50
Agile/Scrum	23.00	7.00	5.25
Spiral	15.00	6.75	4.25
Extreme XP	22.00	6.50	5.00
TickIT	8.75	6.25	3.90
Iterative	13.50	6.00	3.75
RAD	14.50	5.75	3.60
Waterfall + inspections	10.00	5.50	4.75
CMM Level 2	7.50	4.00	2.25
Waterfall	8.00	2.50	1.50
CMM Level 1	6.50	1.50	1.25
<b>Average</b>	<b>13.64</b>	<b>7.03</b>	<b>5.23</b>

Jones, C. (2008). Applied software measurement: global analysis of productivity and quality.

## Errores de Alto Impacto



Mariner I, 1962

DO 3 I = 1.3

en lugar de

DO 3 I = 1,3

DO 3 I = 1.3  
en vez de  
DO 3 I = 1,3

USD 18,5 millones

# Historia de la Ingeniería de Software: Crisis del software

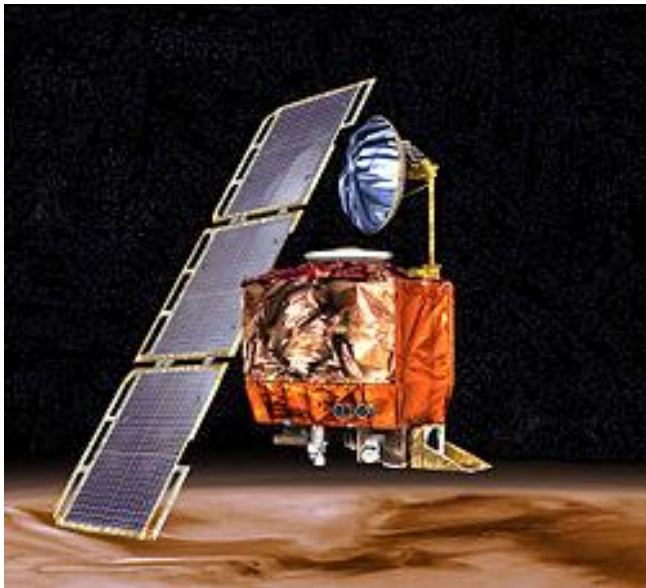
## Errores de Alto Impacto



Por un error de software  
irradió con una dosis  
letal a 6 pacientes, 3  
murieron

## Therac 25, 1985 – 1987

## Errores de Alto Impacto

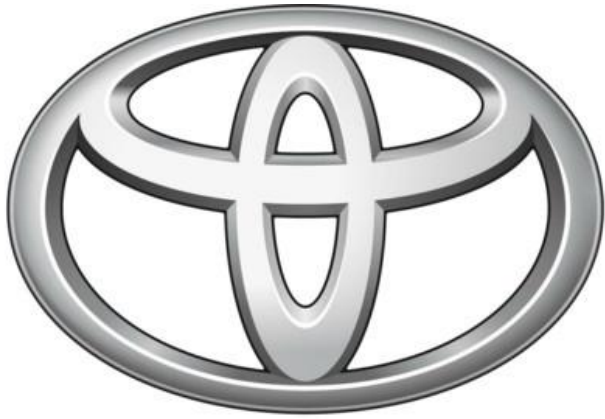


Un error de conversión  
de unidades hizo que  
la Sonda se  
desintegrara al entrar  
a la atmósfera  
USD 327 Millones

## Mars Climate Orbiter, 1998



## Errores de Alto Impacto



**TOYOTA**

Un error en el software  
del sistema de frenos  
provocado por malas  
prácticas (ej. 10000  
variables globales)  
USD 3 Billones

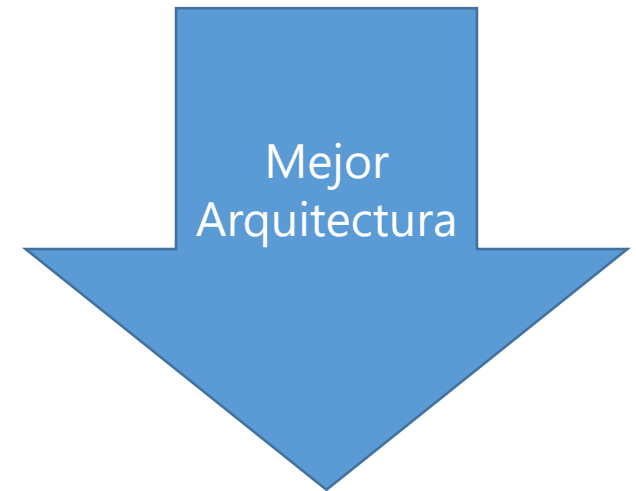
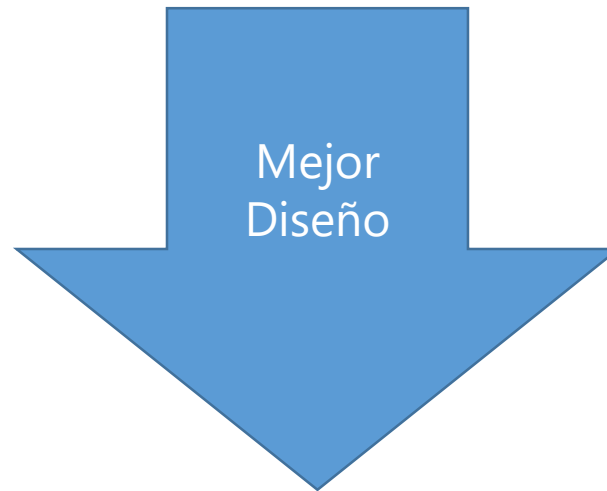
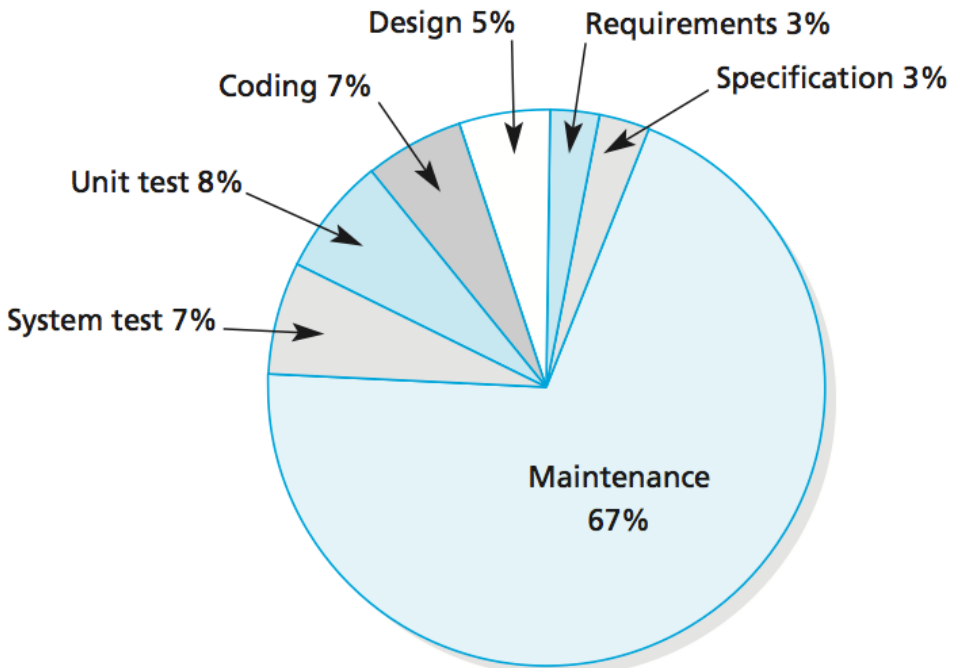
MISRA C, 7134

MISRA 2004, 81514

## Sistema de Frenos, 2010

# Historia de la Ingeniería de Software: Crisis del software

## Alto Costo de Mantenimiento Más del doble del desarrollo



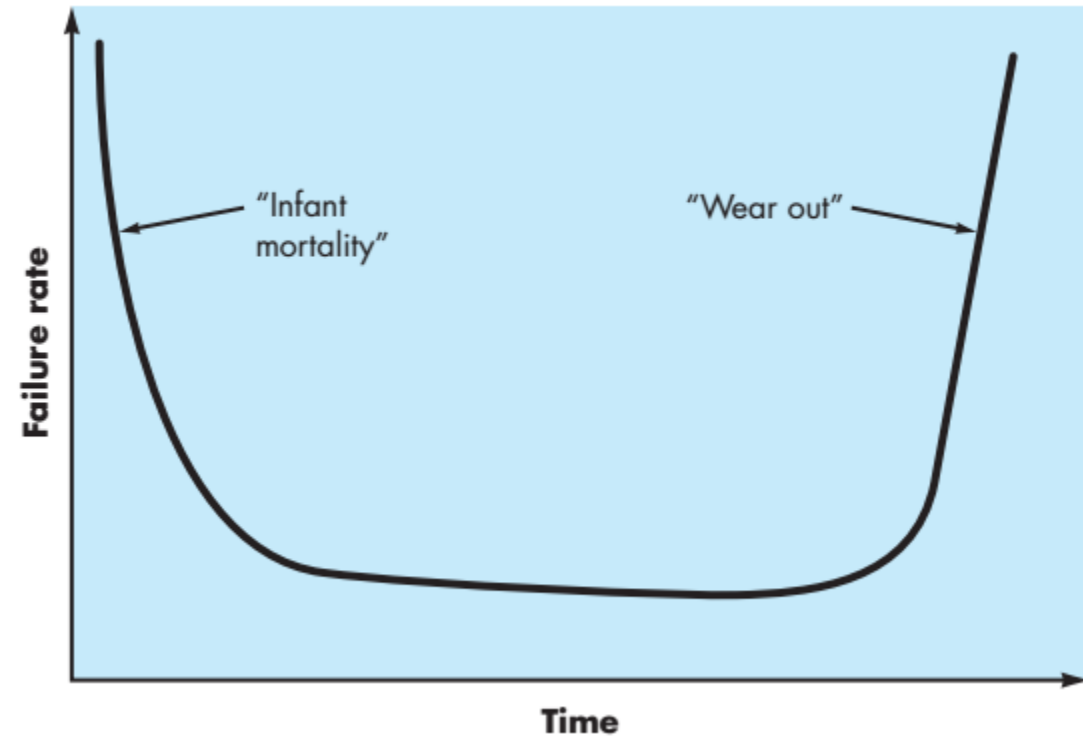
**La medida de la calidad del diseño es el esfuerzo requerido para satisfacer las necesidades del cliente.** Si el esfuerzo es bajo y se mantiene así durante la vida del sistema el diseño es bueno.

Martin, R. C. (2017). *Clean architecture: a craftsman's guide to software structure and design*.

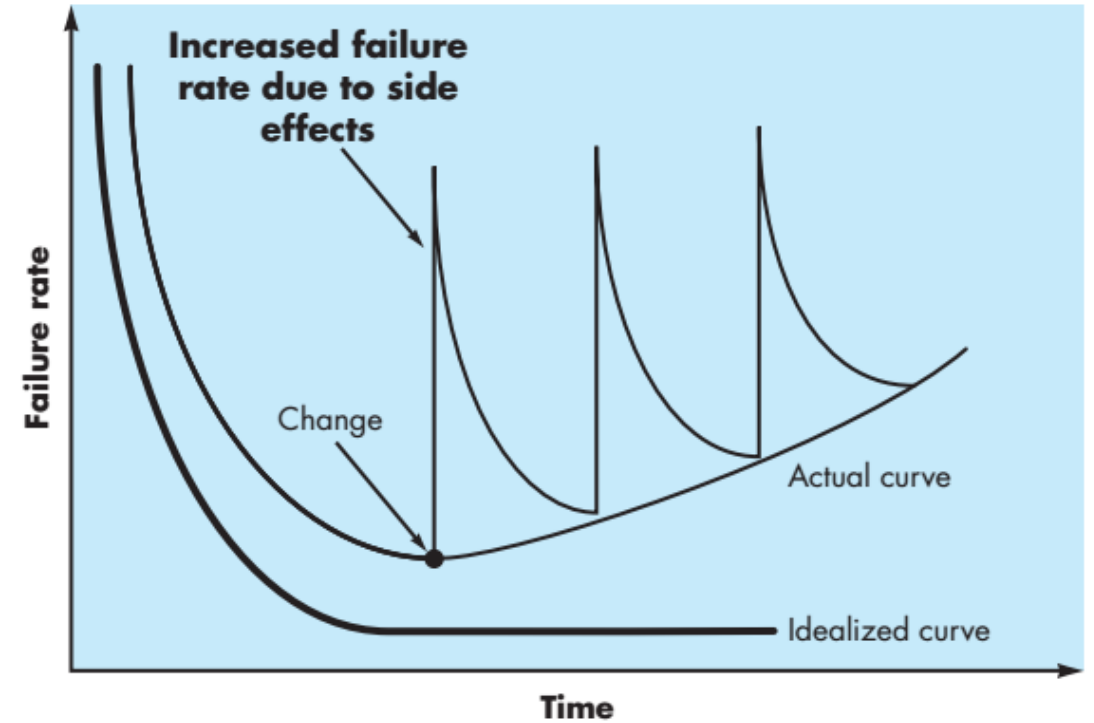


# Historia de la Ingeniería de Software: Crisis del software

## Alto Costo de Mantenimiento

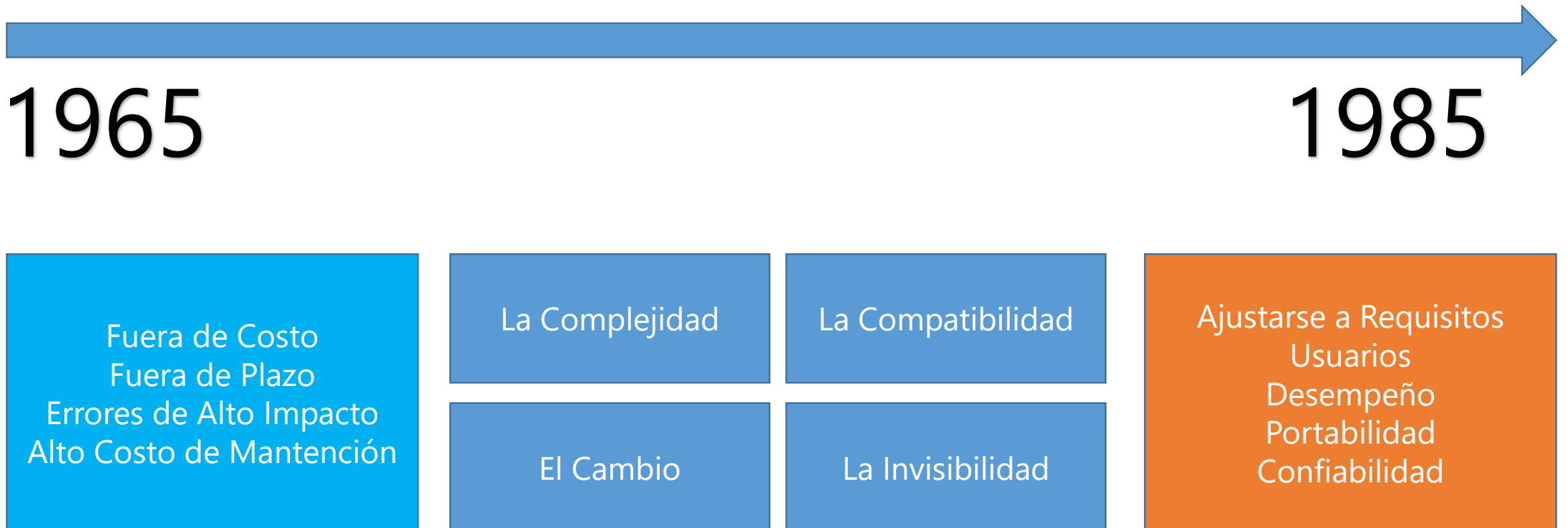


Hardware

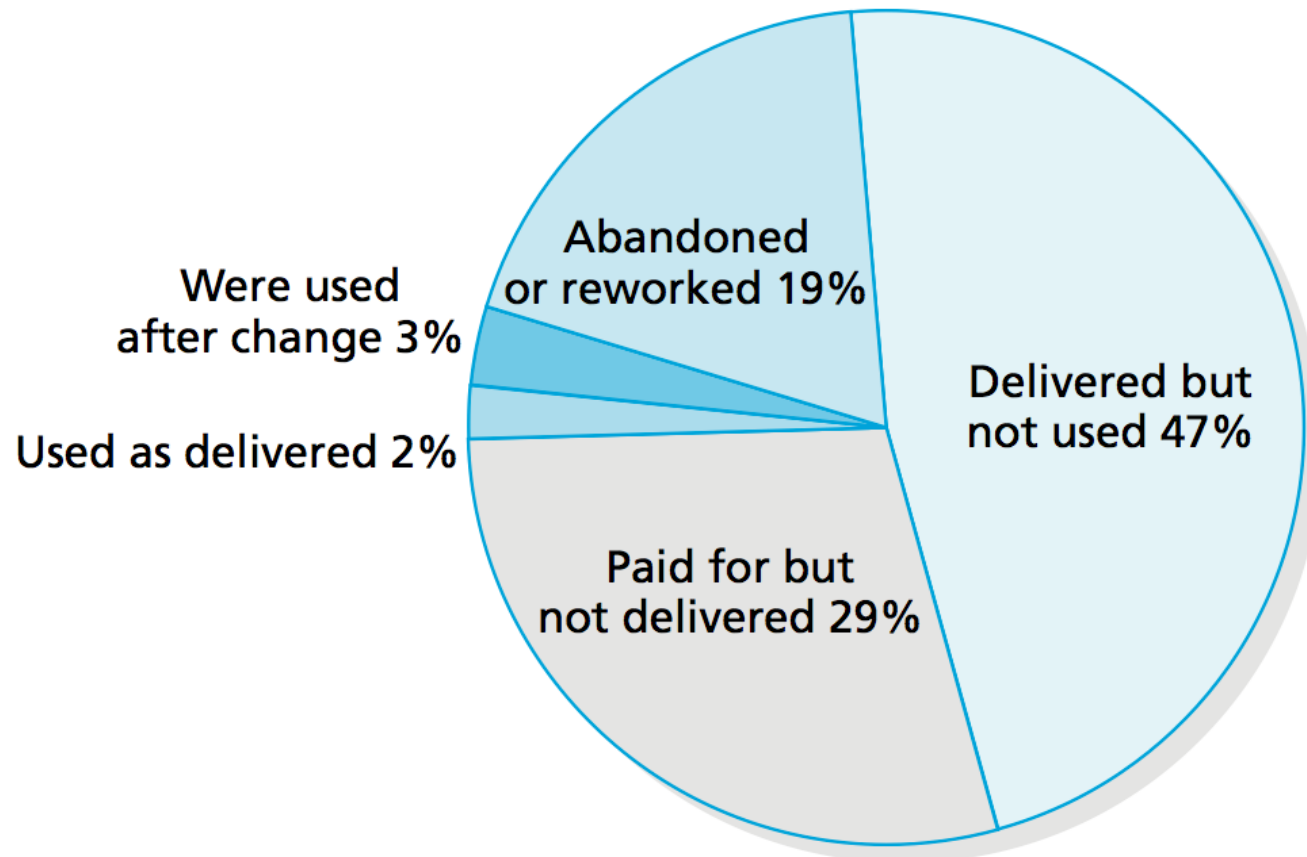


Software

# Historia de la Ingeniería de Software: Crisis del software



## Satisfacer las necesidades del Cliente




Ingeniería de  
Requisitos

# Satisfacer las necesidades del Cliente

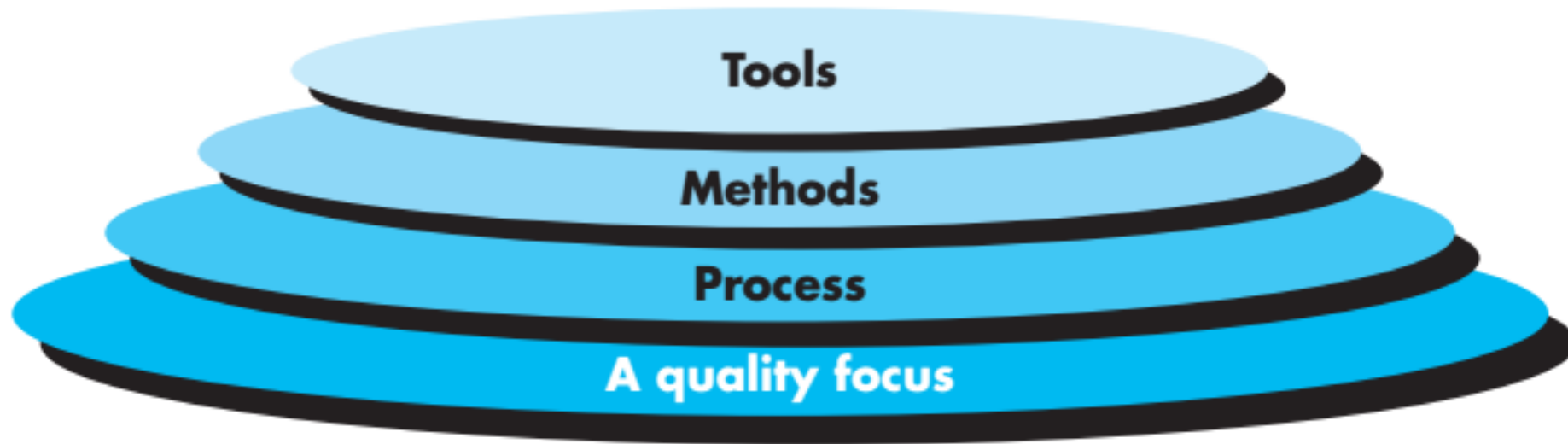
## Stakeholder

Persona que tiene un interés en las características y funciones de un software

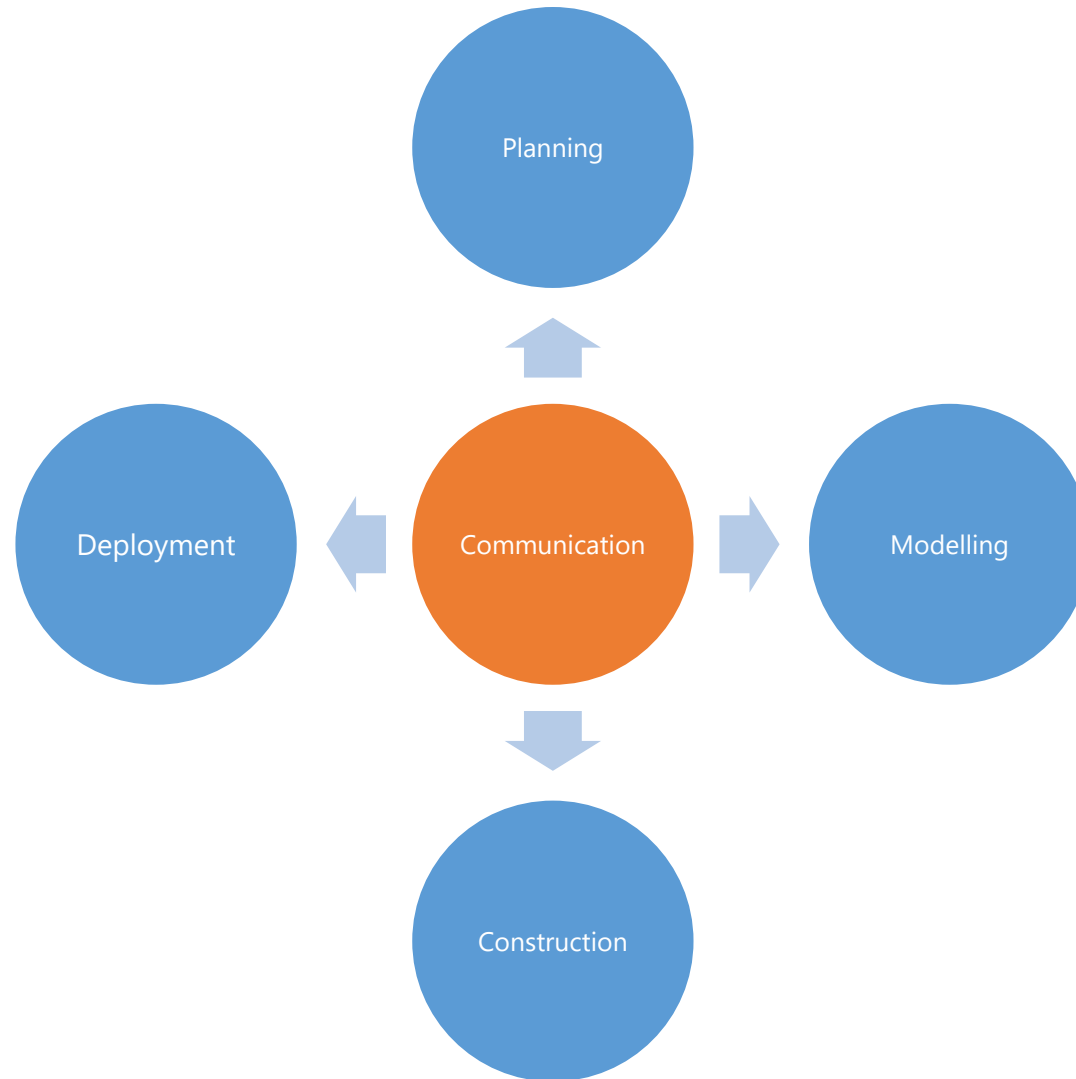


Ingeniería de  
Requisitos

# Historia de la Ingeniería de Software: “No Silver Bullet”

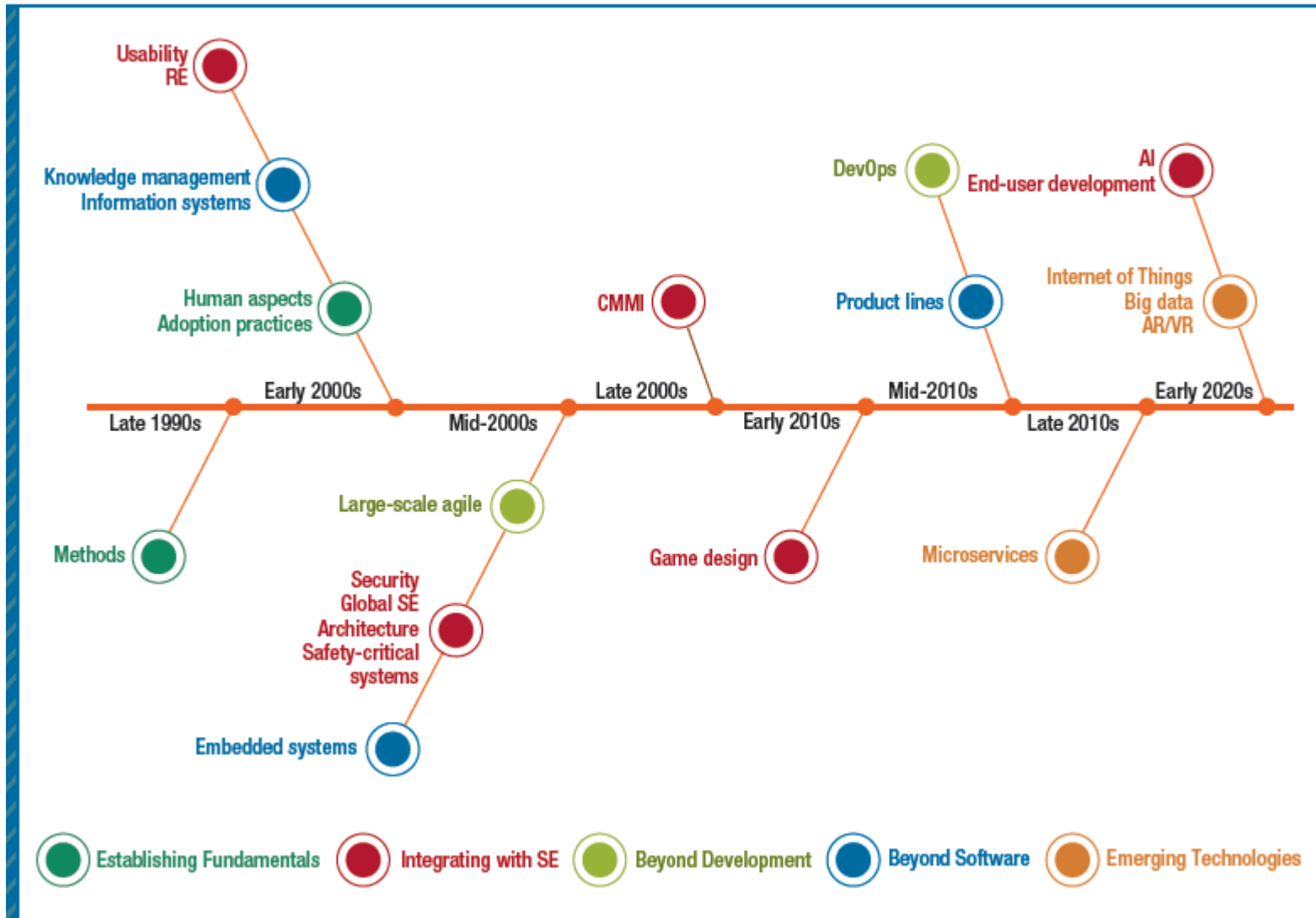


# Historia de la Ingeniería de Software: "No Silver Bullet"



Pressman, R. S. (2015). Software engineering: a practitioner's approach.

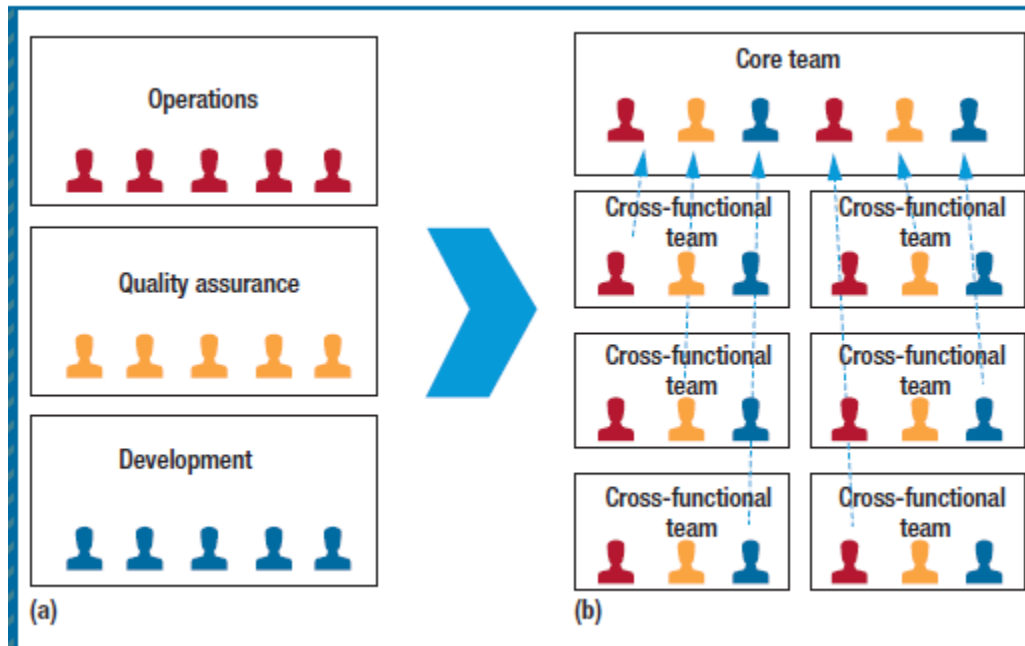
# Historia de la ingeniería de software: Hoy



Hoda, R., Salleh, N., & Grundy, J. (2018). The rise and evolution of agile software development. *IEEE Software*, 35(5), 58-63.



# Historia de la ingeniería de software: Hoy



Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2016).  
Microservices architecture enables devops: Migration to a  
cloud-native architecture. *IEEE Software*, 33(3), 42-52.

# Historia de la ingeniería de software: Hoy

## COMPUTER SCIENCE FUNDAMENTALS

- Abstraction enables the control of complexity.
- Imposing structure on problems often makes them more tractable, and a number of common structures are available.
- Symbolic representations are necessary and sufficient for solving information-based problems.
- Precise models support analysis and prediction.
- Common problem structures lead to canonical solutions.

## ENGINEERING FUNDAMENTALS

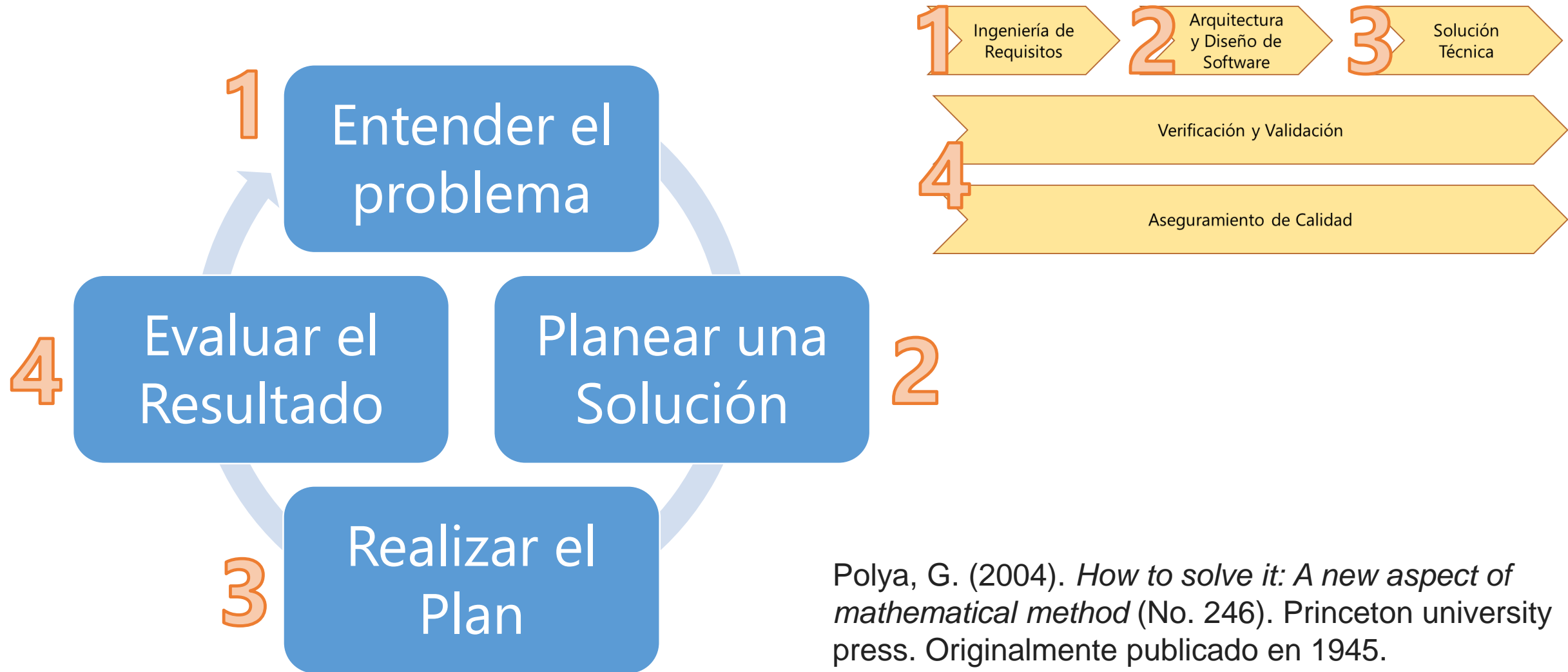
- Engineering quality resides in engineering judgment.
- The quality of the software product depends on the engineer's faithfulness to the engineered artifact.
- Engineering requires reconciling conflicting constraints.
- Engineering skills improve as a result of careful systematic reflection on experience.

## SOCIAL AND ECONOMIC FUNDAMENTALS

- Costs and time constraints matter, not just capability.
- Technology improves exponentially, but human capability does not.
- Successful software development depends on teamwork by creative people.
- Business and policy objectives constrain software design and development decisions as much as technical considerations do.
- Software functionality is often so deeply embedded in institutional, social, and organizational arrangements that observational methods with roots in anthropology, sociology, psychology, and other disciplines are required.
- Customers and users usually don't know precisely what they want, and it is the developer's responsibility to facilitate the discovery of the requirements.

Mead, N. R., Garlan, D., & Shaw, M. (2018). Half a Century of Software Engineering Education: The CMU Exemplar. *IEEE Software*, 35(5), 25-31.

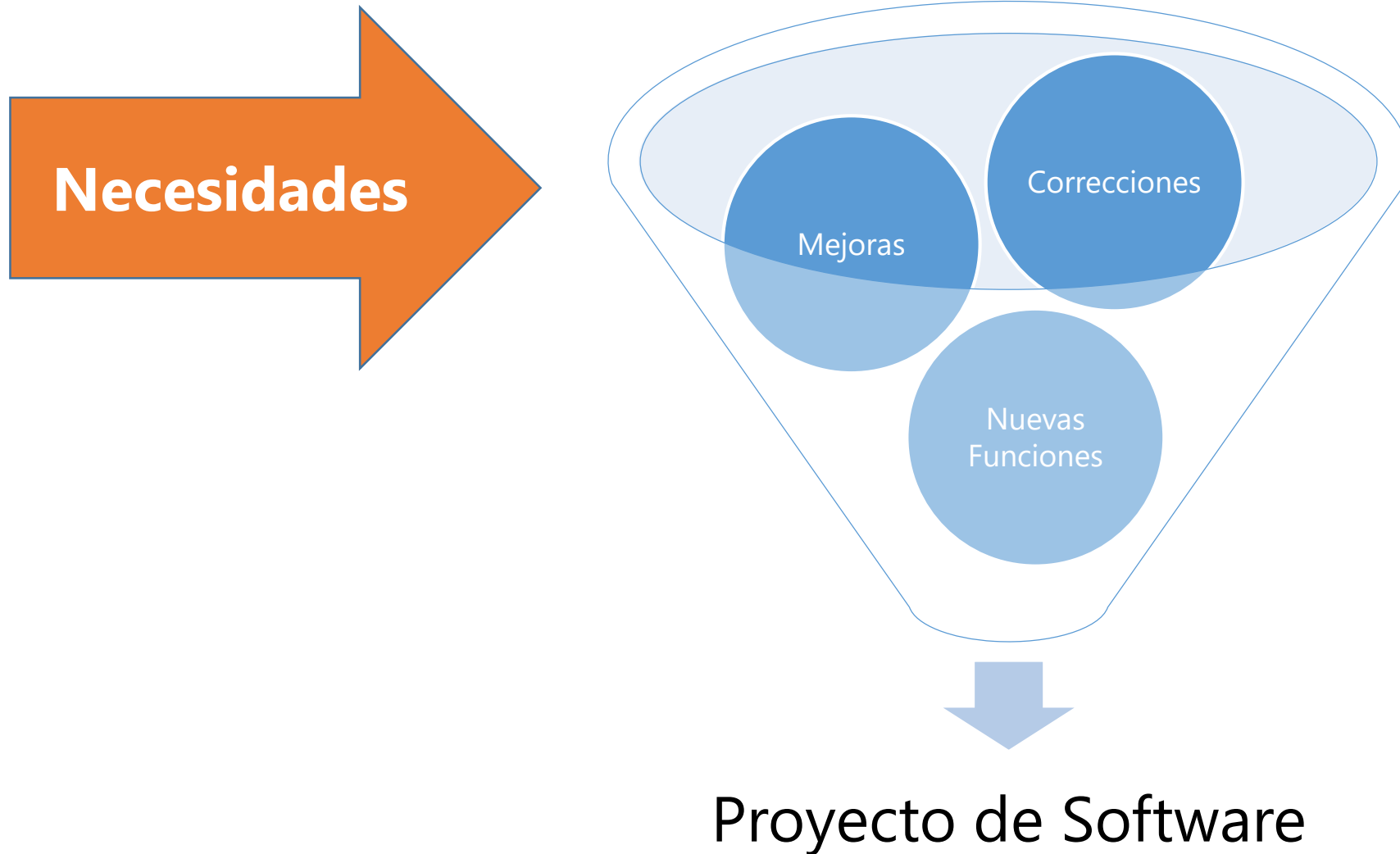
# La base de la ingeniería de software es el proceso de resolución de problemas



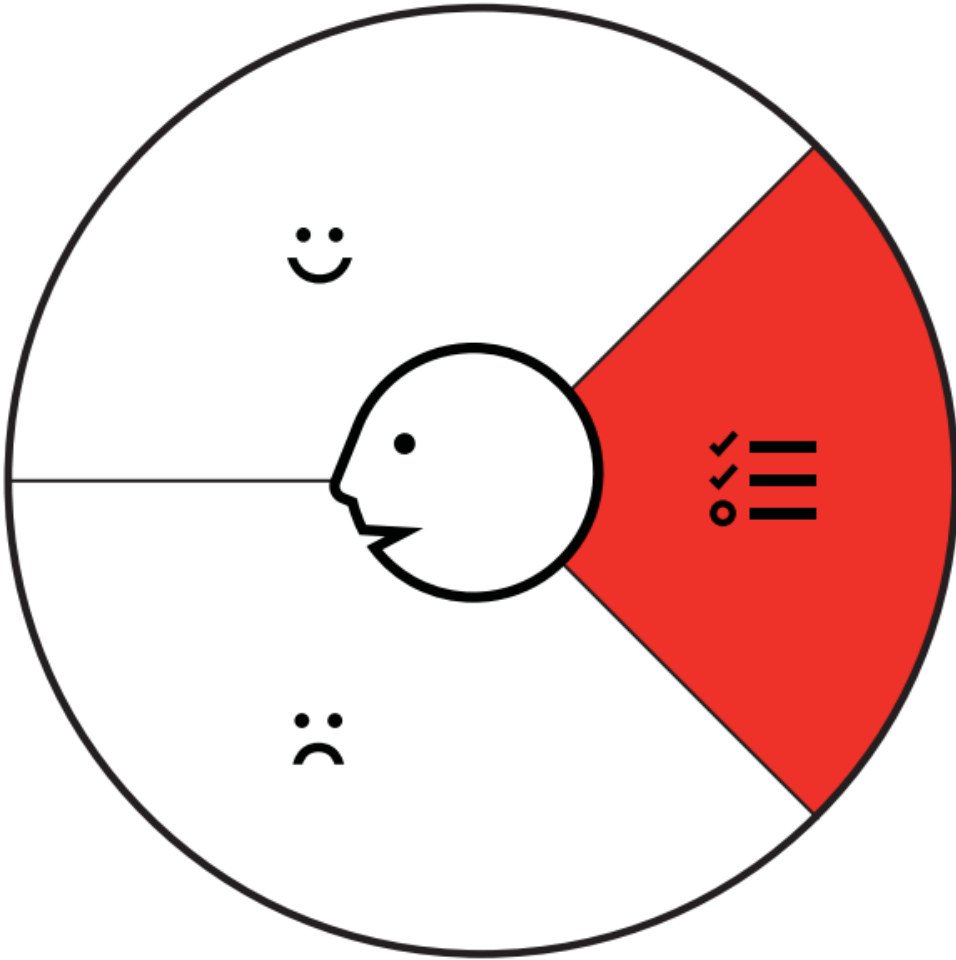


# Origen de un proyecto de software

# El origen de un proyecto de software



# El origen de un proyecto de software: Las necesidades de un cliente



1. Los trabajos
2. Las ganancias
3. Los "dolores"

# Análisis de Caso

## Grupos de 5 integrantes

# Paso 1: Los trabajos

1. What is the one thing that your customer couldn't live without accomplishing? What are the stepping stones that could help your customer achieve this key job?
2. What are the different contexts that your customers might be in? How do their activities and goals change depending on these different contexts?
3. What does your customer need to accomplish that involves interaction with others?
4. What tasks are your customers trying to perform in their work or personal life? What functional problems are your customers trying to solve?
5. Are there problems that you think customers have that they may not even be aware of?
6. What emotional needs are your customers trying to satisfy? What jobs, if completed, would give the user a sense of self-satisfaction?
7. How does your customer want to be perceived by others? What can your customer do to help themselves be perceived this way?
8. How does your customer want to feel? What does your customer need to do to feel this way?
9. Track your customer's interaction with a product or service throughout its lifespan. What supporting jobs surface throughout this life cycle? Does the user switch roles throughout this process?



## Paso 2: Las ganancias

1. Which savings would make your customers happy? Which savings in terms of time, money, and effort would they value?
2. What quality levels do they expect, and what would they wish for more or less of?
3. How do current value propositions delight your customers? Which specific features do they enjoy? What performance and quality do they expect?
4. What would make your customers' jobs or lives easier? Could there be a flatter learning curve, more services, or lower costs of ownership?
5. What positive social consequences do your customers desire? What makes them look good? What increases their power or their status?
6. What are customers looking for most? Are they searching for good design, guarantees, specific or more features?
7. What do customers dream about? What do they aspire to achieve, or what would be a big relief to them?
8. How do your customers measure success and failure? How do they gauge performance or cost?
9. What would increase your customers' likelihood of adopting a value proposition? Do they desire lower cost, less investment, lower risk, or better quality?

# Paso 3: Los dolores

1. How do your customers define too costly? Takes a lot of time, costs too much money, or requires substantial efforts?
2. What makes your customers feel bad? What are their frustrations, annoyances, or things that give them a headache?
3. How are current value propositions under performing for your customers? Which features are they missing? Are there performance issues that annoy them or malfunctions they cite?
4. What are the main difficulties and challenges your customers encounter? Do they understand how things work, have difficulties getting certain things done, or resist particular jobs for specific reasons?
5. What negative social consequences do your customers encounter or fear? Are they afraid of a loss of face, power, trust, or status?
6. What risks do your customers fear? Are they afraid of financial, social, or technical risks, or are they asking themselves what could go wrong?
7. What's keeping your customers awake at night? What are their big issues, concerns, and worries?
8. What common mistakes do your customers make? Are they using a solution the wrong way?
9. What barriers are keeping your customers from adopting a value proposition? Are there upfront investment costs, a steep learning curve, or other obstacles preventing adoption?

# Generación de Propuesta (10 minutos)

1. Características
2. Stakeholders
3. Diferenciadores
4. Alcance
5. Qué requiero
6. Qué riesgos existen

Preguntas

