Ingeniería de Software I (TA046)/Métodos y Modelos en Ingeniería de Software I (95.20)

Capítulo 1: Introducción a la ingeniería de software

Agosto 2024

Objetivo

- Presentar un panorama general de la ingeniería de software y su contexto.

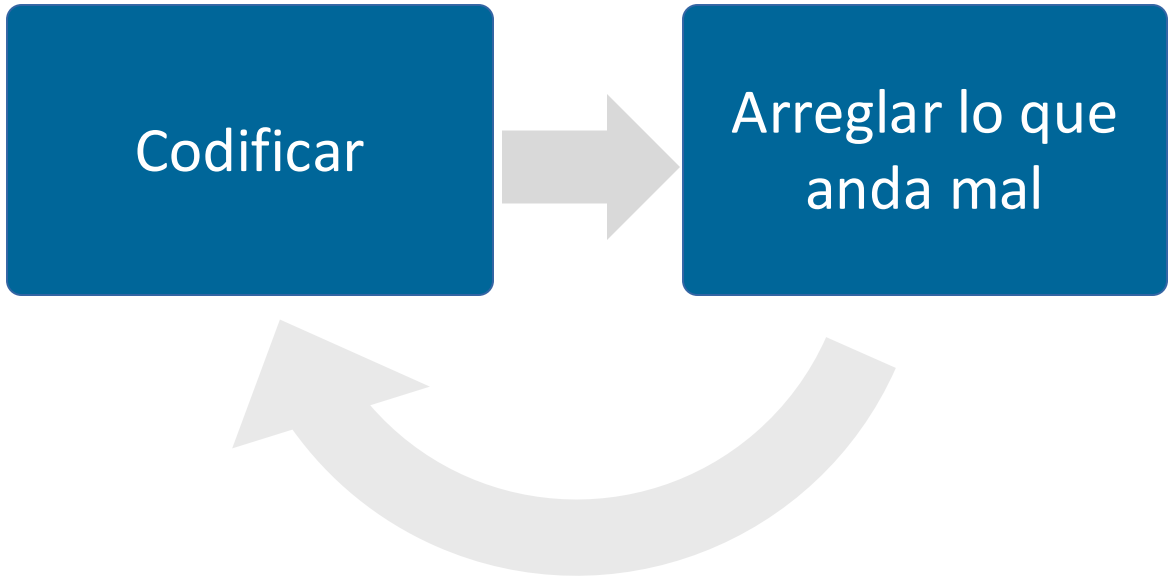
Contenidos

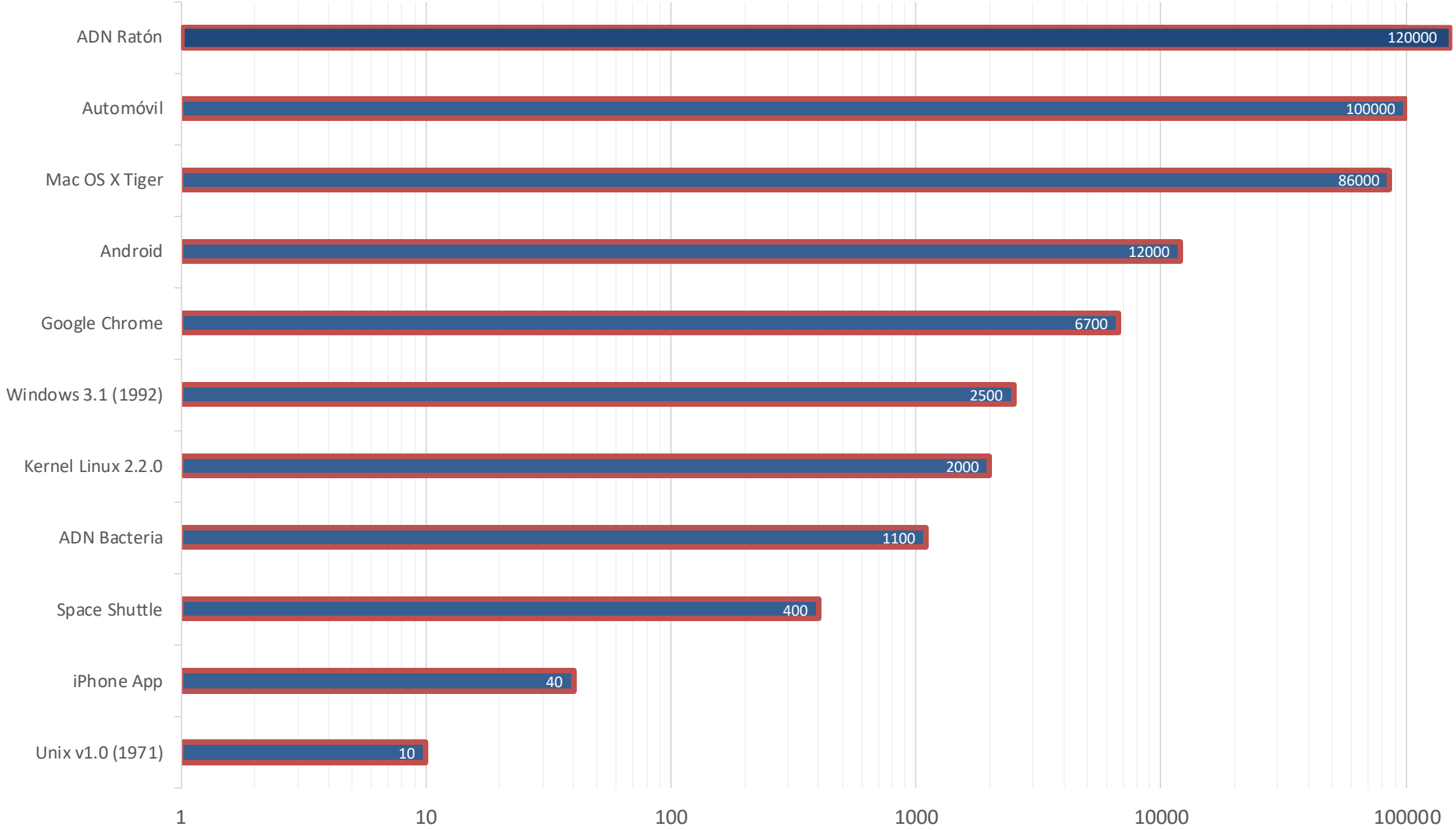
- Introducción
- Una ingeniería distinta
- Cuerpo de conocimiento
- Modelos de proceso
- Modelos en el desarrollo de software
- Cierre

Introducción

Economía digital, ingeniería de software y procesos de desarrollo

¿Qué es la ingeniería de software?





1 MLOC = 18000 páginas de texto

http://www.informationisbeautiful.net/visualizations/million-lines-of-code/

Desarrollo de software a escala industrial

OS/360, una de las primeras experiencias

"The original 1964 budget for OS/360 was **$25M**, enough to fund a team of **twelve "program designers"** who led a team of **sixty programmers** in implementing 40 "functional segments" of code. By October 1965, the team had grown to **150 programmers** and the expected shipping date had been delayed 6 months. By 1966, IBM had **over 1,000 people working on the project**. Between 1963 and 1966, more than **5,000 person-years** were poured into the design, implementation, and documentation of OS/360. Despite these resources, **the project shipped over a year late — with bugs**."

Más información:

https://spectrum.ieee.org/terahertz-2668950242

https://sourcegraph.com/blog/the-ibm-system-360-the-first-modular-general-purpose-computer#

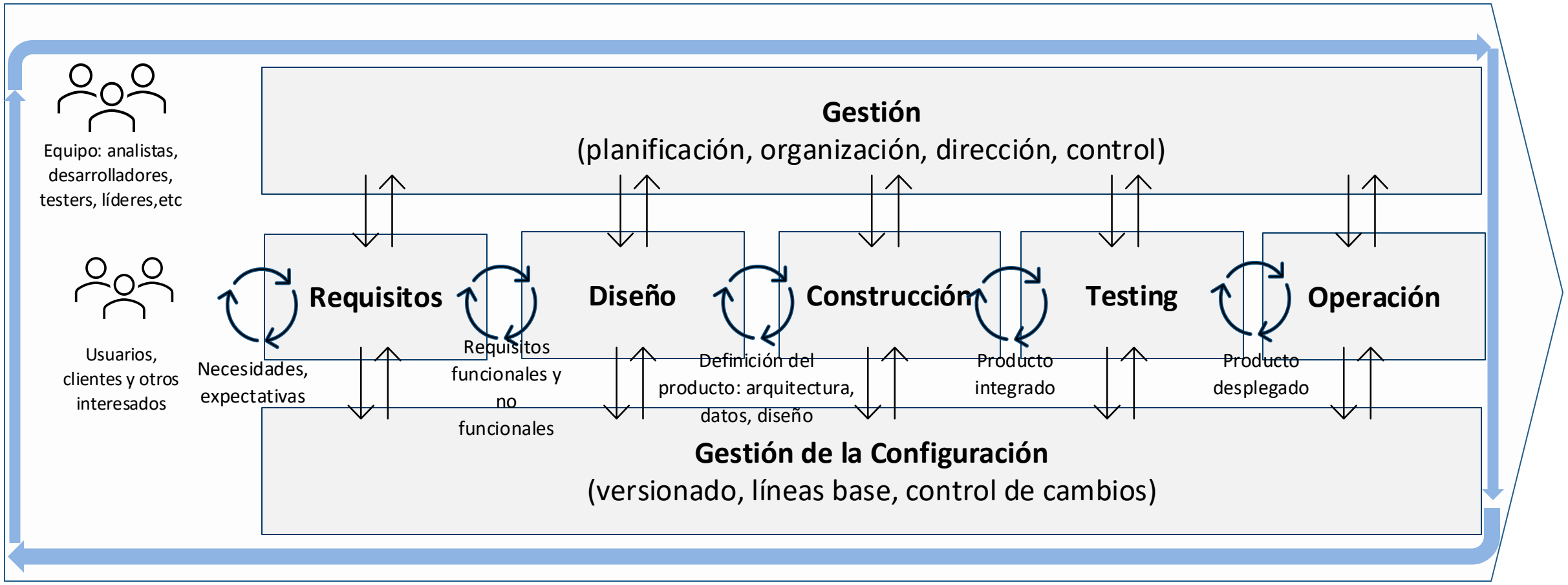
¿Qué es la ingeniería de software?

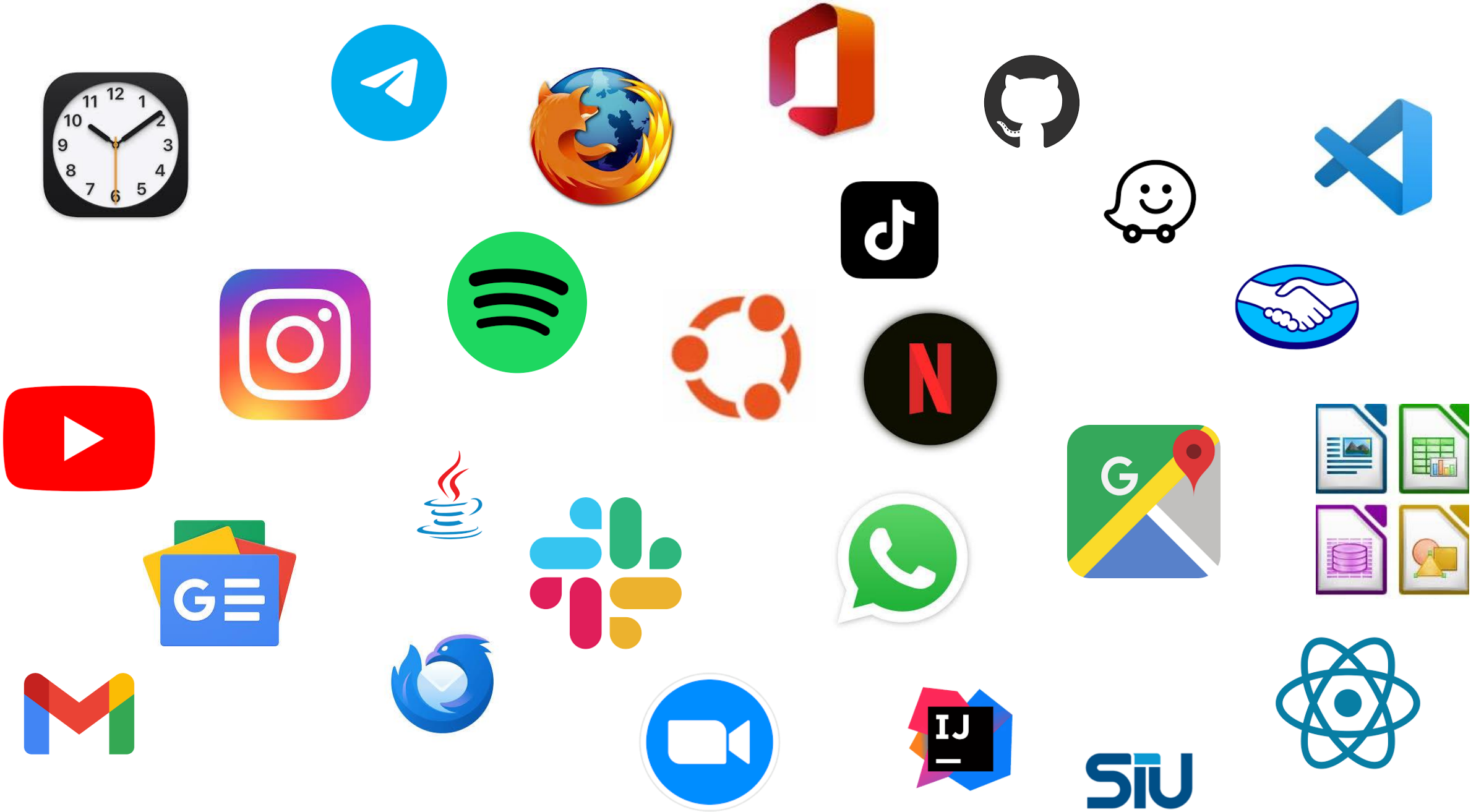
	Pequeño (académico)	Grande (industria)
Cantidad de desarrolladores	1-4	Docenas, centenas, millares
Planificación, organización	Mínima, simple	Elaborada
Duración	3/ 4 meses	A menudo, años
Requisitos	Bien definidos	A menudo, poco claros, conflictivos
Complejidad	Relativamente simple	Usualmente, muy alta
Cambios	Normalmente, no	Muchos a lo largo del tiempo
Testing	Simple, mínimo	Extenso
Documentación	Mínima	Puede ser mucha
Alta Robustez	No necesaria	Necesaria
Alta disponibilidad	No necesaria	Necesaria

https://slideplayer.com/slide/9685686/

¿Qué es la ingeniería de Software?

Múltiples disciplinas, trabajo iterativo e incremental

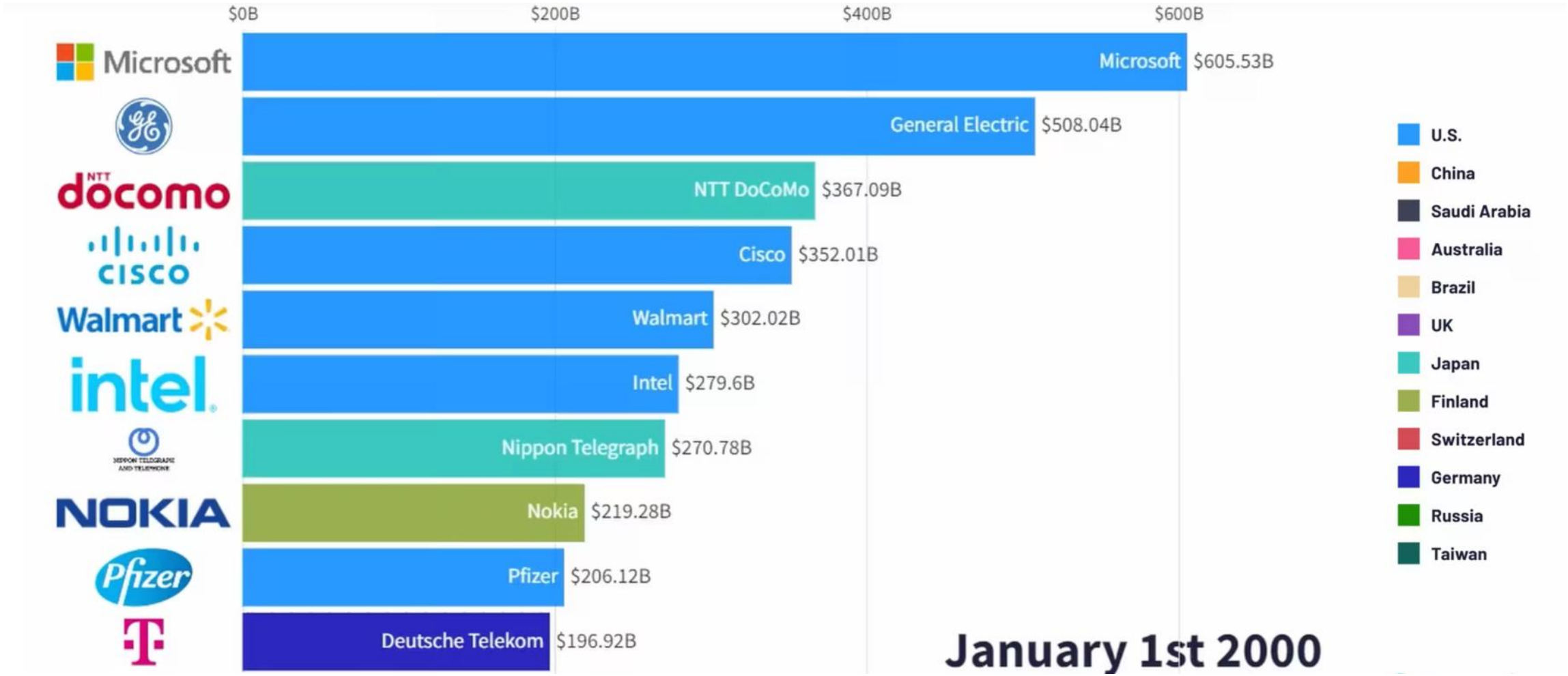




"More and more major businesses and industries are being run on software and delivered as online services — from movies to agriculture to national defense."

Marc Andreessen, creador de Netscape, cofundador y socio de Andreessen-Horowitz, firma con inversiones en Facebook, Groupon, Skype, Twitter, Zynga y Foursquare, entre otros. Accionista de LinkedIn.

https://a16z.com/2011/08/20/why-software-is-eating-the-world/



https://www.visualcapitalist.com/cp/largest-companies-from-2000-to-2022/

https://www.visualcapitalist.com/cp/largest-companies-from-2000-to-2022/

Worldwide IT Spending Forecast (Millions of U.S. Dollars)

	2023 Spending	2023 Growth (%)	2024 Spending	2024 Growth (%)
Data Center Systems	236,098	4.0	293,091	24.1
Devices	692,784	-6.5	730,125	5.4
Software	974,089	11.5	1,096,913	12.6
IT Services	1,503,698	4.9	1,609,846	7.1
Communications Services	1,491,733	3.2	1,537,188	3.0
Overall IT	4,898,401	3.8	5,267,163	7.5

https://www.gartner.com/en/newsroom/press-releases/2024-07-16-gartner-forecasts-worldwide-it-spending-to-grow-7-point-5-percent-in-2024#:~:text=Worldwide%20IT%20spending%20is%20expected,spend%20forecast%20of%20%245.06%20trillion.

VUCA

https://hbr.org/2014/01/what-vuca-really-means-for-you?utm_medium=paidsearch&utm_source=google&utm_campaign=intlcontent_strategy&utm_term=Non-Brand&tpcc=intlcontent_strategy&gad_source=1&gclid=CjwKCAiAvdCrBhBREiwAX6-6UqcaMg-mh3G0ibByx-lgCaoULMHeKxlS49Lvcq6zU3z5w6ASuNOGARoCfO0QAvD_BwE

"…your organization's software delivery capability can in fact provide a **competitive advantage** to your business"

"…high performers were also twice as likely to exceed objectives in quantity of goods and services, operating efficiency, customer satisfaction, quality of products or services, and achieving organization or mission goals. "

Ingeniería de software: una ingeniería distinta

¿Ingeniería o desarrollo de software?

Ingeniería

Creación de soluciones costo efectivas a problemas prácticos mediante la aplicación de conocimiento codificado para construir cosas al servicio de la humanidad.

Ingeniería de software

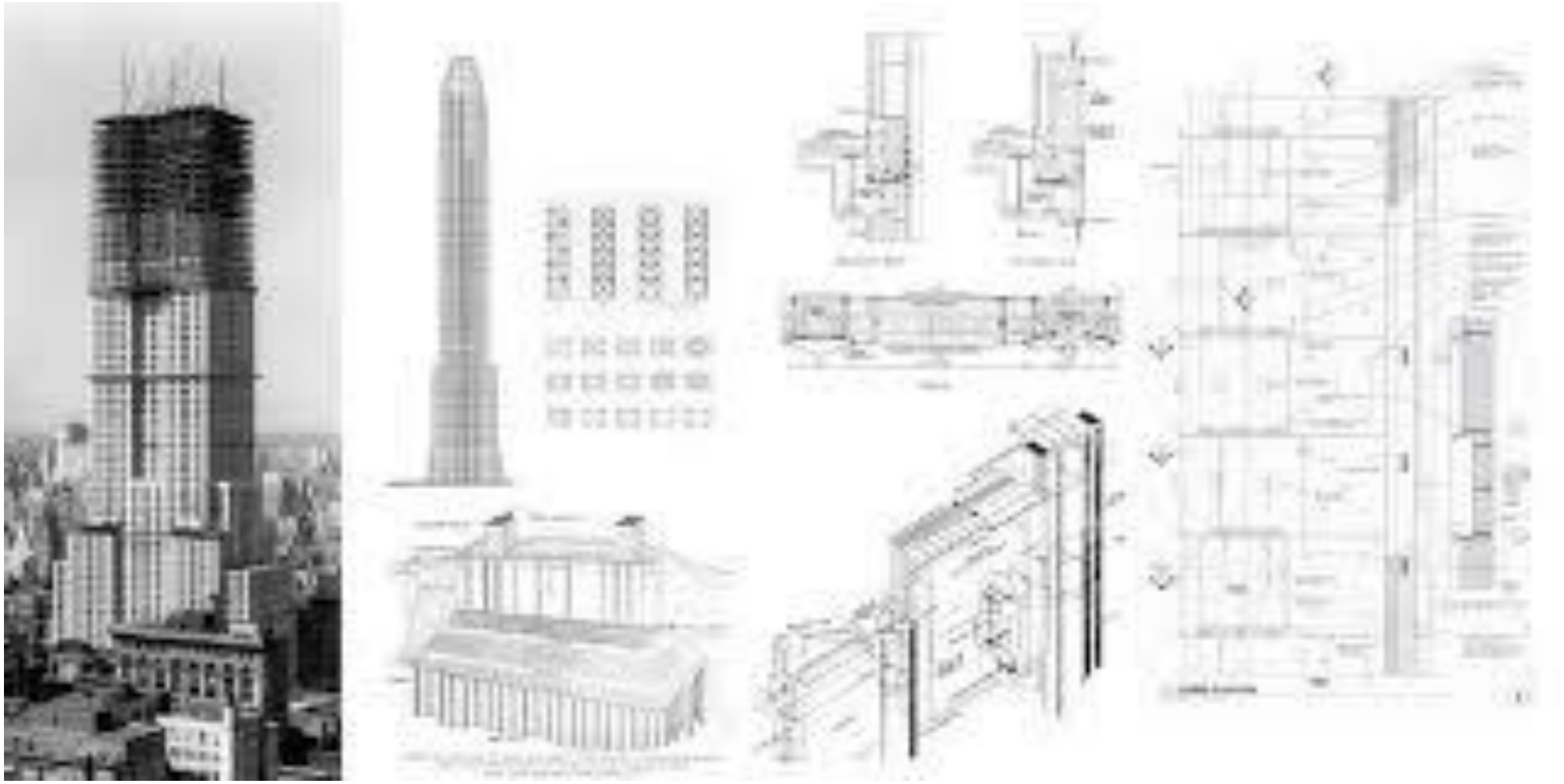
Aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería al software.

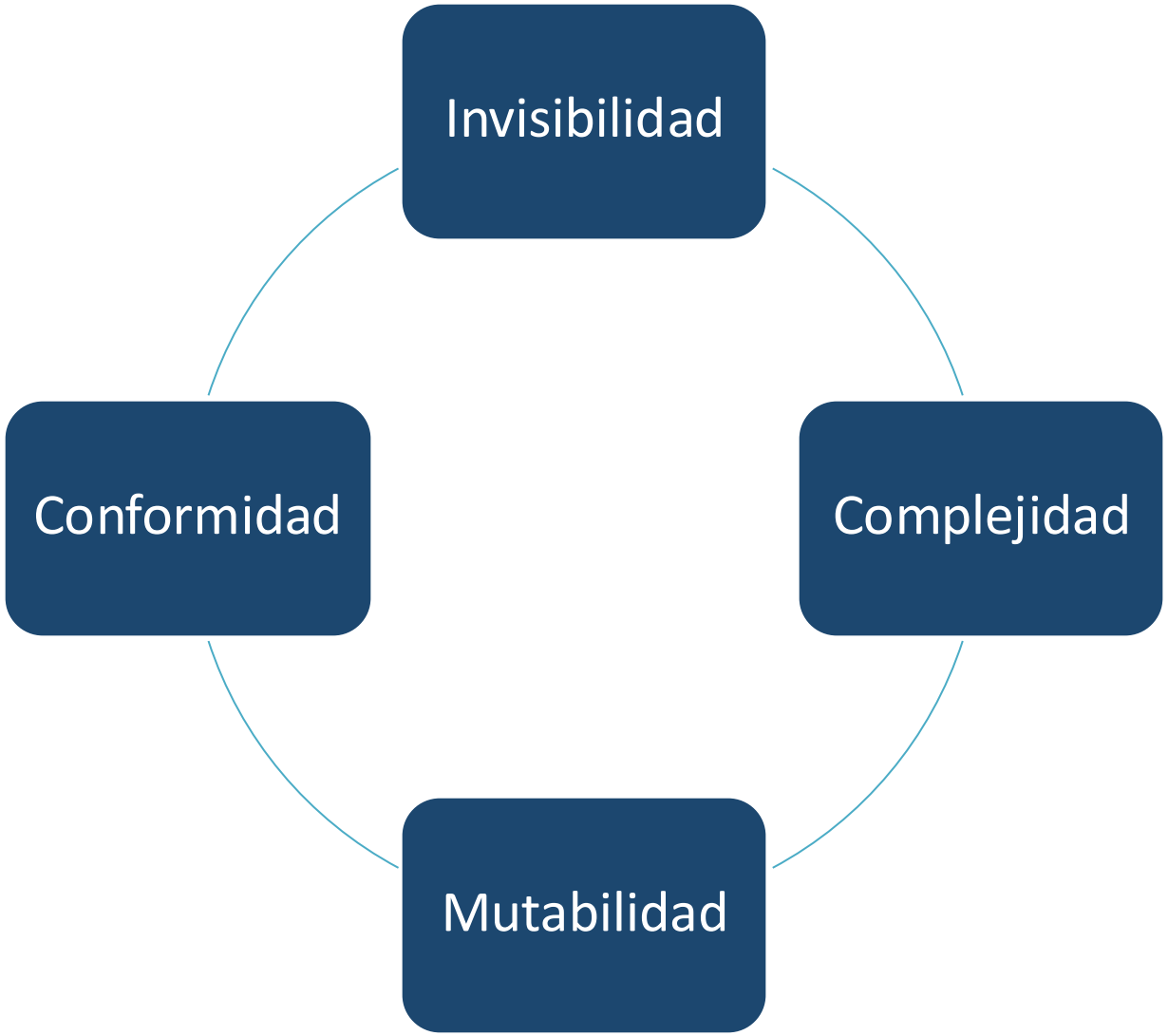
www.computer.org/sevocab

Ingeniería (desarrollo) de Software

Una ingeniería de cosas abstractas.

Las leyes de la física no aplican





Frederick Brooks, *The Mythical Man-Month – Anniversary Edition;* Addison Wesley, 1995

Ingeniería Tradicional	Ingeniería de Software
Producto **tangible**	Producto **intangible**
Ingenieros **diseñan**, **operarios**/máquinas **fabrican**	Ingenieros **diseñan** y **construyen**
Planificación y diseño **completos** antes de fabricación	Planificación y diseño naturalmente **evolutivos**
Replicación de un mismo diseño	Cada **producto** es **único**
Cuerpo de conocimiento basado en física, química, matemáticas	Cuerpo de conocimiento propio

Desarrollo de software

Casi todo es diseño

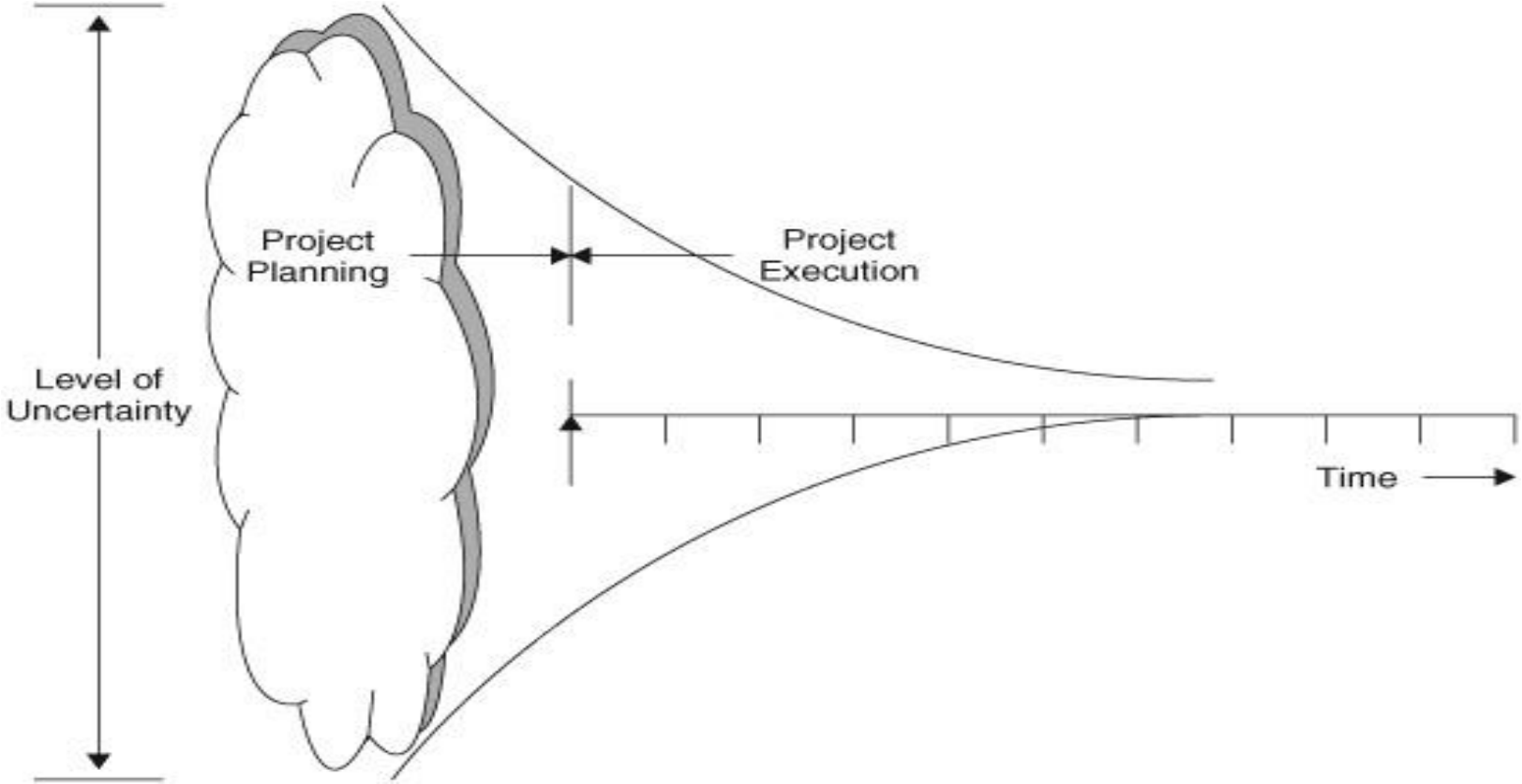
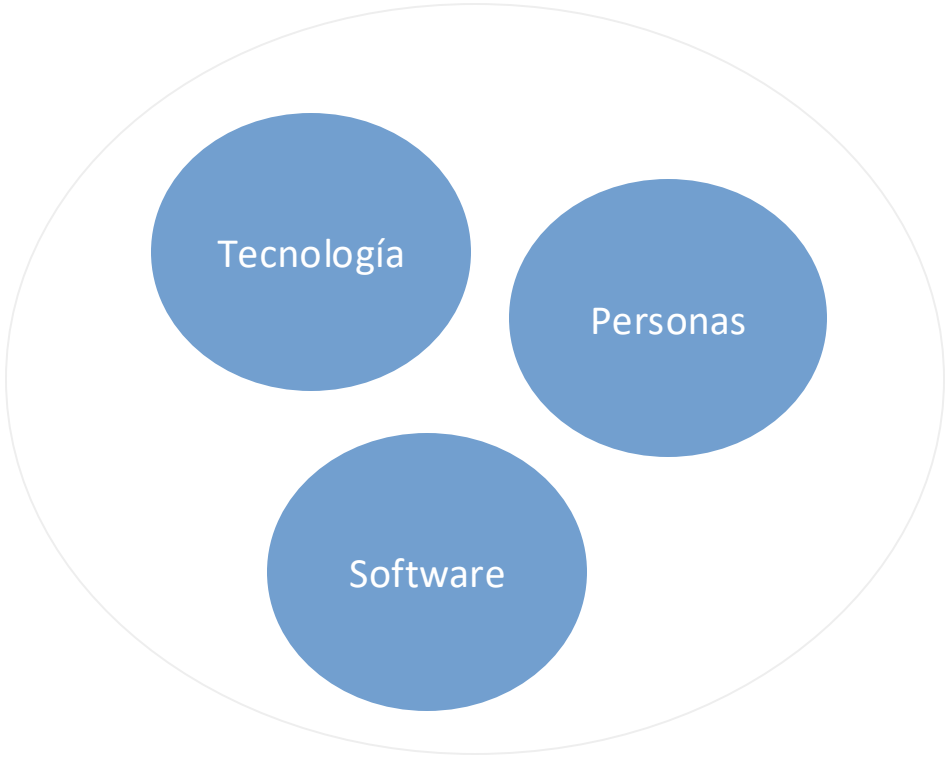


Figura reproducida de Making Sense of Agile Project Management: Balancing Control and Agility por Charles G. Cobb



sistema.
(Del lat. systēma, y este del gr. σύστημα).

1. m. Conjunto de reglas o principios sobre una materia racionalmente enlazados entre sí.

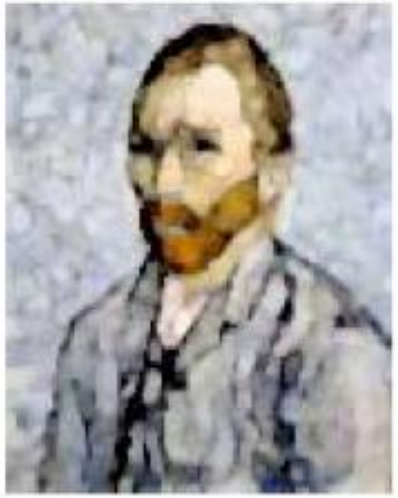
2. m. Conjunto de cosas que relacionadas entre sí ordenadamente contribuyen a determinado objeto.

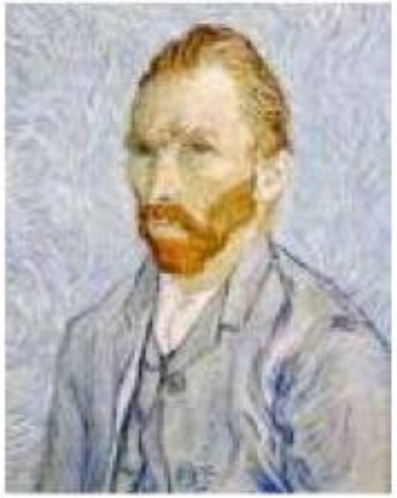
3. m. Biol. Conjunto de órganos que intervienen en alguna de las principales funciones vegetativas. Sistema nervioso.

4. m. Ling. Conjunto estructurado de unidades relacionadas entre sí que se definen por oposición; p. ej., la lengua o los distintos componentes de la descripción lingüística.

Desarrollo de software

Un proceso iterativo e incremental

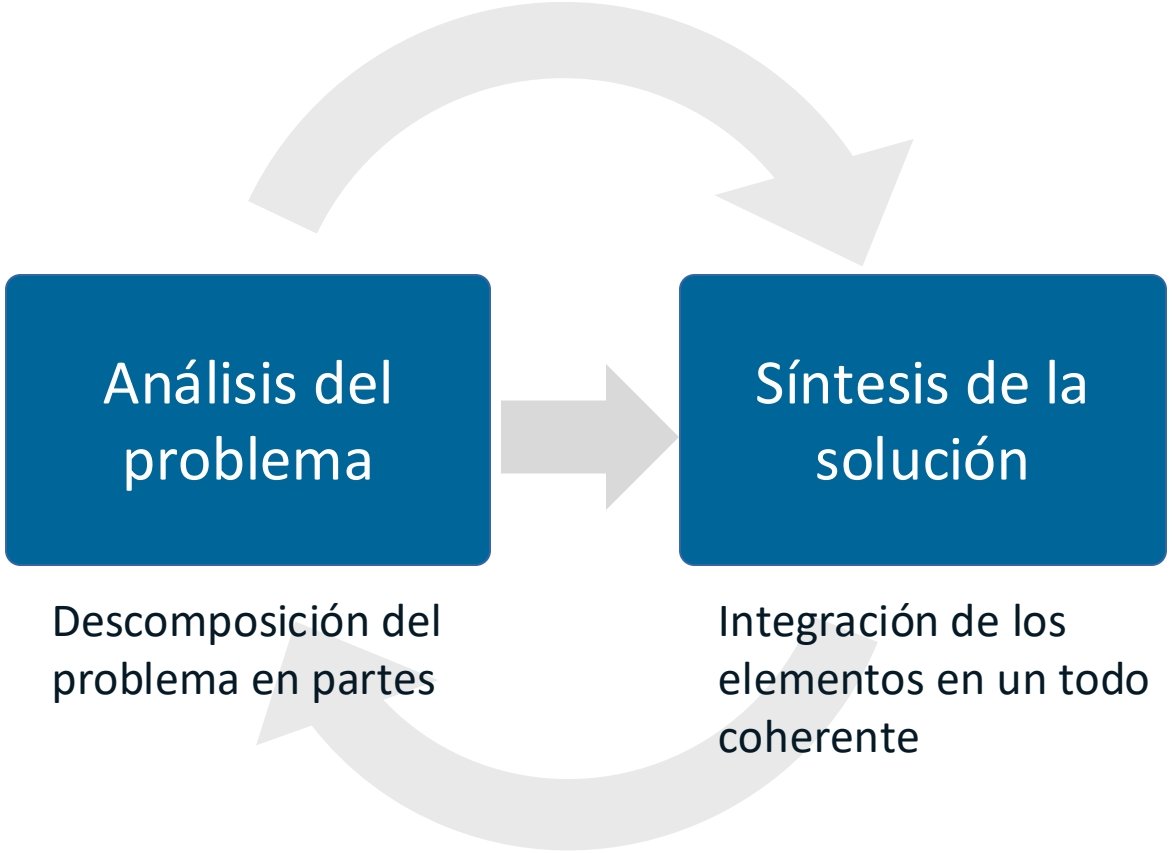




Pictures © Xavier Quesada Allue

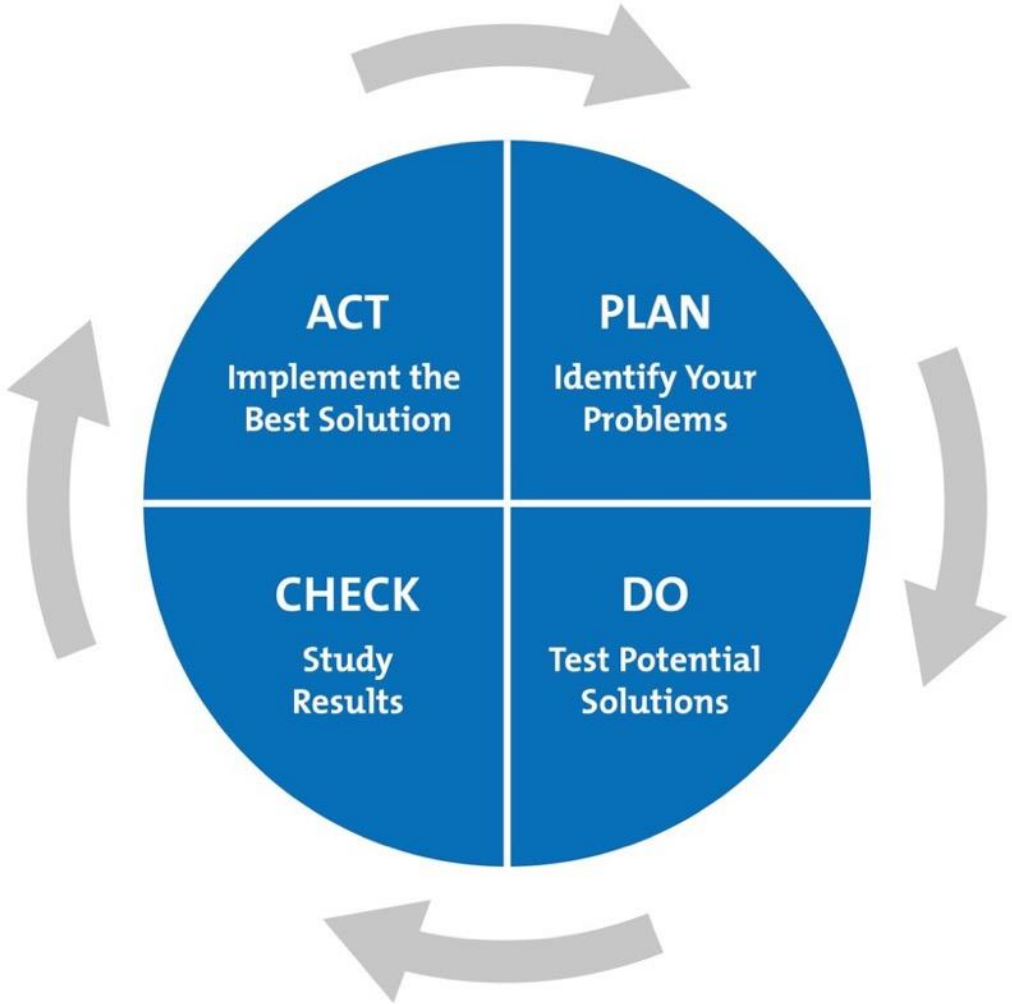
Desarrollo de software

Un proceso iterativo e incremental



Desarrollo de software

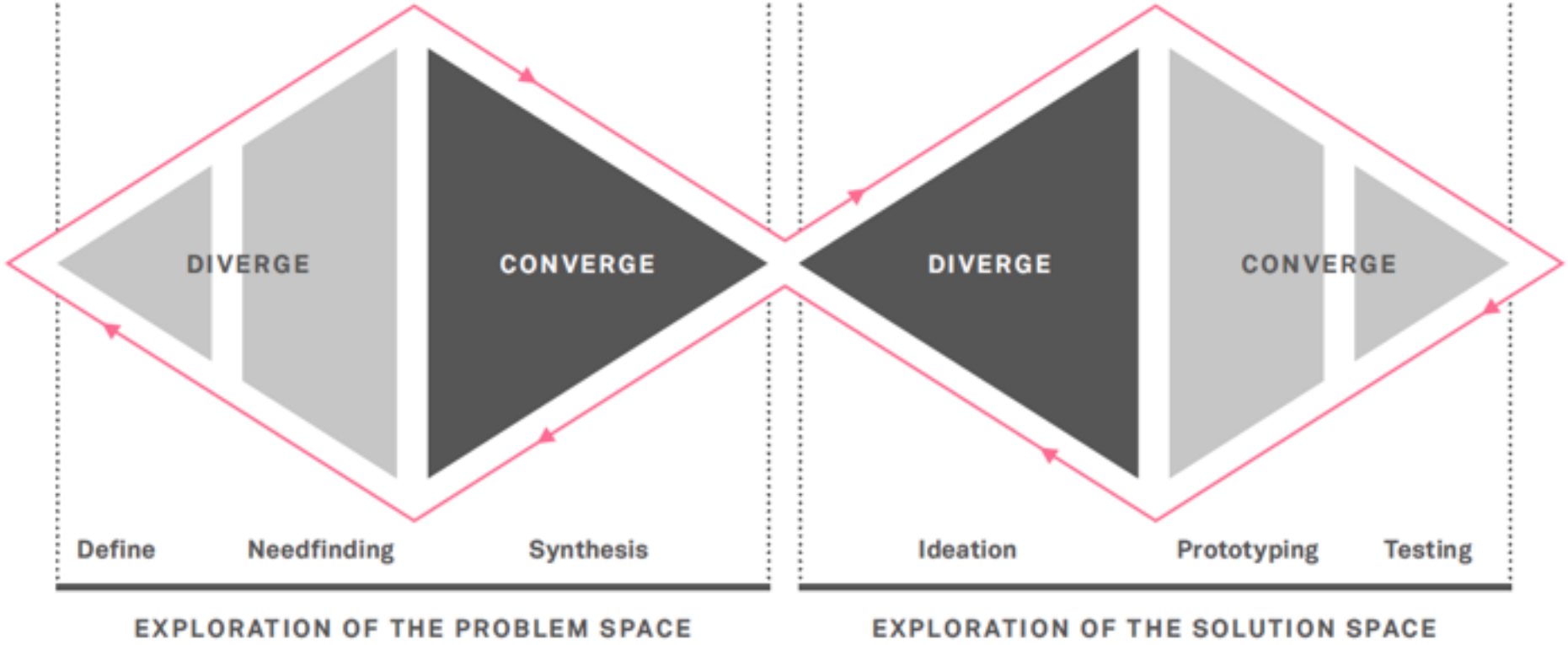
Un proceso iterativo e incremental



Reproducido de https://www.mindtools.com/media/Diagrams/PDCA2017.jpg

Desarrollo de Software

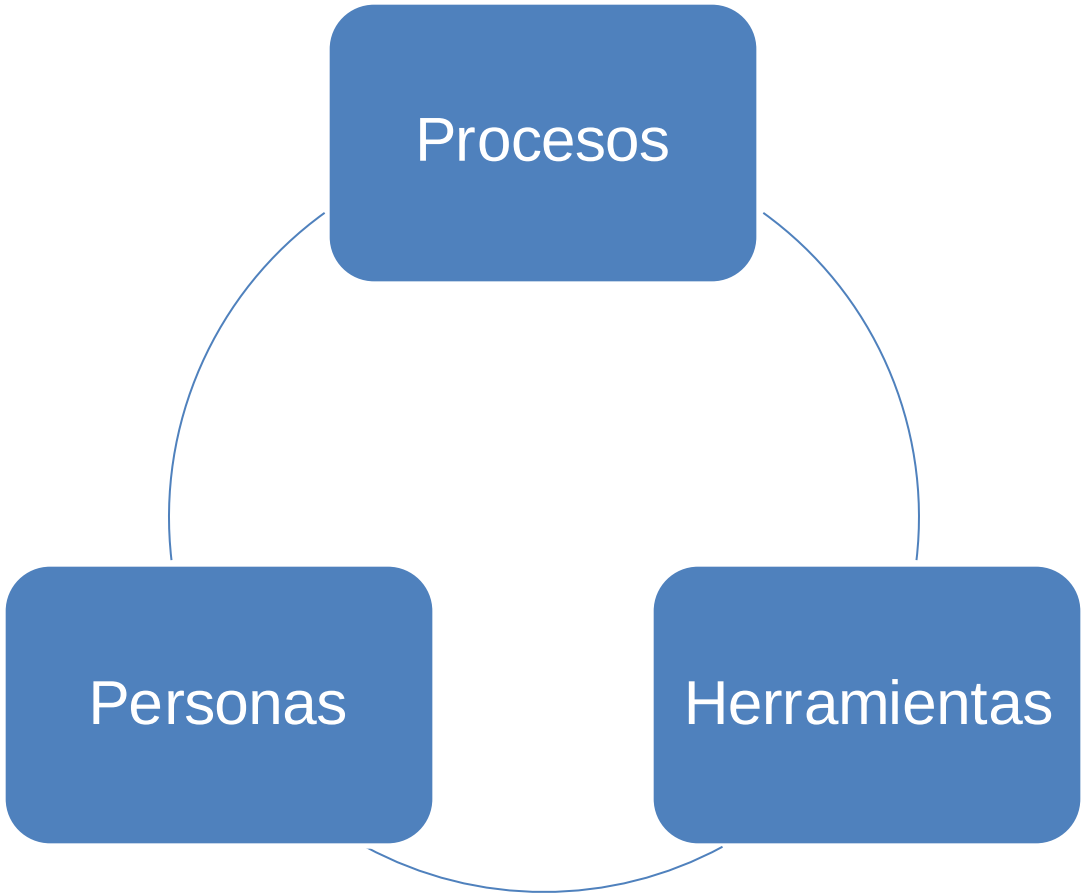
¡Siempre tendremos que lidiar con la **ambigüedad**!



Reproducido del tutorial *DT4RE: Design Thinking for RE,* por Jennifer Hehn, Falk Uebernickel, Daniel Méndez

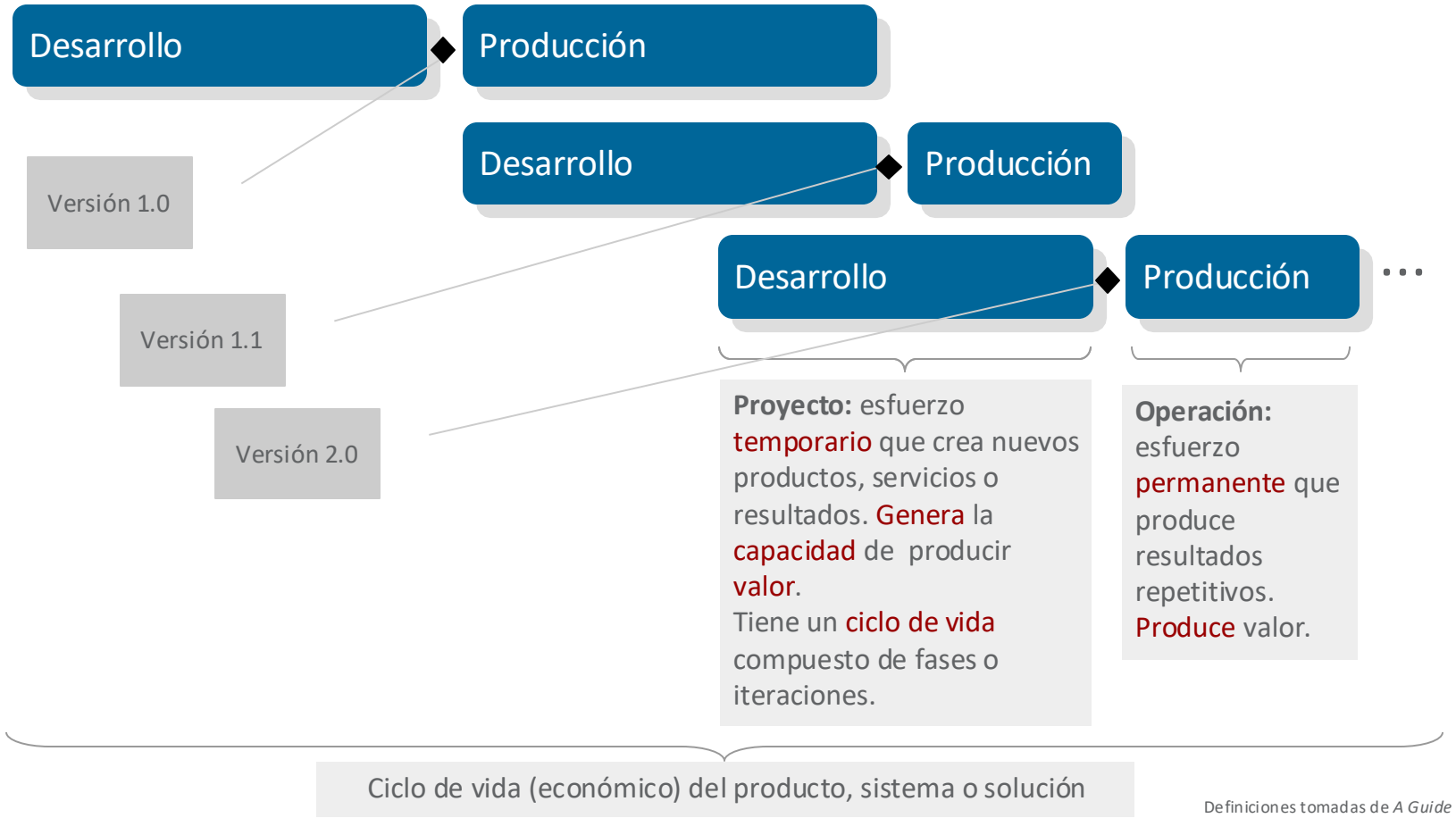
Desarrollo de software

Es el resultado de sistemas sociotécnicos complejos



Desarrollo de software

Enfoque discreto



Definiciones tomadas de *A Guide to the project management body of knowledge (PMBOK)*

Desarrollo de software

Enfoque continuo

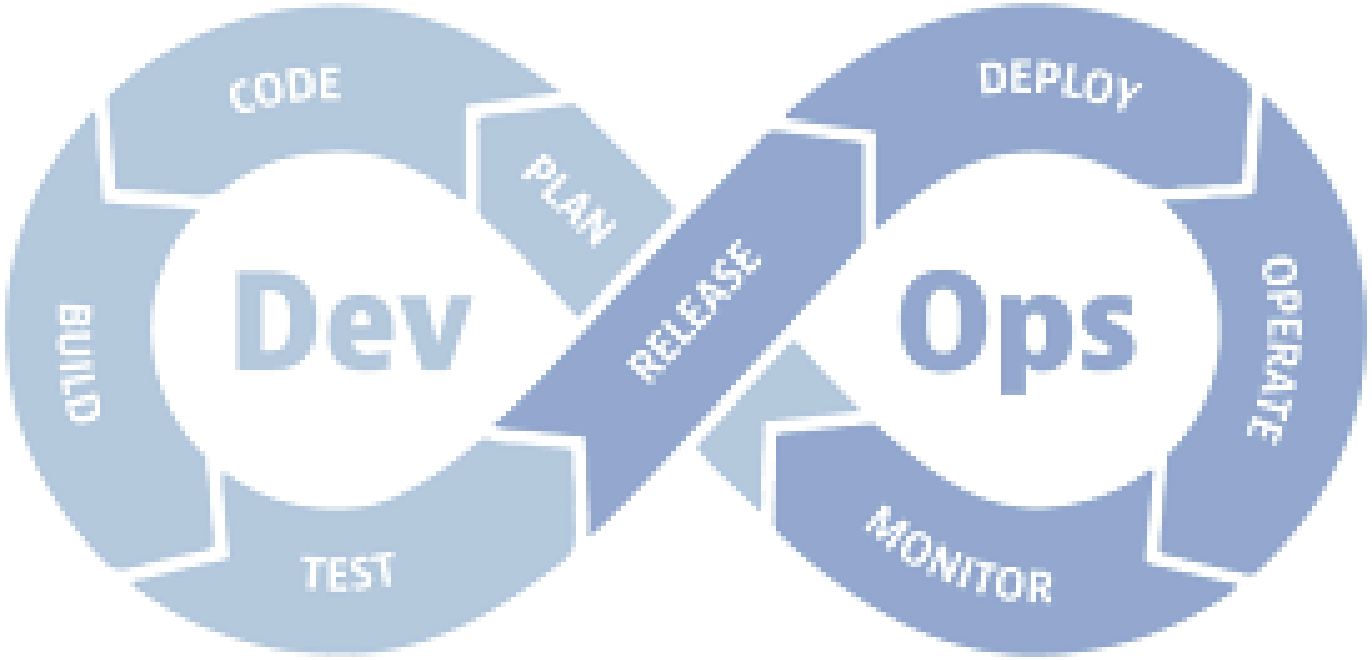


Imagen: https://www.dynatrace.com/news/blog/what-is-devops/

Desarrollo de software

Es el resultado de sistemas sociotécnicos complejos

"Software design is an exercise in human relationships"

Kent Beck, creador de *Extreme Programming*

Cuerpo de conocimiento

Software Engineering Body of Knowledge (SWEBOK) v4.0

Cuerpo de Conocimiento

Conjunto completo de conceptos, términos y actividades que conforman una profesión.

Ingeniería de software

Áreas de conocimiento

SWEBOK®

Requisitos	Operaciones	Modelos y métodos
Arquitectura	Mantenimiento	Calidad
Diseño	Gestión de la configuración	Seguridad
Construcción	Gestión de Ingeniería de Software	Ejercicio de la profesión
Testing	Procesos de Ingeniería de Software	Economía de la Ingeniería de Software
Fundamentos de computación	Fundamentos matemáticos	Fundamentos de ingeniería

Áreas de conocimiento

SWEBOK®

Requisitos

Descubrimiento, análisis, especificación, validación y administración de **requisitos del software**.

Requisito: una descripción de cómo el sistema debe comportarse o de una propiedad o atributo que debe poseer.

Durante una llamada, el teléfono deberá indicar con un tono suave la entrada de otra llamada.

Áreas de conocimiento

SWEBOK®

Arquitectura

Definición de la **arquitectura**, los componentes y las interfaces de un producto de software.

Arquitectura (de un sistema): conceptos y propiedades fundamentales de un sistema en su entorno, plasmados en sus elementos, relaciones y en los principios de su diseño y evolución.

Áreas de conocimiento

SWEBOK®

Diseño

Definición de las características **externas** y de la estructura **interna** del software como base para la construcción, a partir del análisis de los requisitos.

Incluye:

- el diseño de la **arquitectura** (cubierto por el área de conocimiento correspondiente);
- el diseño de alto nivel del sistema y sus componentes;
- el diseño detallado

Áreas de conocimiento

SWEBOK®

Construcción

Creación del software mediante la combinación de actividades de codificación, verificación, pruebas unitarias y pruebas de integración.

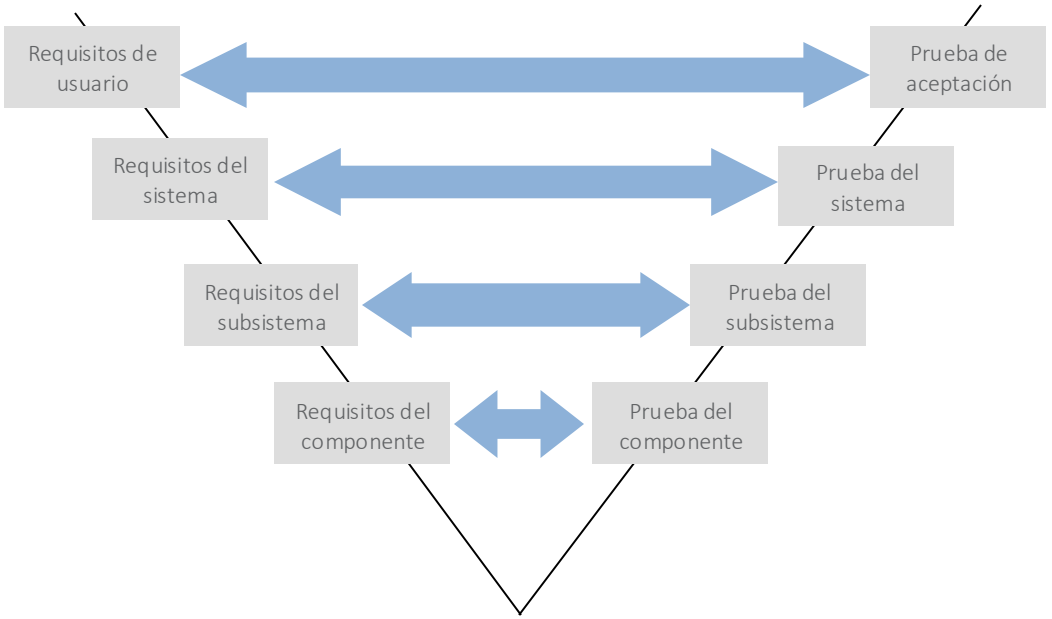
Codificar también es diseñar.

Áreas de conocimiento

SWEBOK®

Testing

Evaluación dinámica del comportamiento esperado del software en un conjunto finito de casos de prueba.



Áreas de conocimiento

SWEBOK®

Operaciones

Despliegue, operación y soporte de sistemas, manteniendo su integridad y estabilidad.

Incluye gestión de incidentes, problemas, cambios, despliegues, monitoreo, observabilidad, operaciones, etc.

Áreas de conocimiento

SWEBOK®

Gestión de Ingeniería de Software

Aplicación de actividades de gestión (planificación, organización, dirección, monitoreo, control) al desarrollo de software con el propósito de **asegurar la entrega de productos y servicios** de manera eficiente y efectiva.

Áreas de conocimiento

SWEBOK®

Gestión de la Configuración del Software

Identificación de la configuración de un producto de software en momentos específicos con el fin de mantener su integridad y trazabilidad a lo largo del ciclo de vida.

Configuración: un conjunto de elementos (no necesariamente software) que forman parte del producto/sistema, combinados de acuerdo a procedimientos específicos.

Áreas de conocimiento

SWEBOK®

Métodos y modelos

Los métodos proporcionan un enfoque sistemático para la especificación, diseño, construcción, prueba y verificación.

Los modelos ayudan a entender, definir y comunicar.

Áreas de conocimiento

SWEBOK®

Procesos de Ingeniería de Software

Un proceso es un conjunto de actividades que transforma una entrada en una salida y que consume recursos.

Áreas de conocimiento

SWEBOK®

Calidad

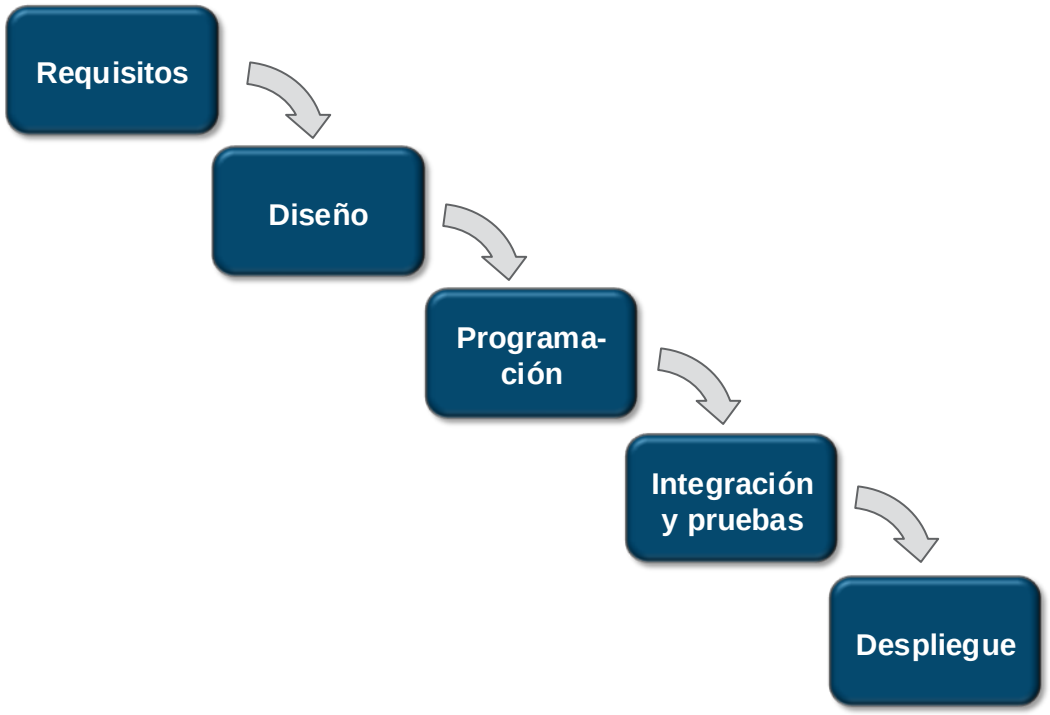
Prácticas, herramientas y técnicas para **comprender** la calidad del software y planificar y evaluar el estado de la calidad del software durante el desarrollo, mantenimiento y operación, tanto desde la perspectiva del producto de software como desde la perspectiva del proceso de software.

Modelos de proceso

Conocidos también como ciclos de vida

Modelos del proceso de desarrollo de software

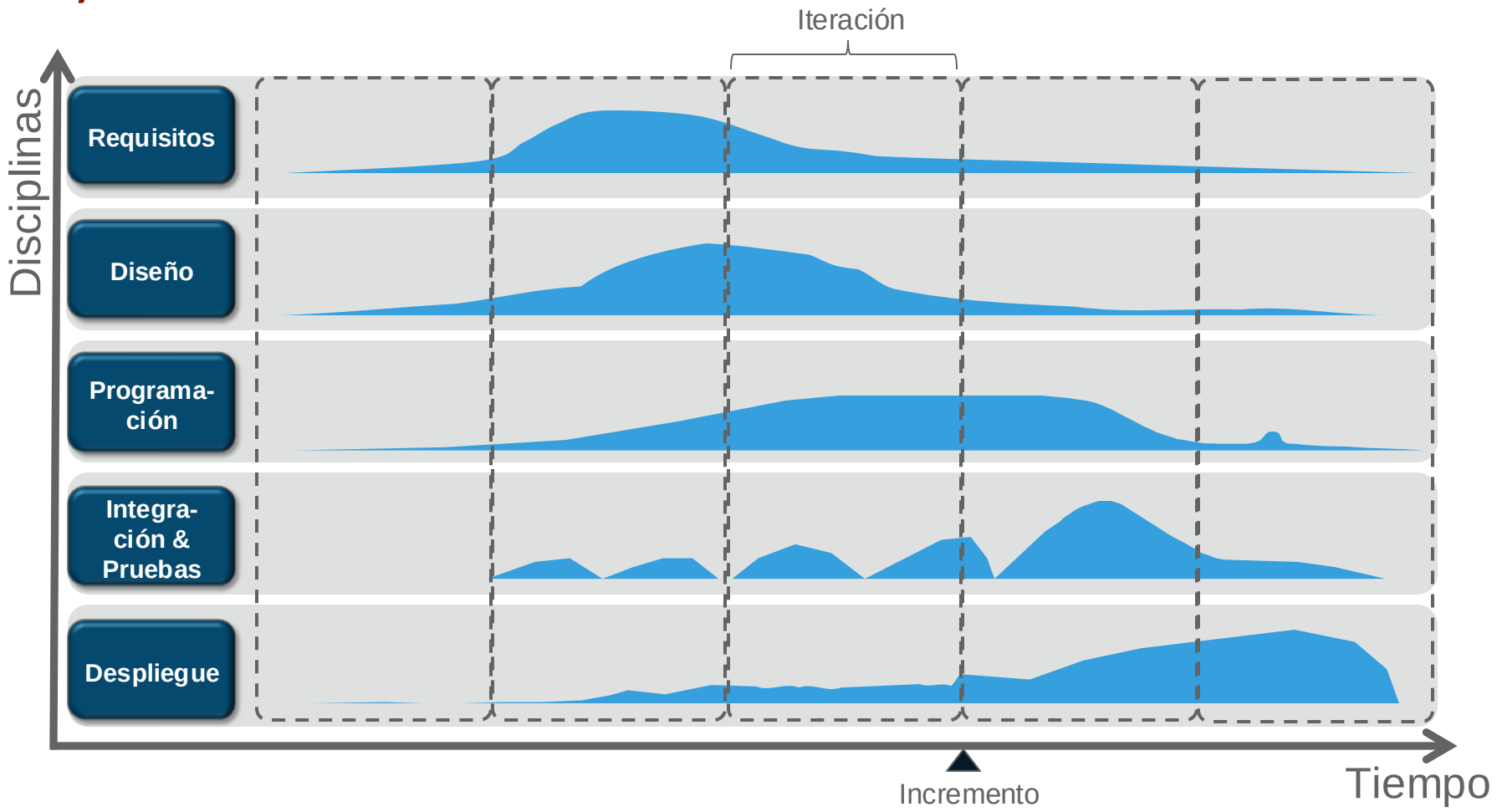
Ciclo de vida en cascada



- Proceso **secuencial**, inspirado en la ingeniería tradicional.
- Primera referencia aparecida en un paper de Herbert D. Benington (1956)
- Atribuido erróneamente a Winston Royce, quien en realidad lo daba como un ejemplo de mal proceso (1970)
- En general, aplicado en la industria sin considerar las propuestas de Royce para mejorarlo.

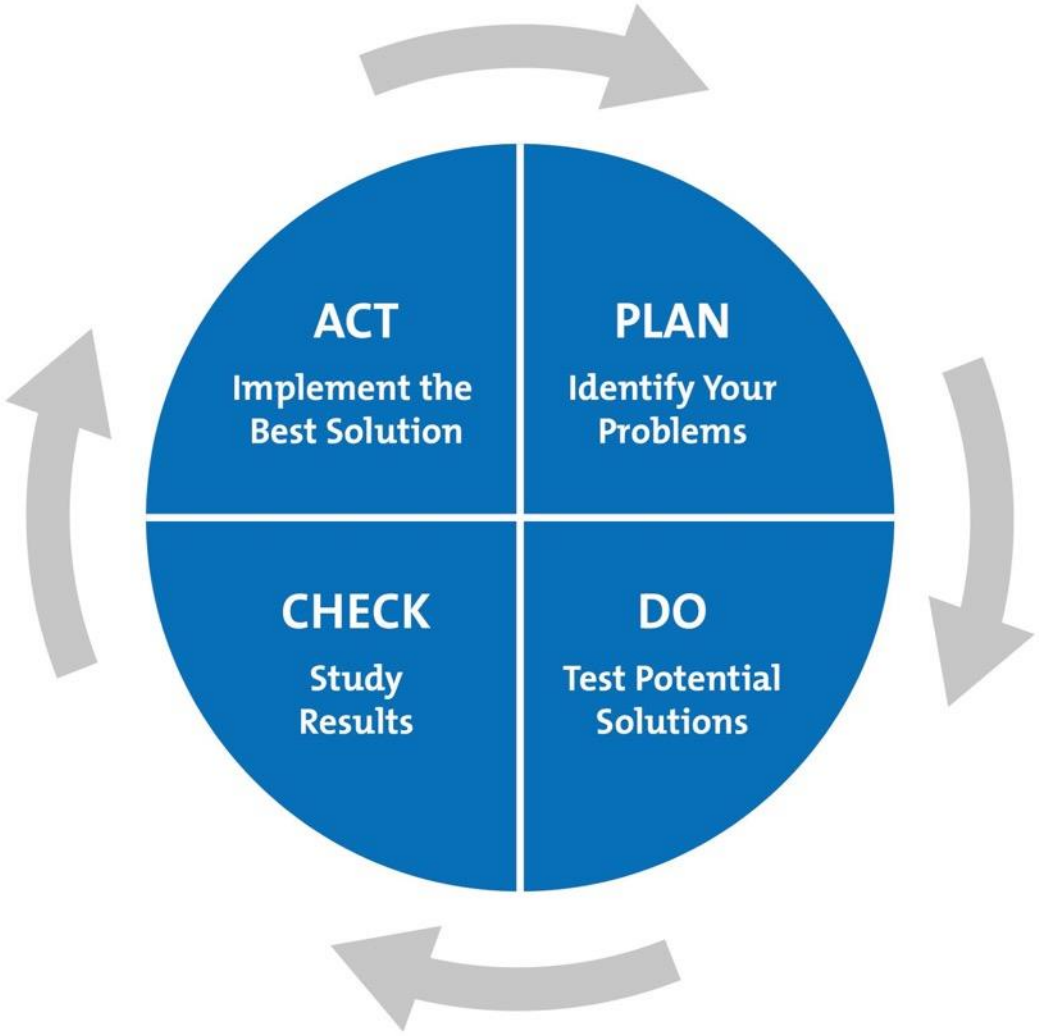
Proceso de desarrollo de software

Disciplinas y ciclo de vida



Modelos del proceso de desarrollo de software

Ciclo PDCA

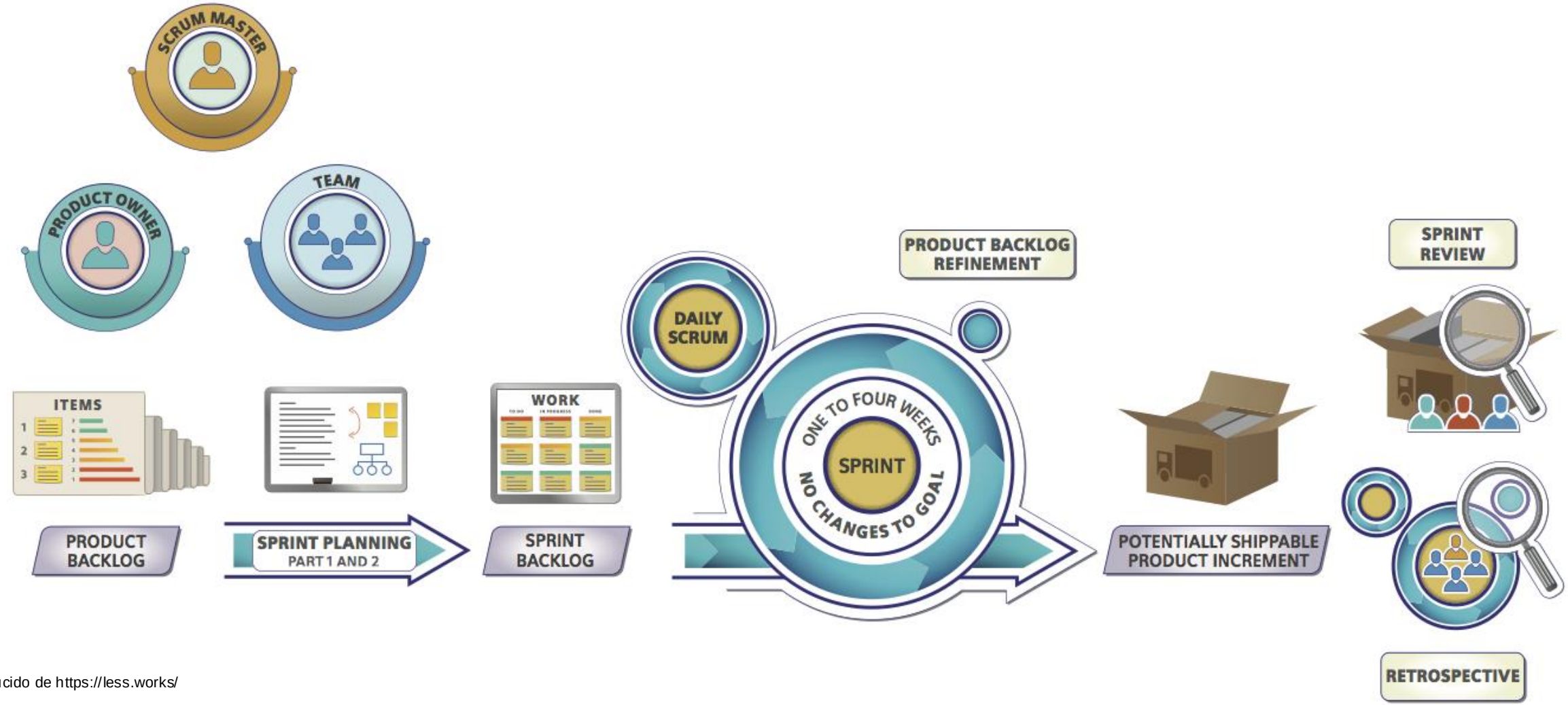


Reproducido de https://www.mindtools.com/media/Diagrams/PDCA2017.jpg

Agile Methodologies

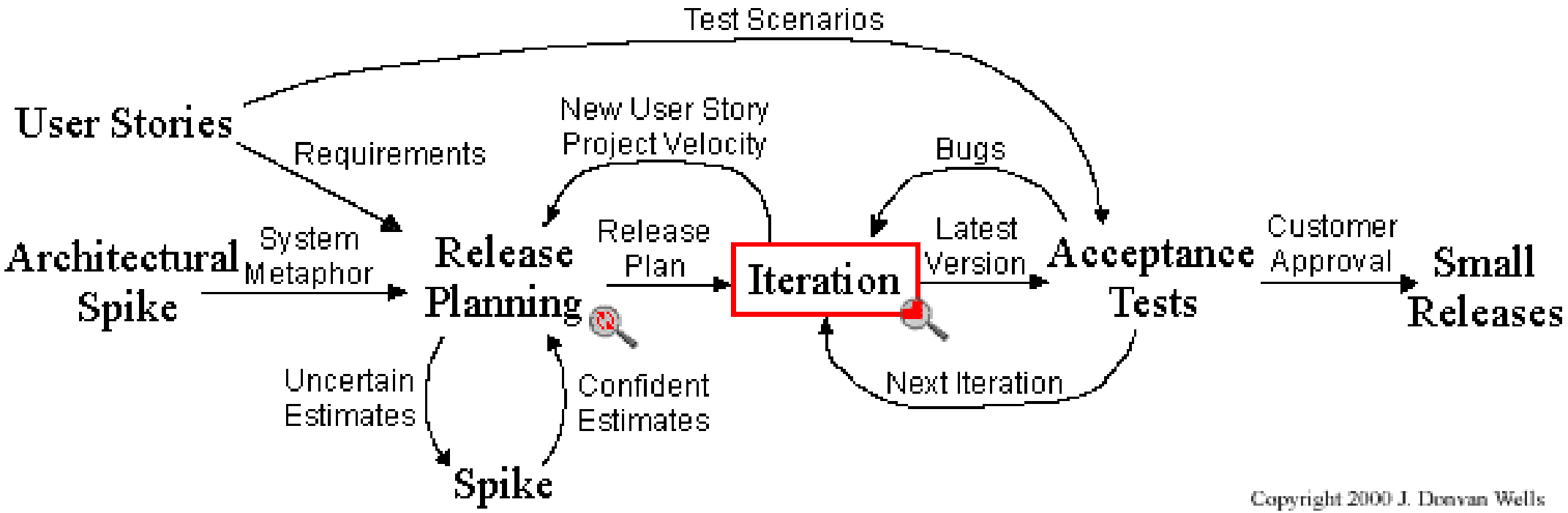
- Scrum
- Extreme Programming (XP)
- Feature Driver Development (FDD)
- Dynamic System Development Method (DSDM)
- Disciplined Agile Delivery (DAD)

Scrum



Reproducido de https://less.works/

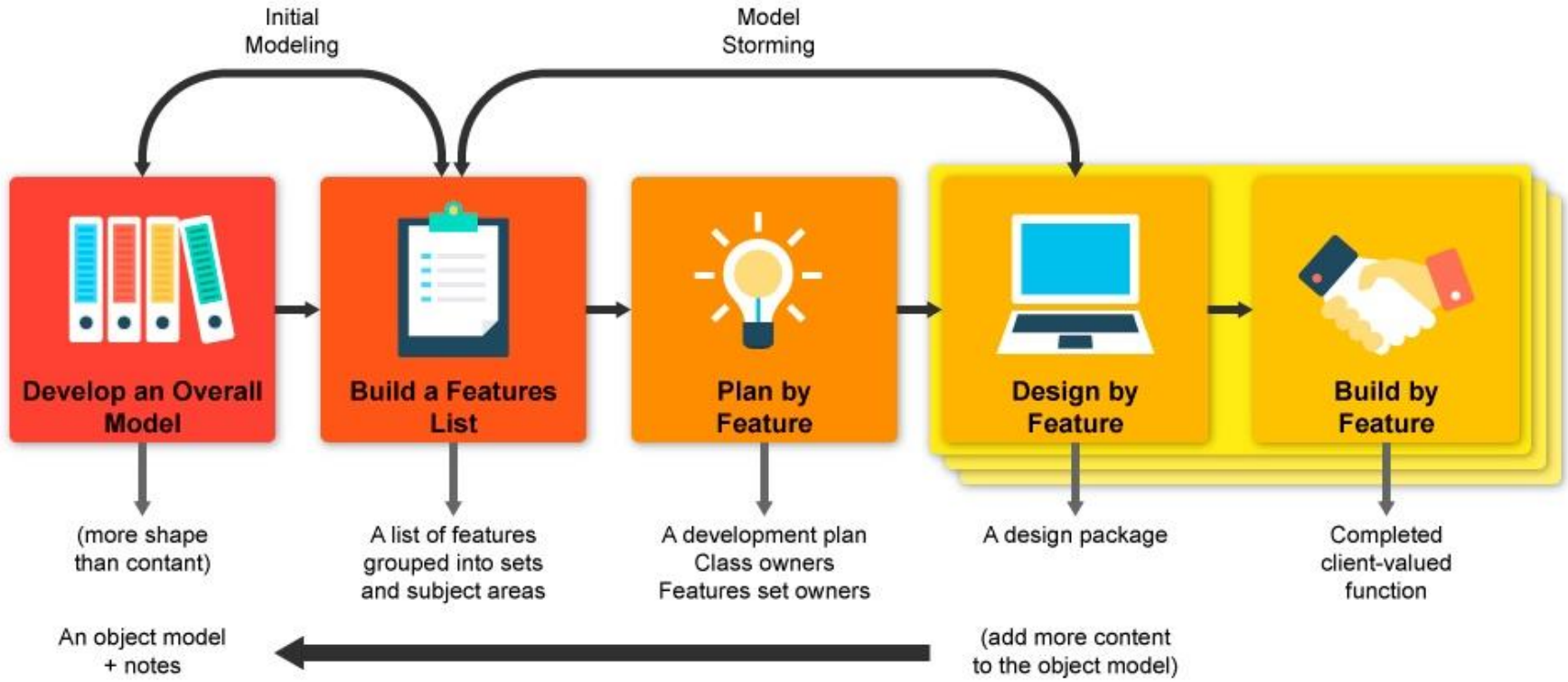
Extreme Programming



- No hay requisitos formalizados en forma escrita.
- Se emplean user stories.
- Un representante del usuario debe formar parte del equipo.
- Las iteraciones son de alrededor de dos semanas.
- No hay actividades de diseño formal al comienzo del proyecto.
- Los desarrolladores están muy preparados.

Reproducido de http://www.extremeprogramming.org

Feature Driven development (FDD)

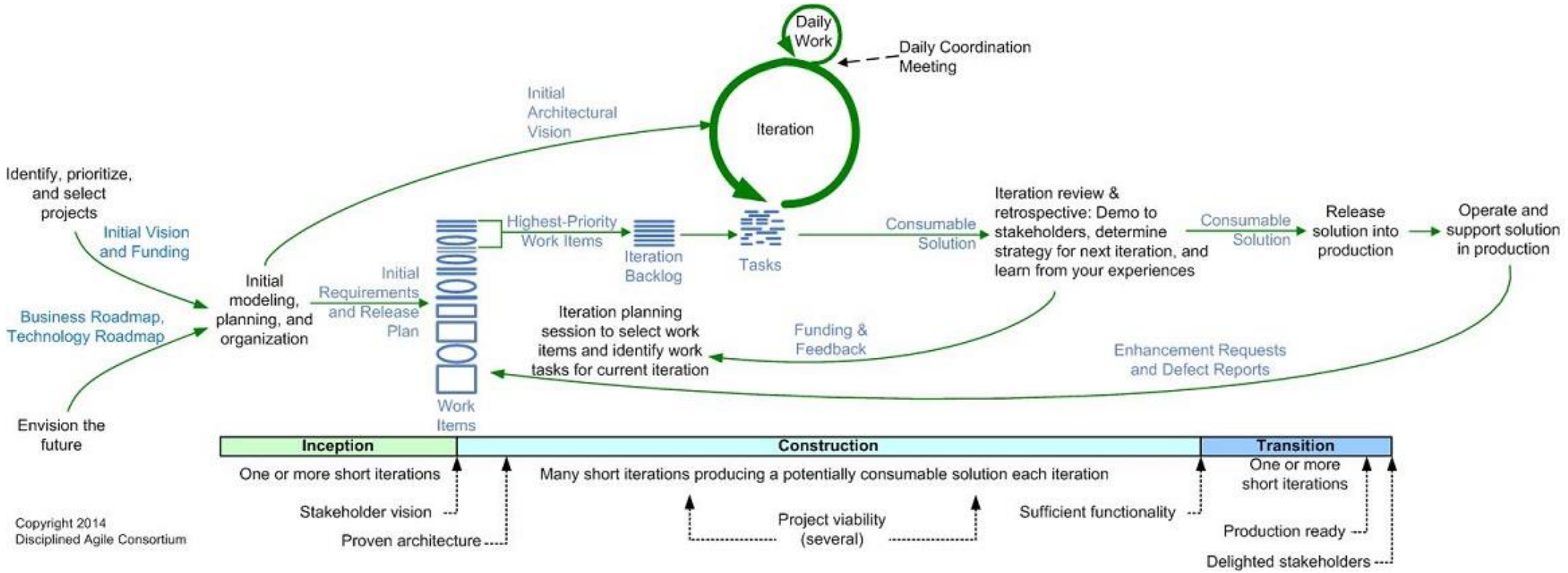


- Para equipos pequeños y grandes.
- Análisis y diseño al comienzo.

- Descomposición funcional: área, actividades de negocio, pasos (features). <acción> <objeto> <resultado>
- P.ej.: Calcular el total de la venta
- El desarrollo de cada *feature* no debe superar las dos semanas.

Reproducido de from http://newline.tech/blog/feature-driven-development-methodology/

Disciplined Agile Delivery (DAD)



- Marco de trabajo híbrido basado en prácticas y estrategias tales como Scrum, XP, Kanban, Unified Process, etc.
- DAD se desarrolló como resultado de la observación de patrones comunes en los que la agilidad se aplicó con éxito a escala.

Reproducido de http://www.ambysoft.com

1956:
Herbert D. Benington publica un paper en el que describe el proceso de desarrollo en cascada.

1970:
Paper de Winston Royce. ¡El ciclo de vida en cascada está mal!

1983:
Grady Booch publica "Software Engineering with Ada", en donde describe un proceso iterativo.

1985:
Barry Boehm publica "A Spiral Model of Software Development and Enhancement"

1990:
James Martin publica "Rapid Application Development"

2010s:
Uso generalizado.
"Slope of enlightenment".
Post-Agilismo.

2001:
Representantes de XP, Scrum, FDD, DSDM y otros se reúnen en Utah.
Origen de la "Agile Alliance"

1975:
Brooks publica The Mythical Man-Month.

1996:
Kent Beck se une al proyecto Chrysler C3, origen de XP.

2003:
Mary & Tom Poppendieck publican Lean Software Development

1950 | 1960 | 1970 | 1980 | 1990 | 2000 | 2010

1968:
Reporte de T.J.Watson Research Center (IBM) recomienda el desarrollo iterativo.

1976:
"Software Metrics", de Tom Gilb, describe EVO.

1987:
DoD cambia el STD-2167 (1985) para permitir desarrollo iterativo e incremental.

1988:
Watts Humphrey publica una versión inicial de CMM.

1989:
Watts Humphrey publica Managing the Software Process.

1993:
SEI publica la versión 1.1 de CMM-SW.

1994:
Sutherland y Schwaber comienzan a aplicar una versión primitiva de Scrum.
Mil-Std-498 reemplaza 2167A. Estimula desarrollo iterativo e incremental.

2002:
SEI publica CMMI v1.1.

2010:
CMMI v1.3 soporta desarrollo ágil.
Humble & Farley publican "Continuous Delivery"

Mid-1990s:
Walker Royce, Philipe Krutchen y otros definen RUP.

1994:
Se reúnen 16 expertos en RAD para discutir la definición de un proceso iterativo e incremental, origen de DSDM.

No todas las actividades agregan valor al producto o servicio

Tiempo

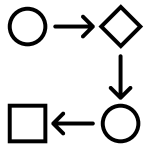
Trabajo

Espera

La mayoría de los procesos agregan valor sólo el 5% del tiempo que lleva entregar el producto o servicio

Introducción

Lean, Agile, DevOps: enfoques complementarios, con puntos de contacto



Lean

Cultura, principios y herramientas orientados a la reducción del desperdicio, la optimización del flujo de trabajo y a la mejora continua

Agile

Desarrollo de software en forma **iterativa e incremental**, con el objetivo puesto en poder adaptarse rápidamente a los cambios

DevOps

Cultura, principios y herramientas basados en Lean y Agile, orientados a optimizar el flujo completo de desarrollo, despliegue y operaciones de software

Modelos en el desarrollo de software

Abstracciones pare entender y construir

"Software is invisible and unvisualizable. Geometric abstractions are powerful tools. The floor plan of a building helps both architect and client evaluate spaces, traffic flows, views (...) A geometric reality is captured in a geometric abstraction.

The reality of software is not inherently embedded in space. Hence, **it has no ready geometric representation** in the way that land has maps, silicon chips have diagrams, computers have connectivity schematics. As soon as we attempt to diagram software structure, we find it to constitute not one, but several, general directed graphs superimposed one upon another. The several graphs may represent the flow of control, the flow of data, patterns of dependency, time sequence, name-space relationships. These graphs are usually not even planar, much less hierarchical."

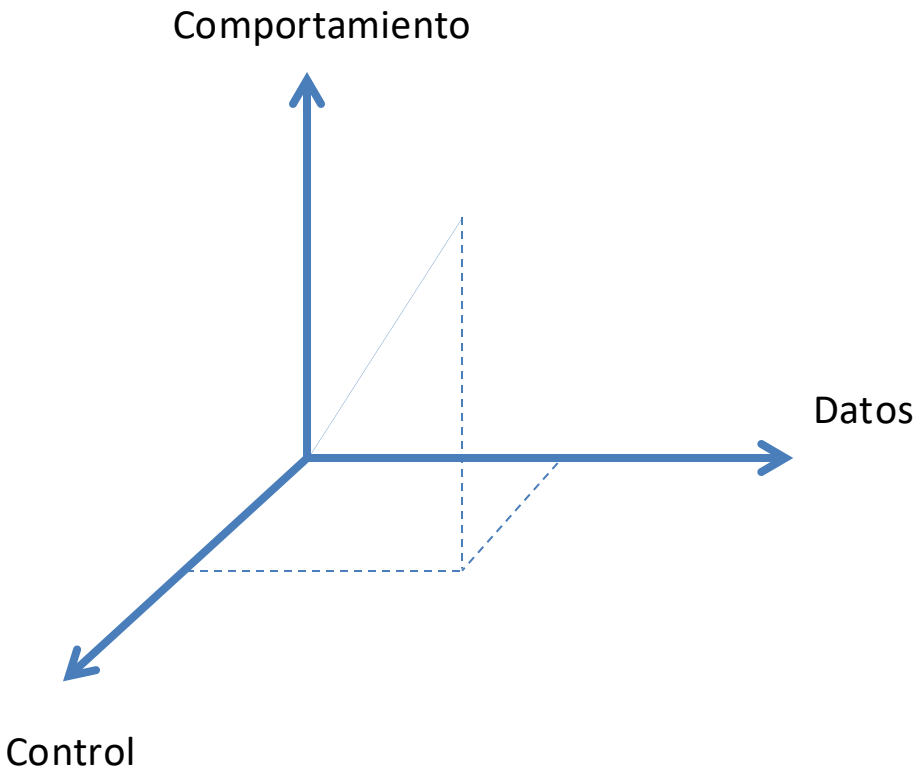
Frederick Brooks, *No Silver Bullet*

- Un modelo es una simplificación de la realidad.
- Construimos modelos para entender mejor el sistema que estamos desarrollando.
- Construimos modelos de sistemas complejos porque no podemos entenderlos en su totalidad.
- Los modelos:
  - Permiten visualizar un sistema existente o que queremos desarrollar.
  - Permiten especificar la estructura o el comportamiento de un sistema.
  - Proveen guía para la construcción de un sistema.
  - Documentan las decisiones tomadas.

[Boock]

Modelos en el desarrollo de software

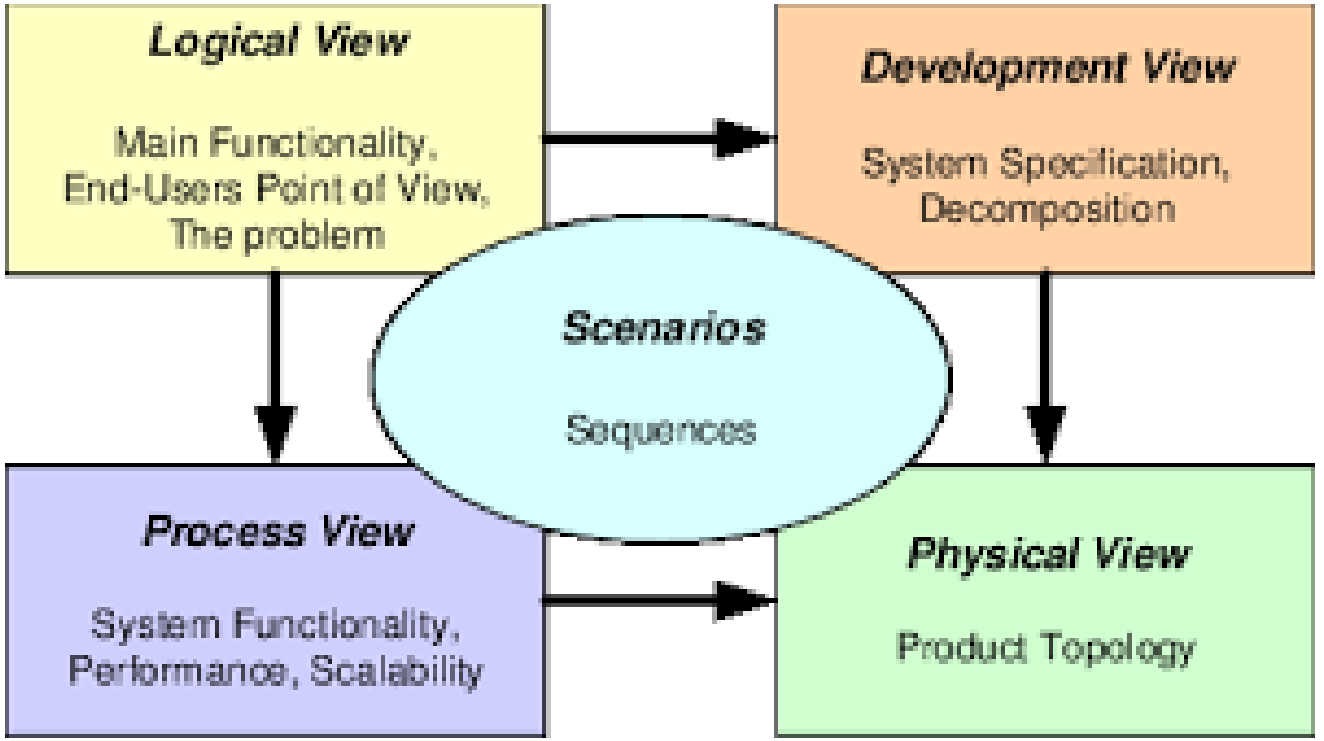
Dimensiones



Jorgensen, Paul C. *Modeling software behavior: a craftsman's approach*. CRC press, 2009.

Modelos en el desarrollo de software

Modelo 4+1 de Krutchen



https://www.researchgate.net/publication/221278125_The_Rational_Unified_Process_with_the_41_View_Model_of_Software_Architecture_-_a_Way_for_Modeling_Web_Applications/figures?lo=1&utm_source=google&utm_medium=organic

Conclusiones

Lean, Agile, DevOps

“Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.”

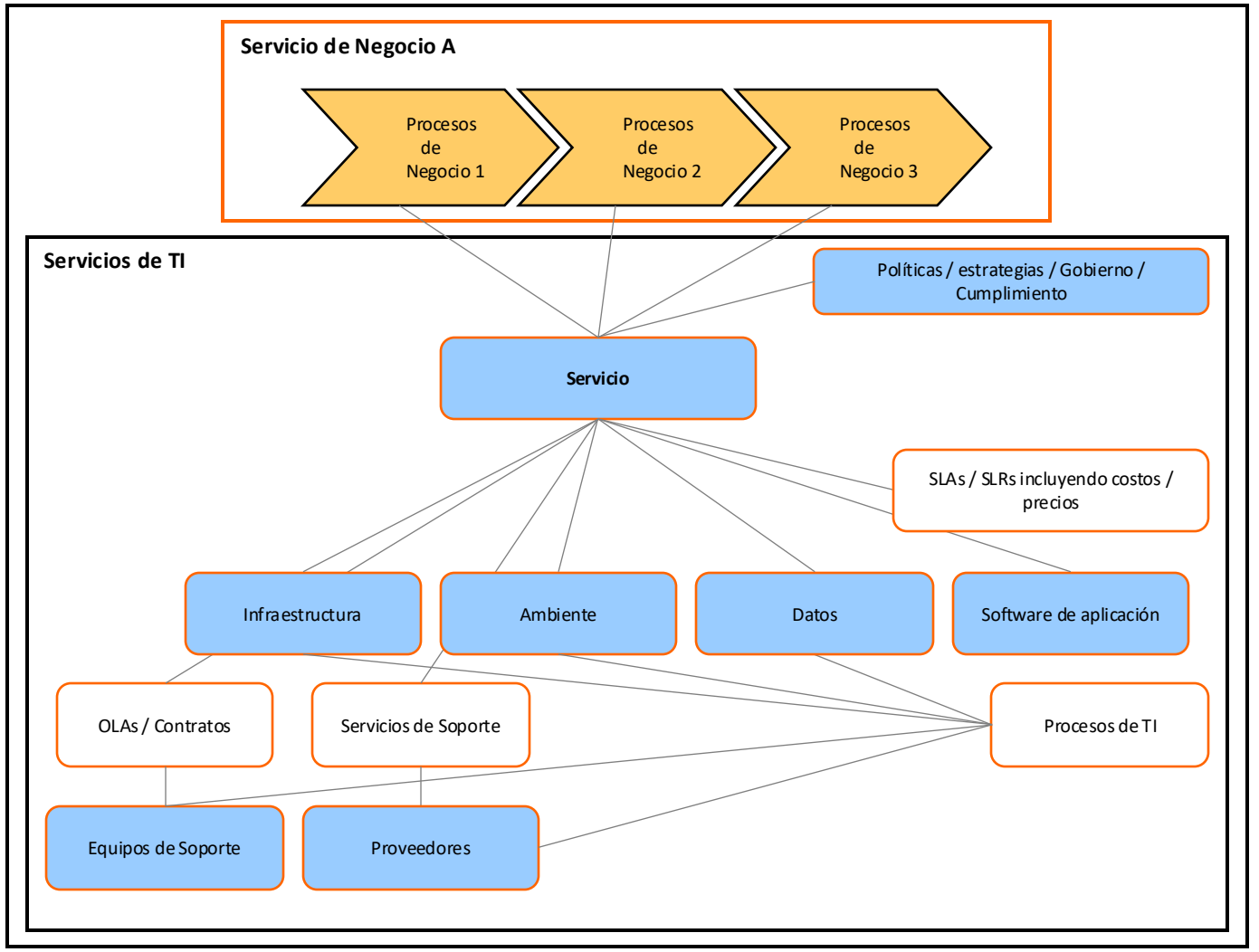
Melvin Conway

http://www.melconway.com/Home/Conways_Law.html

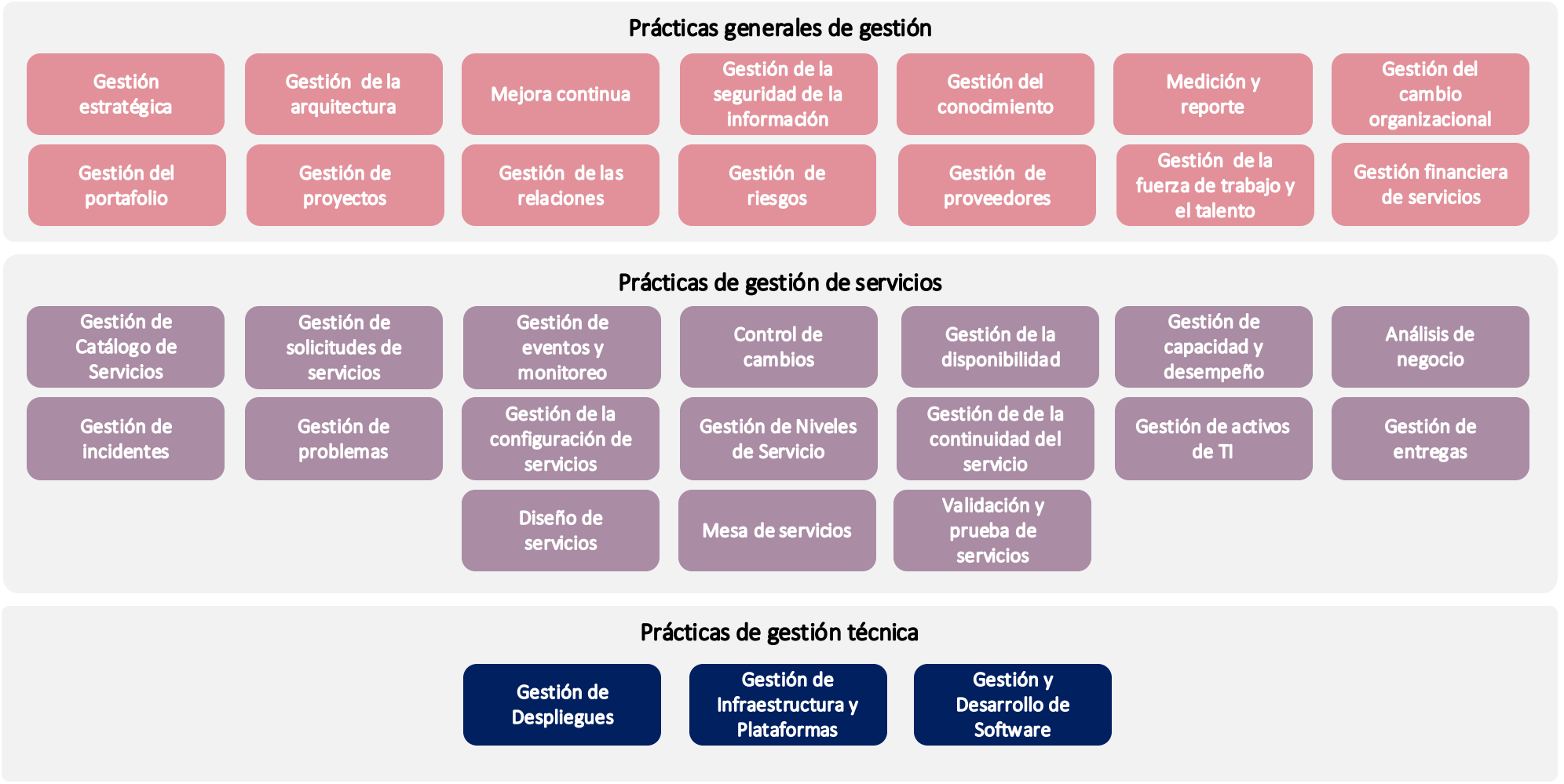
Modelo Organizacional	Modelo de Arquitectura	Modelo de Desarrollo	Modelo de Liberación	Modelo de Prácticas	Modelo de Hosting	Modelo de Externalización	Modelo de Soporte
Organización Jerárquica	Monolítica	Cascada	Anual - cuatrimestral	Orientada a procesos	Centro de cómputos físico	Internalización	Basado en niveles de soporte
Unidades de negocio / modelo funcional	N Capas	Ágil	Mensual	Orientado a prácticas	Máquinas virtuales	Tercerización tradicional (torres)	Basado en conocimientos
Centrado en productos y equipos	Microservicios y plataformas	DevOps	Liberación continua	Flujos de valor	Nubes y contenedores	Servicios tercerización en múltiples proveedores	Auto servicio y auto soporte
Bucles de realimentación basados en datos y potenciados con AI y ML							
Seguridad, privacidad y cumplimiento por diseño							

https://youtu.be/7tLhQLtrWvE?si=rQYZudCIGWKY9p_z

Software como parte de un servicio

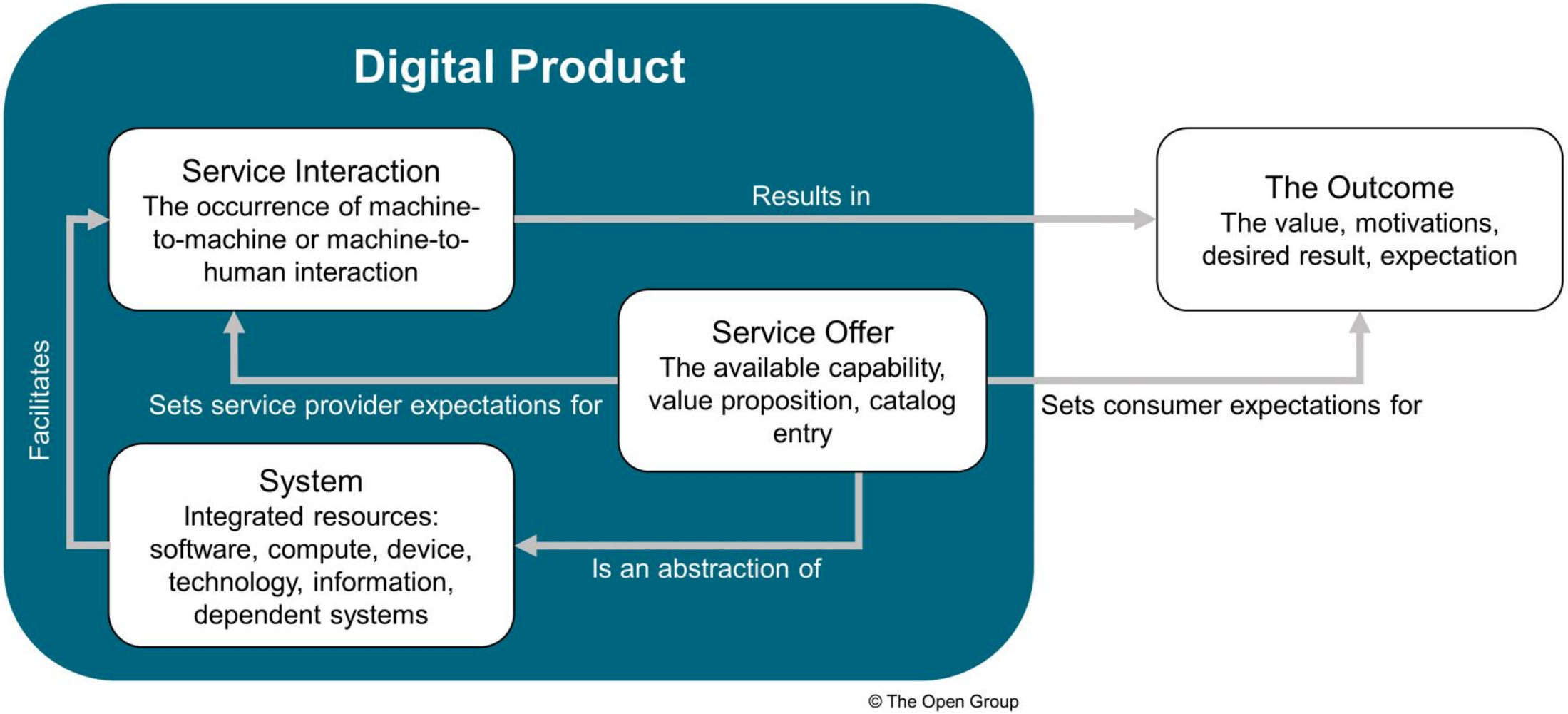


Software como parte de un servicio



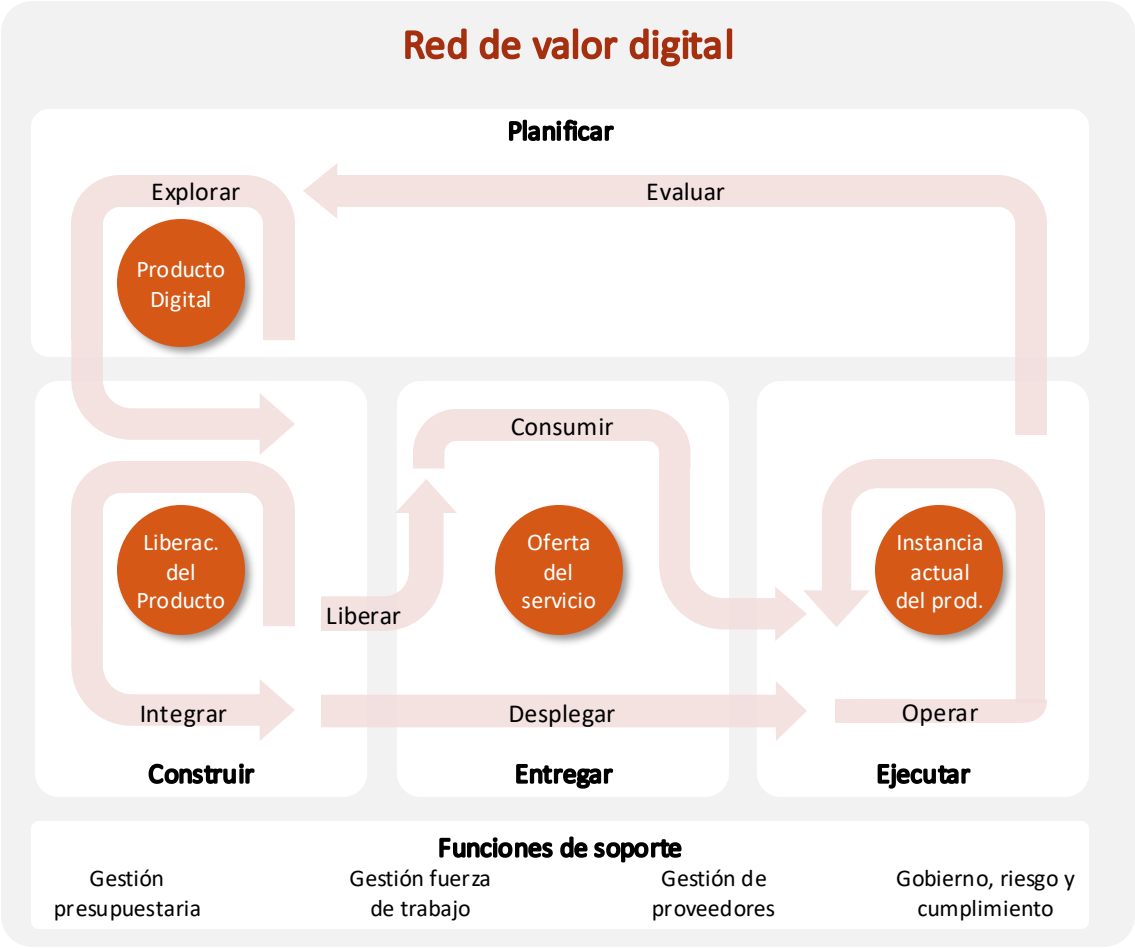
https://www.axelos.com/certifications/itil-service-management

Software como parte de un producto digital



https://www.opengroup.org/it4it

Software como parte de un producto digital

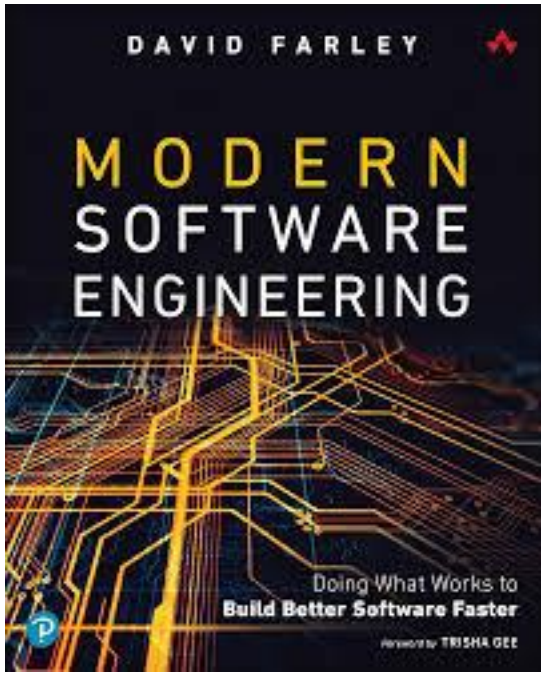


- **Evaluar** el portafolio de productos
- **Explorar** la definición de un nuevo producto o su evolución
- **Integrar** el desarrollo y entrega del producto
- **Desplegar** entrega del producto
- **Liberar** oferta de servicio basada en el producto
- **Consumir** la oferta de servicio
- **Operar** la instancia del producto

https://www.opengroup.org/it4it

Conclusiones

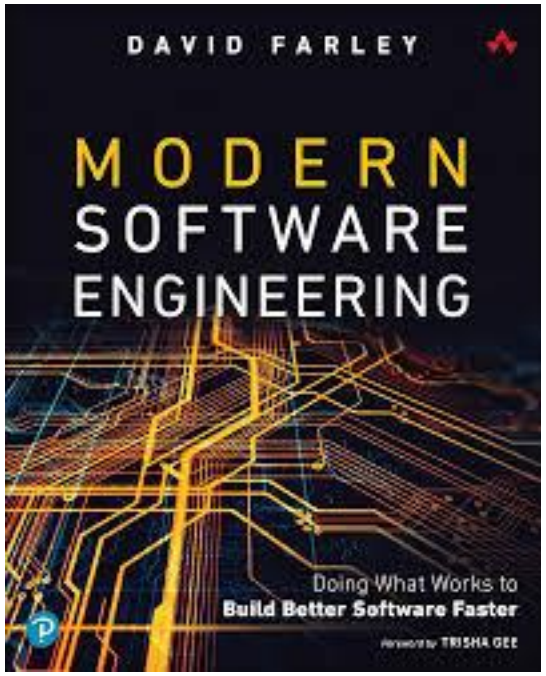
Dave Farley



"Software engineering is the application of an **empirical**, **scientific** approach to finding efficient, economic solutions to practical problems in software."

Conclusiones

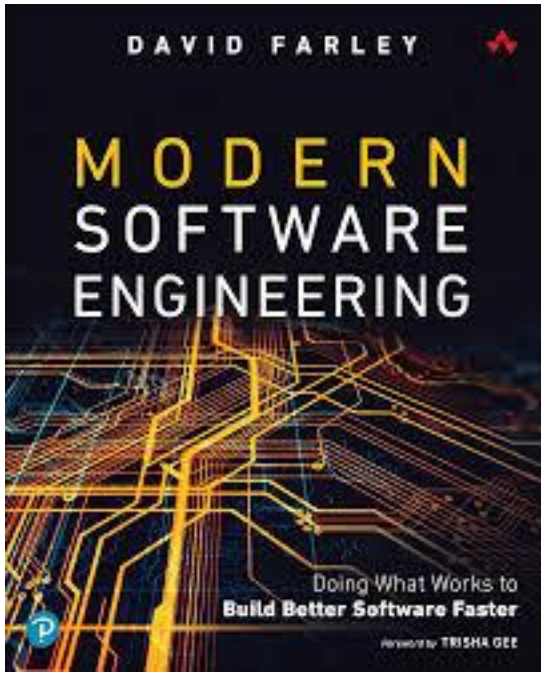
Dave Farley



"Software development is always an exercise in **discovery** and **learning**, and second, if our aim is to be "efficient" and "economic," then **our ability to learn must be sustainable**."

Conclusiones

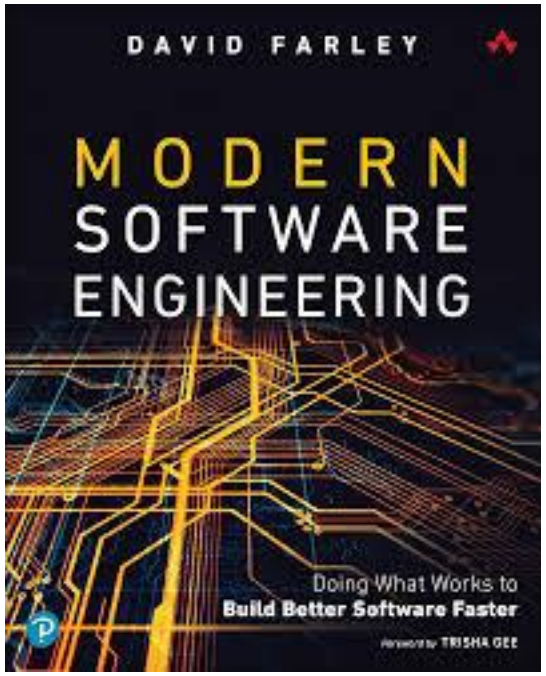
Dave Farley



"We must become **experts at learning** and **experts at managing complexity**"

Conclusiones

Dave Farley

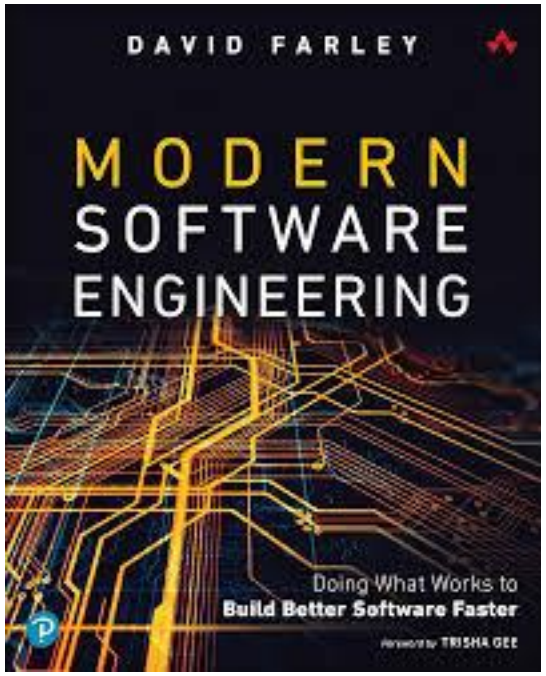


"Five techniques that form the roots of this focus on **learning**:

- Iteration
- Feedback
- Incrementalism
- Experimentation
- Empiricism"

Conclusiones

Dave Farley



"To become **experts at managing complexity**, we need the following:

- Modularity
- Cohesion
- Separation of Concerns
- Abstraction
- Loose Coupling"