

Curso de Ingeniería de Software

# Unidad 1

## Introducción a la Ingeniería de Software

**Guadalupe Ibargüengoitia G.**  
**Hanna Oktaba**

# Programación vs Ingeniería de Software

- ¿Cuál es la diferencia entre la Programación y la Ingeniería de Software?

- La **Ingeniería de Software** es una disciplina reciente, comparada con otras ingenierías.
- Sus inicios datan de finales de los años **sesenta del siglo pasado**, mientras que, por ejemplo, ingeniería civil tiene antigüedad **milenaria**.

# Definiciones de la Ingeniería de Software

- La Ingeniería de Software es *la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, la operación y el mantenimiento de software* (SEVOCAB 2017).

# Definiciones de la Ingeniería de Software

- La Ingeniería de Software es *una disciplina de la ingeniería que se ocupa de **todos los aspectos de la producción del software**, desde sus pasos iniciales de la especificación del sistema, hasta el mantenimiento cuando está en uso* (Sommerville I., 2011).

# Definiciones de la Ingeniería de Software

- La Ingeniería de Software es *la construcción de **productos de software** por grupos de personas, para que sean usados por otras.*
- El **cliente es quien solicita** el desarrollo del producto y plantea el problema a resolver.
- El **equipo de desarrollo construye y entrega** el producto solicitado.

# Objetivo de la Ingeniería de Software.

- El **objetivo** de la Ingeniería de Software según se estableció en la primera conferencia importante de Ingeniería de Software en 1968 fue: *“el establecimiento y uso de principios robustos, orientados a obtener software **económico** que sea **fiable** y funcione de manera **eficiente** sobre máquinas reales”* (Naur P., 1969).

# Campo de acción de la Ingeniería de Software

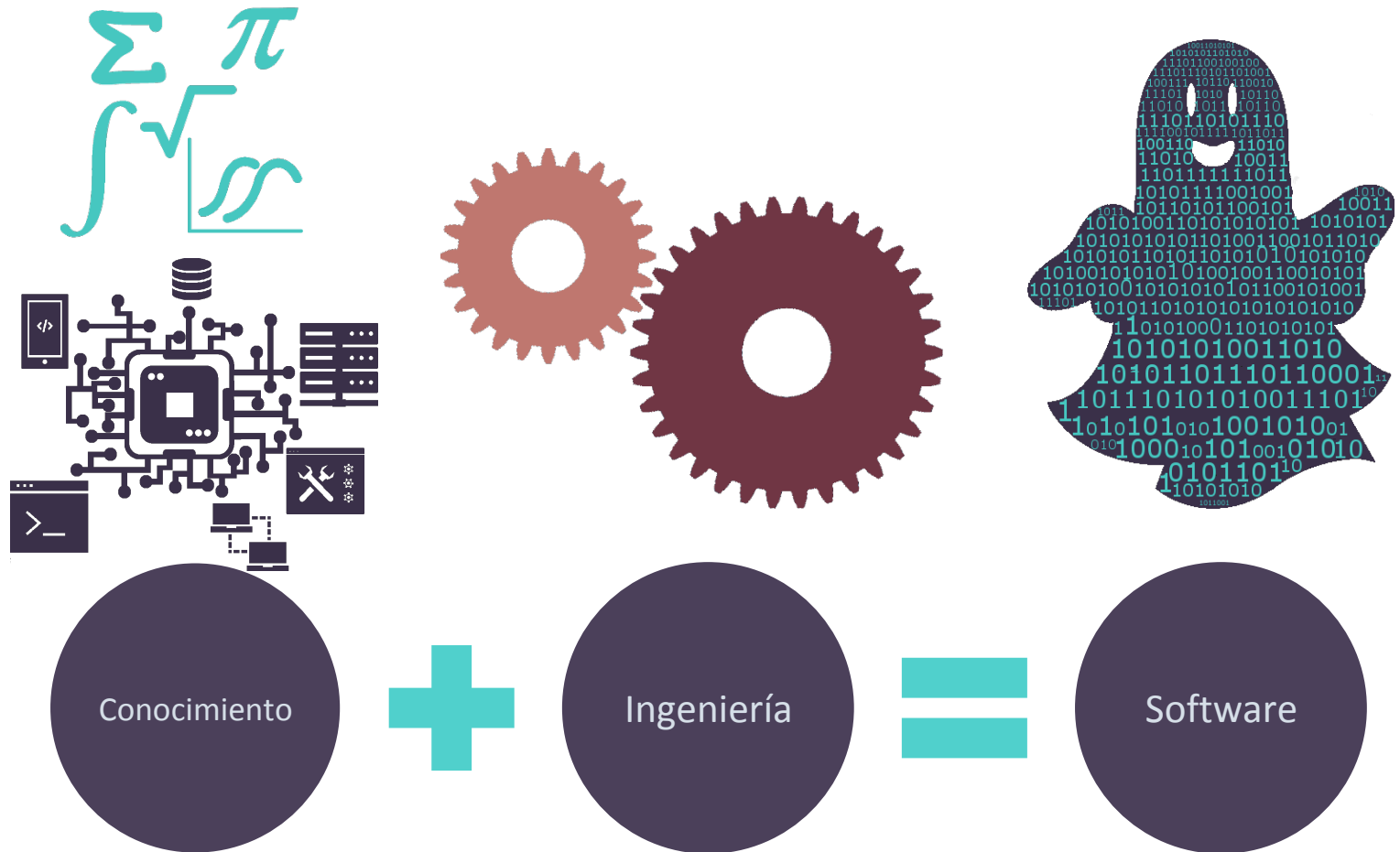
- El campo de acción de la Ingeniería de Software está en el desarrollo de producto de software de calidad, apoyándose en principios aplicados en los procesos de desarrollo.



# Relación entre las Matemáticas, Ciencias de la Computación e Ingeniería de Software

- Las **Matemáticas** te facilitan, a través de la formación mental, la abstracción de conceptos, expresión de algoritmos y construcción de modelos.
- Las **Ciencias de la Computación** te proporcionan conocimientos y herramientas para hacer programas y fundamentos computacionales tales como: conceptos de lenguajes de programación, análisis de algoritmos, sistemas operativos, arquitectura de computadoras, entre otros.
- Las prácticas de la **Ingeniería** que ha incorporado la Ingeniería de Software son: especificación de requisitos, diseño y organización del trabajo en equipos, entre otras.

# Relación entre las Matemáticas, Ciencias de la Computación e Ingeniería de Software



# Software

- ¿Qué es **Software**?

# ¿Qué es el Software?

- Es el conjunto de los *programas* de cómputo, *procedimientos*, *reglas*, *documentación* y *datos asociados*, que forman parte de las operaciones de un sistema de computación. (SEVOCAB , 2017)



# 1.3 ¿Qué es el Software?

- Un producto de software es la suma total de: programas de computadora, procedimientos, reglas, documentación asociada y datos necesarios para la operación de un sistema computarizado (ISO/IEC 12207, 2008).

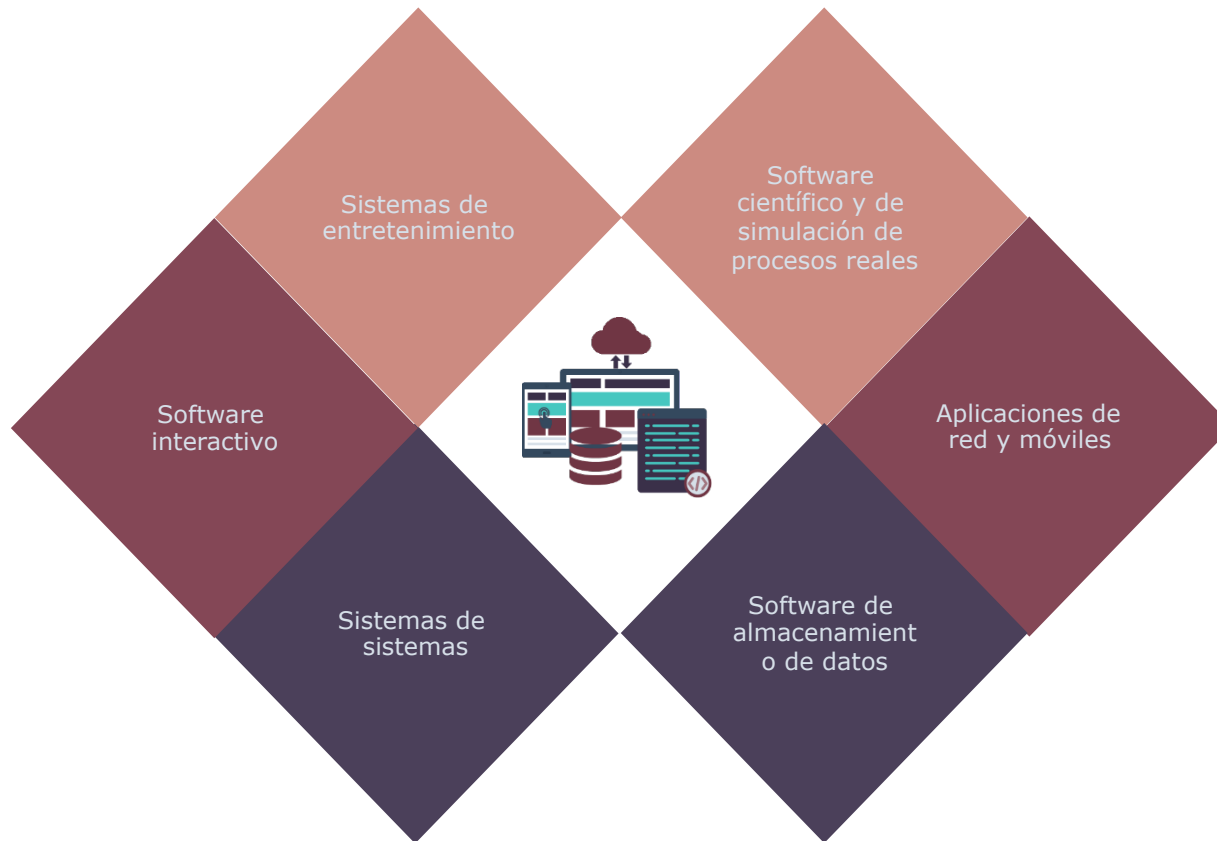
El software hace que el hardware funcione e interactúe con nosotros.



# Características del software

- Es **abstracto e intangible**. No está restringido por las propiedades de los materiales, ni las leyes físicas, ni las reglas de manufactura. (Sommerville I., 2011).
- El software se **desarrolla**, no se fabrica en un sentido clásico (Pressman R.S.).
- Es **fácilmente modificable** y por lo tanto se puede corromper.
- Está **hecho para evolucionar**, pues cambiará según las necesidades de sus usuarios.
- El software **no se desgasta** con el paso del tiempo pero se puede **deteriorar** si al mantenerlo **se le incorporan nuevos defectos**. (Pressman R.S.)

# Tipos de software





# Calidad de software

- *Calidad* de un producto de software es el “*grado en que satisface las necesidades y expectativas del usuario cuando se usa en condiciones específicas*” (ISO/IEC 25010, 2009).
- Para obtener software de calidad es necesario que **todos los productos** que se generen en el desarrollo **sean consistentes y no tengan defectos**.

# Calidad de software

- Un *defecto* es resultado de un error cometido por un desarrollador al generar un producto.
- “Los defectos aún pequeños como faltas de ortografía o de dedo, pueden ocasionar problemas severos en el software al presentar inconsistencias o respuestas impredecibles” (Humphrey W., 1996).

# Practicas de Calidad de software

- Prácticas que se usan para **comprobar la calidad** del software son la *verificación y validación*.
  - *Verificar* un producto de software tiene por objetivo **revisar que no tenga defectos** introducidos por el desarrollador.
  - *Validar* el software es asegurarse **que hace lo que el usuario espera que haga**.

# Cualidades del software (ISO/IEC 25010)

- **Adecuación Funcional** (*Functional suitability*). Representa la capacidad del producto de software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas.
- **Eficiencia de Desempeño** (*Performance efficiency*). Representa el desempeño relativo a la cantidad de recursos utilizados, bajo determinadas condiciones.
- **Compatibilidad** (*Compatibility*). Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno de hardware o software.
- **Usabilidad** (*Usability*). Capacidad del producto de software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones.

# Cualidades del software (ISO/IEC 25010)

- **Fiabilidad** (*Reliability*). Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados.
- **Seguridad** (Security). Capacidad de protección de la información y los datos de manera, que personas o sistemas no autorizados no puedan leerlos o modificarlos.

## Cualidades del software (ISO/IEC 25010)

- **Mantenibilidad** (Maintainability). Capacidad del producto de software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas.
- **Portabilidad** (Portability): Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno operacional o de uso, de hardware y software a otro.

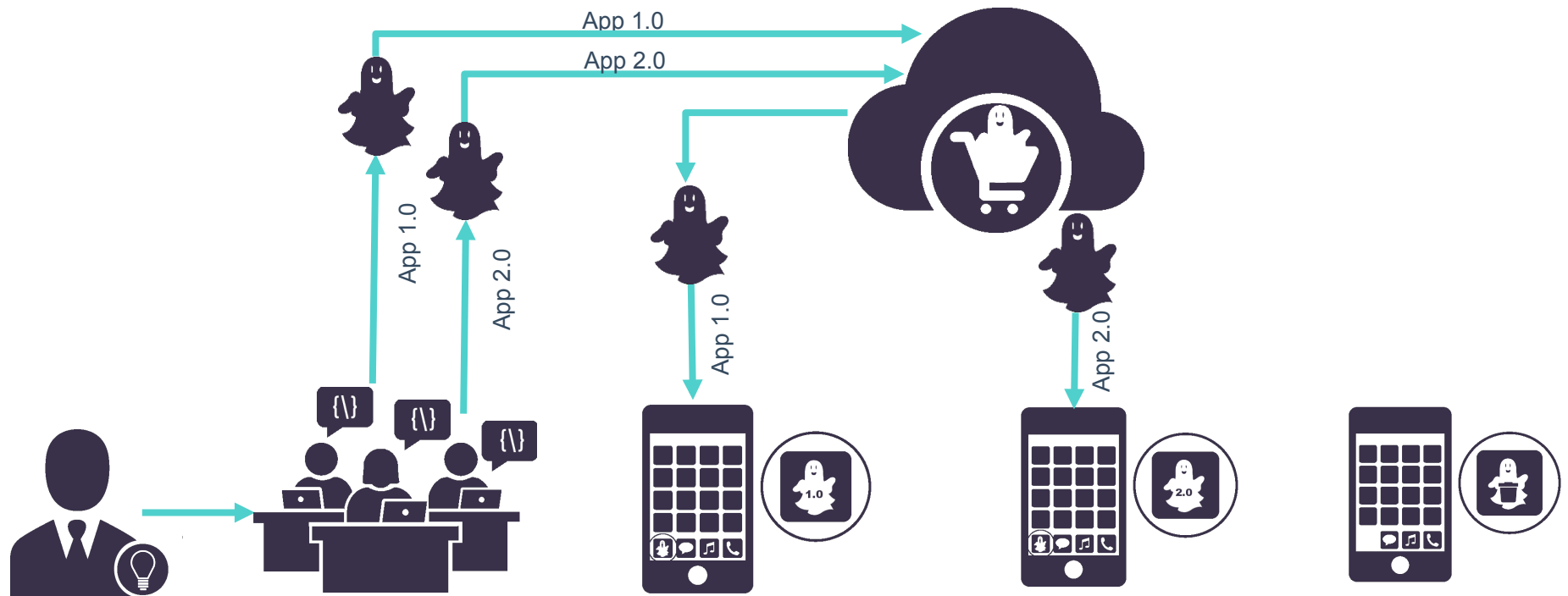
# Definición de cliente y equipo de desarrollo

- En la Ingeniería de Software el *cliente* es quien *solicita* el software. Ese cliente podrías ser tú mismo, otra persona que requiere el desarrollo de software, alguien que está dispuesto a pagarte para que le construyas un software, etc.
- El *equipo de desarrollo* es otro término importante y puede ser una sola persona, 2 o 3 o muchas, que se *encargan de desarrollar y entregar* el software al cliente que lo solicitó.

# Ciclo de vida de software (Software life cycle)

- “La **evolución** que va sufriendo el **software** desde **el momento en que se plantea su construcción**, el tiempo que lleva su desarrollo, la evolución donde se le agregan y modifican funcionalidades, hasta su retiro de **uso**”. (IEEE, 1990).

# Ciclo de vida de software



## Solicitud

Un cliente solicita o adquiere un producto de software.

## Desarrollo

El equipo de desarrollo trabaja en el producto que necesita el cliente.

## Operación

Se opera el producto de software en su entorno real y para sus usuarios.

## Mantenimiento

Incluye las mejoras, adaptaciones, correcciones, migraciones del software.

## Retiro

Cuando el software ya no cumple con sus objetivos, se retira del uso.



# Etapa de Desarrollo

## Análisis de requisitos

Es el paso en el que el cliente expresa sus necesidades para crear el software y el equipo de desarrollo, al analizarlas, las convierte en la especificación de requerimientos del producto de software.

## Diseño

El equipo de desarrollo, en función de la especificación de requerimientos establece la estructura del software, identificando sus componentes principales con sus relaciones y especifica cada componente a detalle.

## Construcción

Se genera el código de cada componente siguiendo el diseño.

## Integración

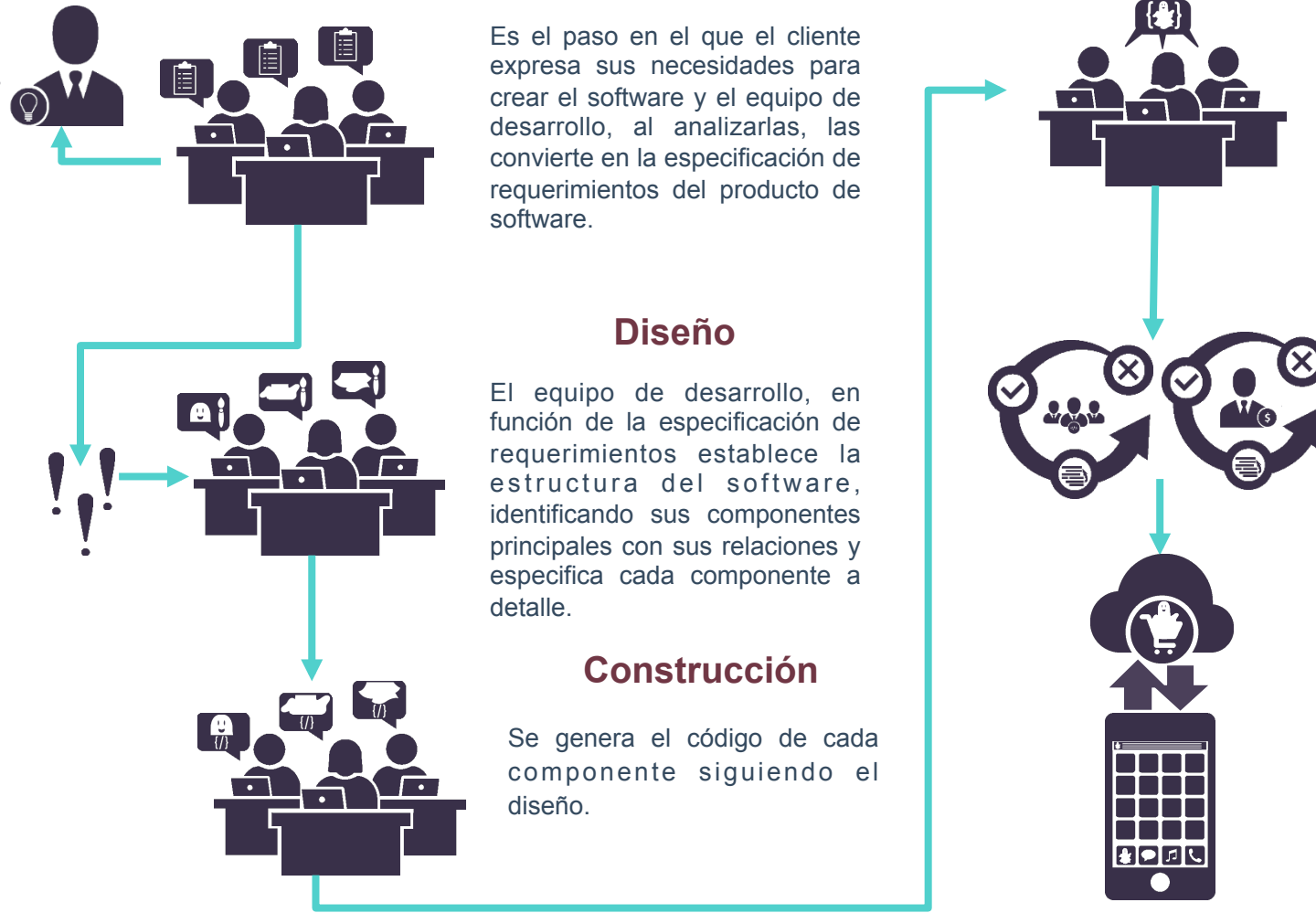
Se asegura que todos los componentes trabajen adecuadamente juntos, haciendo las pruebas de integración necesarias.

## Pruebas

Se revisa que el software cumpla con los requerimientos especificados y se corrigen los defectos.

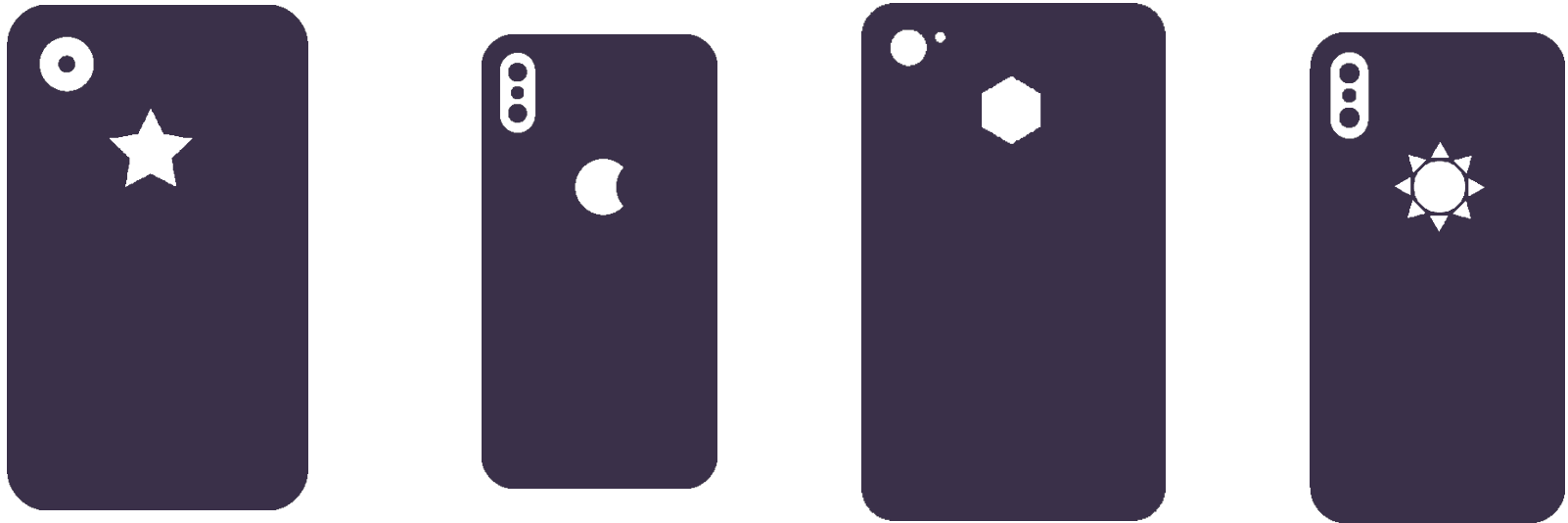
## Entrega

Se entrega el software desarrollado para su puesta en operación.



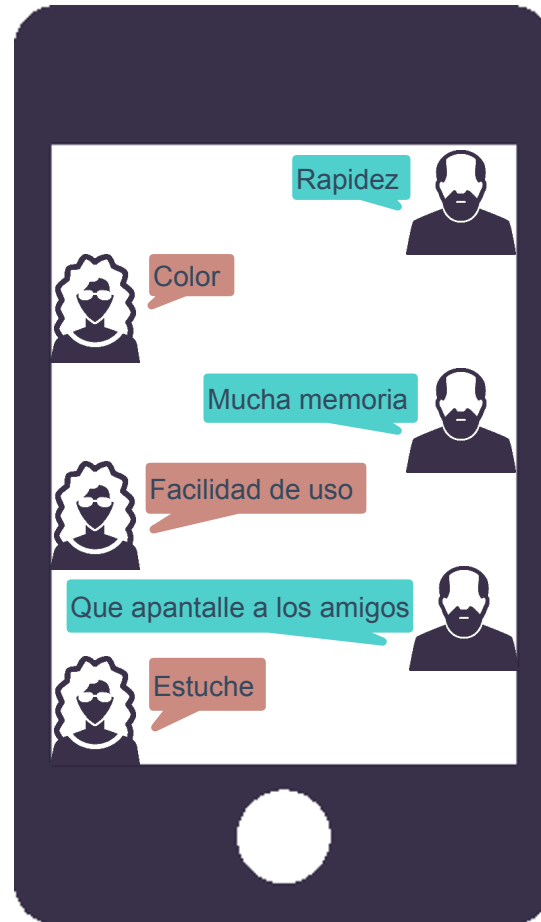
# Principios de la Ingeniería de Software

## Generalidad



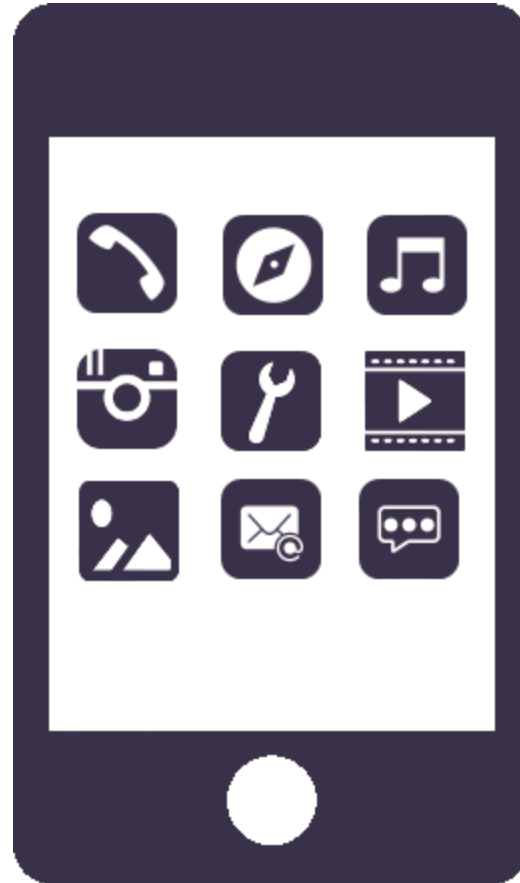
descubrir los aspectos más generales que existen en las necesidades del software para desarrollar el producto de software mas general que los cumpla. Este **principio es fundamental** para desarrollar **herramientas y paquetes genéricos**.

# Abstracción



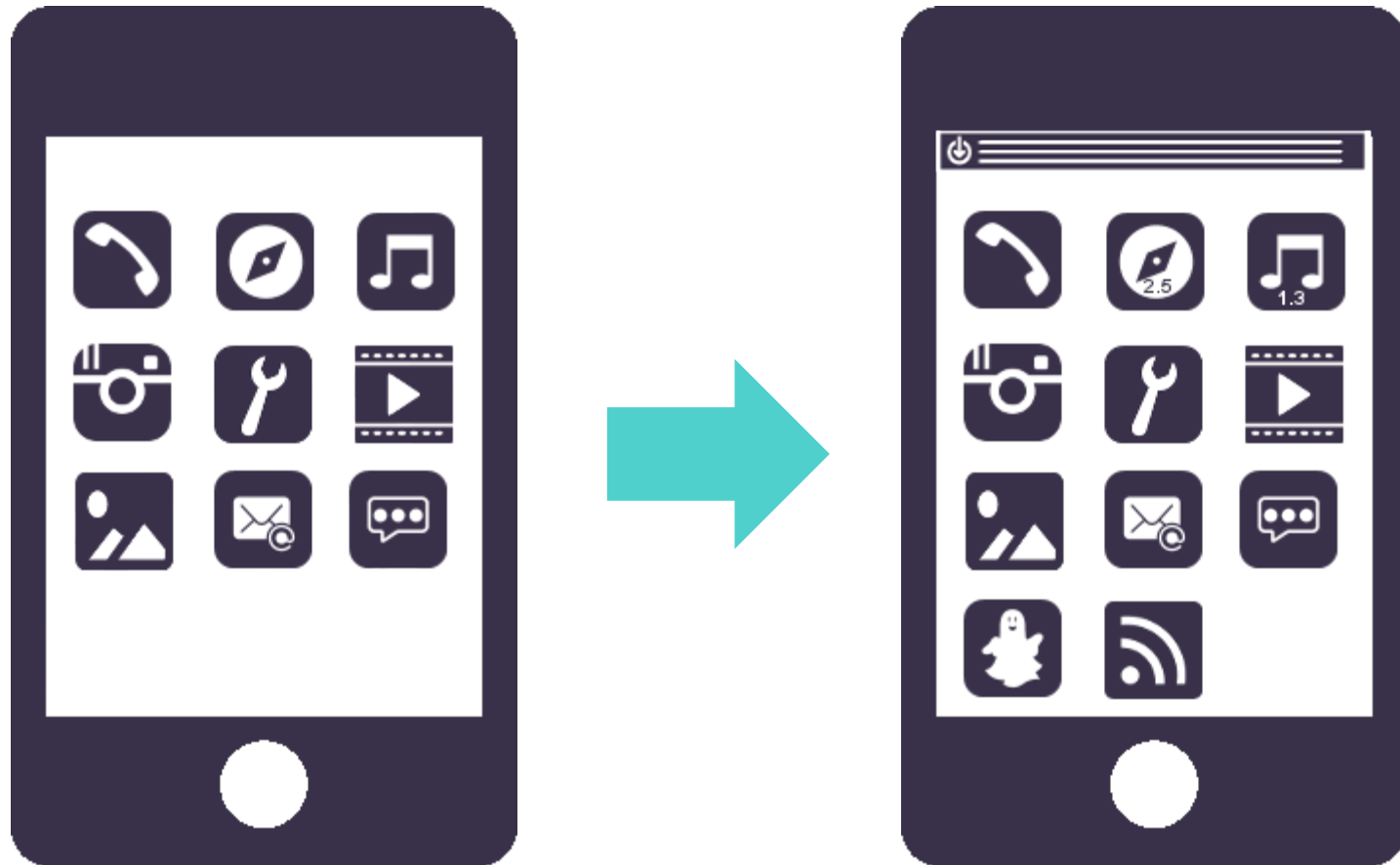
Es identificar los **aspectos más importantes** e ir incorporando los detalles gradualmente.

# Modularidad



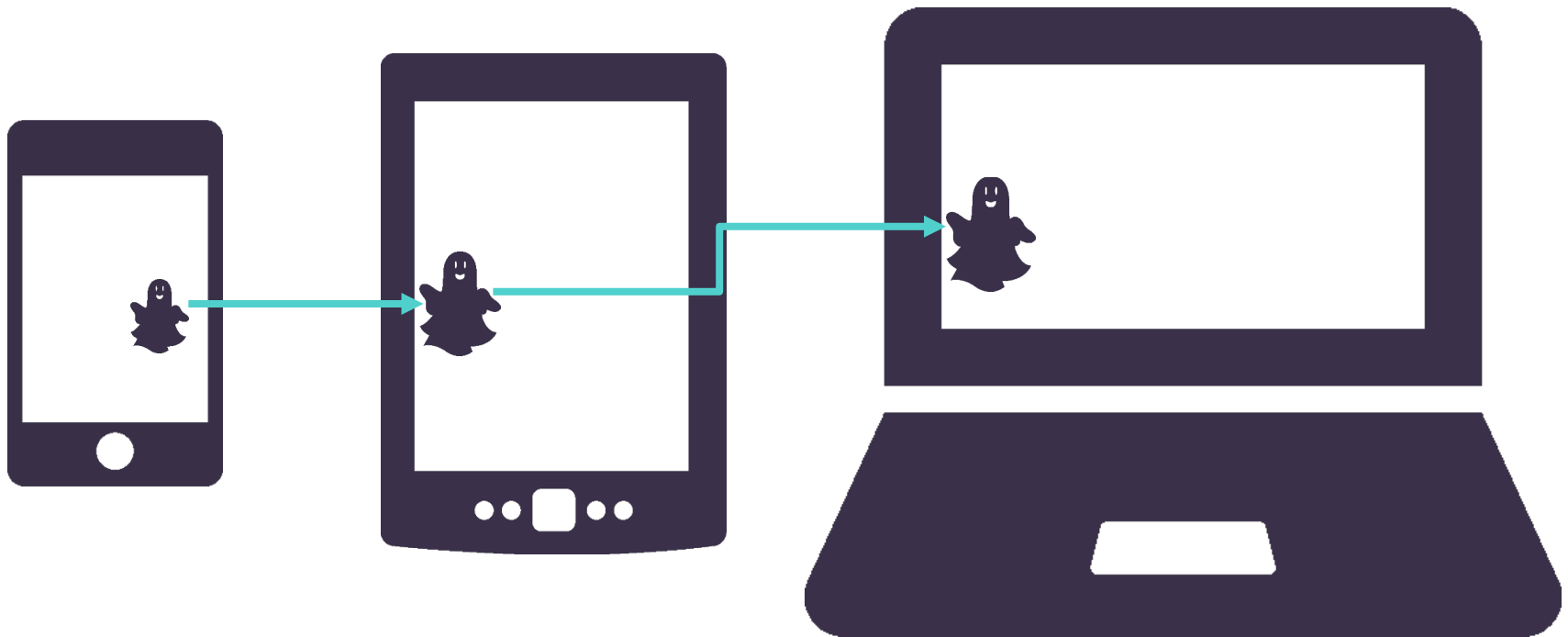
Es **dividir el problema en subproblemas** menos complejos. Incluye los conceptos de **cohesión y acoplamiento**: Los subproblemas deben ser **internamente cohesivos** y, en relación a otros, **débilmente acoplados**.

# Incrementabilidad



Consiste en obtener el producto de **software** incrementando la funcionalidad a través de varias iteraciones de desarrollo.

# Anticipación al cambio



Es **diseñar el software** para que pueda **evolucionar a nuevas versiones**

- **Separación de conceptos:** Es manejar diferentes aspectos de un problema concentrándose en **cada uno por separado**.

Comunicación	Entretenimiento	Redes sociales	Utilidades
Teléfono Mensajería instantánea Mensajes de texto	Juegos Televisión bajo demanda Música	Facebook Twitter Instagram Google +	Cámara Linterna GPS Grabadora

# Historia de la Ingeniería de Software

## La Ingeniería de Software en los 70's

- A fines de los 60's desarrollar software consistía principalmente en **codificar y corregir errores** (code&fix).
- Surge la llamada *crisis del software*, debido a que la mayor parte de sistemas de software:
  - **no respondía a las necesidades de los clientes**
  - **costaba mucho** más caro de lo contratado
  - **no se entregaba en el tiempo planeado.**



# Historia de la Ingeniería de Software

## La Ingeniería de Software en los 80's

- Surgen las *Metodologías de desarrollo de software estructuradas*.
- Su objetivo es desarrollar software siguiendo *el ciclo de vida del software*, que consiste en una serie de pasos:
  - definir los requerimientos para el software,
  - analizarlos,
  - diseñar el software,
  - implementarlo,
  - probarlo y
  - ponerlo a disposición de los usuarios.
- Estos pasos se hacían secuencialmente.

# Historia de la Ingeniería de Software

- Entregado al cliente, se le daba mantenimiento:
  - **correctivo** (para eliminar los defectos)
  - **adaptativo** (para adecuarlo a las nuevas necesidades del cliente)
  - **perfectivo** (para mejorarlo).

## Ejemplos de metodologías estructuradas:

- Yourdon y Constantine (1978), Jackson (1983), Warnier (1981), Orr (1977), Coad (1990).

# Historia de la Ingeniería de Software

## La Ingeniería de Software en los 90's

- En los inicios de esta década surge *Programación Orientada a Objetos*
  - OMT (Rumbaugh, 1991)
  - propuestas de modelado como Casos de uso (Jacobson, 1992)
  - diagramas de clases (Booch, 1991)

# Historia de la Ingeniería de Software

- En 1997, *Lenguaje de Modelado Unificado (Unified Modeling Language - UML)* (Booch, Jacobson y Rumbaugh, 1998).
- UML sigue siendo el estándar de modelado vigente y el más utilizado en la industria de software.
- El *Proceso Unificado* (Booch, Jacobson y Rumbaugh, 1999) utiliza UML.

# Historia de la Ingeniería de Software

– Modelos de procesos como estándares internacionales:

- Modelo de Madurez de Capacidades (Capability Maturity Model - CMM) del Software Engineering Institute (Paulk, 1993) (CMM, 2013),
- **ISO/IEC 12207** (ISO/IEC12207, 2008) y la **ISO/IEC 15504** (ISO/IEC 15504).

# Historia de la Ingeniería de Software

- El CMM evolucionó en la siguiente década a **Capability Maturity Model Integration - CMMI** máximo referente de las empresas de desarrollo de software. **CMMI 2.0 (2018)**

Actualmente **CMMI** tiene prácticas para el desarrollo de software **DEV**, servicios **SER** y adquisiciones **ADQ**

# Historia de la Ingeniería de Software

—*Procesos de software*: “Un proceso de ingeniería consiste en un conjunto de actividades interrelacionadas que transforman una o más entradas en salidas. Al realizar la transformación se consumen recursos” (SWEBOK 3.0, 2014).

# Historia de la Ingeniería de Software

- Watts Humphrey propuso dos modelos de procesos de apoyo al desarrollo de software.
  - Dirigido a individuos, llamado **Personal Software Process (PSP)** ( 1996)
  - Dirigido a equipos, llamado **Team Software Process (TSP)** (1999).



# Historia de la Ingeniería de Software

## La Ingeniería de Software en los 2000's

- *Movimiento ágil* que publica el *Manifiesto por el desarrollo ágil de software*
  - Individuos y sus interacciones sobre procesos y herramientas
  - Software funcionando sobre documentación extensiva
  - Colaboración con el cliente sobre negociación contractual
  - Respuesta ante el cambio sobre seguir un plan

# Historia de la Ingeniería de Software

—Ejemplos de métodos ágiles más populares:

- *eXtreme Programming* (XP)
- SCRUM
- KANBAN

# Historia de la Ingeniería de Software

- Evolucionan los procesos y estándares internacionales para aplicarse a empresas pequeñas surgiendo estándares nacionales como:
  - el estándar mexicano MoProSoft (MoProSoft, 2005) e
  - ISO/IEC 29110 Basic Profile for VSEs basado en MoProSoft .

# Historia de la Ingeniería de Software

- Tendencia para balancear de los procesos con los métodos ágiles. (Bohem, 2004).

# Historia de la Ingeniería de Software

## La ingeniería de Software en los 2010's

- En 2011 se define la norma ISO/IEC 29110 denominada Ingeniería de Software –Perfiles de Ciclo de Vida para Empresas Muy Pequeñas (Software Engineering Lifecycles Profiles for Very Small Enterprises) (ISO/IEC291100, 2011).
- El Perfil Básico, define los procesos que debe llevar a cabo un equipo de hasta 25 personas con un proyecto de desarrollo pequeño

# Cuerpo de conocimiento de la Ingeniería de Software

*Cuerpo de Conocimientos de la Ingeniería de Software (Software Engineering Body of Knowledge)* y por sus siglas en inglés es el SWEBOK (SWEBOK 3.0, 2014).

- Es un compendio recogido por académicos y profesionales, que la ejercen en todo el mundo, para ser utilizado en la educación y en la práctica.

# Ingeniería de Software como profesión

La Ingeniería de Software ha crecido tanto que actualmente se han reconocido como una actividad profesional y existen **carreras profesionales completas** dentro del área de la Computación que se llaman **Ingeniería de Software**.

# Referencias

- Agile Alliance. (2001). *Manifiesto for Agil Software Engineering*. Retrieved from [www.agilealliance.org](http://www.agilealliance.org)
- Beck K. (1999). *Extremme Programming Explained*. Addison Wesley.
- Bohem B., R. T. (2004). *Balancing Agility and Discipline. A guide for de Perplexed*. Addison Wesley.
- Booch B. (1991). *Object Oriented Design. With Applications*. The Benjamin Cummings.
- Booch G., R. J. (2005). *The Unified Modeling Languajes. Users Guide, (Second edition)*. Addison Wesley.
- Clouse A., A. D. (2001). *CMMI distilled: A Practical Introduction to Integrated Process Improvement*. SEI Series in Software Engineering.
- CMM. (2013 йил 28-01). [http://es.wikipedia.org/wiki/Modelo\\_de\\_Capacidad\\_y\\_Madurez](http://es.wikipedia.org/wiki/Modelo_de_Capacidad_y_Madurez).
- CMMI. (2013 йил 28-01). [http://es.wikipedia.org/wiki/Capability\\_Maturity\\_Model\\_Integration](http://es.wikipedia.org/wiki/Capability_Maturity_Model_Integration) .
- Coad P., Y. E. (1990). *Object-Oriented Analysis*. Prentice Hall.
- Ghezzi C., J. M. (1991). *Fundamentals of Software Engineering*. Prentice Hall.
- Humphrey W. (1996). *Introduction to Personal Software Process*. Addison Wesley Professional.
- Humphrey, W. (1999). *Introduction to Team Software Process*. Addison Wesley Professional.
- IEEE. (1990). Standard Glossary of Software Engineering Terminology.
- ISO/IEC 15504. (n.d.). *Software Engineering - Process Assessment*.
- ISO/IEC 25010. (Dic. 2009). *Systems and software engineering -System and software Quality Requirements and Evaluation (SQUaRE)-System and software quality models*.
- ISO/IEC/IEEE 24765. (2010). *Systems and Software engineering Vocabulary*. IEEE.
- ISO/IEC12207. (2008). *Systems and Software Engineering - Software life Cycle Process*.
- ISO/IEC291100. (2011). *29110-5-1-2Software Engineering-lifecycle Profiles for Very Small Entities Management and Engineering Guidde. s.1*. ISO.
- Jackson M. (1983). *System Development*. Prentice Hall.
- Jacobson I., B. G. (1999). *The Unified Software Development Process*. Addison Wesley.
- Jacobson I., C. M. (1992). *Object-Oriented Software Engineering. A Use Case Driven Approach*. Addison-Wesley.
- John\_W.\_Tukey. (2017, 08 11). Retrieved from ([https://es.wikipedia.org/wiki/John\\_W.\\_Tukey](https://es.wikipedia.org/wiki/John_W._Tukey)).
- MoProSoft. (2005). *Modelo de Procesos para la Industris de Software*. Estándar Mexicano Nacional MNX-1-059-NYCE-2005 .
- Naur P., B. R. (1969). *Software Engineering: A report on a conference supported by the NATO Science Committe*. NATO.
- OMG. (n.d.). [www.omg.org](http://www.omg.org). Retrieved 29/09/2012
- Orr K.T. (1977). *Structured Systems Development*. Yourdon Press.
- Paulk, M. (1993). *CApability Maturity Model for Software, version 1.1*. Software Engineering Institute, CArnegie Mellon University.
- Pressman R.S. (n.d.). *Ingeniería de software. Un enfoque práctico*. Mc Graw Hill.
- Rumbaugh J. Blaha M. Premeerlani W., L. W. (1991). *Object-Oriented Modeling and Design*. Prentice Hall.
- SEVOCAB . (2017, 08 11). Retrieved from Software and System Engineering Vocabulary: [https://pascal.computer.org/sev\\_display/index.action](https://pascal.computer.org/sev_display/index.action)
- Sommerville I. (2011). *Software Engineering*. Pearson.
- SWEBOK 3.0. (2014). *Guide to the Software Engineering Body of Knowledge v3.0*. IEEE Computer Society.
- Warnier J.D. (1981). *Logical Construction of Systems*. Van Nostrand Reinhold.
- Yourdon E.N., C. L. (1978). *Structured Design*. Yourdon Press.