Ingeniería de Software I

Jocelyn Simmonds

Departamento de Ciencias de la Computación

Equipo Docente

- Cátedras: Jocelyn Simmonds
  - Oficina BP-P-319
  - Correo: jsimmond@dcc.uchile.cl
- Auxiliar: Florencia Miranda
  - Primera clase auxiliar: miércoles 18 de marzo
  - Apoyada por María José Trujillo y Bárbara Castro (ayudantes)

Consultas generales: Foro U-cursos `https://www.u-cursos.cl/`

Evaluaciones

- (con apuntes) 2 controles + examen. Fechas:
  - Control 1: semana 7 (23 Abr)
  - Control 2: semana 14 (18 Jun)
  - Nota de Control $= 0.5$ Examen $+ 0.5$ promedio (Controles)
- Tareas: proyecto grupal, dividido en 4 tareas ($T_1 - T_4$):
  - entrega de informe, presentación y código (según corresponda)
  - casi todas incluyen una nota de co-evaluación
  - no hay atraso, deben entregar lo que tengan hasta la fecha
  - **no hay tarea recuperativa**
  - Nota de Tareas $= 0.3(T_1 + T_2) + 0.2(T_3 + T_4)$

$$\text{Nota final} = 0.5\text{Nota de Control} + 0.5\text{Nota de Tareas}$$

Proyecto

Graffiti: ilustración en una pared que puede contener texto o una imagen y que intenta trasmitir un mensaje determinado.

Los graffitis han existido como medio de comunicación de posturas políticas o como forma de expresión urbana y anónima que carece de mayor trascendencia.

Ejemplo: Teatro UC

Ejemplo: Teatro UC

Ejemplo: Teatro UC

Museo virtual de graffitis

- Sprint 1 (casi 3 semanas)
  - Como usuario no registrado, quiero crear una cuenta asociada a mi correo, para subir fotos al museo
  - Como usuario registrado, quiero ingresar a mi cuenta, para editar mi perfil
- Sprint 2 (2 semanas)
  - Como usuario registrado, quiero agregar las ubicaciones en que tome mis fotos, para después poder verlas por ubicación
  - Como usuario registrado, quiero agregar etiquetas a mis fotos, para poder categorizarlas
- Sprint 3 (2 semanas)
  - Como usuario registrado, quiero ver las fotos del museo, para poder comentarlas y evaluarlas
  - Como usuario registrado, quiero filtrar las fotos del museo, para poder encontrar las que pertenecen a ciertas categorías o están en torno a cierto punto geográfico

Tareas (1/2)

- Tarea 1 = Sprint 1 (entrega 15 Abr)
  - Código, tests en Github
  - Informe acerca del sprint y el prototipo
  - Retrospectiva en clase del 16 de abril
- Tarea 2 = Sprint 2 (entrega 11 May)
  - Código, tests en Github
  - Informe acerca de sprint y prototipo
  - Demostración/Retrospectiva de 12 min. (12/14 May)
  - 2 presentadores, elegidos al azar

Tareas (2/2)

- Tarea 3 = Sprint 3 (entrega 8 Jun)
  - Código, tests en Github
  - Informe acerca de sprint y prototipo
  - Demostración/Retrospectiva de 12 min. (9/11 Jun)
  - 2 presentadores, elegidos al azar
- Tarea 4: evaluación de calidad de un proyecto ajeno (entrega 26 Jun)
  - Buscamos que hagan una evaluación objetiva del proyecto
  - Su evaluación no afectará las notas de grupo evaluado
  - Informe detallando:
    - tests creados
    - bugs encontrados

Coevaluaciones

- Casi todas las tareas tienen una coevaluación
  - Demuestra compromiso con el proyecto
  - Cumple de manera adecuada con las tareas que le son asignadas
  - Demuestra iniciativa para lograr el éxito del proyecto
  - Mantiene buena comunicación con el resto del equipo
  - Mantiene buena coordinación entre sus tareas y las de sus pares
  - La calidad de su trabajo es la apropiada para lograr el éxito del proyecto
  - Ofrece apoyo en las tareas que van más allá del rol asignado
  - Es capaz de admitir sus equivocaciones y recibir críticas
- Ponderación de nota de coevaluación (C) en una tarea:
  - si $C \geq 5.0$, entonces vale 10% de la nota de la tarea
  - si $C < 5.0$, entonces es la nota que obtienes en la iteración (vale 100%)

Introducción

¿Qué es Software?

¿Qué es Software?

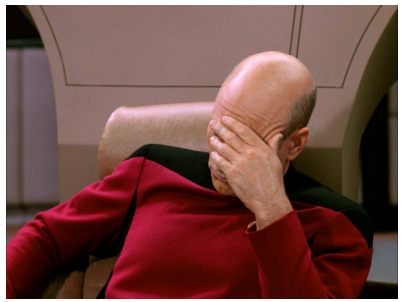
- Software no son sólo los programas que utilizamos
  - código fuente NO es lo único que debemos generar
- Los modelos que describen el problema que resuelven, como fue resuelto, los datos del programa y la documentación para ocupar, instalar o administrar los programas también son software

Ejemplo Doméstico

Una situación cotidiana

- El software se desarrolla para satisfacer una necesidad
- Veamos como un especialista en una materia satisface una necesidad

María, Gásfiter
"Hola Capitán, cuénteme ¿cuál es su problema?"



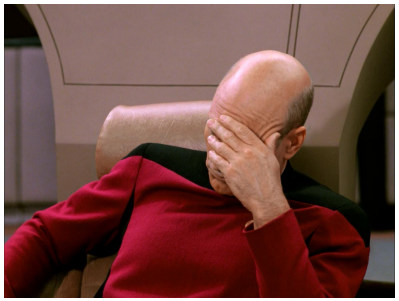
Picard, Cap. USS Enterprise
"Data estaba jugando con los monitos en el baño y ahora el agua no se va"

Ejemplo Doméstico

"Ahh . . . los 'monitos' son de papel . . . de plástico?"

Ejemplo Doméstico

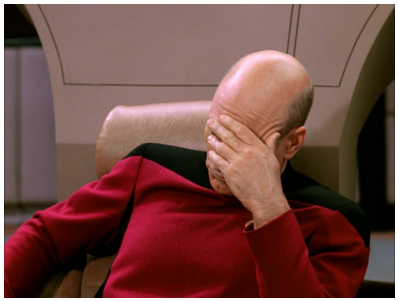
"Ahh . . . los 'monitos' son de papel . . . de plástico?"



"De plástico joven, como el Darth Vader ese"

Ejemplo Doméstico

"Ahh . . . los 'monitos' son de papel . . . de plástico?"

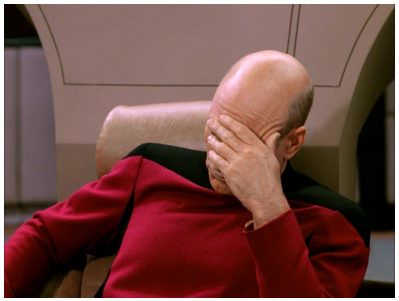


"De plástico joven, como el Darth Vader ese"

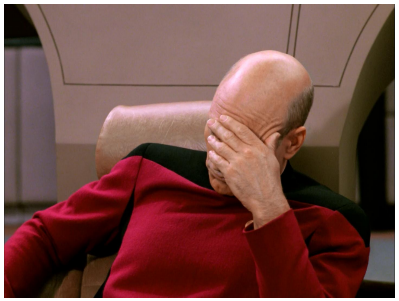
"Comprendo capitán, entonces usted necesita desbloquear la obstrucción del conducto evacuador del baño, que es producido por un elemento sólido"

Ejemplo Doméstico

- ¿Qué ha hecho María?
  - María ha escuchado las necesidades del Capitán Picard
  - María ha preguntado lo que necesita, para saber la gravedad del problema
  - María ha abstraído elementos no importantes del problema (no le importan Data ni Darth Vader)
  - Basado en lo anterior, María ha descrito lo que necesita el Capitán Picard
- María ha analizado el problema y lo que necesita el Capitán Picard

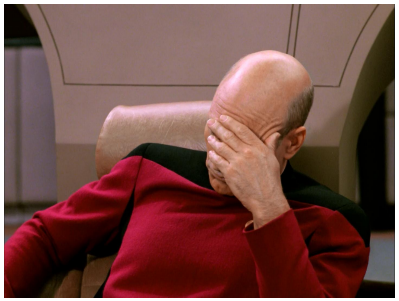


“¿Y lo puede arreglar, joven?”



“¿Y lo puede arreglar, joven?”

“Por supuesto, Capitán”



"¿Y lo puede arreglar, joven?"

"Por supuesto, Capitán"

"Tendré que hacer un corte en el conducto a 10 cm del baño y otro a 20 cm del suelo para sacar el elemento, y luego volveré a poner el mismo tubo. Por último sellaré todo con silicona."

Ejemplo Doméstico

- ¿Qué ha hecho María?
  - María ha pensado en como resolver el problema
  - María ha pensado en qué puede reutilizar en su solución propuesta (podría haber usado otro tubo y cobrar más, pero élla sabe que con el mismo funciona, y María es muy ético)
  - María ha descrito su solución en términos de los elementos que intervienen en la solución
- María ha diseñado una solución

Ejemplo Doméstico

"Voy a entrar a picar . . ."

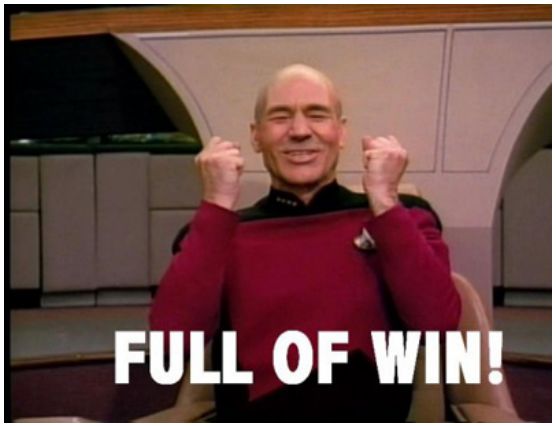
María está implementando la solución

Ejemplo Doméstico

"Listo Capitán, pruebe nomás, lo probé y funciona"

Ejemplo Doméstico

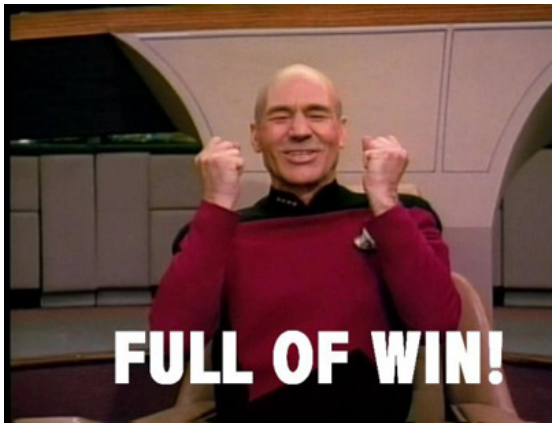
"Listo Capitán, pruebe nomás, lo probé y funciona"



"Funciona perfecto, joven!"

Ejemplo Doméstico

"Listo Capitán, pruebe nomás, lo probé y funciona"



"Funciona perfecto, joven!"

"Muy bien Capitán, esta es mi tarjeta, cualquier problema con el arreglo me avisa"

Ejemplo Doméstico

- ¿Qué ha hecho María?
  - María ha probado su solución
  - María ha ofrecido sus servicios para una eventual mantención de la solución

Ejemplo Doméstico

- En resumen:
  - María analizó el Problema $\rightarrow$ "requisitos"
  - María diseñó la Solución $\rightarrow$ "diseño"
  - María implementó la Solución $\rightarrow$ sistema/programa
  - María probó la Solución $\rightarrow$ certeza
  - María ofreció realizar la mantención de la Solución
- El enfoque usado por María es adecuado también para resolver un problema de software

Ejemplo Doméstico

- En resumen:
  - María analizó el Problema $\rightarrow$ "requisitos"
  - María diseñó la Solución $\rightarrow$ "diseño"
  - María implementó la Solución $\rightarrow$ sistema/programa
  - María probó la Solución $\rightarrow$ certeza
  - María ofreció realizar la mantención de la Solución
- El enfoque usado por María es adecuado también para resolver un problema de software
- ¿Alguna crítica hacia María?

Ejemplo Doméstico

- En resumen:
  - María analizó el Problema $\rightarrow$ "requisitos"
  - María diseñó la Solución $\rightarrow$ "diseño"
  - María implementó la Solución $\rightarrow$ sistema/programa
  - María probó la Solución $\rightarrow$ certeza
  - María ofreció realizar la mantención de la Solución
- El enfoque usado por María es adecuado también para resolver un problema de software
- ¿Alguna crítica hacia María?
  - El uso de lenguaje excesivamente técnico
    - tal vez el Capitán no se convenza de que él entendió el problema
    - tal vez la Capitán no entiende como lo va a resolver

¿Cómo hacemos SW?

- ¿Sabemos hacer software?
  - Tratamos de entender lo que nos piden
  - Ideamos una solución
  - Construimos la aplicación
  - Probamos la aplicación (¿lo hacemos?)
  - Dejamos la tarjeta ☺

¿Cómo hacemos SW?

- ¿Sabemos hacer software?
  - Tratamos de entender lo que nos piden
  - Ideamos una solución
  - Construimos la aplicación
  - Probamos la aplicación (¿lo hacemos?)
  - Dejamos la tarjeta ☺
- ¿. . . Solos? quizás en grupos pequeños

¿Cómo hacemos SW?

- ¿Sabemos hacer software?
  - Tratamos de entender lo que nos piden
  - Ideamos una solución
  - Construimos la aplicación
  - Probamos la aplicación (¿lo hacemos?)
  - Dejamos la tarjeta ☺
- ¿. . . Solos? quizás en grupos pequeños
- Problema: El software moderno . . .
  - Es más difícil de entender y resolver; es complejo, quizás con sub-partes especializadas
  - Requiere de plazos más extensos para ser resuelto
  - Típicamente requiere equipos, no individuos aislados
  - Se deben generar productos intermedios en el desarrollo

Un poco de historia . . .

1 programador-usuario
Problema simple $\rightarrow$
Codificar-corregir

Inicialmente

- El desarrollo de software era tarea de una sola persona.
- El problema (de tipo científico o ingenieril) estaba bien acotado y bien comprendido.
- Generalmente, el usuario final (científico o ingeniero) era el mismo programador, quien desarrollaba software para apoyar su propio trabajo.
- La aplicación era más bien simple y el desarrollo se reducía a la codificación en un lenguaje, típicamente de bajo nivel.
- El modelo usado era de codificar-corregir:
  - Escribir el código; y revisar y eliminar los errores o mejorar/aumentar la funcionalidad.

Historia

1 programador-usuario Problema simple Codificar-corregir	→	Mejor HW PC masificado Problemas más complejos	→

Luego:

- El hardware aumentó sus capacidades y disminuyó sus costos
- Se amplió el ámbito de aplicaciones y se masificó el uso de computadores.
- Se incursionó en áreas donde los problemas no estaban bien acotados (p.ej. administrativos) y el desarrollo se tornó más complejo.

Historia

1 programador-usuario
Problema simple
Codificar-corregir
→
Mejor HW
PC masificado
Problemas más complejos
→
Mantención de SW
Demanda por calidad
→

Más tarde:

- Pasó a haber una gran masa de software pre-existente
- Aparece un problema aún mayor: mantener los sistemas.
- Usuarios-programadores ya no manejan todo el proceso de SW
- Surgen demandas por mayor calidad y confiabilidad de las aplicaciones y del proceso de SW.

Historia

1 programador-usuario Problema simple Codificar-corregir	→	Mejor HW PC masificado Problemas más complejos	→	Mantención de SW Demanda por calidad	→	Grupos de desarrollo Calidad

Finalmente:

- El desarrollo se convierte en una actividad de grupo
- Exige planificar, organizar y estructurar el trabajo en torno a "proyectos".
- La comunicación entre humanos (usuario-desarrollador y desarrollador-desarrollador) se convierte en un problema.

"Crisis del software" a fines de los '60

Crisis del Software

Problemas identificados:

- Los proyectos no terminaban en plazo.
- Los proyectos no se ajustaban al presupuesto inicial.
- La calidad del software generado era baja.
- Software no cumplía con las especificaciones.
- El código era inmantenible $\rightarrow$ dificultaba la gestión y evolución del proyecto.

Y cuando el SW funciona la mayoría de las veces . . . aún puede mejorar

“Ingeniería” de Software

Propuesta de la disciplina

> *Establecimiento y uso de principios con caracteres de ingeniería apropiados para obtener, eficientemente, software confiable, que opere eficaz y eficientemente en máquinas reales*

En una Conferencia de la OTAN en Alemania (1968), para abordar la crisis del software.

Percepciones del Desarrollo de SW

Aun así, siguen los problemas en los proyectos de desarrollo:

- Retraso respecto al potencial de hardware
- Insatisfacción de la demanda
- Bajo cumplimiento de compromisos – costos, plazos
- Baja calidad y satisfacción de clientes/usuarios
- Mantención de sistemas legados
- Ineficiencia
- Altos costos
- Baja confiabilidad
- Escasa ingeniería

Algunos mitos

- Estándares y procedimientos bastan
- Tecnología de punta basta
- Más gente para ponerse al día
- Programación inmediata
- Fácil acomodo de los cambios
- Programación: fin del trabajo
- Calidad: sólo del ejecutable
- Código es el único producto

Ingeniería de Software

- Objetivos
  - Maximizar calidad
  - Maximizar productividad
  - Minimizar riesgos
- La Ingeniería de Software es la disciplina que se ocupa de la construcción sistemática, eficaz y eficiente de software eficaz y eficiente.
  - Sistemática – principios, métodos y técnicas
  - Eficaz – . . . para lograr los objetivos
  - Eficiente – . . . usando los recursos adecuadamente

ISW como Gestión de Riesgo

Dificultades en Producir Software

- Brooks ("No silver bullet", 1986) arguyó que hacer software siempre será muy complejo
- Problemas accidentales
  - Solubles con avances de investigación
- Problemas esenciales
  - Complejidad
  - Conformidad
  - Necesidad de cambios
  - Invisibilidad

Proyectos atrasados y cancelados

Estudio de 8000 proyectos en EEUU [Standish 2000]

- En plazo y costo: 9%-16%
- Retrasados/excedidos: 53%-60%
- Cancelados (abortados): 31%
  - Requisitos incompletos: 13%
  - Falta de participación del usuario: 12%
  - Falta de recursos: 11%
  - Expectativas descontroladas: 10%
  - Falta de apoyo gerencial: 9%
  - Requisitos inestables: 9%
  - Falta de planificación: 8%
  - Obsolescencia pre-parto: 7%

Costo de problemas

Cuantificación de Impacto

- Costo de reparar errores en los requisitos . . .
  - Detectado durante análisis: $X
  - Detectado durante diseño: $5X
  - Detectado durante construcción: $10X
  - Detectado durante prueba: $20X
  - Detectado después de entregado: $100X-$200X

Proyectos ineficientes e inefectivos

- Riesgos de construcción de software
  - (R1) riesgo de hacer algo inútil (inefectividad)
  - (R2) riesgo de sub-/sobre-estimar recursos (ineficiencia)
- Clasifiquemos las causas de cancelación
  - Requisitos incompletos: 13% (R1)
  - Falta de participacion del usuario: 12% (R1)
  - Falta de recursos: 11% (R2)
  - Expectativas descontroladas: 10%
  - Falta de apoyo gerencial: 9%
  - Requisitos inestables: 9%
  - Falta de planificación: 8% (R2)
  - Obsolescencia pre-parto: 7% (R1)

Ing. de SW como Gestión de Riesgo

- Propósito: Reducir el riesgo de construcción
  - Inefectividad: riesgo de hacer algo inútil
  - Ineficiencia: riesgo de sub-/sobre-estimar recursos
- Para efectividad:
  - Ingeniería de Requisitos: caracterización del problema correcto
- Para eficiencia:
  - Diseño de Software: resolución del problema correctamente (p.ej. reusando trabajo previo)
  - Calidad de Productos: para minimizar retrabajo
- Enfoque sistemático: empacado en procesos

Procesos de Software

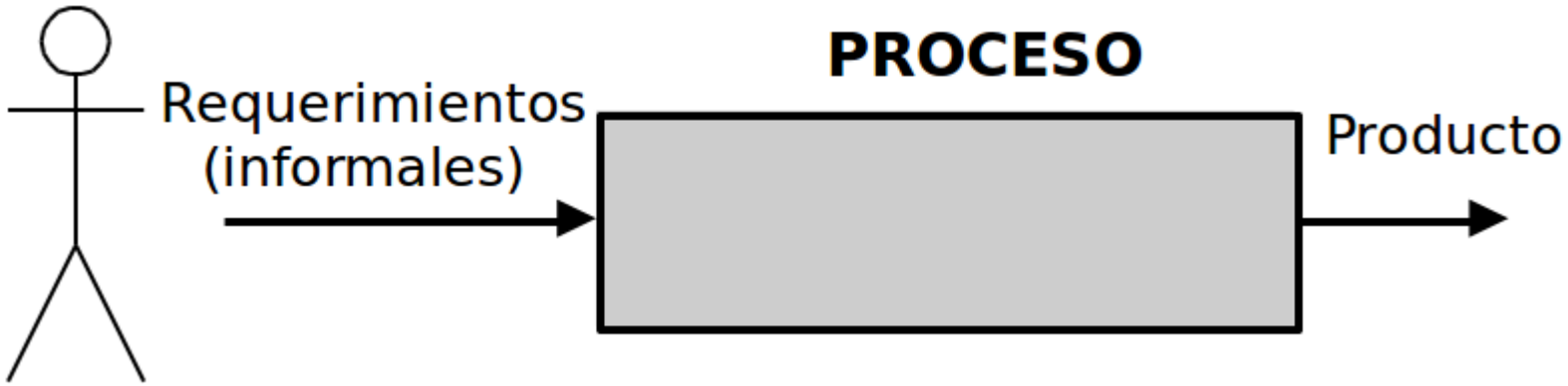
- Proceso: serie de pasos que involucran actividades, restricciones y recursos, y que producen una salida.
  - Impone consistencia y estructura al conjunto de actividades.
  - No es un procedimiento, sino un conjunto organizado de ellos.

Procesos de Software

- Proceso: serie de pasos que involucran actividades, restricciones y recursos, y que producen una salida.
  - Impone consistencia y estructura al conjunto de actividades.
  - No es un procedimiento, sino un conjunto organizado de ellos.
- Un proceso permite/sugiere técnicas y herramientas.
- La estructura del proceso guía la acción, permitiendo examinar, entender, controlar y mejorar actividades.
- Permite capturar y transmitir experiencia de un proyecto a proyectos futuros.

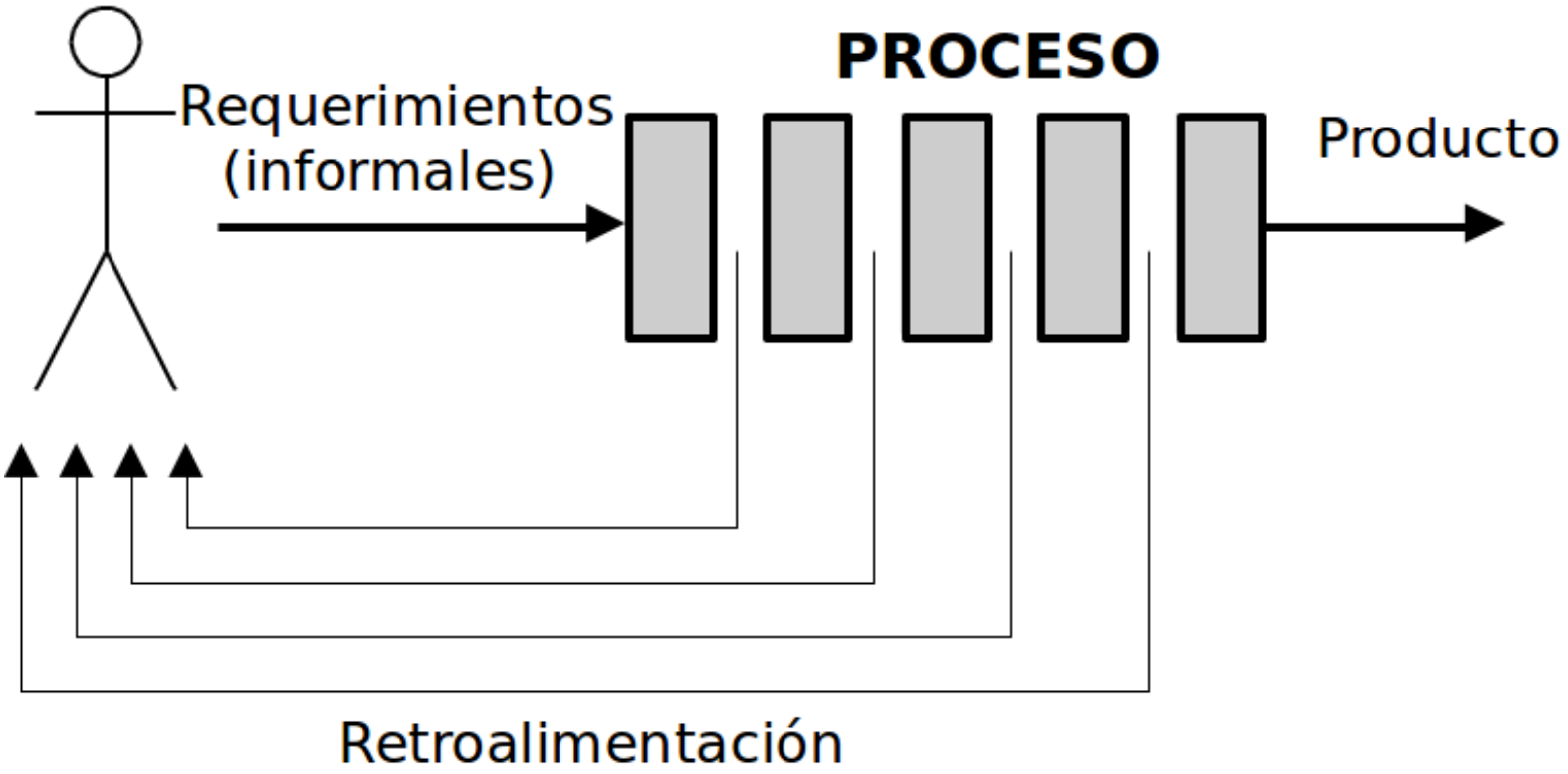
Proceso de software

El proceso de desarrollo a la antigua, como “caja negra”.



Proceso de software

El proceso de desarrollo como se concibe ahora.



Actividades de desarrollo

Estrategias de Desarrollo

- Las actividades para el desarrollo de software pueden ser organizadas de acuerdo a distintas estrategias, conocidas como Modelos de Proceso o Paradigmas de Desarrollo

Actividades de desarrollo

Estrategias de Desarrollo

- Las actividades para el desarrollo de software pueden ser organizadas de acuerdo a distintas estrategias, conocidas como Modelos de Proceso o Paradigmas de Desarrollo
- Versión simplista
  - Especificación $\rightarrow$ Diseño $\rightarrow$ Validación $\rightarrow$ Evolución

Actividades de desarrollo

Estrategias de Desarrollo

- Las actividades para el desarrollo de software pueden ser organizadas de acuerdo a distintas estrategias, conocidas como Modelos de Proceso o Paradigmas de Desarrollo
- Versión simplista
  - Especificación $\rightarrow$ Diseño $\rightarrow$ Validación $\rightarrow$ Evolución
- Pero hay más actividades para hacer Software
  - Formalizar las actividades en un Proceso de Desarrollo
  - Administrar cambios y versiones de los productos de trabajo (Gestión de la configuración)
  - Gestionar, planificar y medir la ejecución del desarrollo
  - Obtener, desarrollar o adaptar herramientas y métodos para el desarrollo
  - Asegurar la Calidad del Software

En este curso hablaremos de . . .

- Requisitos
- Procesos de desarrollo
- Diseño (arquitectura vs detallado, interacciones)
- Reuso
- Mantención y evolución
- Verificación y validación