# Calidade e probas en robótica

## 2022/23

Métricas aplicadas al desarrollo de software


Marcos Boullón Magán

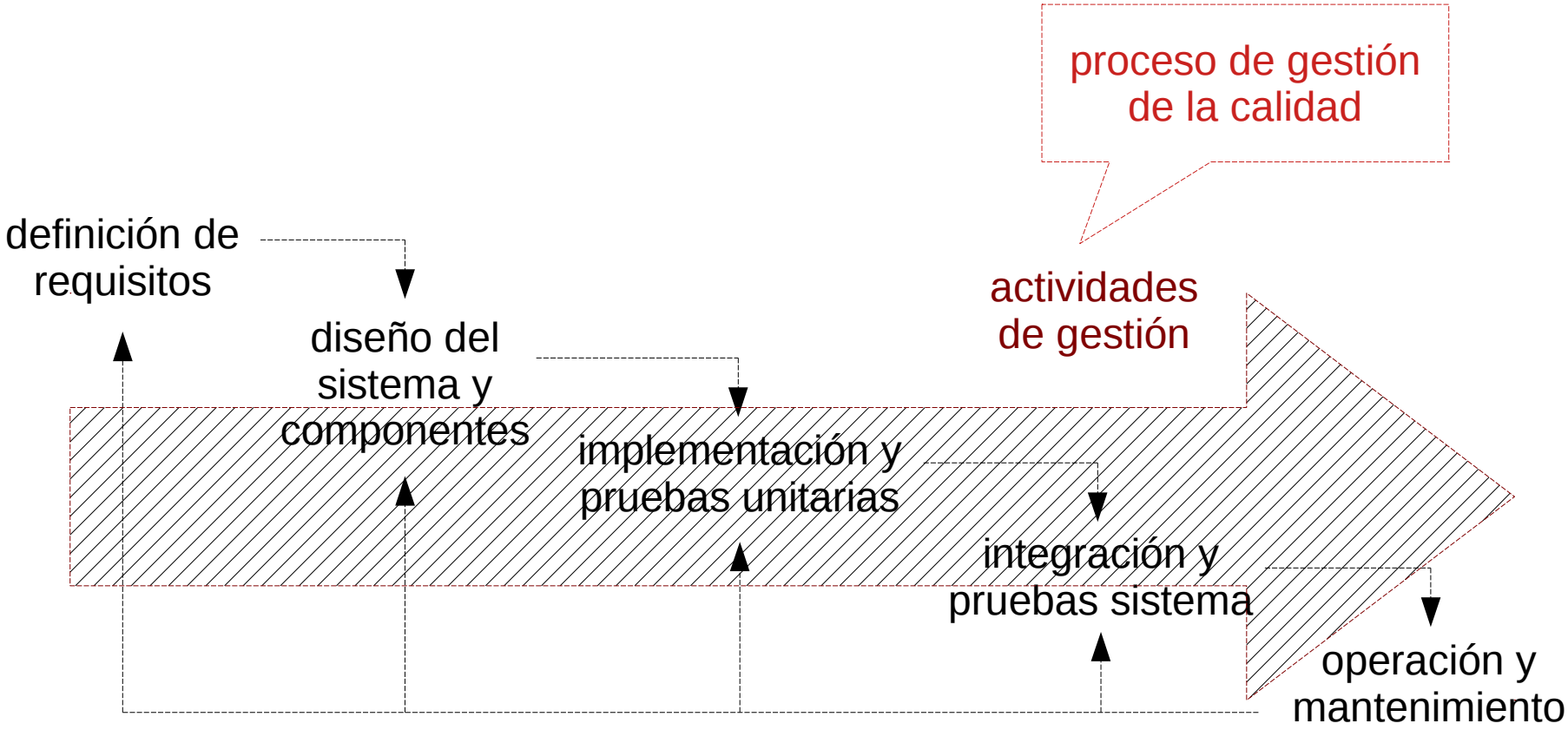
*Depto. Electrónica e Computación*

# Modelos de proceso

Hay muchos procesos propuestos para desarrollar sistemas software, pero todos incluyen las mismas **actividades fundamentales**:

- especificación de funcionalidades
- desarrollo
  - diseño del sistema y de componentes
  - implementación
- validación
- mantenimiento y evolución

# Modelos de proceso (2)



# Problemas, problemas, problemas

Los primeros grandes desarrollos software, en la década de los 60, van a presentar problemas de *calidad*:

- son lentos
- poco fiables
- difíciles de mantener y de reutilizar

El problema fundamental es la inexistencia de métodos (formales) para convertir el proceso artesanal de construcción de programas en uno industrial.

*¡Crisis del software!*

# Problemas, problemas, problemas (2)

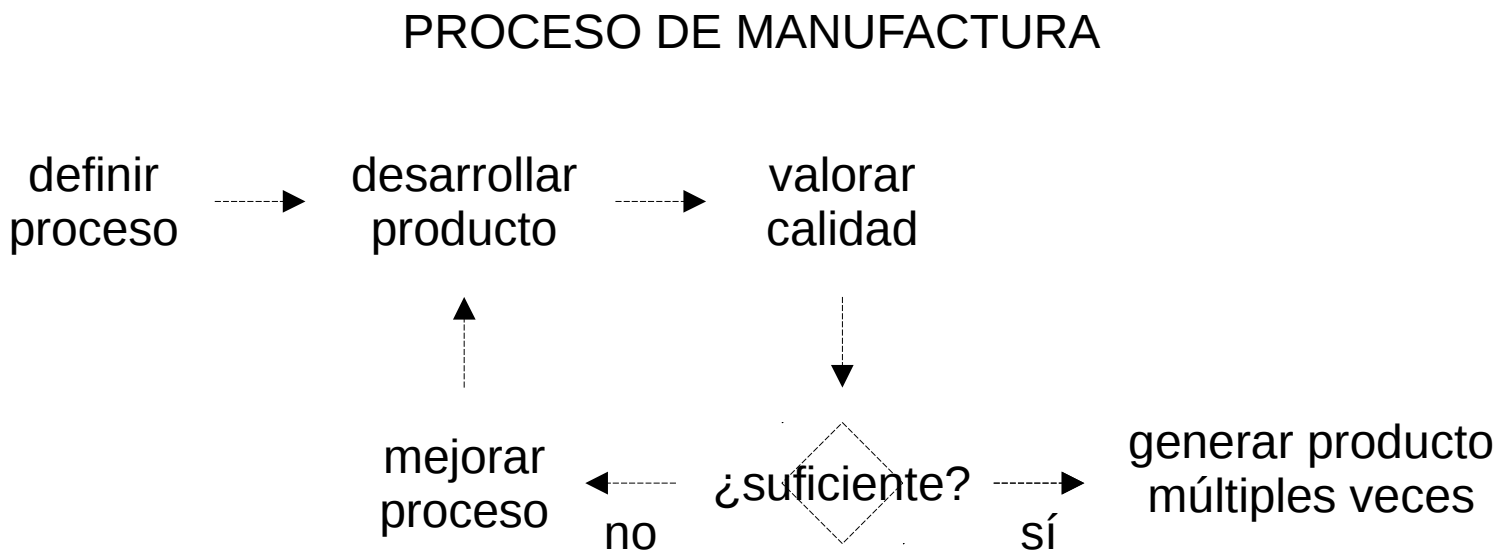
Las técnicas que se manejan en aquel momento para la gestión de la calidad provienen de las **industrias de manufactura**.

Definen la *calidad* como la conformidad con una especificación detallada del producto ideal, dentro de unos márgenes de error (tolerancia).

Se apoya en dos ideas fundamentales:

- un producto puede ser especificado perfectamente, a la vez que se pueden establecer procedimientos para comprobar el nivel de conformidad del producto terminado (si no llega a las especificaciones perfectas)
- la calidad del proceso de manufactura está estrechamente relacionada con la calidad del producto… así que se puede ir corrigiendo el funcionamiento de las máquinas para optimizar la calidad del producto

# Problemas, problemas, problemas (3)



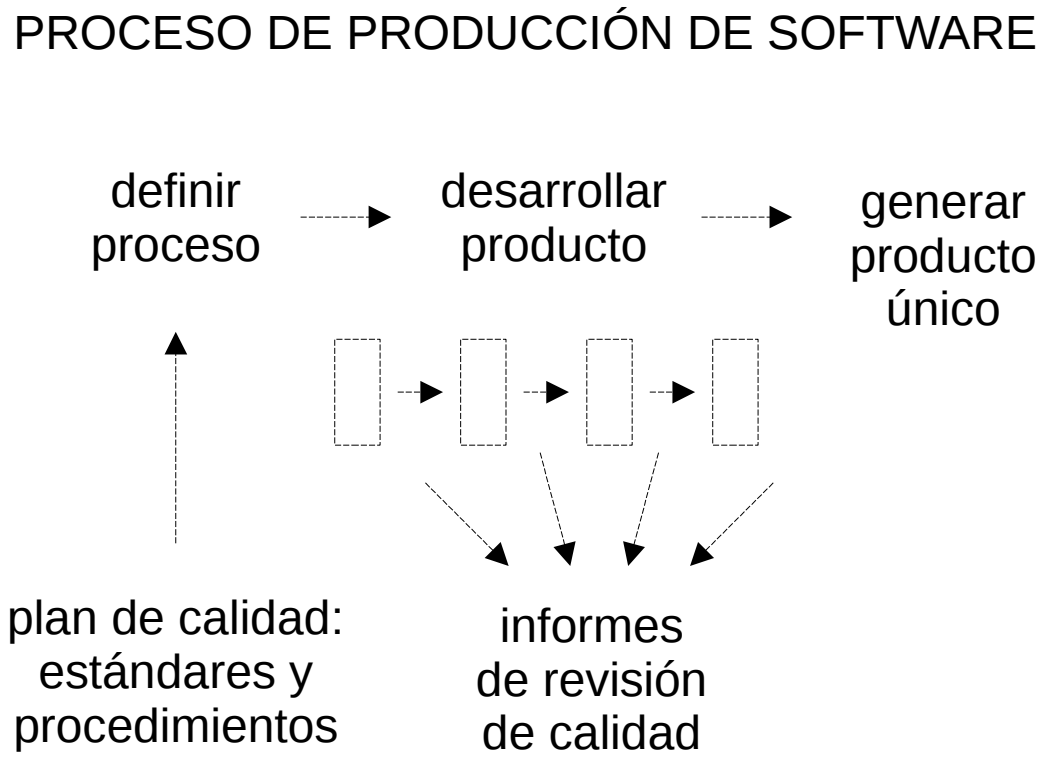
# Problemas, problemas, problemas (4)

La **industria del software** tiene características propias que no se corresponden con las manufacturas:

- el software se diseña, no se manufactura
- se genera un único producto, no múltiples copias del mismo
- no son aplicables los conceptos de especificación perfecta o tolerancia
- muchas veces ni siquiera podemos saber si se cumplen las especificaciones:

  imprecisión en la especificación de requisitos; no hay una evaluación única porque se trata de contentar a múltiples usuarios con diferentes requerimientos (contradictorios), y se busca un compromiso; ni siquiera se pueden medir todos los requisitos: no hay unidades (*adaptabilidad*), sólo se pueden evaluar una vez está construido (*mantenibilidad*)...

# Problemas, problemas, problemas (5)



# Proceso de producción de software

El software es un *proceso creativo*; demasiada estandarización puede ser contraproducente. Influyen tanto las habilidades y experiencias individuales, como la calidad de las herramientas, la gestión de plazos y recursos…

Existe una *relación compleja* entre la calidad del proceso y la calidad del producto. Sin embargo, sabemos que una buena calidad del proceso produce menores defectos en el producto.

# Proceso de producción de software (2)

Por tanto la valoración de la calidad es un *proceso subjetivo*.

El equipo encargado de esta valoración debe decidir si el producto se ajusta al propósito pretendido. Para ello evalúa:

- seguimiento de estándares y procesos definidos
- pruebas de desarrollo y sus registros, para asegurar que se efectúan correctamente y así:
  - que las funcionalidades estén bien implementadas
  - que se consideren los atributos no funcionales del sistema
- ¿es el software confiable? ¿tiene un rendimiento aceptable? ¿es comprensible? … ¿se puede usar?

# Proceso de producción de software (3)

Atributos (no funcionales) relacionados con la calidad del sistema (son contradictorios: mejorar algunos, degrada a otros):

| | | |
|---|---|---|
| protección | comprensibilidad | portabilidad |
| seguridad | comprobabilidad | usabilidad |
| fiabilidad | adaptabilidad | reusabilidad |
| flexibilidad | modularidad | eficiencia |
| robustez | complejidad | facilidad de uso |

Los atributos más importantes del sistema son los de *confiabilidad*, seguidos de los de *eficiencia*.

# Calidad en la organización

La gestión de la calidad se debe realizar a dos niveles:

**Organización**. Se establecen los estándares a seguir para todas las etapas del proceso (captura de requisitos, modelos de diseño, codificación, plan de pruebas, documentación…), se definen los procesos de desarrollo del software, y los métodos de verificación y validación de la calidad.

**Producto/proyecto**. Se definen procesos específicos, adaptados, para la validación de la conformidad del producto con las especificaciones. Se registran en el *plan de calidad* del proyecto.

# *QA team*

La organización define un equipo de aseguramiento de la calidad (*quality assurance team, QA team*), que es el equipo responsable de revisar que el producto se esté construyendo de acuerdo a las directrices dadas:

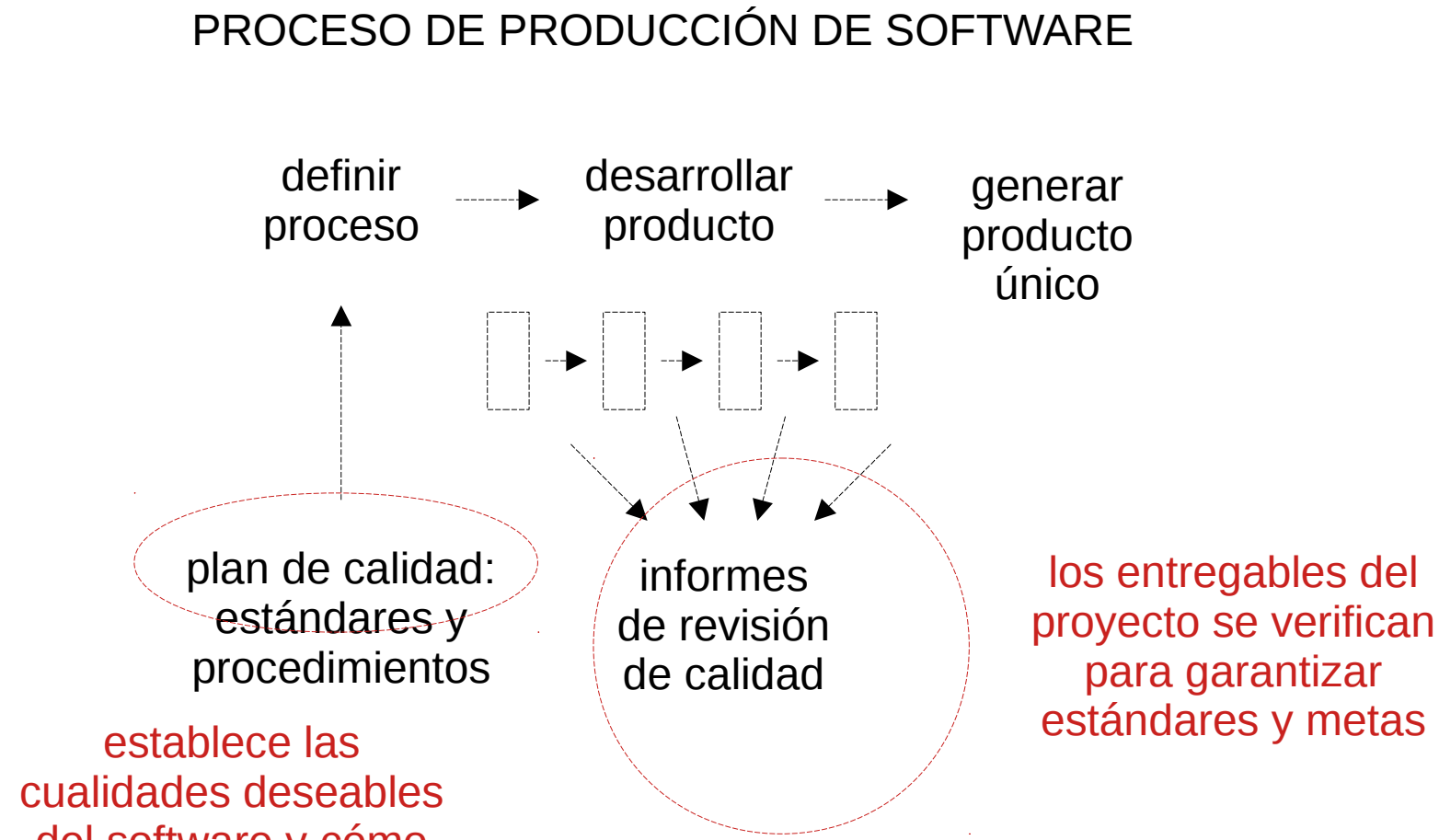
- uso de los procedimientos definidos
- aplicación de estándares
- revisar las pruebas y sus resultados
- verificar los requisitos funcionales y no funcionales
- mantener registros del proceso de validación

# *QA team* (2)

El *QA team* realiza una comprobación **independiente** y **pública** sobre cada etapa del desarrollo de software (el plan de calidad indica qué y cómo valorar). Para evitar suspicacias, este equipo normalmente estará compuesto por *usuarios que no participan en el desarrollo* del producto.

*NOTA: En procesos de desarrollo software ágiles, los procesos de aseguramiento de la calidad son mucho más informales.*

# *QA team* (3)


PROCESO DE PRODUCCIÓN DE SOFTWARE
definir proceso
desarrollar producto
generar producto único
plan de calidad: estándares y procedimientos
informes de revisión de calidad
establece las cualidades deseables del software y cómo valorarlas: priorización, métricas...
los entregables del proyecto se verifican para garantizar estándares y metas

# Cultura de calidad

¿Se puede optimizar la calidad del producto sólo a través de procedimientos y estándares (ya que los planes de calidad parecen estar centrados en la codificación)?

*Probablemente NO.*

Hay aspectos intangibles en el desarrollo:

- legibilidad
- simplicidad
- …

Sólo se pueden aplicar a través de una *cultura de calidad* en la organización.

# Plan de calidad

Secciones del documento:

- introducción al producto (expectativas)
- planes del producto (fechas críticas, responsabilidades, planes de servicio…)
- descripción de procesos (procesos y estándares a utilizar)
- metas de calidad (identificación y justificación de los atributos de calidad)
- riesgos y gestión de riesgos (identificación de riesgos y acciones a tomar)

# Plan de calidad (2)

Aspectos más importantes del proceso de gestión de la calidad:

- estándares de desarrollo

  entregan valor en forma de aumento de calidad del producto

- revisiones e inspecciones

  comprobar la calidad de los entregables del proyecto

- métricas

  asignar valores numéricos a los requisitos funcionales y no funcionales

# Estándares software

La organización hará una selección de los estándares software, métodos y herramientas para aplicar los anteriores y monitorizar su aplicación.

Los **estándares** no son más que buenas prácticas del sector, destiladas a lo largo de los años y depuradas por expertos.

- permiten definir escenarios donde evaluar la calidad: si estos estándares reflejan las expectativas del cliente en confiabilidad, usabilidad, rendimiento…
- crean una continuidad en el proyecto más allá de desarrolladores individuales; al adoptar todos las mismas prácticas, hay un menor esfuerzo de aprendizaje

# Estándares software (2)

**Estándar de proceso**. Establecen las buenas prácticas a seguir:

*definición de especificaciones, procesos de diseño, procesos de validación, herramientas de soporte, envío de código, proceso de liberación de una versión, proceso de control de cambios, proceso de registro de prueba...*

**Estándar de producto**. Concreción de los anteriores. Diseñados de manera que puedan aplicarse y comprobarse de manera efectiva (tiempo, coste) sobre el producto concreto:

*estructura del documento de requisitos, formatos de documentación del código, estilo de programación, formato del plan de proyecto, lenguaje, herramientas y modelos de codificación, formato de solicitud de cambio...*

# Estándares software (3)

Existen diferentes organismos internacionales de estandarización software: US DoD, OTAN, ANSI, IEEE, ISO…

Algunas administraciones pueden exigir estándares, procedimientos y metodologías propias:

- AMTEGA (Axencia para a Modernización Tecnolóxica de Galicia)

  *https://amtega.xunta.gal/documentacion-de-interese-para-provedores/estandares-de-desenvolvemento-e-adquisicion-de-aplicacions*

- MADEJA, Marco de desarrollo de software de la Junta de Andalucía

  *https://www.juntadeandalucia.es/servicios/madeja/*

- METRICA3, Consejo Superior de Informática

  *https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html*

# Estándares software (4)

Estos estándares deben evolucionar y adaptarse para reflejar las circunstancias de la empresa y tecnologías cambiantes: distintos tipos de software requieren distintos procesos de desarrollo.

> *los estándares aplicados dependerán de la negociación entre el administrador del proyecto y el administrador de calidad*

La organización utilizará aquellos mejor soportados por herramientas, plantillas, estilos… Y puede adoptar nuevos estándares en respuesta a requisitos del cliente o del proyecto.

# Estándar ISO 9001

La familia de estándares ISO 9000 sirve para desarrollar sistemas de administración de la calidad.

El estándar ISO 9001 es el más general. Se aplica a organizaciones que diseñan, desarrollan y mantienen productos (incluidos productos software).

Define nueve procesos centrales que deben ser documentados, definiendo y manteniendo registros que demuestren la aplicación de estos procesos de calidad. PERO NO SE DEFINEN LOS PROCESOS CONCRETOS.

# Estándar ISO 9001 (2)

Procesos de entrega de producto:

- adquisición empresarial
- diseño y desarrollo
- prueba
- producción y entrega
- servicio y soporte

Procesos de soporte:

- administración empresarial
- administración del proveedor
- administración del inventario
- administración de la configuración

# Estándar ISO 9001 (3)

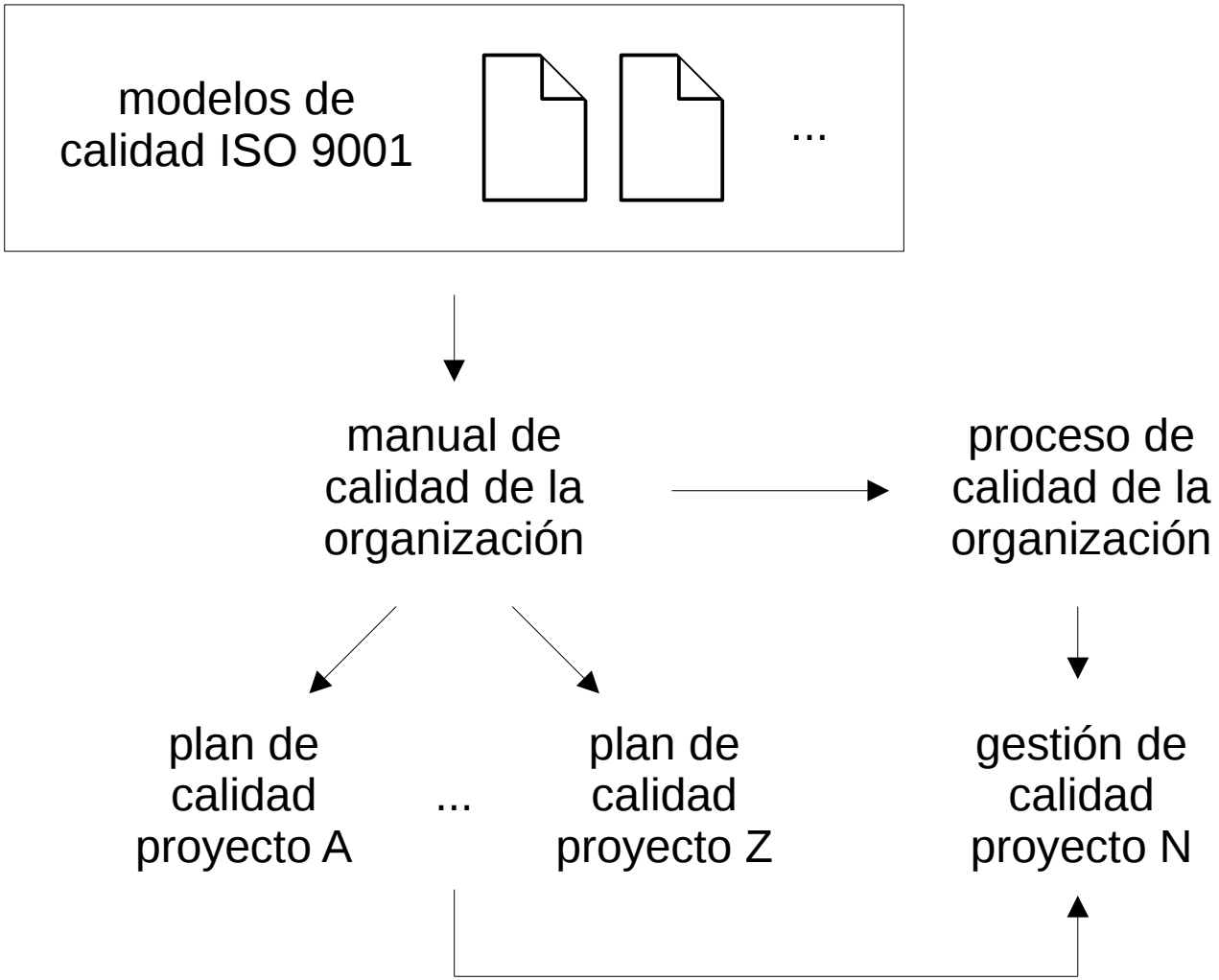
Existen autoridades de acreditación independientes que certifican que los procesos de calidad de una organización concuerdan con el estándar ISO 9001 (ENAC, AENOR…).

Pero no existe la seguridad de que las organizaciones acreditadas con este ISO 9001 empleen las mejores prácticas para el desarrollo de software o produzcan software de mayor calidad: para ello hay que usar los mejores procesos; ¡el ISO 9001 sólo comprueba que se siguen los procesos anunciados, no su validez!

*NOTA: Los procesos de desarrollo software ágiles se llevan bastante mal con los estándares ISO 9001.*

# Estándar ISO 9001 (4)

Relación con el proceso de calidad de la organización:



# Inspecciones y revisiones

Inspecciones y revisiones se usan junto con las pruebas como parte del proceso de verificación y validación.

Objetivo: comprobar la calidad de los entregables del proyecto, examinando software, documentación y registros del proceso, para descubrir errores, problemas y omisiones, y comprobar que se siguen los estándares de calidad.

# Proceso de revisión

Es un proceso **público** para la mejora de la calidad:

Un grupo de personas examinan software y documentación (especificaciones, documentos de diseño, código, modelos del proceso, planes de prueba, procesos de gestión de la configuración, estándares de proceso, manual de usuario…)

buscando problemas potenciales (errores, problemas, omisiones…) y falta de conformidad con estándares

y realizan juicios informados sobre el nivel de calidad de un entregable o del proyecto en su conjunto

para tomar decisiones de planificación y asignación de recursos.

# Proceso de revisión (2)

Las conclusiones de la revisión deben registrarse formalmente como parte del proceso de gestión de la calidad.

Los comentarios de los revisores deben pasar a los responsables.

Esta revisión confirma la coherencia e integridad de la documentación del proyecto.

# Proceso de revisión: actividades previas

Establecer el equipo de revisión

*roles: moderador, autor del código, secretario, inspectores, diseñador ejecutivo que tome las decisiones técnicas significativas, otros miembros del proyecto (participan a través de comentarios escritos), administrador del proyecto autorizado a hacer cambios al plan del proyecto*

Selección herramientas

*edición colaborativa de documentos con aprobación/rechazo de comentarios*

Organizar el tiempo

Destinar un lugar

Distribuir la documentación a revisar

# Proceso de revisión: reunión de revisión

Un autor (código, documento) repasa el producto con el equipo línea a línea

Se registran formalmente decisiones y acciones a tomar

Se consideran todos los comentarios escritos previamente

El registro de comentarios y decisiones es firmado por los responsables

# Proceso de revisión: actividades posteriores

Se resuelven los conflictos y problemas surgidos durante la revisión:

- corrección de bugs
- refactorización para adaptarse a estándares
- reescritura

Puede realizarse una revisión posterior para comprobar que se han procesado todos los comentarios.

# Proceso de revisión en metodologías ágiles

En procesos de desarrollo software ágiles estas revisiones son bastante informales: una junta de revisión se convoca después de cada incremento para exponer conflictos y problemas de calidad.

Sin embargo no es un proceso formal (se considera que estas reuniones ralentizan el ritmo de desarrollo).

# Proceso de inspección

En las inspecciones (revisión por pares) los miembros del equipo colaboran para encontrar bugs en el producto en desarrollo: defectos, problemas, anomalías…

Sirven como un complemento de las pruebas.

No requieren la ejecución de código, así que se pueden inspeccionar versiones incompletas del producto.

Se pueden usar los casos de prueba para la inspección, pero lo normal es revisar los documentos básicos: modelos de diseño, código fuente...

# Proceso de inspección (2)

Durante las inspecciones se utiliza una lista de verificación de errores comunes para enfocar la búsqueda.

Esta lista ha sido compilada mediante experiencia, dependen del lenguaje y del dominio de la aplicación, y es específica de la organización que la usa.

Esta lista se actualiza regularmente según se encuentran nuevos tipos de defectos.

| | |
|---|---|
| Fallos de datos | ¿Todas las variables del programa se inician antes de usar sus valores? ¿Todas las constantes tienen nombre? ¿La cota superior de los arreglos es igual al tamaño del arreglo o Valor – 1? Si se usan cadenas de caracteres, ¿se asigna explícitamente un delimitador? ¿Hay posibilidad de desbordamiento de buffer? |
| Fallos de control | Para cada enunciado condicional, ¿la condición es correcta? ¿Hay certeza de que termine cada ciclo? ¿Los enunciados compuestos están correctamente colocados entre paréntesis? En caso de enunciados, ¿se justifican todos los casos posibles? Si después de cada caso en los enunciados se requiere un paréntesis, ¿éste se incluyó? |
| Fallos de E/S | ¿Se usan todas las variables de entrada? ¿A todas las variables de salida se les asigna un valor antes de que se produzcan? ¿Entradas inesperadas pueden causar corrupción? |
| Fallos de interfaz | ¿Todas las llamadas a función y método tienen el número correcto de parámetros? ¿Los tipos de parámetro formal y real coinciden? ¿Los parámetros están en el orden correcto? Si los componentes acceden a memoria compartida, ¿tienen el mismo modelo de estructura de memoria compartida? |
| Fallos gestión de almacenamiento | Si se modifica una estructura vinculada, ¿todos los vínculos se reasignan correctamente? Si se usa almacenamiento dinámico, ¿el espacio se asignó correctamente? ¿El espacio se cancela explícitamente después de que ya no se requiere? |
| Fallos gestión de excepciones | ¿Se tomaron en cuenta todas las posibles condiciones de error? |

# Proceso de inspección en metodologías ágiles

Los procesos ágiles apenas usan estos procesos de inspección formal, sino que prefieren apoyarse en las buenas prácticas de los otros desarrolladores (técnica de programación en grupos).

# Métricas

La **medición** es la derivación de valores numéricos para un atributo de un componente, sistema o proceso. Al compararlos con valores esperados (a partir de líneas base, valores de referencia, históricos), tenemos una medida de la calidad del software o una valoración de la efectividad del sistema de producción.

El objetivo principal es *utilizar estas mediciones en lugar de revisiones* para realizar juicios sobre la calidad, para evitar realizar estas revisiones cuando los valores entran en el umbral de calidad establecido.

Objetivo secundario: identificar áreas susceptibles de mejora.

# Métricas (2)

Una **métrica** es una característica del sistema o del proceso que puede ser medida de manera objetiva:

- tamaño del programa en líneas de código
- índices de complejidad de texto
- número de fallos detectados por el cliente
- días-hombre requeridos para desarrollar un módulo
- ...

# Métricas (3)

Por sus características, distinguimos entre

- **métricas de control** (métricas de proceso), aquellas que apoyan la gestión del proceso de desarrollo

  *esfuerzo promedio (días-hombre) para reparar los errores reportados por el cliente...*

- **métricas de predicción** (métricas de producto), aquellas que permiten predecir las características del software desarrollado

  *complejidad ciclomática de un módulo, longitud promedio de identificadores en el código, número de operaciones disponibles en un objeto...*

Todas estas métricas van a influir en la administración del proyecto. Permiten saber si hay que hacer cambios (métricas de control) y estimar el esfuerzo para ello (métricas de predicción).

# Métricas y calidad

Hay atributos de calidad que no pueden medirse de manera objetiva (mantenibilidad, usabilidad... se refieren a cómo se experimenta el software, que depende de la experiencia del usuario).

Tenemos que derivarlos de características internas del software que sí podamos medir. Condiciones:

- el atributo debe poder medirse con exactitud (tal vez usando herramientas específicas)
- tiene que haber una relación entre el atributo interno medible y el atributo de calidad externo
- esta relación debe poder expresarse como fórmula matemática o modelo (a través del análisis de los datos se extraerán los parámetros y se calibrarán específicamente)

  *por ejemplo, aún después de tantos años no podemos relacionar numéricamente la complejidad ciclomática de un componente y el número de errores esperados...*

# Métricas y calidad en la organización

Las grandes compañías recogen información sobre sus propios productos (errores descubiertos, cambios en los requerimientos de usuario...) para mejorar los procesos de verificación y validación, pero también el desarrollo y para evaluar los cambios sobre procesos y herramientas: *telemetría*.

Buscan una aproximación empírica a la ingeniería de software: recolección de datos, hipótesis, validación/refutación de las mismas y actuación.

El objetivo de estas mediciones es doble:

- calcular numéricamente un valor para la calidad a partir de los valores de todas las métricas
- identificar componentes fuera de los márgenes de seguridad, ya que estos tienen más posibilidades de contener errores (peor diseñados)

# Métricas y calidad en la organización (2)

Dificultades:

- introducir un programa de valoración sistemática del software en la organización es costoso
- no hay estándares para las métricas ni procesos estandarizados de medición y análisis
- una alta variabilidad en los procesos dentro de la organización como para que las métricas sean consistentes
- las métricas y su investigación están orientados a metodologías de desarrollo basado en plan, cuando ahora se utilizan muchas metodologías ágiles; ¿siguen siendo aplicables las mismas conclusiones?
- la introducción de mediciones supone una carga extra al desarrollo, que precisamente es lo que quieren eliminar las metodologías ágiles

# Métricas de producto

También llamadas *métricas de predicción*. Están basadas en características internas del producto que pueden ser medidas directamente (tamaño, peso, número de funciones...).

Distinguimos entre **métricas dinámicas**, medidas durante la ejecución (en pruebas, en la vida útil del programa...), que sirven para caracterizar atributos de eficacia y fiabilidad, y **métricas estáticas**, extraídas desde representaciones del sistema (código, diagramas, documentación), que sirven para los atributos de complejidad, comprensibilidad y mantenibilidad.

# Métricas de producto (2)

El problema básico es que mientras que sí parece haber una relación clara y directa entre **métricas dinámicas** y atributos de rendimiento o de fiabilidad, aquellas **métricas estáticas** que pueden medirse fácilmente no tienen una relación tan evidente con atributos como mantenibilidad o comprensibilidad, que dependen más del tipo de sistema.

En este caso los predictores más fiables (empíricamente) son *tamaño del programa* y *complejidad del control*.

# Posibles métricas de producto

Posibles métricas genéricas:

- Fan-in/Fan-out
- Longitud de código
- Complejidad ciclomática
- Longitud de identificadores
- Profundidad de anidado condicional
- Índice Fog

Métricas específicas de la programación orientada a objetos:

- Métodos ponderados por clase
- Profundidad de árbol de herencia
- Número de hijos
- Acoplamiento entre clases de objetos
- Respuesta por clase
- Falta de cohesión en métodos

# Posibles métricas de producto (2)

**Fan-in/Fan-out.** Fan-in (abanico de entrada) es una medida del número de funciones o métodos que llaman a otra función o método (por ejemplo, X). Fan-out (abanico de salida) es el número de funciones a las que llama la función X. Un valor alto para fan-in significa que X está estrechamente acoplado con el resto del diseño y que los cambios a X tendrán extensos efectos dominó. Un valor alto de fan-out sugiere que la complejidad global de X puede ser alta debido a la complejidad de la lógica de control necesaria para coordinar los componentes llamados.

**Longitud de código.** Ésta es una medida del tamaño de un programa. Por lo general, cuanto más grande sea el tamaño del código de un componente, más probable será que el componente sea complejo y proclive a errores. Se ha demostrado que la longitud del código es una de las métricas más fiables para predecir la proclividad al error en los componentes.

**Complejidad ciclomática.** Ésta es una medida de la complejidad del control de un programa. Tal complejidad del control puede relacionarse con la comprensibilidad del programa

# Posibles métricas de producto (3)

**Longitud de identificadores.** Ésta es una medida de la longitud promedio de los identificadores (nombres para variables, clases, métodos, etcétera) en un programa. Cuanto más largos sean los identificadores, es más probable que sean significativos y, por ende, más comprensible será el programa.

**Profundidad de anidado condicional.** Ésta es una medida de la profundidad de anidado de los enunciados condicionales en un programa. Los enunciados condicionales profundamente anidados son difíciles de entender y proclives potencialmente a errores.

**Índice Fog.** Ésta es una medida de la longitud promedio de las palabras y oraciones en los documentos. Cuanto más alto sea el valor del índice Fog de un documento, más difícil será entender el documento.

# Posibles métricas de producto (4)

**Métodos ponderados por clase (WMC).** Éste es el número de métodos en cada clase, ponderado por la complejidad de cada método. Por lo tanto, un método simple puede tener una complejidad de 1, y un método grande y complejo tendrá un valor mucho mayor. Cuanto más grande sea el valor para esta métrica, más compleja será la clase de objeto. Es más probable que los objetos complejos sean más difíciles de entender. Tal vez no sean lógicamente cohesivos, por lo que no pueden reutilizarse de manera efectiva como superclases en un árbol de herencia.

**Profundidad de árbol de herencia (DIT).** Esto representa el número de niveles discretos en el árbol de herencia en que las subclases heredan atributos y operaciones (métodos) de las superclases. Cuanto más profundo sea el árbol de herencia, más complejo será el diseño. Es posible que tengan que comprenderse muchas clases de objetos para entender las clases de objetos en las hojas del árbol.

# Posibles métricas de producto (5)

**Número de hijos (NOC).** Ésta es una medida del número de subclases inmediatas en una clase. Mide la amplitud de una jerarquía de clase, mientras que DIT mide su profundidad. Un valor alto de NOC puede indicar mayor reutilización. Podría significar que debe realizarse más esfuerzo para validar las clases base, debido al número de subclases que dependen de ellas.

**Acoplamiento entre clases de objetos (CBO).** Las clases están acopladas cuando los métodos en una clase usan los métodos o variables de instancia definidos en una clase diferente. CBO es una medida de cuánto acoplamiento existe. Un valor alto para CBO significa que las clases son estrechamente dependientes y, por lo tanto, es más probable que el hecho de cambiar una clase afecte a otras clases en el programa.

# Posibles métricas de producto (6)

**Respuesta por clase (RFC).** RFC es una medida del número de métodos que potencialmente podrían ejecutarse en respuesta a un mensaje recibido por un objeto de dicha clase. Nuevamente, RFC se relaciona con la complejidad. Cuanto más alto sea el valor para RFC, más compleja será una clase y, por ende, es más probable que incluya errores.

**Falta de cohesión en métodos (LCOM).** LCOM se calcula al considerar pares de métodos en una clase. LCOM es la diferencia entre el número de pares de método sin compartir atributos y el número de pares de método con atributos compartidos. El valor de esta métrica se debate ampliamente y existe en muchas variaciones. No es claro si realmente agrega alguna información útil además de la proporcionada por otras métricas.

# Proceso de medición de componentes

Se puede integrar un proceso de medición dentro del proceso de valoración de la calidad del software. Cada componente se evalúa mediante un rango de métricas. A través de comparaciones entre varios de estos componentes y refiriéndose a series históricas (incluso de otros proyectos) pueden identificarse problemas de calidad en los componentes:

- elegir mediciones (que respondan a las preguntas que buscamos)
- seleccionar componentes
- medir las características internas de los componentes (automatización)
- identificar mediciones anómalas
- analizar componentes para decidir si hay un problema de calidad
- almacenar los datos recogidos para hacer futuras comprobaciones

# Proceso de medición de componentes (2)

Por supuesto, es posible malinterpretar los datos y hacer inferencias incorrectas. ¡Hay que considerar el contexto donde se han recogido los datos!

Los cambios en la métricas pueden deberse a cambios en la forma de uso del sistema por causas propias o ajenas. Para simplificar el análisis hay que reducir el número de factores que afectan a las mediciones.

# Extra. Complejidad ciclomática

Corresponde con el número de ramas independientes que puede tomar el código en un módulo. Se calcula a partir de un diagrama de flujo (simplificado), contabilizando aristas, nodos y sistemas desconectados:

*M = E - N + 2P*

*M: complejidad; E: número de aristas; N: número de nodos; P: número de sistemas*

*(alternativamente, M = P + 1; P: número de nodos con condicionales)*

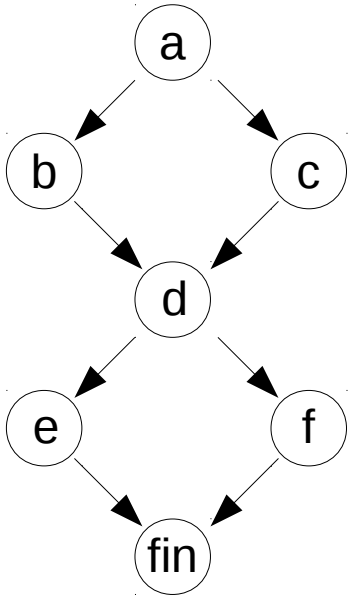
Valores aproximados:

| | |
|---|---|
| *1 < M <= 10* | código simple |
| *11 < M <= 20* | código moderadamente complejo |
| *20 < M <= 50* | código complejo |
| *M > 50* | el código no puede testearse |

# Extra. Complejidad ciclomática (2)

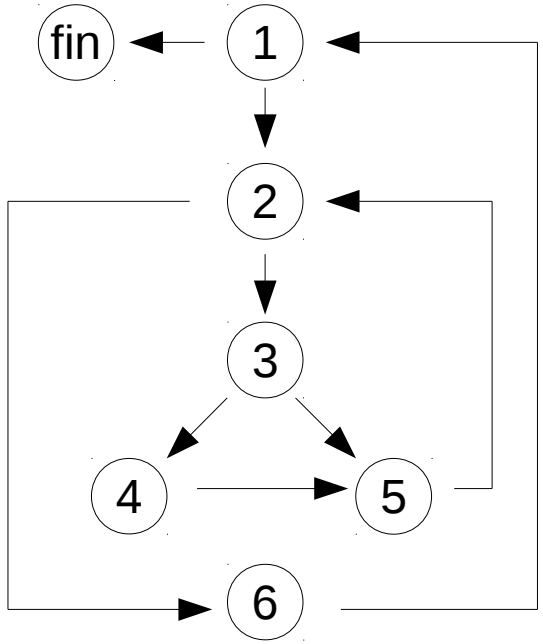
```
IF A:
    B
ELSE:
    C

IF D:
    E
ELSE:
    F
```




*M = 8 - 7 + 2*1 = 3*
*(alter.) M = 2 + 1*

```
      A
1     WHILE B:
          C
2         WHILE D:
3             IF E:
4                 F
5             ...
6         G
7         fin
```




*M = 9 - 7 + 2*1 = 4*
*(alt.) M = 3 + 1*

# Métricas de proceso

Existe la idea *intuitiva* de que mejorar la calidad del proceso influye en la calidad del producto. En ese caso, se puede modificar el proceso de desarrollo para reducir las posibilidades de introducir defectos y mejorar la detección.

Esto es correcto en las industrias de manufactura, donde se puede aplicar un control estadístico de procesos para identificar problemas y recalibrar las máquinas.

En producción intelectual, sin embargo, las habilidades y experiencia del personal es tan importante como el proceso concreto de desarrollo.

# Métricas de proceso (2)

Se identifican los siguientes factores que influyen en el proceso de construcción de software:

- la calidad del proceso
- la calidad del personal
- coste, tiempo, calendarización de recursos
- tecnología

En proyectos grandes, con muchos subsistemas, personal trabajando en remoto... el factor principal es el proceso; las tareas más intensivas son la integración de módulos, la gestión del proyecto, la comunicación entre equipos...

En proyectos pequeños, los son la habilidad del equipo y las herramientas que usan (en este caso, un buen proceso sirve para limitar los daños, pero no para reducirlos).

# Métricas de proceso (3)

También llamadas *métricas de control*. Estas mediciones del proceso son datos cuantitativos acerca del proceso del desarrollo (tiempos de ejecución, esfuerzo...) y su eficiencia. Pero no sirven para saber si mejora la calidad del producto.

Tipos de métricas de proceso:

- tiempo que tarda en completarse una tarea
- recursos empleados (tiempo, dinero...)
- número de ocurrencias de un mismo evento (detección de errores, solicitud de cambios, líneas modificadas ante un error...)

# Métricas de proceso en la organización

Al igual que antes, hay que considerar en conjunto la propia variación de valor en estos cambios más todos los factores extra que influyen en el proceso, para saber si los cambios propuestos/probados han mejorado el proceso:

- medición
- análisis

  *identificar las actividades implicadas en el proceso; entender las relaciones entre actividades y mediciones; relacionar el proceso con otros procesos comparables en otras partes de la organización... en resumen: construir un modelo del proceso con actividades del proceso y transferencia de información entre actividades*

  *cuestionarios y entrevistas; estudios etnográficos*

- implementación de cambios

  *identificación de mejoras para atacar los problemas de calidad, ineficiencias, cuellos de botella...; priorización de mejoras; integración de cambios (actualizar procedimientos, métodos y herramientas); y capacitación del personal*