Principalmente se hará foco en:

- Proceso (definición)
- QA (definición, objetivos, beneficios)
- QC (definición, objetivos, beneficios)
- PDCA (definición, objetivos, beneficios)
- Calidad de producto (definición, objetivos, beneficios)
- Métrica (definición)
- Indicador (definición)
- La calidad, (responsabilidades)
- ISO 9000 (que es?)
- Plan de calidad (para que sirve?)

sean libres de modificar o agregar contenido

Proceso

¿Qué es un proceso?

- Es una manera concreta y repetible de lograr un objetivo.
- Es el conjunto de actividades, métodos y herramientas, las cuales permiten el desarrollo de un producto de manera controlable y repetible
- Es el conjunto de actividades/eventos que se realizan con un determinado fin
- Sistema por el cual un conjunto de recursos y actividades interrelacionados transforman elementos de entrada en elementos de salida.

Los resultados del negocio son producidos por procesos del negocio con objetivos, políticas y restricciones
El workflow de los procesos es un factor determinante en la productividad del negocio.

Los procesos permiten estandarizar, exigen coordinación y clarificación de responsabilidades. Facilitan la comunicación y comprensión.

Cuando una empresa asegura calidad, está garantizando una adecuada visibilidad sobre el proceso de software que se utiliza, y los productos que son construidos. Incluye las revisiones y auditorías de los productos y actividades para verificar que se cumplen con procedimientos/estándares.

Atributos del Proceso:

- Definido
- Documentado
- Soportado
- Conocido
- Practicado
- Medido
- Mejorable

¿Qué es QA (Quality Assurance)?

QA hace referencia a la calidad presente en los **procesos** mediante los cuales se desarrollan los productos de Software.
Estos procesos, deben estar bien definidos en la metodología de la organización, junto con los estándares, políticas, templates, capacitaciones.

Una metodología define un conjunto de artefactos que deben ser utilizados para dar soporte al proceso de desarrollo.

El **objetivo** de QA es asegurar que se cumplan los procesos, estándares y planes, haciendo **foco** en los procesos del proyecto, guiando y monitoreando los mismos. Así como el QC encuentra defectos, el QA aprovecha los resultados de QC para evaluar y mejorar los procesos con los que se desarrolla el producto.

¿Qué hace un QA?

- Participar en el desarrollo de la definición del proceso de software para el proyecto
    - Reuniones de adaptación de procesos junto a líderes de proyecto y principales referentes de distintas áreas.
    - Se adapta la metodología de la organización a las necesidades concretas del proyecto; Se definen prácticas, artefactos entregables, métricas, herramientas de seguimiento y control.
- Preparar un plan de QA
    - Auditorías y revisiones que se realizan
    - Estándares aplicables al proyecto
    - Procedimientos para el informe y seguimiento de incidentes
    - Los resultados que se deben obtener
- Revisar las actividades de ingeniería de software para verificar que se ajusten al proceso de software
    - Auditar productos seleccionados para verificar que se ajustan con los definidos en el proceso del proyecto
    - Identificar, documentar y realizar seguimiento de desviaciones
- Garantizar que las desviaciones estén documentadas y se gestionen de acuerdo con los procedimientos definidos
    - El grupo de QA debe verificar que se realicen las correcciones necesarias para resolver incidentes
    - Elaborar un reporte para informar a las distintas áreas sobre resultados de la revisión
- Reportar resultados
    - Los resultados se pueden reportar a través de reportes cualitativos (ej, lista de incidentes) y cuantitativos (métricas. Promedio de incidentes identificados por revisión; Tiempo medio de demora en resolución de incidentes)
- Proceso de cierre de proyectos (Post Mortem)
    - Se identifican las buenas y malas prácticas presentes en el proyecto
    - Se recolectan sugerencias

- El feedback que pueda tener un impacto en el proceso de desarrollo, sirve para el proceso de mejora continua.

¿Cuáles son los beneficios de contar con estas actividades de QA?

- Mejora en los procesos: La calidad de un producto se basa en la calidad del proceso que se usa para producirlo.
- Es un pilar para el proceso de mejora continua.
- Permite capitalizar las mejores prácticas implementadas en distintos proyectos a lo largo del tiempo.
- Cuando existe un proceso formal de control de calidad, le aporta mucho valor a los clientes.

¿Qué es QC (Quality Control)?

El **objetivo** del QC, es detectar problemas en los productos, haciendo **foco** en el contenido haciendo revisiones de productos. El QC sabe como funciona o como debería funcionar el producto.
El principal **beneficio** que tiene contar con QC, es asegurarnos que cuando un producto llega al cliente o usuario final, esté libre de fallos (o con la menor cantidad de errores posibles).

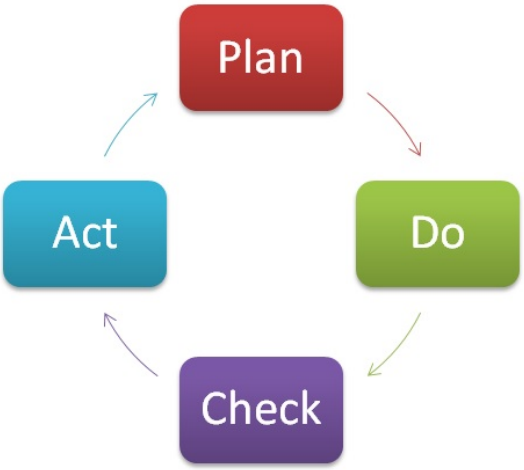
Atributos del Producto:
- Requerimientos
- Mantenibilidad
- Confiabilidad
- Seguridad
- Rendimiento
- Disponibilidad

QA: Procesos. Proactivo. Prevenir Defectos
QC: Producto. Reactivo. Encontrar Defectos

Ciclo de Deming (Plan - Do - Check - Act)

La metodología consta de cuatro pasos esenciales que se deben llevar a cabo de forma sistemática para lograr la **mejora continua**. Una vez finalizados estos cuatro pasos esenciales, se debe volver al inicio y volver a repetirlo. Se entiende por mejora continua, el constante mejoramiento de la calidad:

- Disminución de fallos
- Aumento de eficacia y eficiencia
- Solución de problemas
- Previsión y eliminación de riesgos potenciales



Las cuatro etapas son:

- **Planificar - Plan:**
  - Se buscan las actividades susceptibles de mejora y se establecen los objetivos a alcanzar. Para buscar posibles mejoras se pueden realizar grupos de trabajo, escuchar opiniones de los trabajadores, buscar nuevas tecnologías mejores a las que se están usando ahora.

- **Hacer - Do:**
  - Se realizan los cambios para implantar la mejora propuesta. Generalmente conviene hacer una prueba piloto para probar el funcionamiento antes de realizar los cambios a gran escala.

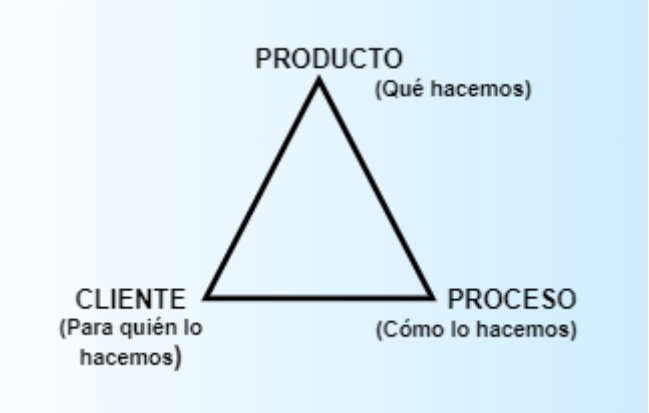
- **Controlar o Verificar (Check)**
  - Una vez realizada la mejora, se deja un periodo de prueba para verificar su correcto funcionamiento. Si la mejora no cumple con las expectativas iniciales, habrá que modificarla para ajustarla a los objetivos esperados

- **Actuar (Act)**
  - Una vez finalizado el periodo de prueba, se deben estudiar los resultados y compararlos con el funcionamiento de las actividades antes de haberse realizado la mejora. Si los resultados son satisfactorios, se implanta la mejora de forma definitiva, y sino, se ajustan los resultados o se desecha.

Calidad de Producto:

¿Qué se entiende por Calidad?

- “Es satisfacer los requerimientos del cliente”
- “Calidad total es un modo de vida corporativa, un modo de administrar una organización. Abarca toda la organización e involucra la puesta en práctica de actividades orientadas hacia el cliente”
- “Concordancia del software producido con los requisitos funcionales explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente”



Asegurar calidad en software (SQA)

Es el establecimiento de un marco de trabajo de procedimientos y estándares, tanto para Producto como para Proceso.

- **Estándares del Producto:** Estructura del documento de requerimientos, documento que define cómo utilizar un lenguaje de programación o estándares de documentos.
- **Estándares del Proceso:** Definen los procesos a seguir durante el desarrollo. Especificaciones de diseño, validaciones, descripción de documentación a generar.

Para garantizar la calidad del software, se necesita:

- Enfoque a la gestión de calidad
- Tecnología (métodos y herramientas) de ing. de Software y un equipo de SQA
- Revisiones técnicas formales durante todo el proceso
- Estrategia de pruebas
- Control de documentación del software y de los cambios realizados
- Procedimientos
- Mecanismos de medición y generación de informes.

Actividades:

- Establecimiento de un plan de calidad para el proyecto
- Participación del proceso del desarrollo de software a utilizar en un proyecto
- Revisión de las actividades de software para asegurar su ajuste al proceso planificado
- Auditoría de los productos de software
- Aseguramiento de la gestión y documentación de desvíos
- Registrar desvíos e informarlos.
- Gestionar los cambios
- Recopilar y analizar métricas del software

Garantía de calidad estadística:
Se utiliza para establecer calidad cuantitativamente; Se agrupan y clasifica información sobre los defectos. Se encuentran las causas de cada uno y se aíslan los focos vitales para corregirlos con prioridad.

Las causas más comunes de estos defectos:

- Especificaciones incompletas (IEE)
- Mala interpretación de la comunicación con el cliente (MCC)
- Desviación deliberada de la especificación (DDE)
- Incumplimiento de los estándares de programación (IEP)
- Error en la representación de datos (ERD)
- Interfaz de módulo inconsistente (IMI)
- Error en la lógica del diseño (ELD)
- Prueba incompleta o errónea (PIE)
- Documentación imprecisa/incompleta (DII)
- Error en la traducción del diseño al lenguaje de programación (TLP)
- Interfaz hombre máquina imprecisa/inconsistente (IHM)

Asegurar la calidad, tiene un **costo**:
Costo de calidad = Costos de conformidad + Costos de no conformidad

El costo de conformidad, incluye los costos para cumplir con los requerimientos:

- Reporte y tracking de defectos, pruebas del sistema e integración, test de aceptación de usuario
- Costos incurridos para mantener a un nivel mínimo todos los otros tipos de costos, como planificación y aseguramiento de calidad, capacitación, reuso, revisiones, administración de configuración.
- Proyectos de mejoras y certificaciones (CMMI, ISO)

El costo de **no** conformidad:

- Costo de fallas detectadas **antes** de entregar el producto al cliente. Como el diseño de acciones correctivas, re-revisiones, fixing de defectos, 2dos test del sistema/integración, retrabajo.
- Costo de fallas detectadas **luego** de entregar el producto. Retrabajo, Reingeniería, No satisfacción, Impacto en la percepción del servicio del cliente, Penalidades contractuales.

¿Por qué invertir en calidad?

- Exigencia de los clientes
- Competir a nivel internacional
- Utilizar las mejores prácticas
- Ser más productivos, flexibles
- Anticiparnos a problemas
- Cometer menos errores y aprender de ellos
- Para ser más eficientes
- Ser predecibles

Métricas:

- “No se puede predecir lo que no se puede medir”
- Las métricas son un buen medio para entender, monitorear, controlar, predecir y probar el desarrollo y mantenimiento del software
- La medición persigue 3 objetivos:
    - Entender que ocurre durante el proceso de desarrollo
    - Controlar que ocurre en nuestros proyectos
    - Mejorar nuestros procesos y productos

¿Para qué se mide el software?
La medida de software permite cuantificar cronogramas de trabajo, esfuerzo de desarrollo, tamaño de producto, estado de proyecto y desempeño de calidad.

Medir el software puede mostrarnos incumplimientos de tiempos previstos o excedentes en presupuestos.
La medición constante permite mejorar estimaciones futuras.
También, las métricas ayudan al aprendizaje más profundo de las organizaciones y procesos.

Atributo: Propiedad mensurable, física o abstracta que comparten todas las entidades de una categoría de entidad. Un atributo solo puede pertenecer a una categoría de entidad.

La medición, se realiza sobre atributos. Un atributo, puede tener definida 0, 1 o más métricas.

Medida: Valor asignado a un atributo de una entidad mediante una medición. Es el resultado de una medición.

Métrica: Medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. Una métrica de software relaciona de alguna manera las medidas individuales.

Una métrica está definida para uno o más atributos.

Indicador: Interpretación de una métrica. Los indicadores representan aquella parte de la realidad que ha sido más relevante por quienes lo diseñaron.

Los indicadores son utilizados en función del interés del observador. Uno óptimo, será aquel que prevea de la información necesaria para la toma de decisiones.

La **MEDIDA**, captura una característica individual.
La **MÉTRICA**, permite relacionar y comparar mediciones
El **INDICADOR**, es la interpretación de una métrica.

Las métricas pueden ser de Procesos, Proyecto o Producto:

Métricas de Proceso: La medición del proceso de desarrollo de software consiste en medir las actividades relacionadas con el software, y sus atributos típicos pueden ser:

- Esfuerzo
- Costo
- Defectos encontrados
- Métricas sobre los errores detectados antes de la entrega
- Defectos detectados e informados por los usuarios finales
- Productos de trabajo entregados
- Esfuerzo humano y tiempo consumido
- Ajuste con la planificación

Algunas métricas de Proceso:

- Soporte a clientes:
    - Tamaño del backlog de defectos
    - Tiempo de respuesta ante un incidente
- Esfuerzo:
    - Distribución del esfuerzo por fase
    - Distribución del esfuerzo por actividad
- Testing:
    - Efectividad en remover defectos entre fases
    - Costo de remover defectos
- QA:
    - Cantidad de revisiones de calidad de software
    - Cantidad de issues identificados en las revisiones de QA
- Reutilización
    - Cantidad de código reusado

Métricas de Proyecto: En los proyectos de software se utilizan las métricas y los indicadores que se deducen de ellas para adaptar flujo de trabajo. Tiene doble finalidad, minimizar tiempo de desarrollo a través de ajustes y el de valorar la calidad del producto.

Los indicadores de proyecto permiten:

- Evaluar el estado del proyecto en curso
- Realizar un seguimiento de los riesgos potenciales
- Detectar las áreas de problemas antes de que se conviertan en críticas

- Ajustar el flujo y las tareas de trabajo

Algunas métricas de Proyecto:
- Costo:
    - Costo de desarrollo
    - Costo de soporte
- Esfuerzo:
    - Horas trabajadas
    - Tiempo transcurrido
    - Distribución del esfuerzo por fase
    - Distribución del esfuerzo por actividad
- Seguimiento:
    - Cronograma vs estimado
    - Porcentaje de funcionalidad implementada por unidad de tiempo
    - Esfuerzo estimado vs esfuerzo real

Métricas de Producto: Se realizan por varias razones:
- Indicar calidad del producto
- Evaluar productividad de la gente que lo desarrolla
- Evaluar los beneficios en términos de productividad y de calidad
- Establecer una línea de base para la estimación
- Ayudar a justificar el uso de nuevas herramientas o de formación adicional

Algunas métricas de Producto:
- Complejidad:
    - Complejidad de diseño (acoplamiento)
    - Complejidad de código
    - Métodos por clase
    - Profundidad y ancho de jerarquía
- Calidad:
    - Densidad de defectos
    - Cantidad de problemas detectados
- Tamaño:
    - Líneas de código
    - Puntos de función
    - Bytes

Además de estas métricas mencionadas, se pueden distinguir **métricas directas e indirectas** en el proceso de ingeniería.

Las directas pueden ser Costo, Esfuerzo, Lineas de código producidas, Velocidad de ejecución, defectos observados en un cierto periodo de tiempo.
Las indirectas, pueden ser medidas como Funcionalidad, Calidad, Complejidad y Eficiencia.

Plan de Calidad:
El plan de Calidad es un documento a través del que se detalla cómo debe ser el proceso que garantice la calidad de los proyectos, productos o procesos. Este plan debe dar respuesta a cuestiones como qué acciones se llevarán a cabo, qué recurso serán necesarios o quienes serán los encargados de aplicar el plan.

Se creó una norma ISO 10005:2005, en donde se establecen las directrices para diseñar, revisar y aplicar un plan de calidad.

El plan de calidad, consta de 6 fases:

- **Identificación de la necesidad de un plan de calidad en la organización** (determinar si se necesita y el por qué). Es útil para aquellas empresas que necesiten demostrar a terceros cómo desarrollan la gestión de la calidad.
- **Identificación de las entradas para el plan de calidad.** Definir requisitos que serán necesarios para elaborarlo, relacionados a legislación, requerimientos de clientes, proveedores o inversores, recursos disponibles.
- **Definir el alcance del plan de calidad.** Definir si está enfocado a un determinado proyecto o proceso, y descomponerlo en acciones para analizar/describir las características.
- **Preparación del plan de calidad.** Necesita de una persona encargada del proyecto, un responsable que se encargue de la coordinación y elaboración del mismo, y de un equipo de trabajo que colabore con él. Recopilar toda la documentación necesaria para describir el plan
- **Contenido del plan de calidad**. Se plasma sobre papel el plan de calidad que se va a desarrollar. En este documento se van a reflejar datos necesarios para el desarrollo posterior del plan. Alcance, elementos de entrada, responsabilidad, recursos necesarios, requisitos necesarios, comunicación interna/externa, métodos de control
- **Revisión, aceptación e implementación del plan de calidad**. Tras la revisión y aceptación, el plan podrá comenzar a implementarse en la organización.

2do parcial

Temas:
19/04: Gestión de la configuración del software OK
26/04: Parcial
03/05: RTF OK
10/05: Modelos de calidad
17/05: - / Feriado
24/05: - / Feriado
31/05: - / Clase especial (seguridad)
07/06: Gestión del riesgo OK
14/06: - / Repaso (?)
21/06: - / Feriado

Gestión del riesgo

Gestión de riesgos: Es el proceso de identificación, análisis y determinación de riesgos asociados a eventos para poder tomar acciones.

Objetivo: reducir los riesgos hasta un nivel tolerable para la organización

Identificación, análisis y gestión
Podemos entonces entender un riesgo como un problema potencial: puede ocurrir, o puede no ocurrir.
Pero, sin importar el resultado, realmente es una buena idea identificarlo, valorar su probabilidad de ocurrencia, estimar su impacto y establecer un plan de contingencia para el caso de que el riesgo se convierta en un problema.

Ejemplos de riesgos comunes en un proyecto software
- Cambio de requisitos
- Meticulosidad en requerimiento o desarrolladores
- Escatimar (minimizar) en la calidad (deuda técnica)
- Planificaciones demasiado optimistas
- Diseño inadecuado
- Síndrome de la panacea
- Trabajo de baja calidad (mediocre)
- Error en la contratación
- Diferencias entre el personal de desarrollo y los clientes

Conceptos relacionados:
Riesgo: pérdida o daño potencial
Activo: Todo lo que tenga valor para el negocio
Amenaza: Presencia de un evento que puede impactar en forma negativa
Vulnerabilidad: Ausencia o debilidad de un control

Salvaguarda: Contramedida para reducir el riesgo
Exposición: Estadio en que se está expuesto a pérdidas debido a una amenaza

¿De qué nos protegemos?
- Ataques: Externos, Internos
- Accidentes: Naturales, Humanos

Etapas en la gestión de riesgos:
1) Identificación y evaluación de los activos (tangibles e intangibles)
2) Análisis de amenazas (identificación de amenazas que pueden impactar en la organización)
3) Análisis de vulnerabilidades (identificación de las vulnerabilidades que pueden permitir la concreción de amenazas)
4) Evaluación del riesgo (evaluación de toda la información recopilada)

Acciones sobre el riesgo:
1) Reducción:
    a) Basado en contramedidas
    b) No se puede hacer cero
2) Transferencia:
    a) Traslado el riesgo
    b) Póliza de seguros
3) Aceptación:
    a) Asumir probabilidad de pérdida
    b) Debe ser formal
4) Eliminación:
    a) Eliminar elemento de riesgo
    b) Intenta evitar llegar a esto

Fórmulas conceptuales:

Riesgo TOTAL = Amenaza * Vulnerabilidades * Valor del recurso
(El que se asume en caso de no implementar contramedidas)

Riesgo RESIDUAL = Riesgo total - Control Gap
(El remanente una vez implementadas las contramedidas)

Control GAP = Riesgo total - Riesgo residual
(Riesgo reducido por implementación de las contramedidas)

Fases: Análisis y evaluación de riesgos

1) Análisis de riesgos:
Conceptos:
- Estudio de ambientes en relación a atributos que representan riesgos
- Como herramienta para la toma de decisiones

- Debo identificar las vulnerabilidades y amenazas del sistema
- Quiero saber a qué están expuestos
- Debo conocer previamente cuáles son los activos, su valor y sus relaciones

Consideraciones:
- Contemplar todos los activos críticos
- Involucrar a los dueños
- Metodológicamente
- En forma periódica o ante un cambio significativo

2) Evaluación de riesgos

Conceptos:
- Asignación de valores (activos, frecuencias, consecuencias y otros)
- El reporte final incluye una medida del riesgo y recomendación para reducirlo
- Determinación del impacto de potenciales amenazas al negocio

Resultado:
- Riesgos identificados
- Justificación económica de controles (costo/beneficio)

Métodos:
- Cuantitativo
- Cualitativo

Método cuantitativo para evaluación de riesgos:

Asigna valores objetivos cuantificados
Requiere de un plan detallado
Se aplica a: activos a proteger, amenazas y su riesgo, potenciales pérdidas, frecuencia de ocurrencia, controles, predicción de pérdidas
Proceso anterior: preliminary security examination (PSE): ayuda a recopilar elementos para la evaluación

Método cuantitativo para evaluación de riesgos:

Ventajas: Evaluación y resultados basados en métricas, cuantificación de los parámetros de la CIA en términos monetarios, provee análisis costo/beneficio, permite realizar seguimiento y evaluación de la gestión del riesgo
Desventajas: cálculos complejos, necesidad de recopilar mucha información, no está basado en estándar de conocimientos.

Análisis Cualitativo – Ejemplo de criterios

Probabilidad de ocurrencia	Impacto		
	Alto	Medio	Bajo
Alto	Alto	Alto	Medio
Medio	Alto	Medio	Bajo
Bajo	Medio	Bajo	Bajo

Probabilidad de ocurrencia	Impacto				
	[0 ; 1)	[1 ; 2)	[2 ; 3)	[3 ; 4)	[4 ; 5]
[0 ; 1)	Bajo	Bajo	Bajo	Medio	Medio
[1 ; 2)	Bajo	Bajo	Medio	Medio	Medio
[2 ; 3)	Bajo	Medio	Medio	Medio	Alto
[3 ; 4)	Medio	Medio	Medio	Alto	Alto
[4 ; 5]	Medio	Medio	Alto	Alto	Alto

Análisis Cualitativo – Amenaza X

Personal consultado	Probabilidad de ocurrencia	Impacto	Efectividad de la contramedida		
			Guardia	Control biométrico	CCTV
Gerente de TI	3	4	2	4	3
Jefe de datacenter	2	5	1	5	2
Jefe de Seguridad	2	4	1	4	3
Responsable de SegInf	4	5	1	4	2
Promedio	2.75	4.5	1.25	4.25	2.5

RIESGO ALTO

Cuantitativo vs Cualitativo

Característica	**Cualitativo**	**Cuantitativo**
Cálculos	Simples	Complejos
Aplicable	Siempre	No siempre
Análisis costo/beneficio	Subjetivo	Concreto
Objetividad	Baja	Alta
Comprensible por la dirección	Menos	Más
Herramientas automatizadas	No aplicable	Aplicable
Utiliza métricas claras	No	Sif

Revisiones técnicas formales (RTF)

Definición: Las RTF son **revisiones** metódicas y estructuradas. Son un medio efectivo para mejorar la calidad del software.

Las RTF también son conocidas como **Revisiones por Pares** o **Peer Reviews**.

Objetivo:

- Identificar y eliminar defectos en los productos desarrollados, en forma temprana y eficiente.
- Encontrar sugerencias de mejora.
- Descubrir errores en la función, la lógica o la implementación de cualquier representación del software/*artefacto interviniente en el proceso de desarrollo de software*.
- Verificar que el software/los artefactos bajo revisión satisfacen los requisitos/requerimientos.
- Garantizar que el software/los artefactos se han desarrollado de acuerdo con ciertos estándares predefinidos
- Conseguir un software desarrollado de forma uniforme/obtener **uniformidad** en los artefactos elaborados
- Hacer que los proyectos sean manejables

El **principal objetivo** de una RTF es: **Detectar los errores antes de que pasen a otra actividad de la ingeniería del software o antes que se entregue al usuario final**

Aplican a diferentes tipos de productos

- Código
- Documentación Funcional
- Documentación Técnica

¿Por qué es importante incorporar RTFs en el proceso de desarrollo?

- Las revisiones purifican actividades de ingeniería de software
- La RTF es un medio efectivo para **descubrir errores y mejorar la calidad**
- El Software Engineering Institute (SEI) señala a las RTF como una de las prácticas industriales esenciales para la gestión del proceso software

Beneficios de las RTF

Beneficios directos

- Detección temprana y eficiente de defectos
- Disminución de los tiempos de testing
- Mejora en la calidad de los productos
- Aumento de la productividad

Beneficios Indirectos:

- Entrenamiento para los participantes

- Mejora en la comunicación y el trabajo en equipo
- Mejora en la calidad de estándares y métodos
- Aumento de la visibilidad sobre los procesos
- Incremento del soporte y la continuidad en el trabajo diario

¿Qué productos se pueden revisar en una RTF?
Durante los procesos del desarrollo de Software, se producen y utilizan distintos *artefactos* que pueden ser revisados en una RTF
Ejemplos típicos:

- Análisis: Definición de requerimientos, consistencia y corrección de la especificación
- Diseño: Revisión de la arquitectura o del diseño detallado
- Codificación: Traducción correcta del diseño al código, Errores de tipeo, Cumplimiento de estándares de codificación
- Prueba: Validación de la estrategia de pruebas, Revisión de los casos de prueba

Observaciones importantes:
Es necesario evaluar la relación costo/beneficio de realizar RTF: “Los resultados de las inspecciones no deben, bajo ninguna circunstancia, ser usados para la evaluación de los programadores”, “Los resultados de las inspecciones son para el uso y el beneficio de los programadores” (Fagan).
La revisión debe estar enfocada en los defectos del producto, no en el autor. Se debe ser muy cuidadoso al iniciar un proceso de RTF para que esto quede claro a todos los que participan

Tipos de RTF
Inspecciones:

- El producto es revisado por un grupo de pares que reportan los incidentes identificados
- Más formal y estructurada. Requiere mayor preparación
- Implica “leer” el componente durante una reunión de revisión
- Usada para documentación

Walkthrough:

- Presentación de un producto por su autor a un grupo de revisores
- Más informal
- Usada para revisión de modelos gráficos o código

Proceso del RTF
El proceso está compuesto por cuatro etapas:

Planificación
- Identificación de productos
- Identificación de tipos de revisiones a realizar

Preparación
- Selección de revisores
- Distribución del material
- Capacitación necesaria sobre el proceso

Ejecución y seguimiento
- Reunión de revisión

- Seguimientos de los incidentes identificados

Control

- Controlar la correcta aplicación del proceso
- Asegurar la recolección de mediciones

1) RTF - Planificación

- Se hace al inicio del proyecto o etapa
- Es responsabilidad conjunta del líder del proyecto y el responsable de SQA
- Se define qué productos serán revisados y los métodos que se utilizarán para las distintas revisiones
  . Se deben revisar los productos con mayor criticidad o complejidad
  . El tipo de revisión a realizar depende principalmente del tipo de producto

- Roles intervinientes en una revisión
  . *Inspección Formal:* Autor, Moderador, Revisores
  . *Walkthrough:* Autor / Presentador, Moderador, Revisores

2) RTF - Preparación

- Confirmación de la revisión:
  . Confirmar con el autor que el producto está listo para revisión
    . La revisión no puede hacerse si el autor no considera que el producto está listo.
    . "Listo" no quiere decir que no tiene errores
  . Confirmar participantes de la revisión
  . Confirmar fecha a todos los que participarán de la revisión

- Preparación de la revisión:
  . Autor: Prepara y distribuye el material para la revisión
  . Responsable QA: Colabora en la preparación de la revisión
  . Revisores: Revisan materiales antes de la reunión de revisión (si aplica)

3) RTF - Ejecución y seguimiento

Inspección formal	**Walkthrough**
Más estructurado y formal	Más informal
Objetivos: - Detectar defectos y desviaciones de estándares o especificaciones. - No se discuten soluciones concretas.	**Objetivos**: - Encontrar defectos, omisiones y contradicciones - Proponer mejoras - Considerar alternativas para los problemas detectados - Capacitación de los participantes e intercambio de ideas
- Varios miembros (3 a 5), incluido un moderador	- Llevada a cabo por el autor del producto y uno o más revisores

- Preparación formal obligatoria por parte de los revisores	- No exige una preparación previa extensiva por los revisores
- Las corrección de defectos o mejoras detectadas es obligatoria para el autor - Es necesario hacer un seguimiento formal de la ejecución y los resultados de las mismas	- Consiste en una exposición/ descripción del producto por el autor a los revisores, que dan feedback al respecto

Clasificación de incidentes según criticidad

La **clasificación** se refiere al **impacto**, no a la dificultad para corregir el incidente identificado.

Ejemplo de clasificación

- **Crítico**
  . *Código:* cualquier aspecto del sistema **no consistente** con las especificaciones del sistema o que **provocaría una falla** en el mismo
  . *Documentación:* error que hace que la solución sea **inconsistente** con los objetivos o requerimientos del cliente
- **Moderado**: Defecto o error que **si no se arregla no generará retrasos** en el proyecto **ni errores o problemas de funcionamiento** del sistema
- **Cosmético**: Errores de **tipeo, formato, gramática,** etc., que **no afectan el contenido** del documento

Documentación

Los resultados de las RTF deben quedar documentados con el objetivo de:

- Permitir realizar un **seguimiento** sobre los incidentes identificados
- Proveer de **input** a la **etapa de control**

RTF - Directrices

. Revisar al producto, no al autor
Una RTF involucra personas y egos
Si se lleva a cabo de manera inadecuada, la RTF puede tomar un efecto negativo
Los errores se deben señalar con gentileza, el tono de la junta debe ser relajado y contractivo.
La finalidad no debe ser avergonzar o menospreciar
. Establecer una agenda y respetarla
Mantener el rumbo y seguir el programa
No vacilar en llamar la atención de la gente cuando se empiece a divagar
. Limitar el debate
. Enunciar áreas de problemas, pero no se debe intentar resolver los que se hayan señalado. No es una sesión para resolver problemas.

4) RTF - Control

El proceso de RTF puede dar lugar a distintas **métricas**
Ejemplos: cantidad de revisiones por mes, cantidad de incidentes identificados por revisión, defectos identificados con mayor frecuencia

RTF - Conclusiones

Incorporar RTF en el proceso de desarrollo es **útil para**:

- Mejorar la calidad de los productos que se desarrollan
- Identificar problemas en forma temprana
- Capacitar recursos y mejorar el trabajo en equipo
- Es importante seguir el proceso y respetar los roles para lograr buenos resultados

Modelos de calidad

Tres tipos de modelos importantes:

- **Calidad del producto:** propiedades del producto según usuario y según desarrollador (valor técnico)
- **Calidad del proceso:** actividades que influyen en calidad del producto (valor técnico)
- **Calidad de uso:** relación del producto con el ambiente donde se emplea (valor comercial)

Calidad basada en el proceso
Se busca analizar las actividades del proceso que más influyen en la calidad del producto.
Se modela el proceso para analizarlo mejor
Se pueden hacer preguntas como: ¿dónde y cuándo se puede hallar un tipo de defecto? ¿cómo hallar los defectos antes? ¿Existen actividades alternativas que proporcionen mayor calidad?

Modelos basados en proceso:

- CMM y CMMI
- ISO 15504 SPICE
- ISO 9000

ISO 9000:
La ISO 9000 es una norma de Gestión de Calidad que contiene las directrices que permiten aumentar la eficiencia de un negocio y la satisfacción del cliente. El objetivo de esta norma, es implementar un sistema de gestión de calidad en la organización, aumentar su productividad y reducir los costos que no sean necesarios, para así, garantizar la calidad de procesos y productos.

La norma es aplicable a empresas u organizaciones sin importar su sector. El enfoque está orientado a procesos, permitiendo que sirva tanto para productos como para servicios.

Apunta a ver cómo interactúan los procesos, si se puede integrar con otros, y cuales son los aspectos más importantes de los productos/servicios.

La norma consiste en realizar varias auditorías para verificar la eficacia del sistema de gestión de calidad. (1ra, 2da y 3ra parte).

- La primera parte es una auditoría interna
- La segunda parte es permitir al consumidor final evaluar el rendimiento de la organización
- La tercera parte, es una alternativa a la segunda, y consta de que un organismo de certificación audite la organización y lo evalúe.

Si la organización cumple con los requisitos de la norma, se convierte en certificada y se le entrega un sello de calidad reconocido en todo el mundo.

Es importante la norma, ya que permite destacarse frente a los ojos del cliente, además de mejorar la eficiencia y aumentar ingresos.

Principios de la ISO 9000:

- Enfoque al cliente
- Buena dirección
- Participación de personas
- Enfoque a la gestión de calidad
- Enfoque al sistema de gestión
- Mejora continua
- Enfoque para la toma de decisiones
- Relaciones con proveedores

Gestión de configuración

Cuando se construye software los **cambios** son **inevitables**. Los **cambios aumentan el nivel de confusión** en el equipo de desarrollo, debido a diferentes razones: no se han analizado los cambios antes de realizarlos, no se han registrado antes de implementarlos, no se les ha comunicado a aquellas personas que necesitan saberlo, no se han controlado de manera que mejoren la calidad y reduzcan los errores.
Tener un control de versiones va a afectar directamente en la calidad del producto.

¿Qué es la gestión de configuración?

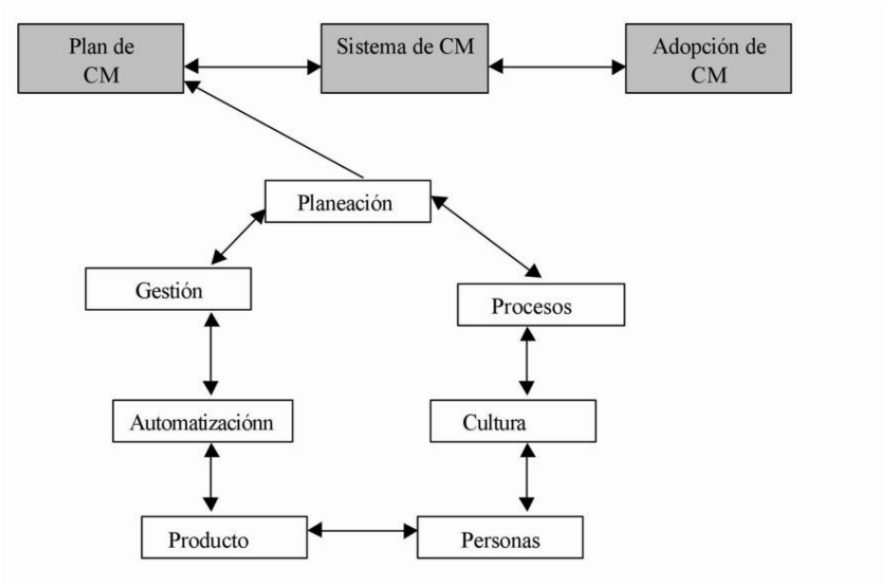
- La GCS es un **conjunto de actividades** diseñadas para **identificar y definir los elementos en el sistema que probablemente cambien**, controlando el cambio de estos elementos a lo largo de su ciclo de vida, estableciendo relaciones entre ellos, definiendo mecanismos para **gestionar distintas versiones** de estos elementos, y **auditando** e **informando** de los cambios realizados.
- Es una **actividad de protección** que **gestiona el cambio** a lo largo del ciclo de vida del software
- Es una actividad de protección que puede considerarse **dentro de la SQA**.

¿Cuál es el propósito de la gestión de configuración? Establecer y mantener la **integridad** de los productos de software a través del ciclo de vida del proceso de software.

¿Por qué es necesaria la gestión de configuración? Los requerimientos del sistema siempre cambian durante su desarrollo y su uso, y se tienen que incorporar estos requerimientos en nuevas versiones del sistema

¿Por qué es importante la gestión de configuración? Los cambios incontrolados aplicados a un proyecto de software lo llevan al fracaso.

Elementos de Configuración de Software (ECS)



Todas las cosas que son versionadas son elementos de la configuración.
Los **elementos que componen toda la información generada como parte del Software** se denominan colectivamente elementos de la configuración del software.
Son elementos de la configuración: Datos - Programas - Documentos
A medida que progresa el software el número de **Elementos de Configuración Software (ECS)** crece rápidamente. Los ECS producen otros ECSs para crear una **jerarquía de información.**

Las fuentes fundamentales del cambio son:

- Fallos.
- Nuevos negocios o condiciones comerciales que dictan cambios en los requisitos del producto.
- Nuevas necesidades del cliente que demandan la modificación de los datos, funciones o servicios.
- Reorganización y/o reducción del volumen comercial que provoca cambios en el proyecto.
- Restricciones presupuestarias o de planificación que provocan una redefinición del producto.

Actividades de GCS

La Gestión de la Configuración Software (GCS) es una **actividad de protección** que gestiona el cambio a lo largo del **ciclo de vida del software.** El cambio se puede producir en cualquier momento y por cualquier razón. Por tanto, las **actividades de GCS** son:

- **Identificar el cambio (planificación)**: Cómo identificar y gestionar las versiones de un programa para permitir modificaciones.
- **Controlar el cambio (monitorización y control)**: Cómo controlar los cambios antes y después de distribuir el software al cliente. Esto incluye saber quien es el responsable de aprobar y de asignar prioridades a los cambios.
- **Garantizar la correcta implementación del cambio (realización de auditorías)**: Cómo podemos garantizar que los cambios se han llevado a cabo adecuadamente.
- **Informar del cambio a todos aquellos que lo necesiten (elaboración de informes/reportar):** Qué mecanismos se usan para avisar a otros de los cambios realizados.

1) Identificación de ECS/Planificación

Cada objeto tiene un conjunto de características que lo identifican:

- Nombre.
- Descripción (Versión, proyecto, tipo de ECS).

Generalmente cada empresa tiene una propia convención de nombres para cada tipo de ECS

2) Control de versiones/Monitorización y control

El control de versiones permite gestionar la versión del sistema. A su vez, la versión del sistema viene identificada por las versiones de los ECSs

• Ejemplo: versión 1.0 = { SRS3.2, diseño2.0, código4.1, casos de prueba2.4.... }*

Cada versión puede tener distintas variantes

Las variantes suelen darse cuando tenemos una misma versión para diferentes plataformas

• Ejemplo:

versión 1.0 Windows = { SRS3.2, diseño2.0, código4.1, casos de prueba2.4.... }
versión 1.0 Mac OS X = { SRS3.2, diseño2.0, código4.5, casos de prueba3.0.... }

3) Auditoría

La identificación, control de versiones y control de cambios promueven un seguimiento hasta la generación del pedido de cambio.

Podemos asegurar que el cambio se ha efectuado correctamente gracias a:

- Las RTFs. (Revisión técnica formal), que implican una corrección técnica del cambio
- Las auditorías de configuración software, que tiene un carácter complementario y se preocupa de si:
  - Se ha hecho el cambio especificado en el pedido de cambio.
  - Se han incorporado modificaciones adicionales.
  - Se ha llevado a cabo una RTF.
  - Se han seguido adecuadamente los estándares de ISO.

- Se han reflejado los cambios en el ECS, incluidos fecha de cambio y autor.
- Se han seguido procedimientos de GCS para gestionar el cambio.
- Se han actualizado convenientemente todos los ECSs relacionados.

4) Informe de estado/Elaboración de informes/Reportar

Los Informes de Estado de la Configuración (IECs) informan sobre: qué pasó, quién lo hizo, cuándo pasó, qué más se vio afectado.
Se debe generar un IEC:
- Cada vez que se asigna una nueva identificación a un ECS.
- Cada vez que la autoridad de cambio expide un pedido de cambio.
- Cada vez que se lleva a cabo una auditoría de configuración.
- Regularmente, para mantener informados a los desarrolladores de los cambios importantes.

Línea de base (Baseline)

Una línea base es un concepto de GCS que nos ayuda a controlar los cambios sin perjuicio de aquellos que sean necesarios.

Definición:

- **Especificación o producto que se ha revisado formalmente y sobre los que se ha llegado a un acuerdo**, y que de ahí en adelante **sirve como base para un desarrollo posterior** y que **puede cambiarse solamente a través de procedimientos formales** de control de cambios.
- Un **punto de referencia** en el desarrollo del software que queda **marcado por el envío de uno o más elementos de configuración del software y la aprobación del ECS obtenido mediante una RTF (Revisión técnica formal) o un comité de cambio.**

Antes de que un ECS se convierta en línea base el cambio puede llevarse a cabo de manera rápida e informal.Sin embargo, una vez que se ha establecido una línea base solo se pueden efectuar los cambios si se aplica un procedimiento formal para evaluarlos y verificarlos.

ECSs que forman un **conjunto de líneas base** (El IEEE Std. 1028-1997 incluye una lista de ECSs denominados productos software): Plan del proyecto del software, Especificación, Diseño, Código, Casos de prueba, Manuales de operación e instalación, Manual de usuario, Documentos de mantenimiento, Estándares y procedimientos de IS, Declaración de Objetivos, Indicadores.
Además de estos ECSs pueden inmovilizarse las herramientas de software (e.g., editores, compiladores, herramientas CASE, etc.)

Ventajas y desventajas

Ventajas:
- Resolución más rápida de los problemas.
- Gestión de cambios más eficiente.
- Reducción de costos
- Más seguridad
- Control de licencias
- Mayores niveles de seguridad

- Mayor rapidez en la restauración del servicio

Desventajas:

- Una incorrecta planificación
- Herramientas inadecuadas
- Falta de coordinación con la gestión de cambios y versiones
- Falta de organización
- Falta de compromiso

GCS en Metodologías ágiles

Identificación y control son **actividades críticas** para un concepto clave en Agile: **rápida adaptación a los cambios.**

La gestión de configuración es importante en desarrollos ágiles porque necesita soportar: cambios frecuentes (hasta diarios), **múltiples baselines,** y múltiples espacios de trabajo.

Principios de CM en Metodologías ágiles

- **Identificar, controlar, auditar y reportar** son principios independientes de la metodología
- La implementación de GCS se debe ajustar para asegurar que los principios ágiles permanecen intactos sin dañar los principios de la gestión de la configuración.
- Mientras que los cuatro principios se respeten, CM se puede aplicar de diferentes maneras y en diferentes metodologías.

Prácticas de CM en Metodologías ágiles

- Establecer ambiente de CM
- Integración continua
- Testing unitario
- Automatización del despliegue
- Automatización de los builds
- Gestionar línea base
- Control de versiones

Implementación del **Plan de Gestión de Configuración** en Metodologías ágiles

El **plan de gestión de configuración** es el **documento** que **guía la estrategia** de la gestión de configuración de un proyecto.

Toda la estrategia de GCS se define en e**l plan de GCS** que **contiene**: definición de roles y responsabilidades, identificación de los ítems de configuración, repositorios, permisos, herramientas, ambientes, convención de nombres, estrategias, control de cambios, definición de los baselines, back ups, recuperación, etc.

Las prácticas de GCS se aplican durante toda la vida del proyecto.

Conclusión:

GCS permite mantener la trazabilidad de todos los componentes y saber qué se entrega en un release dado (historia de un ítem de trabajo, cuándo fue entregado y qué artefactos fueron modificados por el mismo).

Las prácticas que se utilizan para implementar los principios de GCS son las mismas que se aplican a la implementación de GCS en proyectos ágiles.

La **automatización es base** para la implementación de un GCS en un proyecto ágil.