FUNDAMENTOS DE INGENIERÍA DE SOFTWARE

Elaboró:

M.T.I. Edwin León Bojórquez

M.C. Carlos Benito Mojica Ruiz

M.C. Luis Ramiro Basto Díaz

Gestión del proceso software

Gestión de la Ingeniería de Software

- La gestión de la Ingeniería de software puede definirse como la aplicación de actividades de gestión: planeación, coordinación, medición, monitoreo, control y reporte; para asegurar que el desarrollo y mantenimiento de software sea sistemático, disciplinado y cuantificable.

- El área de la gestión de la ingeniería de software consiste del proceso de gestión del proyecto de software y de la medición en la ingeniería de software. La gestión del proceso se refiere a las actividades que se realizan para asegurar que los procesos de ingeniería de software se llevan a cabo de manera consistente con las políticas objetivos y estándares de la organización. La medición se refiere a la asignación de valores y niveles a aspectos de ingeniería de software y los modelos derivados.

Gestión de la Ingeniería de Software

- Aspectos específicos que dificultan la gestión en la ingeniería de software:
  - La falta de conocimiento por parte de los clientes sobre la complejidad inherente en la ingeniería de software, particularmente cuando hay cambios en los requerimientos.
  - Es casi inevitable que los procesos de ingeniería de software acepten nuevos o cambios en los requerimientos.
  - Como resultado, los procesos de ingeniería de software son procesos iterativos, más que una secuencia de tareas cerradas.
  - La ingeniería de software necesariamente incorpora aspectos de creatividad y disciplina.
  - El grado de novedad y complejidad de software es extremadamente alto.
  - Existe un rápido cambio en la tecnología.

Proceso de ingeniería de software

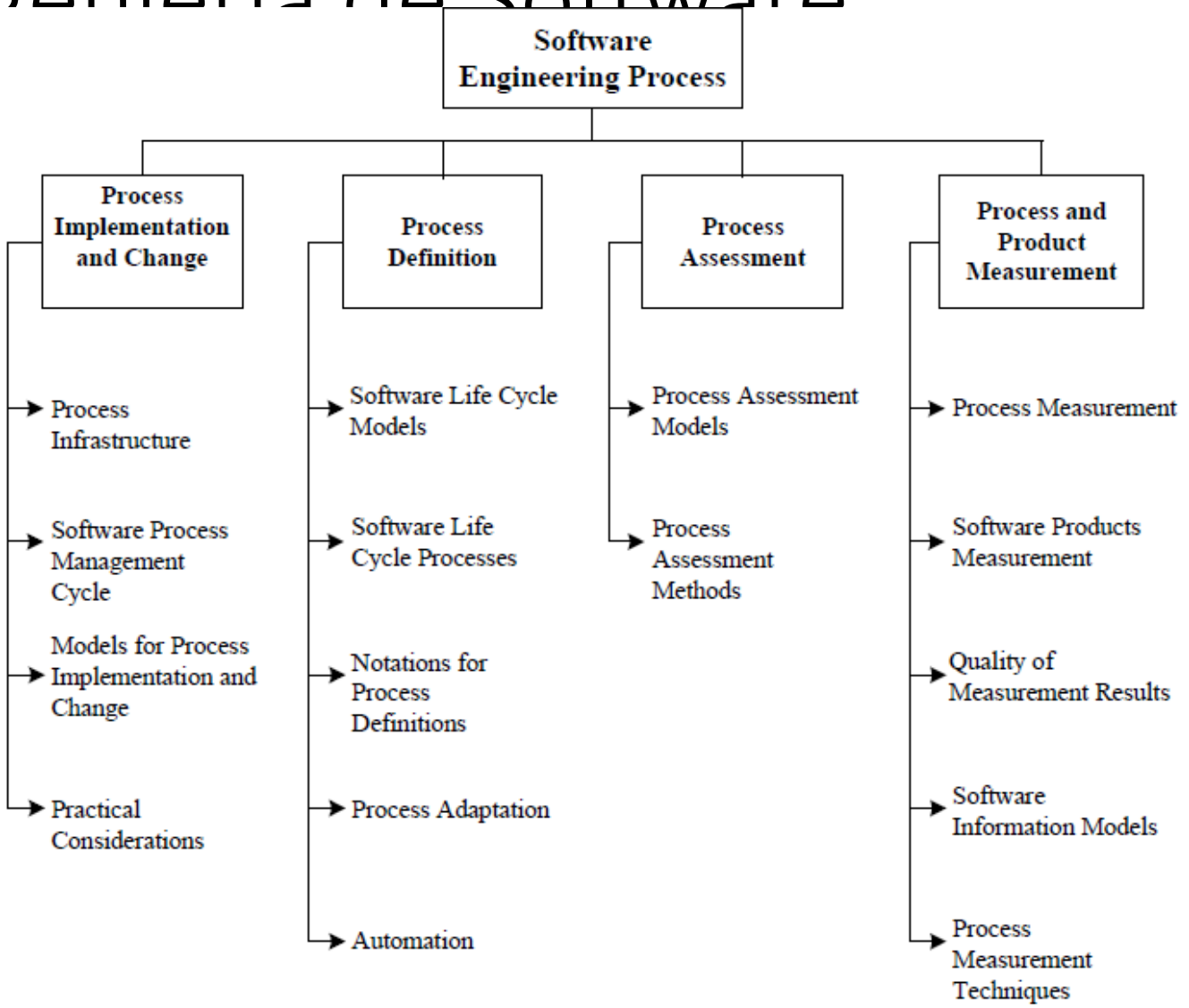
- El proceso de ingeniería de software puede ser examinado desde dos niveles:

  - El primer nivel abarca las actividades técnicas y de gestión involucradas dentro del proceso de ciclo de vida, las cuales son realizadas durante la adquisición, desarrollo, mantenimiento y retiro del software.

  - El segundo nivel, se interesa por la definición, evaluación, medición, gestión, cambio y mejora de los mismos procesos del ciclo de vida de desarrollo.

Proceso de Ingeniería de Software

- Está área de conocimiento aplica a cualquier parte de la gestión del proceso del ciclo de vida de desarrollo de software donde se han realizado cambios procedimentales o tecnológicos para mejorar algún proceso o producto.



Implementación y cambios del proceso

- Esta área se enfoca en el cambio organizacional. Describe la infraestructura, actividades, modelos y consideraciones prácticas para la implementación y cambios en los procesos.

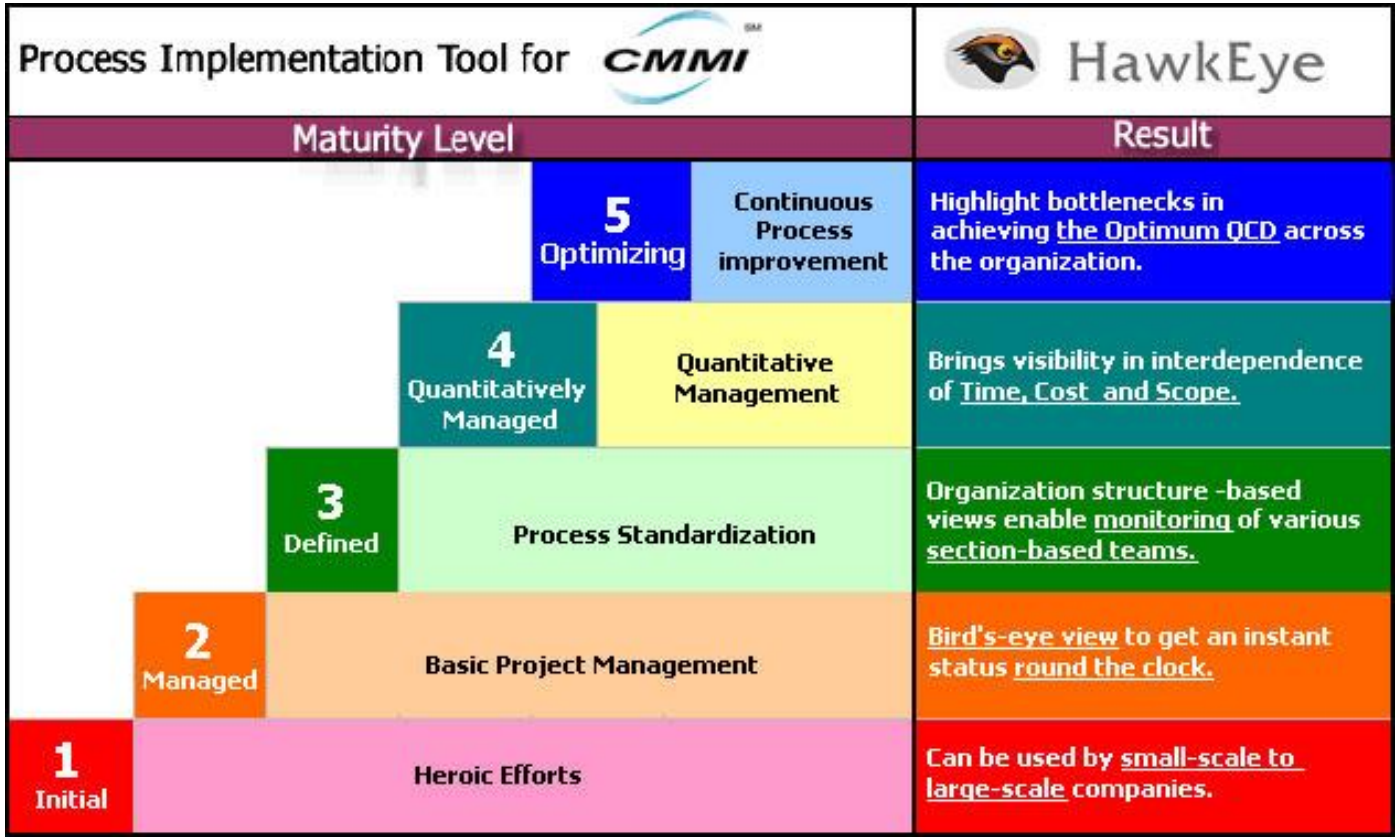
- Se refiere a la situación en la que los procesos son implementados por primera vez, y cuando los procesos actuales son cambiados. En ambos casos, las prácticas existentes tienen que ser modificadas.

Definición del proceso

- Una definición del proceso puede ser un procedimiento, una política o un estándar. Los procesos del ciclo de vida de software están definidos por varias razones, entre las que se encuentran incrementar la calidad del producto, facilitar la comunicación y el entendimiento entre personas, soportar los procesos de mejora y de gestión, proveer una guía automática del proceso, y proveer un soporte de ejecución automática.

Evaluación del proceso

- Se lleva a cabo mediante un modelo de evaluación y un método de evaluación. Algunos modelos disponibles son CMMI, SW-CMM, Bootstrap.



Medición del proceso y producto

- La medición puede realizarse como soporte al inicio de la implementación y cambios del proceso o para evaluar las consecuencias de la implementación y cambios en dicho proceso, o puede realizarse en el producto mismo.

- Mientras la aplicación de la medición en la ingeniería de software puede ser compleja, existen aspectos en la medición que son fundamentales para el desarrollo de procesos de medición y de análisis más avanzados.

5.2. gestión de la configuración

Gestión de la Configuración del SW

- Es un conjunto de actividades diseñadas para administrar el cambio mediante la identificación de los productos de trabajo que es probable que cambien, el establecimiento de relaciones entre ellos, la definición de mecanismos para administrar diferentes versiones de dichos productos de trabajo y el control de los cambios impuestos, así como la auditoría y reporte de los cambios realizados.

- Según Babich (1986) la Gestión de Configuración del Software (GCS) es el arte de identificar, organizar y controlar las modificaciones que sufre el software que construye un equipo de programación. *El objetivo es maximizar la productividad minimizando los errores*.

Gestión de la Configuración del SW

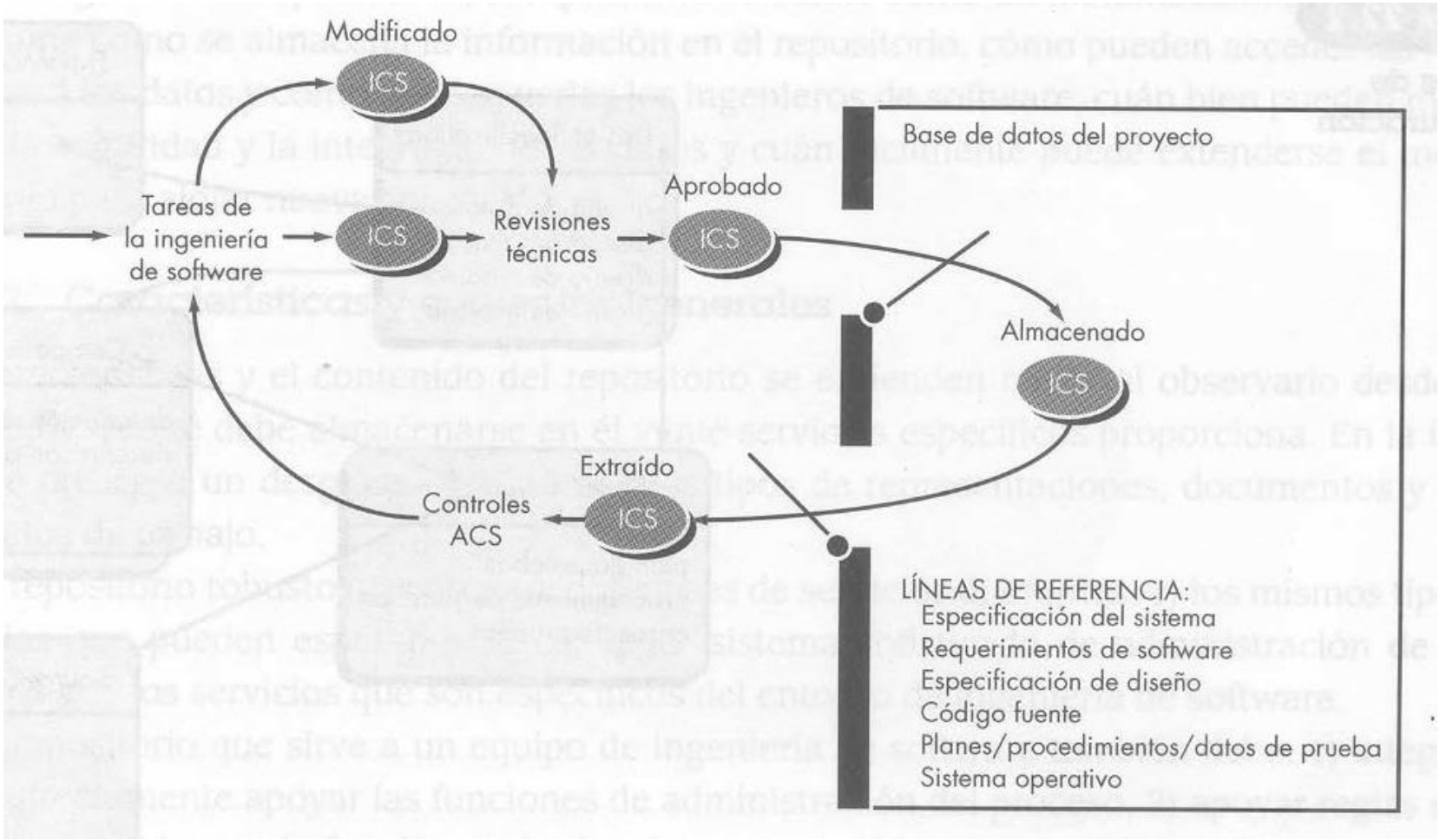
- La salida del proceso de software es información que puede dividirse en tres categorías amplias:
  - Programas de cómputo.
  - Documentos.
  - Datos.
- Los ítems que comprenden toda la información producida como parte del proceso de software se llaman colectivamente ***configuración del software***.

Gestión de la Configuración del SW

- Un cambio es un hecho de vida en el desarrollo de software. Una ***línea de referencia*** es un concepto de administración de la configuración del software que le ayuda a controlar el cambio sin impedir seriamente cambios justificados.

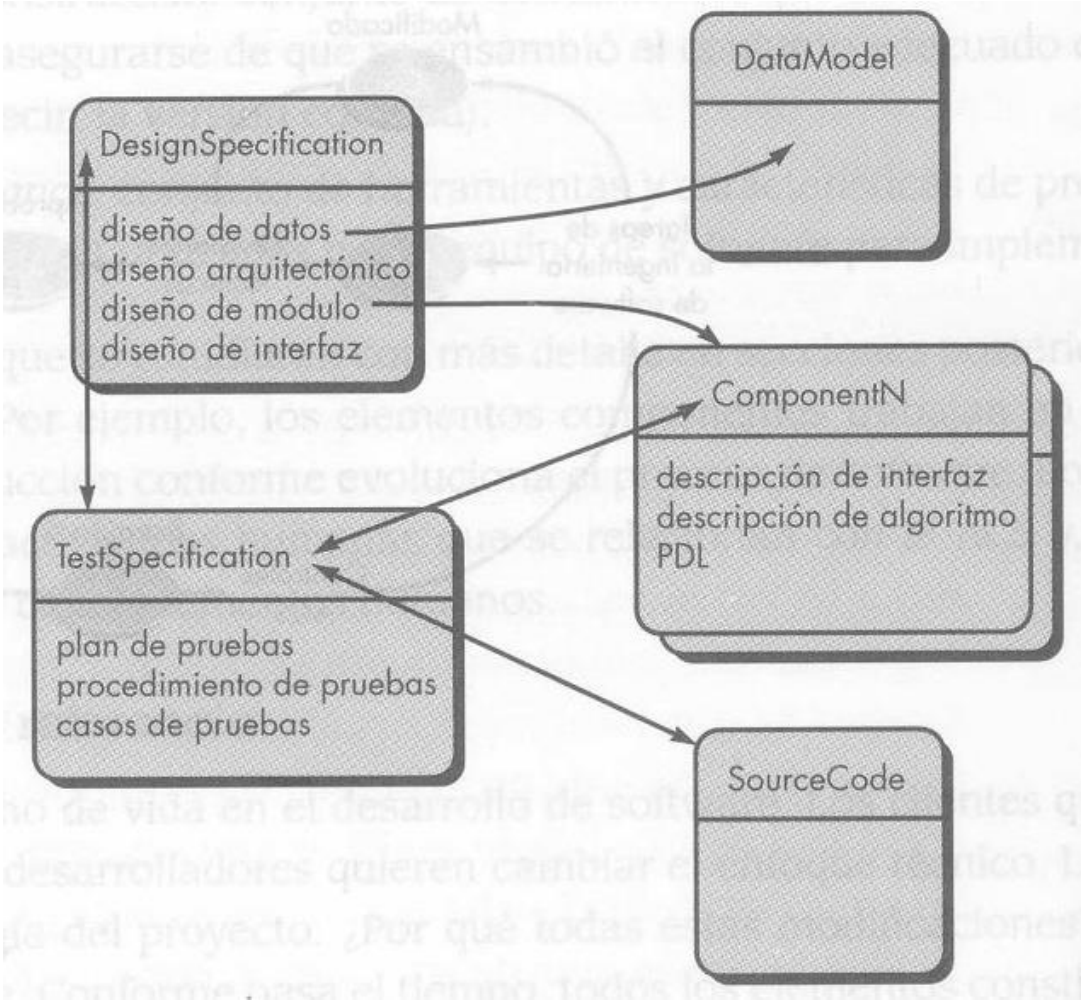
- Una línea de referencia es una especificación o producto que se revisó formalmente y con el estuvo de acuerdo, que a partir de entonces sirve de base para un mayor desarrollo y que puede cambiar sólo a través de procedimientos de control de cambio formal (IEEE Std. No. 610.12-1990).

Gestión de la Configuración del SW



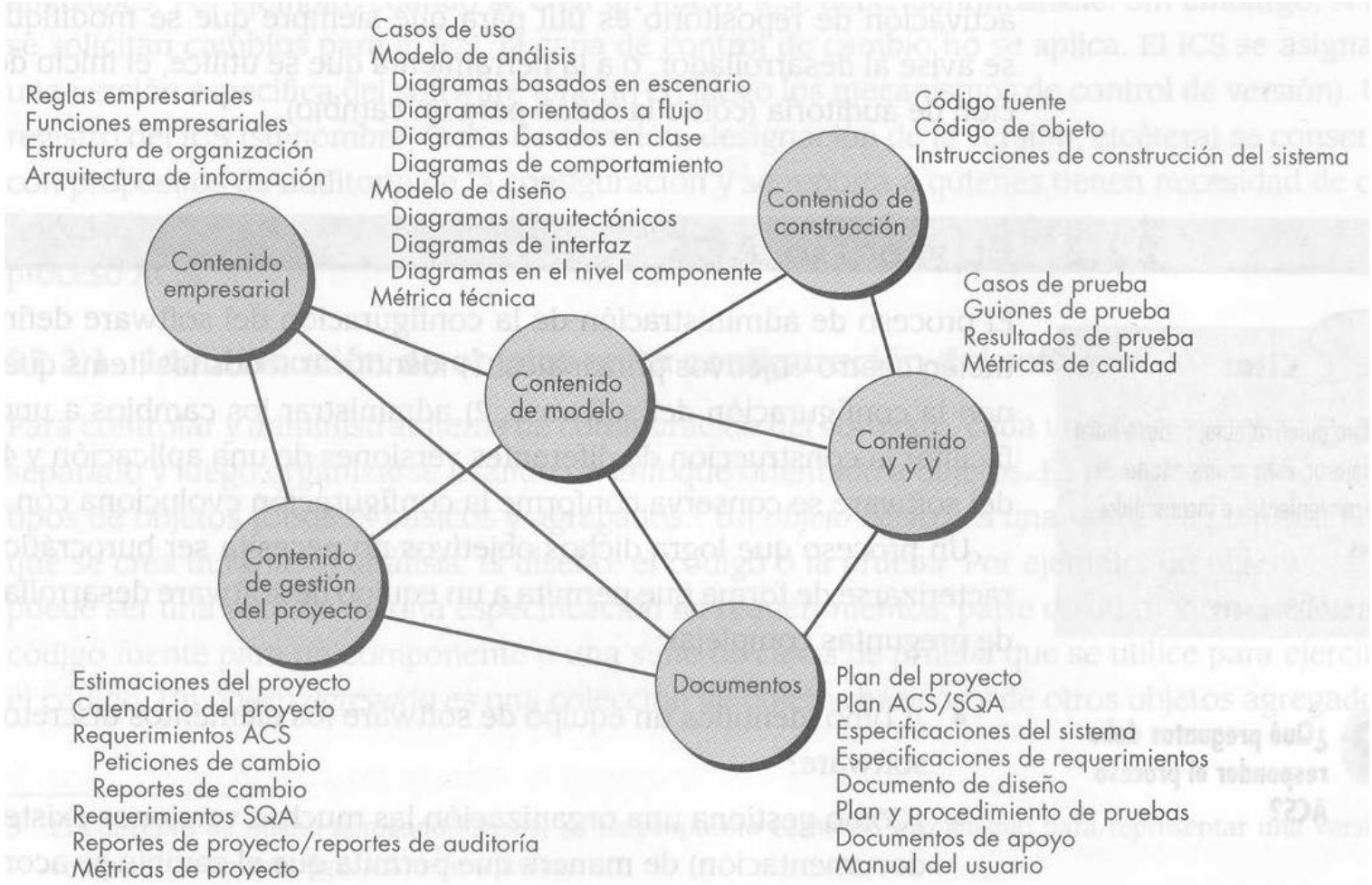
Gestión de la Configuración del SW

- Como se mencionó anteriormente, un ***ítem de configuración de software*** (ICS) es cualquier información de salida del proceso de software, que puede dividirse en tres categorías amplias: programas de cómputo, documentos o datos.



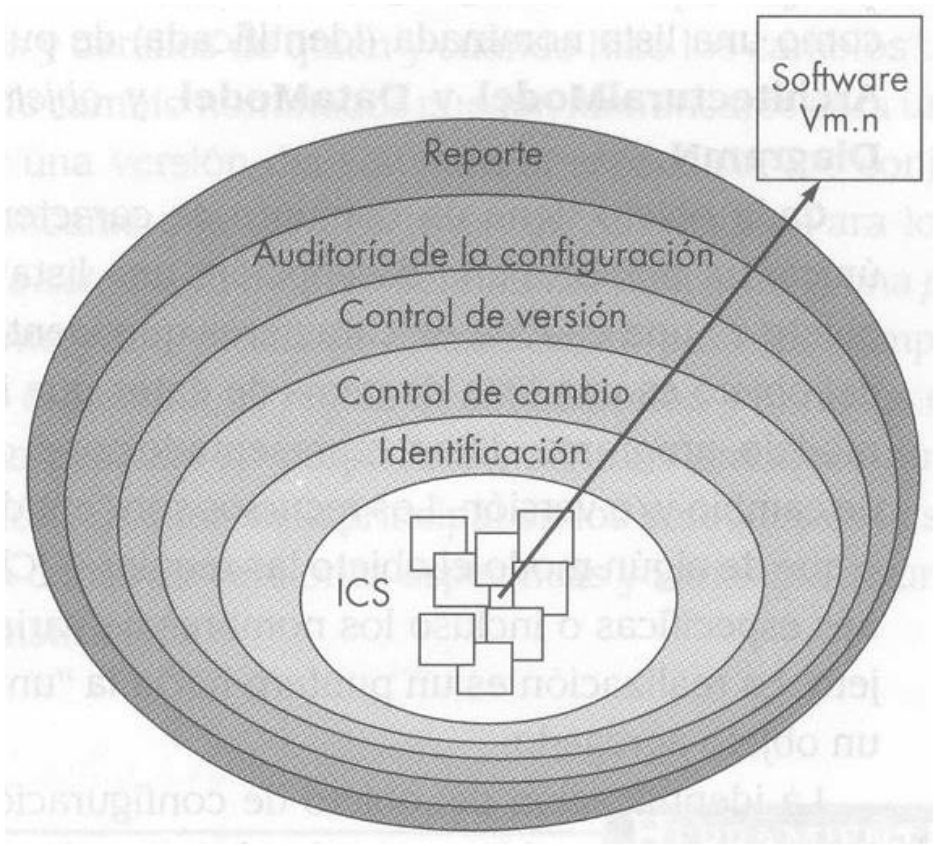
Gestión de la Configuración del SW

- El repositorio GCS es el conjunto de mecanismos y estructuras de datos que permite a un equipo de software administrar el cambio en forma efectiva.



Gestión de la Configuración del SW

- El proceso de gestión de la configuración del software define una serie de tareas que tienen cuatro objetivos principales:
  1. Identificar todos los ICS.
  2. Administrar los cambios en esos ítems.
  3. Facilitar la construcción de diferentes versiones de una aplicación.
  4. Garantizar la calidad del software conforme evoluciona la configuración.



5.3. gestión de calidad

Gestión de la Calidad

- La gestión de la calidad del software para los sistemas de software tiene tres intereses fundamentales:

  - A nivel organización, la gestión de calidad se ocupa de establecer un marco de proceso y estándares de organización que conducirán a software de mejor calidad.

  - A nivel proyecto, la gestión de calidad implica la aplicación de procesos específicos de calidad y la verificación de que continúen dichos procesos planeados.

  - A nivel de proyecto, la gestión de calidad se ocupa también de establecer un plan de calidad para un proyecto. El plan de calidad debe establece metas de calidad para el proyecto y definir cuáles procesos y estándares se usarán.

Aseguramiento de la Calidad (QA)

- Es la definición y estándares que deben conducir a la obtención de productos de alta calidad y, en el proceso de fabricación, a la introducción de proceso de calidad.

- Se refiere a la Validación y Verificación, y los procesos de que la comprobación de los procedimientos de calidad se aplicó de manera adecuada.

Calidad del Software

- La valoración de la calidad del software es un proceso subjetivo en que el equipo de gestión de calidad tiene que usar su juicio para decidir si se logró un nivel aceptable de calidad. El equipo de gestión de calidad debe considerar si el software se ajusta o no a su propósito pretendido.

  - ¿Se siguieron lo estándares de programación y documentación en el proceso de desarrollo?
  - ¿El software se verificó de manera adecuada?
  - ¿El software es suficientemente confiable?
  - ¿El rendimiento del software es aceptable para su uso normal?
  - ¿El software es utilizable?
  - ¿Está bien estructurado y es comprensible?

Atributos de calidad de Software según Boehm

- Protección
- Seguridad
- Fiabilidad
- Flexibilidad
- Robustez
- Comprensibilidad
- Comprobabilidad
- Adaptabilidad
- Complejidad
- Portabilidad
- Usabilidad
- Reusabilidad
- Eficiencia
- Facilidad para que el usuario aprenda a utilizarlo

Estándares de Software

- Los estándares de software tienen una función muy importante en la gestión de la calidad del software. Un aspecto muy importante del aseguramiento de la calidad es la definición o selección de estándares que deben aplicarse al proceso de desarrollo de software o producto de software. Los estándares de software son importantes por tres razones:

  - Reflejan la sabiduría que es de valor para la organización.
  - Proporcionan el marco para definir, en un escenario particular, lo que significa el término “Calidad”.
  - Auxilian la continuidad cuando una persona retoma el trabajo iniciado por alguien más.

Tipos de estándares de Software

- Estándares de productos.-
  - Se aplican al producto de software a desarrollar. Incluyen estándares de documentos, estándares de documentación y estándares de codificación.

- Estándares de procesos.-
  - Establecen los procesos que deben seguirse durante el desarrollo de software. Deben especificar cómo es una buena práctica de desarrollo. Pueden incluir definiciones de especificación, procesos de diseño y validación, herramientas de soporte de proceso y una descripción de los documentos que deben escribirse durante dichos procesos.

Componentes de calidad

- Debido a que la calidad es una composición de varias características, la noción de calidad se define usualmente a través de un modelo, los cuales no muestran distinción entre atributos internos y atributos externos. Ejemplos:

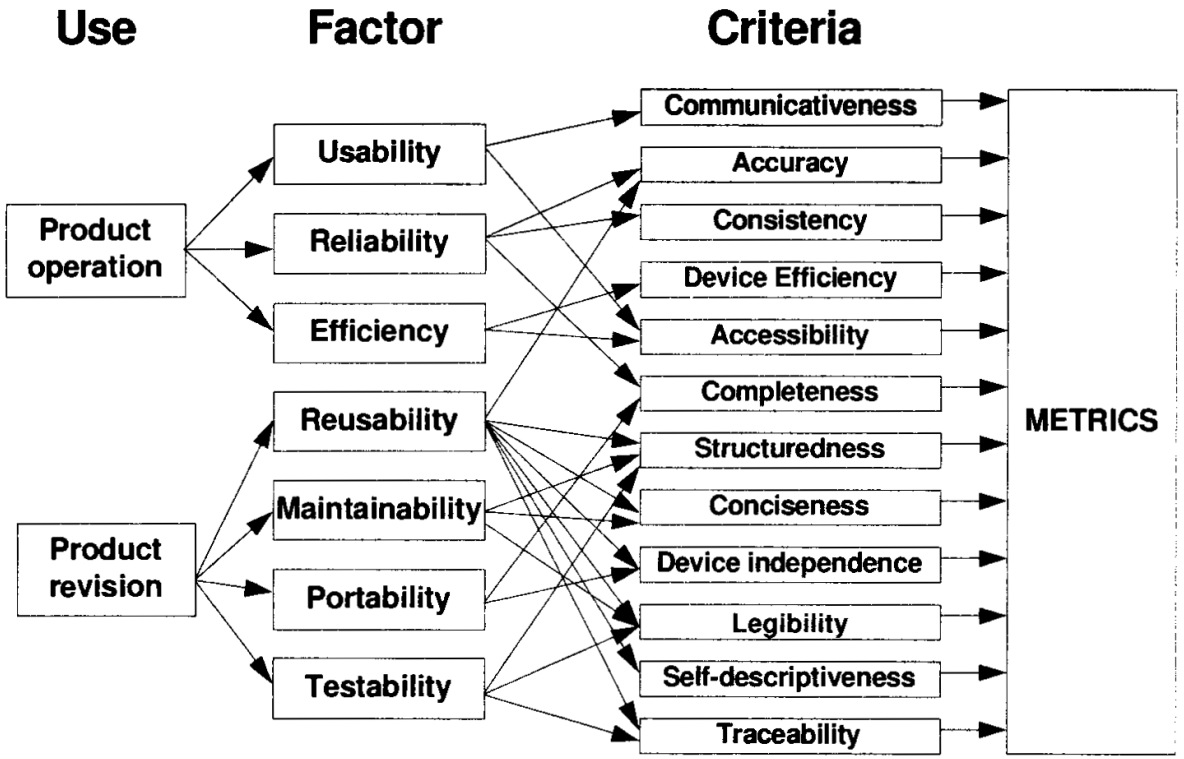


Figure 1.5: Software quality model

Componentes de calidad

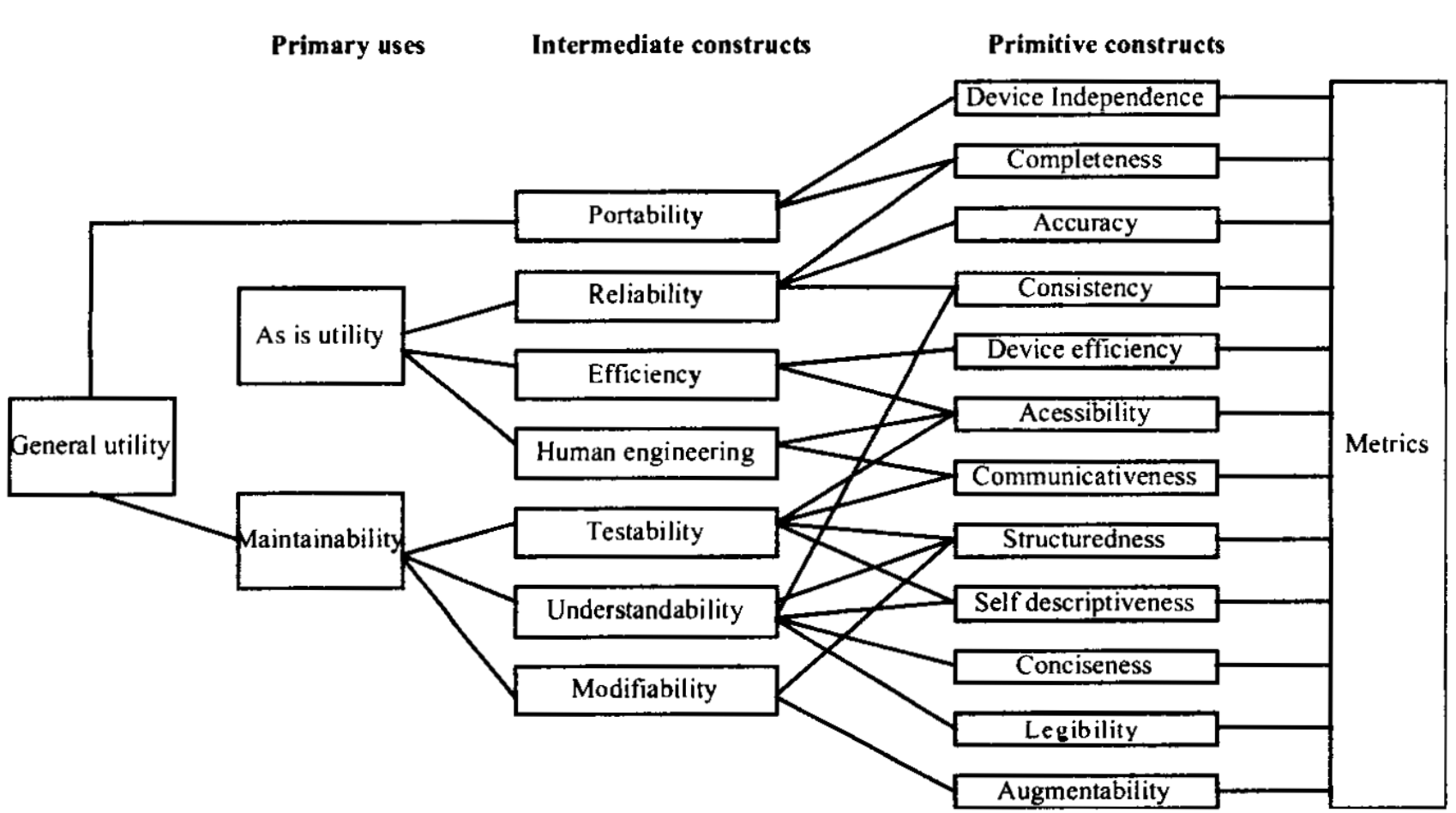


Figure 9.1: Boehm software quality model

Componentes de calidad

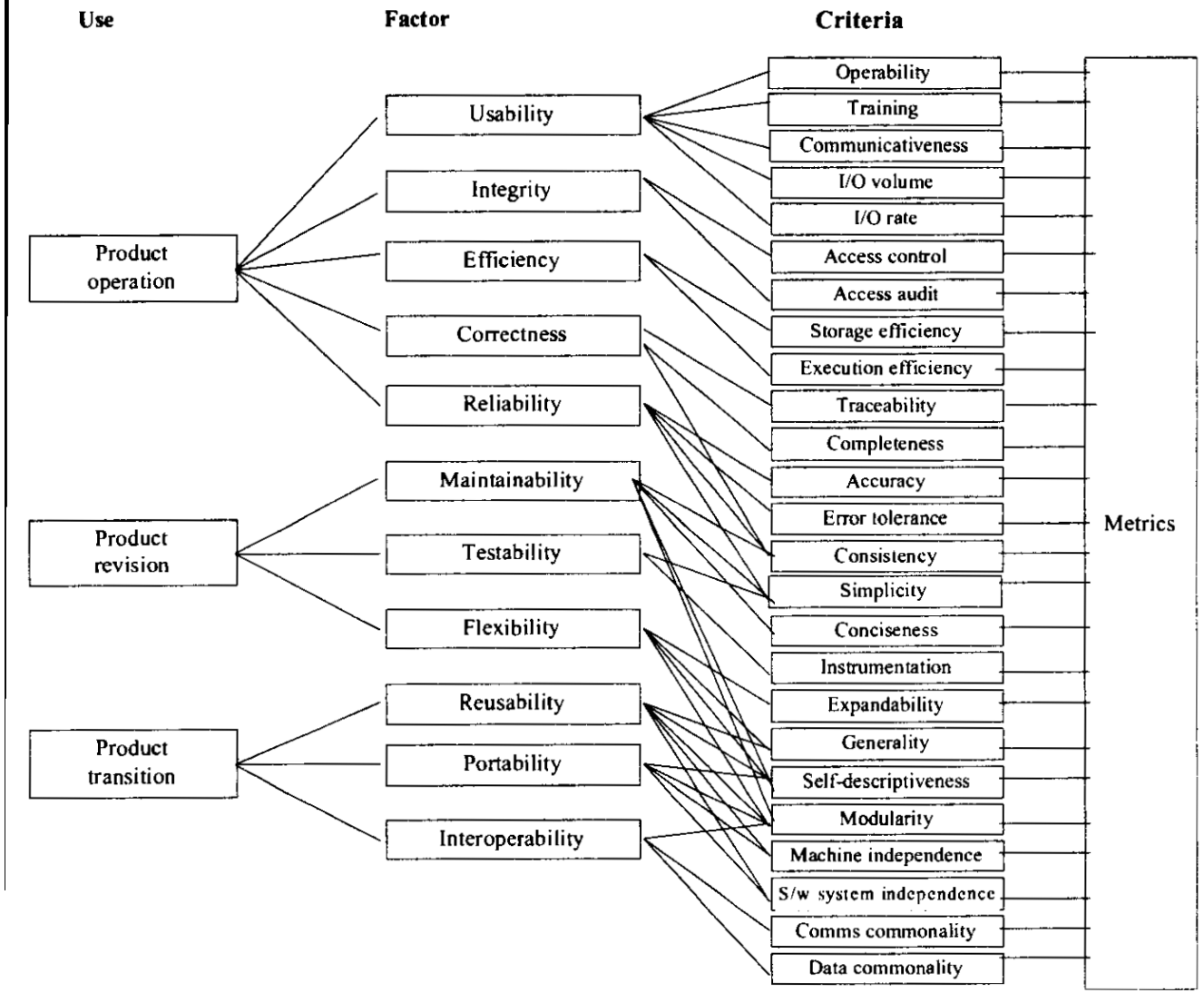


Figure 9.2: McCall software quality model

Definiendo un modelo propio

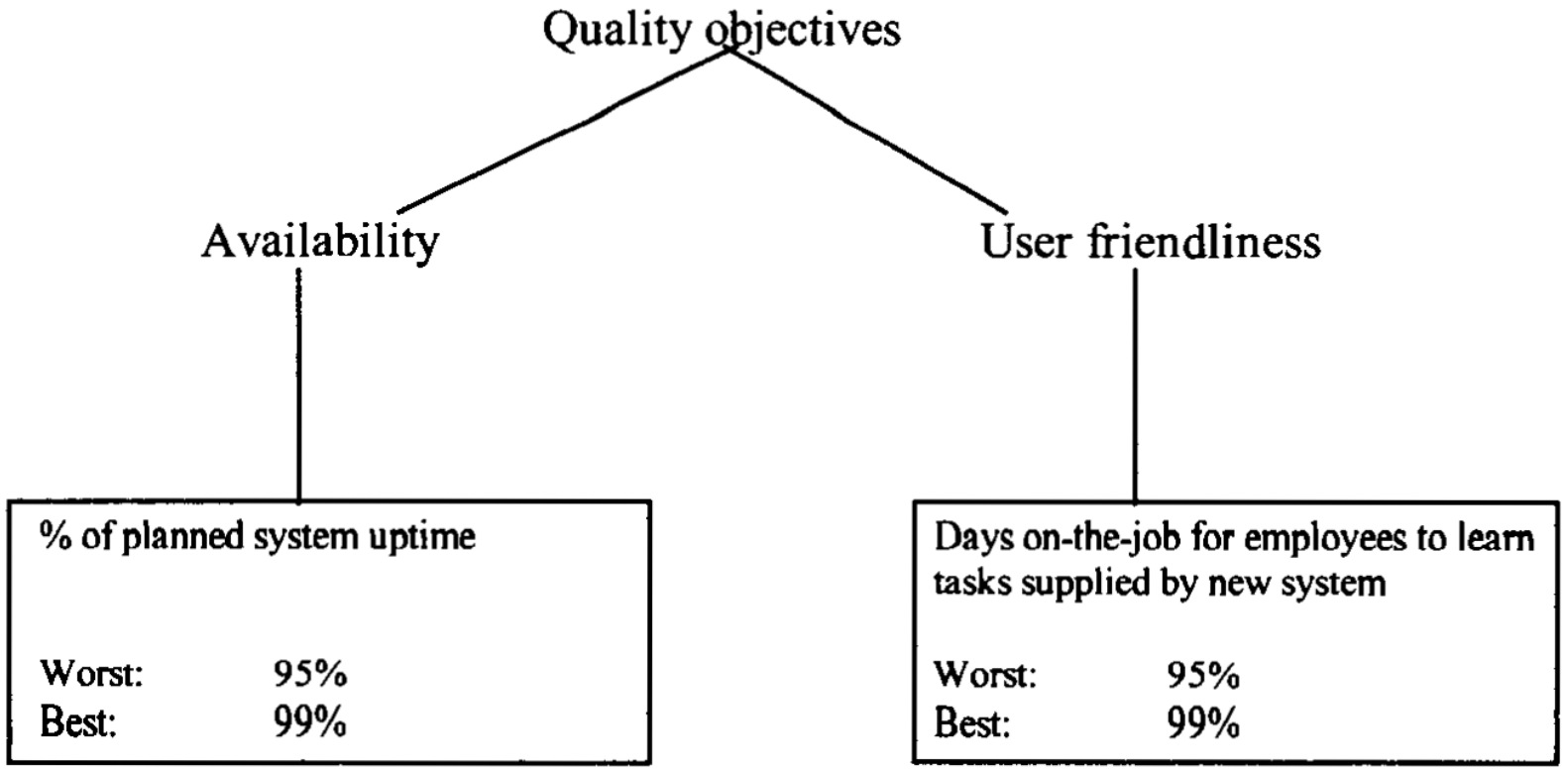


Figure 9.4: Gilb's attribute expansion approach

5.4. Herramientas case para la gestión del software

Herramientas CASE para la gestión de software

- Enterprise Architect
  - Empresa: sparxsystems (http://www.sparxsystems.com.ar)
  - Los administradores de proyectos lo utilizan para:
    - Asignar recursos a los elementos.
    - Medir el esfuerzo y riesgos.
    - Estimar el tamaño y complejidad del proyecto.
    - Implementar control de cambios y procedimientos de mantenimiento.

Herramientas CASE para la gestión de software

- PlanWise
  - Empresa: Duedil (https://www.duedil.com/)
  - Soporta:
    - Nivel empresarial y nivel proyecto.
    - Gráficas de Gantt.
    - Cambios.
    - Riesgos.
    - Decisiones.
    - Manejo de recursos humanos.
    - Presupuestos.
    - Alineación del negocio.
    - Reportes.

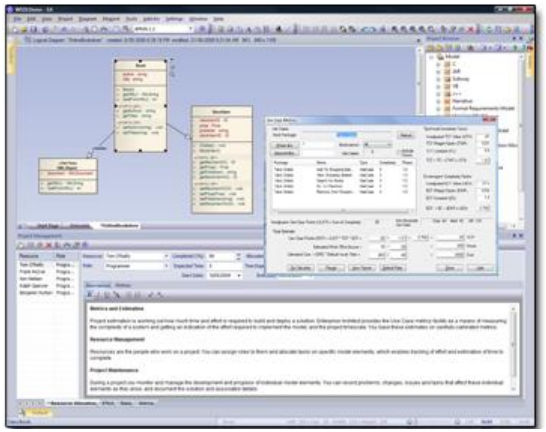
Herramientas CASE para la gestión de software

- IntelligenceSoft
  - Empresa: Intelligence-Soft (http://www.intsoft.spb.ru/)
  - Soporta:
    - Planeación de aplicaciones.
    - Reportes y herramientas de comunicación.
    - Lista de tareas y su calendarización.
    - Monitoreo del avance de las tareas.
    - Aseguramiento de la calidad mediante la planificación del testeo.

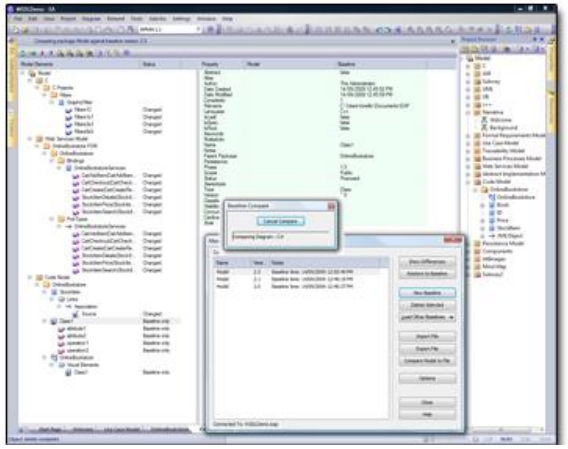
Herramientas CASE para la gestión de software

- Enterprise Architect

Administración de proyectos



Administración de la complejidad



UNIDAD 6
MODELOS DE MEJORA DEL PROCESO SOFTWARE

OBJETIVO: Al finalizar la unidad, el alumno describirá las características fundamentales de los principales modelos de mejora del proceso software y aproximaciones para el desarrollo software individual y por equipo.

CONTENIDO:

6.1. Estándares IEEE.

6.2. Modelo de madurez de la capacidad integrado.

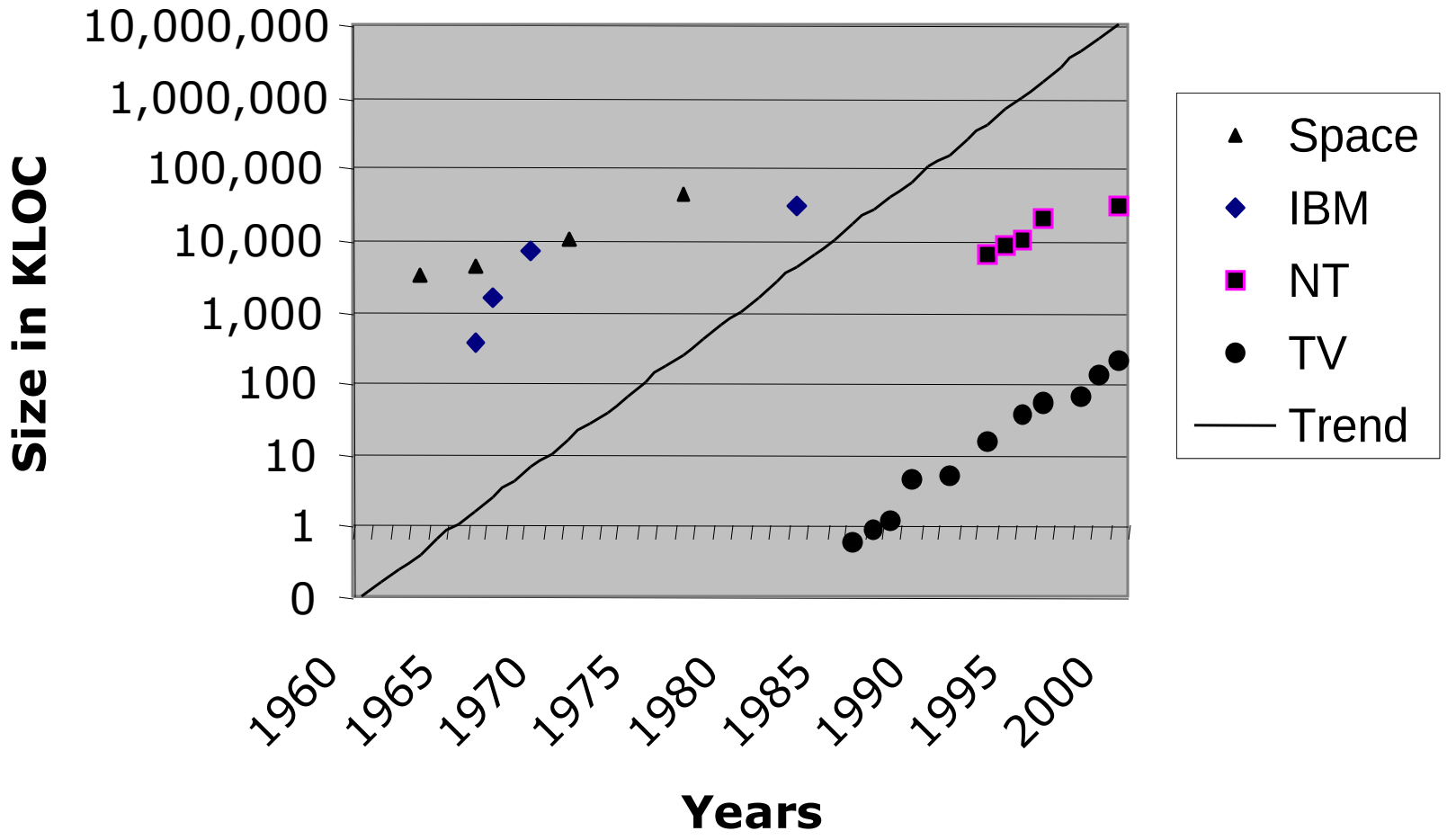
6.3. PSP.

6.4. TSP.

Estándares IEEE

Introducción

Los productos de software son más grandes



Introducción

- Cuando aumenta el tamaño, los productos de software tienen más problemas.

Project Size	People	Time (Months)	Success Rate
Less than $750K	6	6	55%
$750K to $1.5M	12	9	33%
$1.5M to $3M	25	12	25%
$3M to $6M	40	18	15%
$6M to $10M	+250	+24	8%
Over $10M	+500	+36	0%

Standish: Chaos Reports, 1999

¿Por qué fallan los proyectos?

- Los proyectos de software fallan generalmente por cuatro razones:
  - Los compromisos de software son con frecuencia irreales:
    - Mientras más grande el proyecto, se tiene menor influencia.
    - Sino tenemos nada que decir, nadie escuchará.
  - Los grandes proyectos son difíciles de controlar:
    - Actualmente, pocos desarrolladores tienes planes personales.
    - Sin plan, no se puede conocer el estado del trabajo.
    - Sino se sabe dónde se está, la administración no puede entender el estado del trabajo.
    - Si la administración no entiende el estado del trabajo, no puede administrar el proyecto.

¿Por qué fallan los proyectos?

- Los problema de calidad empeoran con el tamaño de los proyectos:
  - En sistemas de software, si alguna parte tiene problemas de calidad, el sistema tendrá problemas de calidad.
  - Si los desarrolladores no administran la calidad, sus equipos no administran la calidad.
  - Sin administración de la calidad, esta será siempre peor.

- Para ser efectivo, los equipos necesitan liderazgo y entrenamiento:
  - Los líderes construyen un equipo motivado y comprometido.
  - El entrenamiento desarrolla cohesión en el equipo.
  - Los equipos motivados, comprometidos y unidos realizan el mejor trabajo.

Consideraciones importantes

- El software es una parte crítica en la sociedad actual.
- Los métodos de software generalmente utilizados en la actualidad son aceptables, pero por no tenerse otras alternativas.
- Los profesionales de software crean sus propios métodos y técnicas de desarrollo.
- Muchos de los productos de software terminados, usualmente funcionan perfectamente, pero sólo después de un profundo proceso de pruebas y mantenimiento.

Proceso de Software

- Un proceso de software es una serie de pasos a seguir para el desarrollo y mantenimiento de software. Una definición de un proceso de software es la descripción de dicho proceso.

- Específicamente, un proceso de software establece un marco de trabajo técnico y administrativo para el empleo de métodos, herramientas y personal en las tareas de software, mientras que la definición del proceso identifica roles y tareas específicas.

Procesos definidos

- Los procesos definidos proporcionan los siguientes beneficios:
  - Propician una comunicación efectiva entre usuarios, administradores, clientes e investigadores dentro del proceso.
  - Permiten entender mejor la administración, proveen una base precisa para la automatización del proceso y facilitan la movilidad del personal.
  - Facilitan el proceso de reusabilidad.
  - Apoyan la evolución del proceso proporcionando un medio efectivo para el aprendizaje del proceso y un fundamento sólido para la mejora del proceso.
  - Ayudan a la administración del proceso.

Mejora de procesos

- En la actualidad existe una constante demanda de la industria por un mejor y más barato software, que debe entregar en plazos cada vez más cortos. Por consiguiente, numerosas compañías de software han dirigido la atención hacia la mejora de procesos de software como una forma de aumentar la calidad de su software, reducir sus costos o acelerar los procesos de desarrollo.

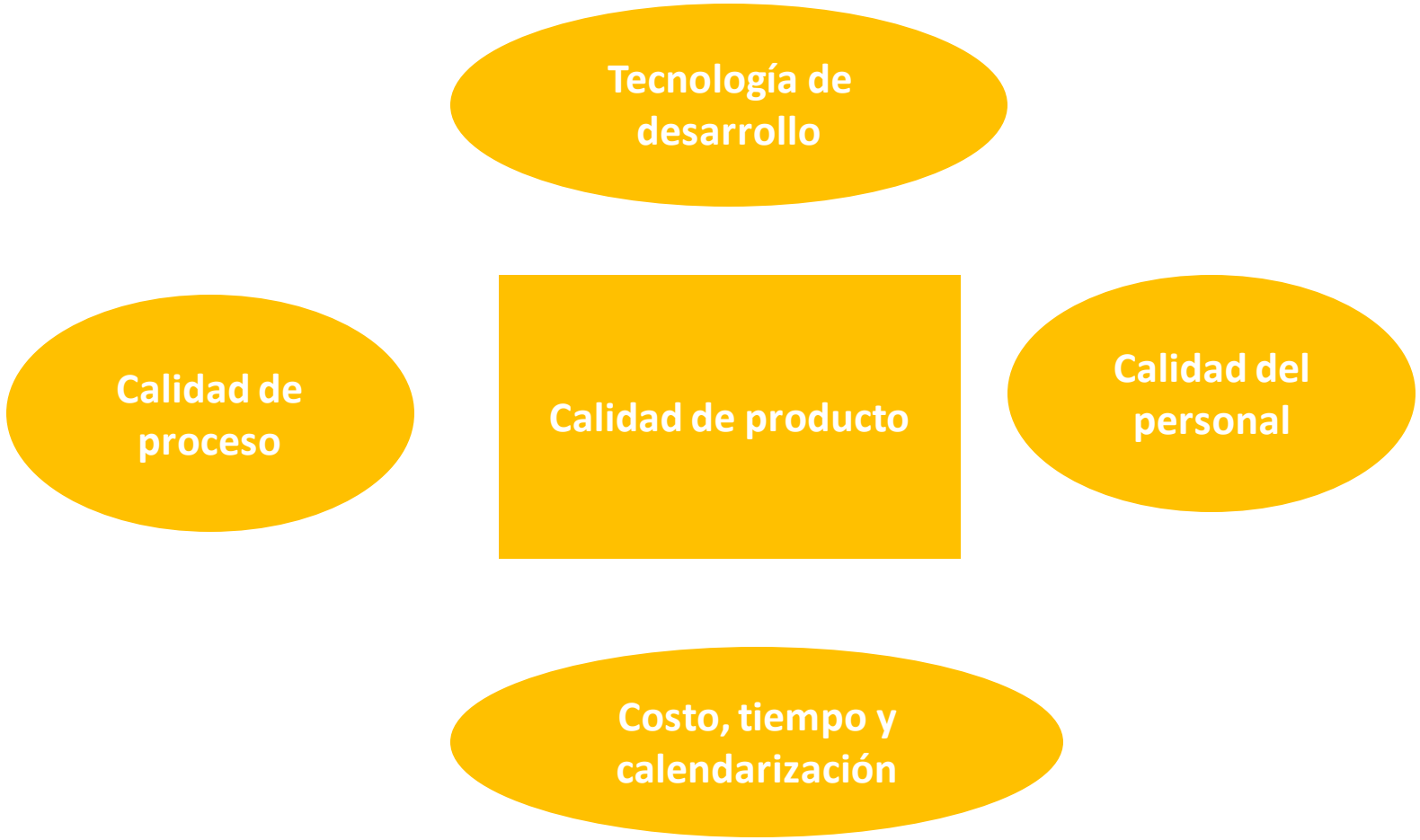
- La mejora de procesos significa comprender los procesos existentes y cambiarlos para incrementar la calidad del producto o reducir los costos y tiempo de desarrollo.

Mejora de procesos

- Se usan dos enfoques diferentes para la mejora y el cambio de procesos:
  - El enfoque de madurez de procesos, que se ha orientado en mejorar el proceso y la gestión del proyecto e introducir en una organización buenas prácticas de ingeniería de software. El nivel de madurez del proceso refleja la medida en que se adoptan buenas prácticas técnicas y administrativas en los procesos de desarrollo de software organizacional.

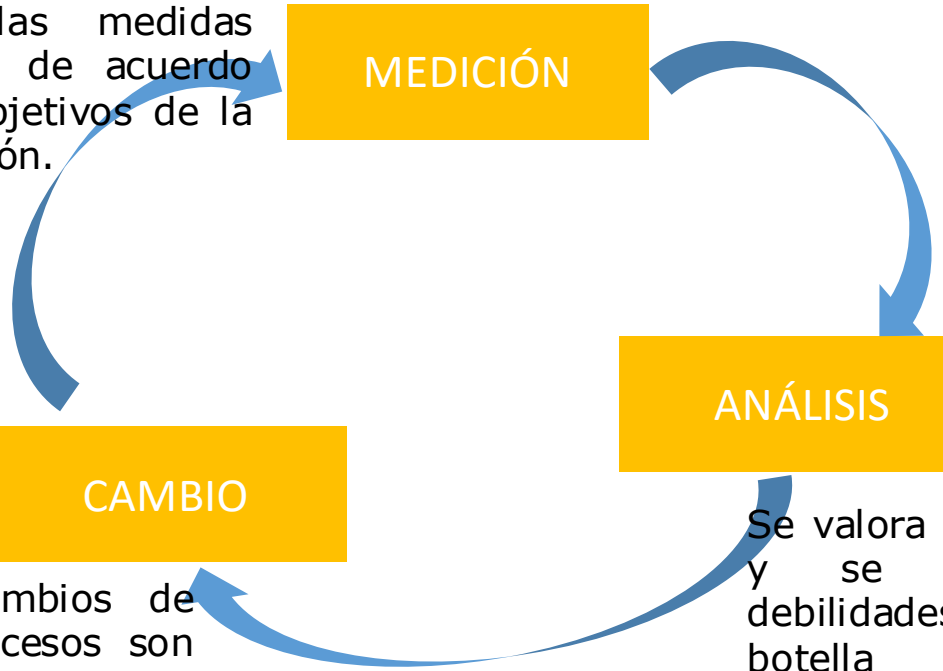
  - El enfoque ágil, orientado al desarrollo iterativo y la reducción de las sobrecargas en el proceso de software. Las características primarias de los métodos ágiles son la entrega rápida de funcionalidad y la capacidad de respuesta ante los cambiantes requerimientos del cliente.

Factores que afectan el producto de software



Proceso de mejora de procesos

Se miden atributos del proyecto actual o el producto. La meta es mejorar las medidas implicadas de acuerdo con los objetivos de la organización.

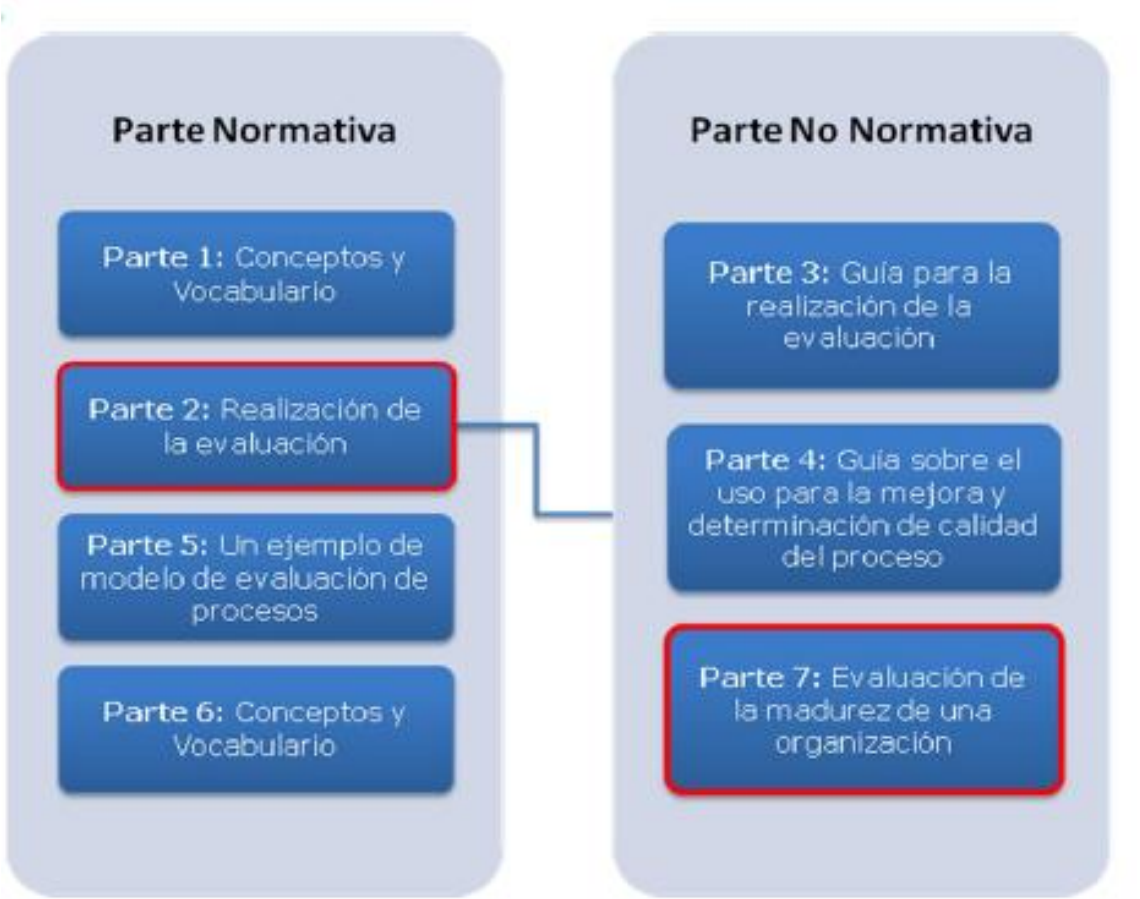


Se valora el proceso actual y se identifican las debilidades y los cuellos de botella del proceso. Durante esta etapa pueden desarrollarse modelos de proceso que describen el proceso.

Los cambios de los procesos son propuestas para atacar algunas de las debilidades identificadas en el proceso.

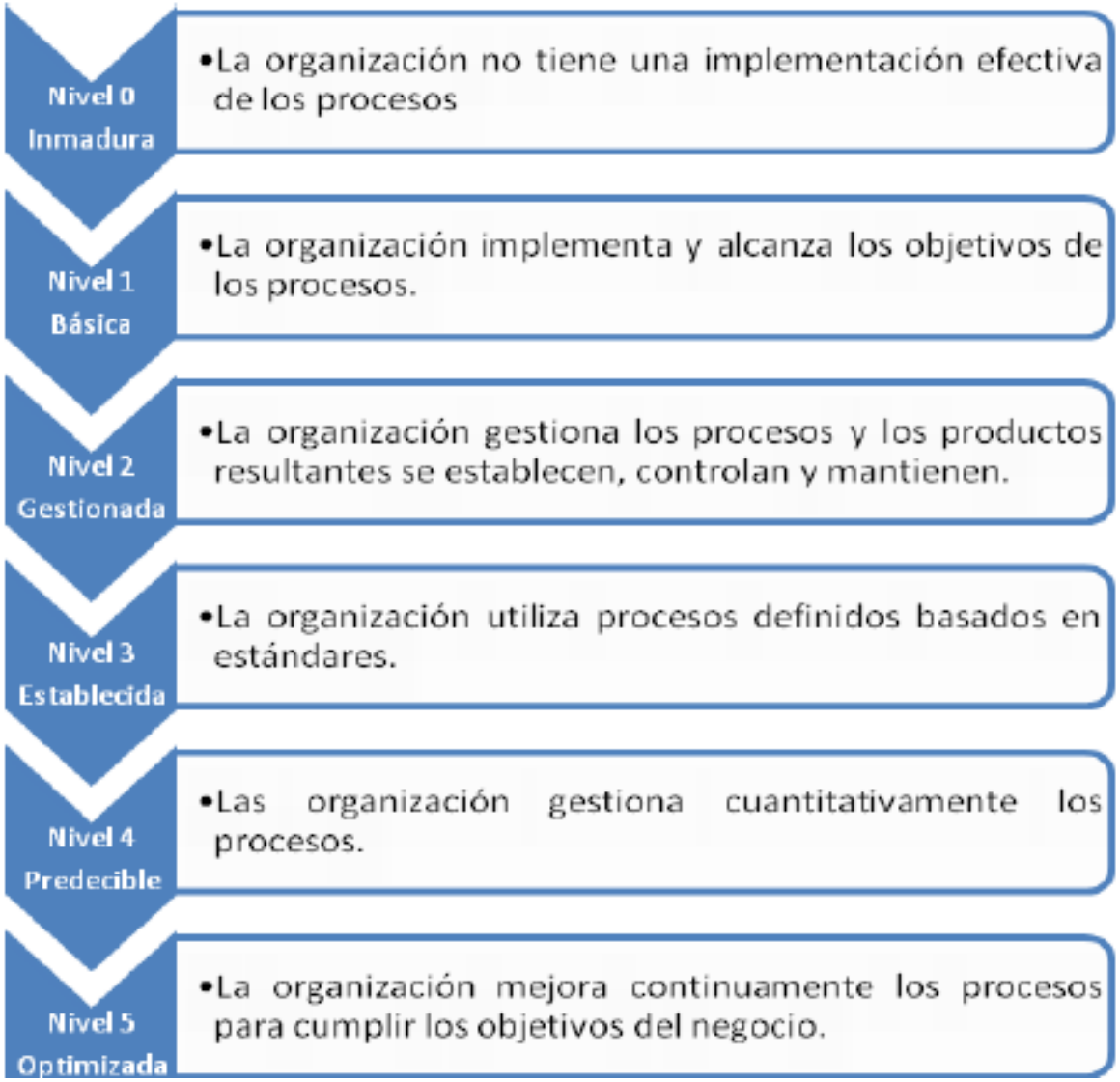
ISO (*International Organization for Standardization*) /IEC (*International Electrotechnical Commission*) 15504 IEC

- Es un modelo para la evaluación de los procesos del ciclo de vida de software que presenta la siguiente estructura:



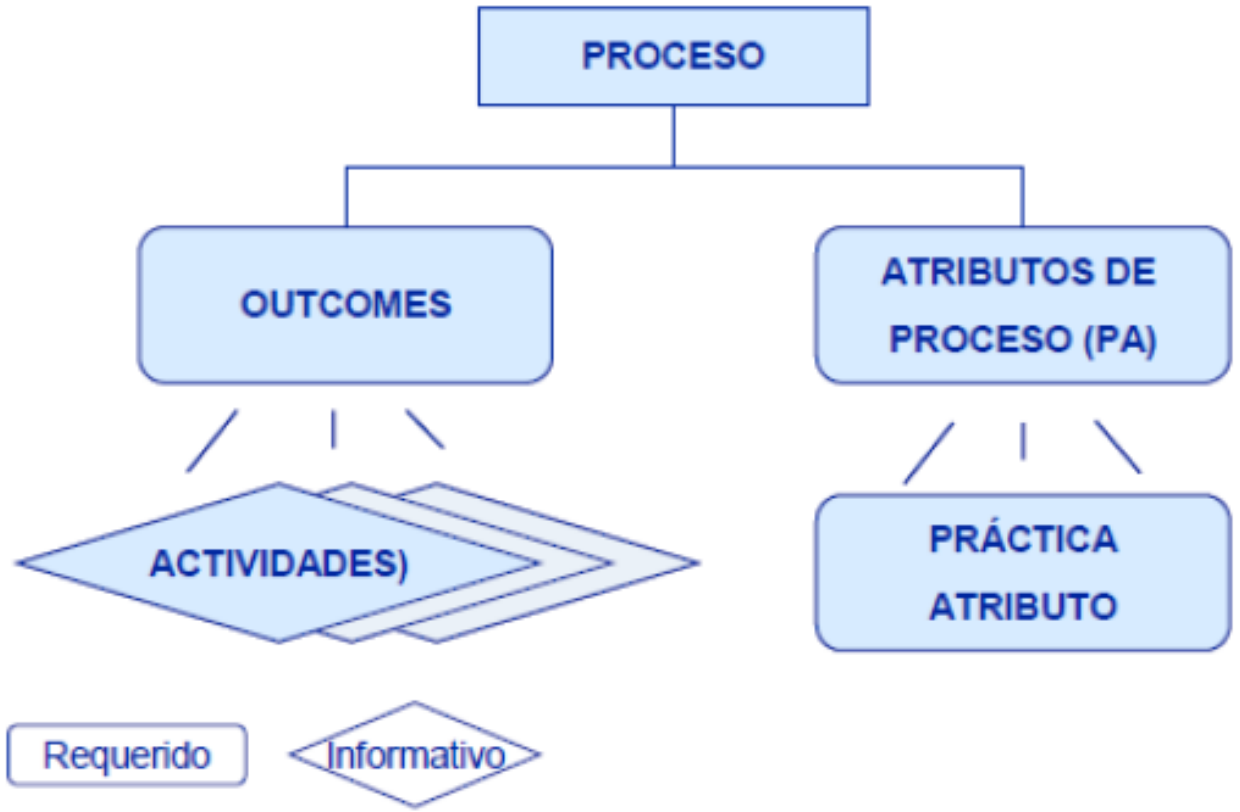
Contempla las partes normativas (1, 2, 7), que se refieren a aquellas donde se definen los requisitos mínimos para realizar una mejora de procesos de desarrollo y para medir el nivel de madurez de la organización en cuanto al desarrollo de software, y por otro lado, las no normativas (3, 4, 5, 6), en donde se dan las guías de interpretación de los requisitos mínimos y en sí sobre la norma.

Niveles de madurez según ISO/IEC 15504



Componentes del modelo

- Los procesos pertenecientes a cada nivel serán evaluados según los atributos del proceso, y los resultados del proceso conocidos como outcomes.



6.2. Modelo de Madurez de la capacidad integrado (CMMI)

Modelo de Madurez de la Capacidad Integrado (CMMI)

- El modelo CMMI tiene la intención de ser un marco para la mejora de procesos con amplia aplicabilidad.

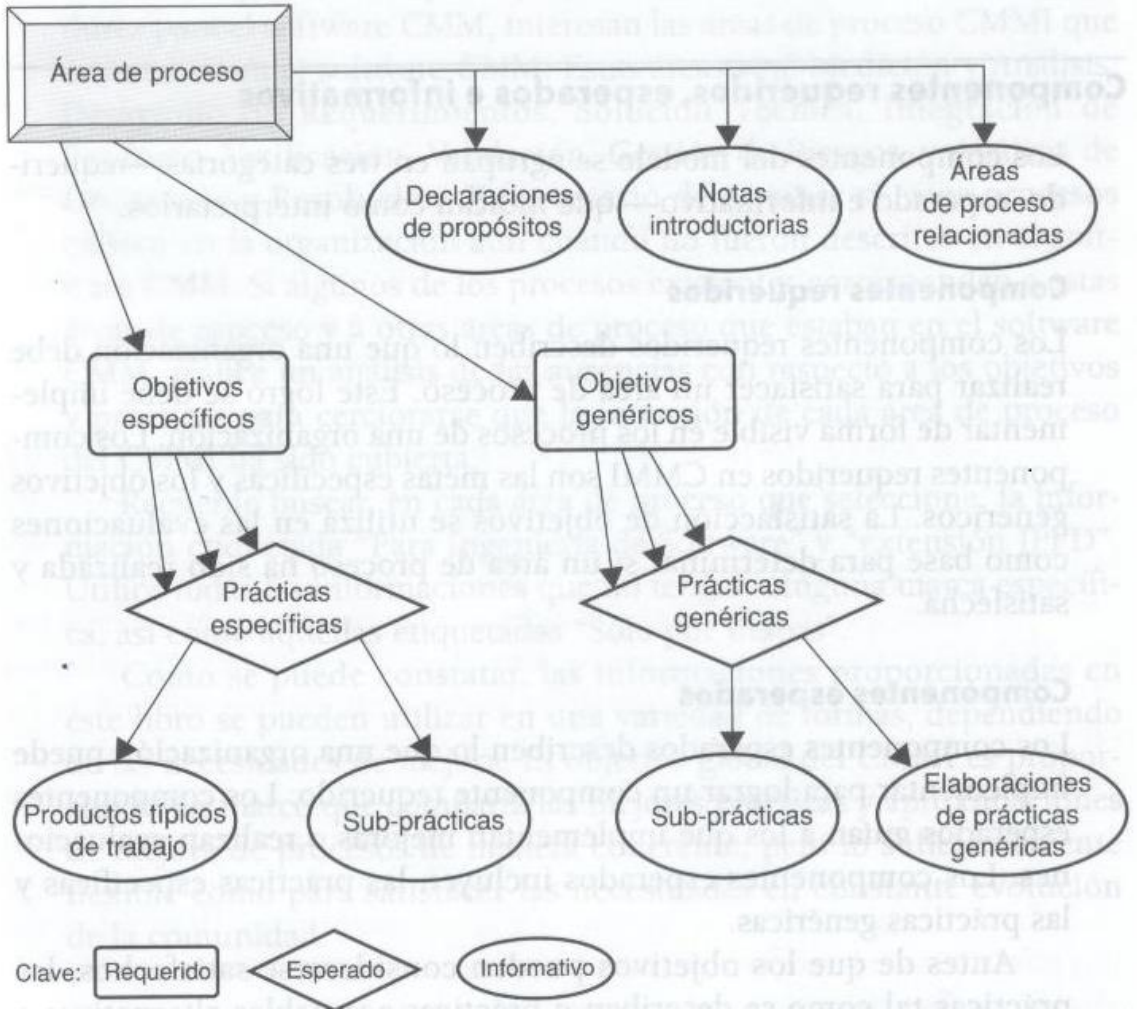
- El propósito del CMMI para desarrollo es proporcionar un modelo de referencia (conjunto de buenas prácticas) que cubra las actividades del desarrollo y mantenimiento aplicadas tanto a los productos como a los servicios.

Componentes del modelo CMMI

1. Un conjunto de 22 áreas de proceso que se relacionan con las actividades de proceso del software.

2. Algunas metas, las cuales son descripciones abstractas de un estado deseable que debe lograr una organización.

3. Un conjunto de buenas prácticas, las cuales son descripciones de formas para lograr una meta.



Áreas del modelo CMMI

Nivel	Enfoque	Áreas de Proceso
5 Optimizado	Mejora de procesos continua	CAR – Análisis y Resolución Causal OID – Innovación y Despliegue Organizacional
4 Cuantitativamente Administrado	Gestión cuantitativa de los procesos	QPM – Gestión de Proyectos Cuantitativa OPP – Desempeño de Procesos Organizacionales
3 Definido	Estandarización de procesos	DAR – Análisis de Decisión y Resolución IPM – Gestión Integral de Proyectos + IPPD OPD – Definición Organizacional de Procesos + IPPD OPF – Enfoque Organizacional de Procesos OT – Capacitación Organizacional PI – Integración de Productos RD – Desarrollo de Requerimientos RSKM – Gestión de Riesgo TS – Solución Técnica VAL – Validación VER – Verificación
2 Administrado	Gestión de proyectos básica	CM – Gestión de la Configuración MA – Medición y Análisis PMC – Monitoreo y Control de Proyectos PP – Planeación de Proyectos PPQA – Aseguramiento de Calidad de Productos y Procesos REQM – Administración de Requerimientos SAM – Gestión de Acuerdo con Proveedores
1 Inicial	Dependencia por personal competente ("héroes") y sus herramientas. Este nivel no es evaluado por CMMI.	

Modelos CMMI por etapas y continuo

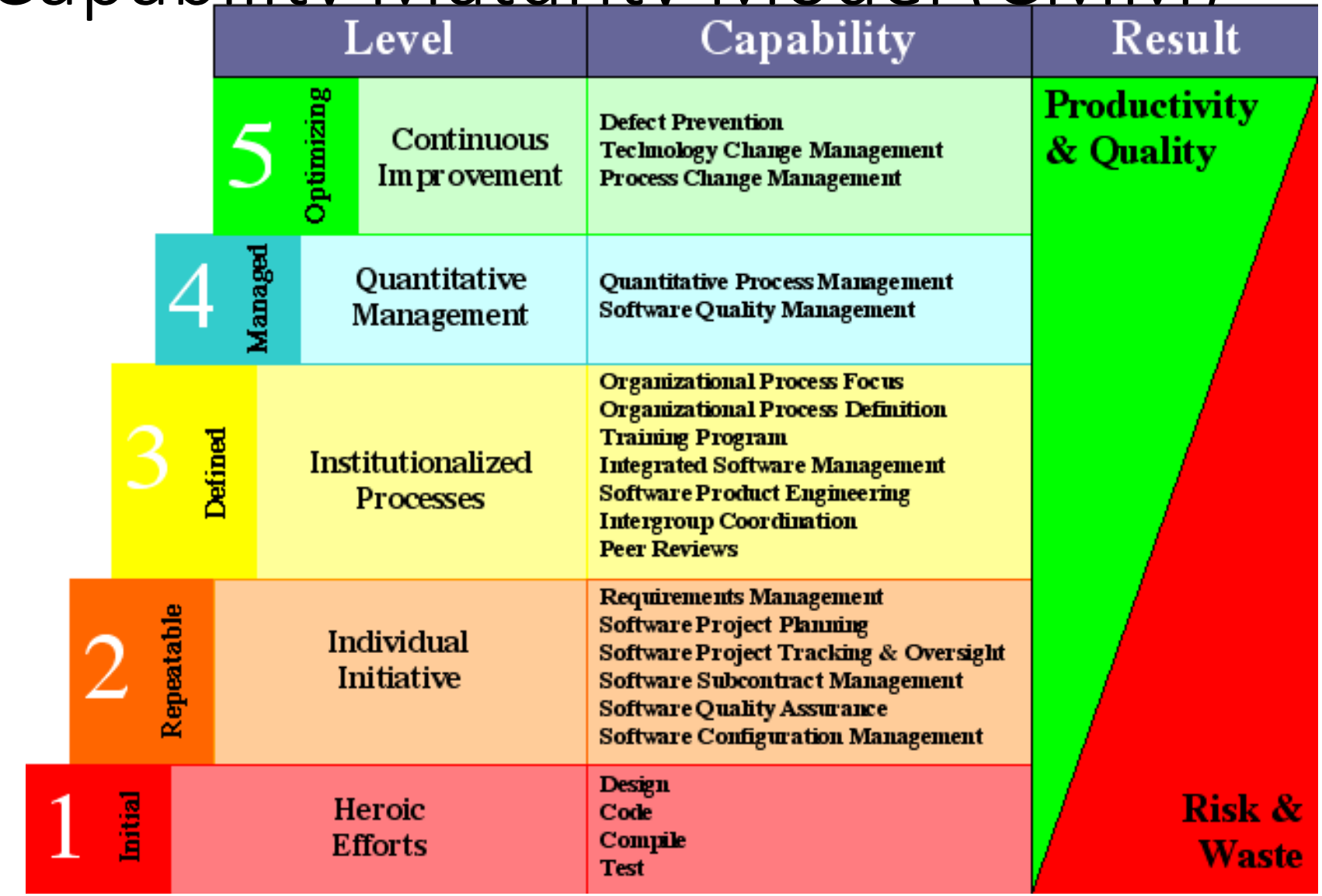
- El modelo CMMI se puede emplear desde dos puntos de vista:
  - El modelo CMMI en etapas que ofrece un medio para valorar la capacidad del proceso de una organización en uno de cinco niveles.
  - Modelo CMMI continuo se usa para medir la madurez de las áreas de proceso específicas dentro de la organización. Son modelos de grano más fino que considera prácticas individuales o en grupo y valoran el uso de la buena práctica dentro de cada grupo de procesos.

Modelo CMMI en etapas

Modelo CMMI continuo

	Representación Continua	*Representación Escalonada*
	Nivel de Capacidad	Nivel de Madurez
Nivel 0	Incompleto	-
Nivel 1	Realizado	Inicial
Nivel 2	Manejado	Manejado
Nivel 3	Definido	Definido
Nivel 4	Manejado cuantitativamente	Manejado cuantitativamente
Nivel 5	Optimizando	Optimizando

The Capability Maturity Model (CMM)



Personal Software Process (PSP)

¿Qué es PSP?

- El Proceso Software Personal (PSP) es un modelo que fue diseñado para ayudar a los ingenieros del software a hacer bien su trabajo. Muestra cómo aplicar métodos avanzados de ingeniería a sus tareas diarias. Proporciona métodos detallados de planificación y estimación, muestra a los ingenieros cómo controlar su rendimiento frente a estos planes y explica cómo los procesos definidos guían su trabajo.

¿Qué es PSP?

- El proceso de desarrollo de software personal (PSP) es un proceso diseñado para apoyar el control, gestión y mejora de la forma personal de trabajo.

- Es un marco de trabajo estructurado con formas, guías y procedimientos para el desarrollo de software.

- Provee un conjunto de datos históricos que apoyan al cumplimiento de compromisos, y hace a la rutina diaria de trabajo más predecible y eficiente.

Propósito

- El PSP Como CMMI, están basados en principios de mejoramiento de procesos. Mientras que CMMI está enfocado en mejorar las capacidades organizacionales, PSP esta puesto en el desarrollo individual.

- Para fomentar el mejoramiento en el nivel personal, PSP se extiende a la administración y control de procesos del desarrollador. Utilizando PSP los desarrolladores producen software usando un enfoque estructurado y disciplinado.

Propósito

- Siguen un proceso definido, planifican, miden, supervisan su trabajo, administran la calidad de los productos y aplican un feedback cuantitativo para mejorar sus procesos personales de trabajo, llevando a:

  - Mejores estimaciones.
  - Mejor planificación y seguimiento.
  - Protección contra compromisos que nunca se cumplen.
  - Un compromiso personal hacia la calidad.
  - Involucrarse en un proceso de mejoramiento continuo.

Enfoque del PSP

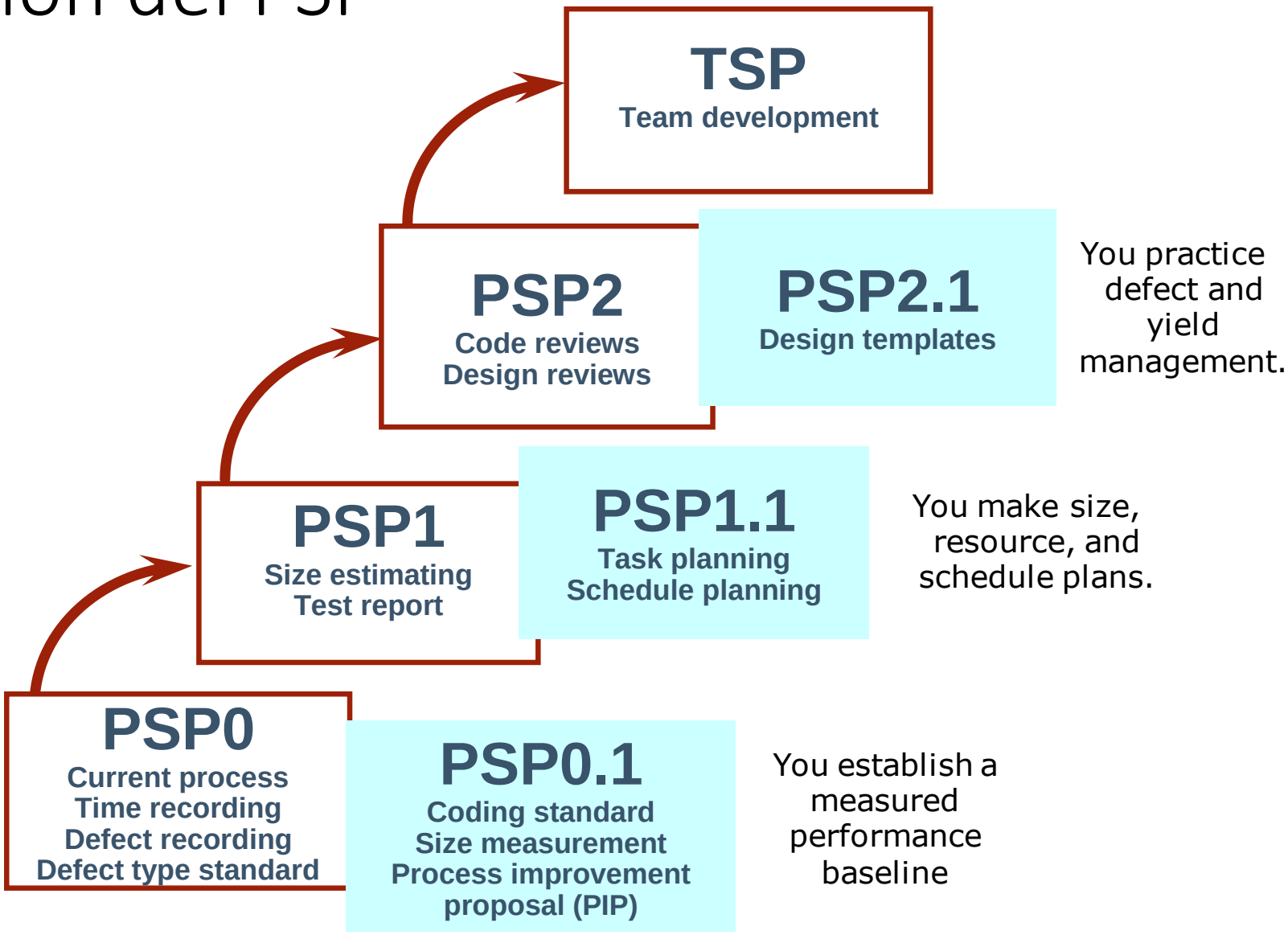
- Identifica aquellos métodos y prácticas utilizados en el desarrollo de sistemas de software a gran escala que pudieran emplearse individualmente.

- Define un subconjunto de esos métodos y prácticas que puedan ser aplicados en el desarrollo de programas pequeños.

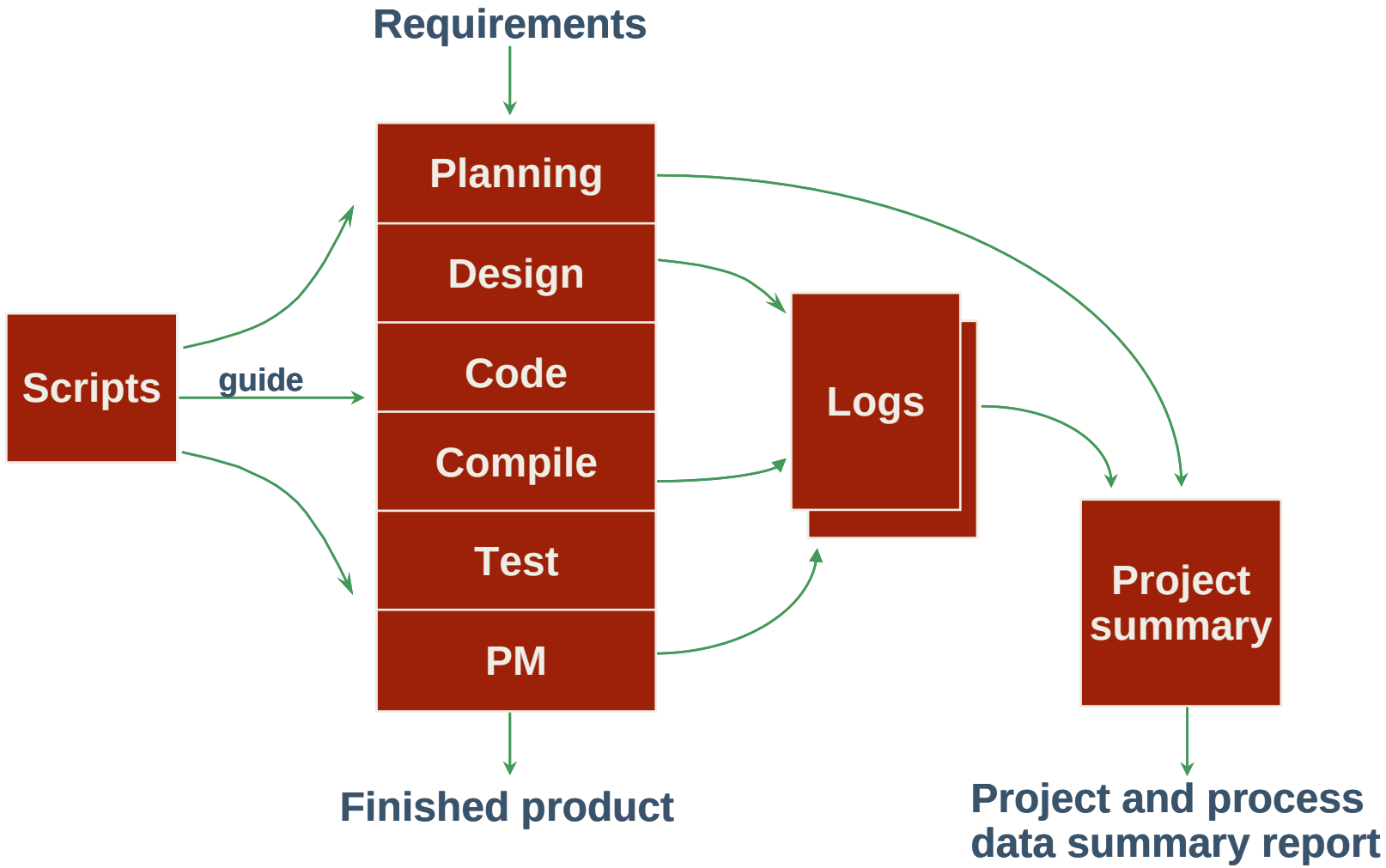
- Estructura dichos métodos y practicas de manera que pudieran ser introducidas gradualmente.

- Provee ejercicios adecuados para practicar dichos métodos en un entorno educativo.

Evolución del PSP

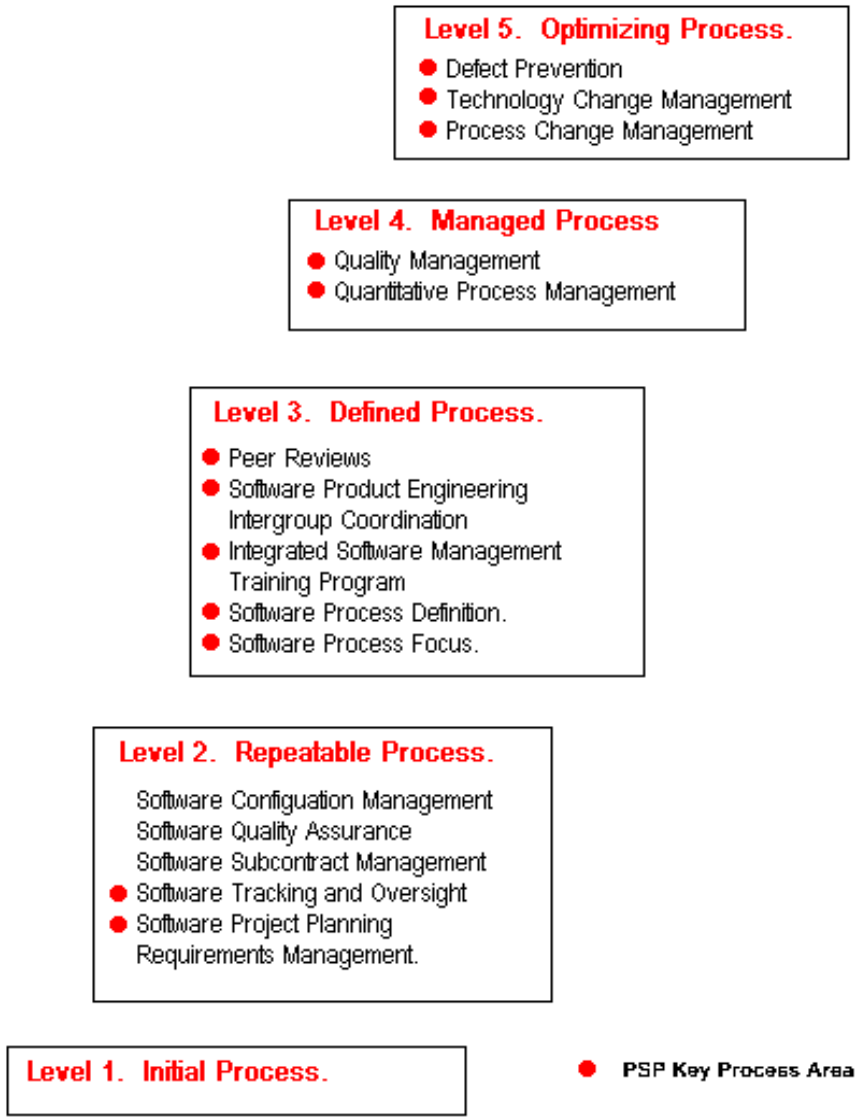


Flujo básico del PSP



CMM y PSP

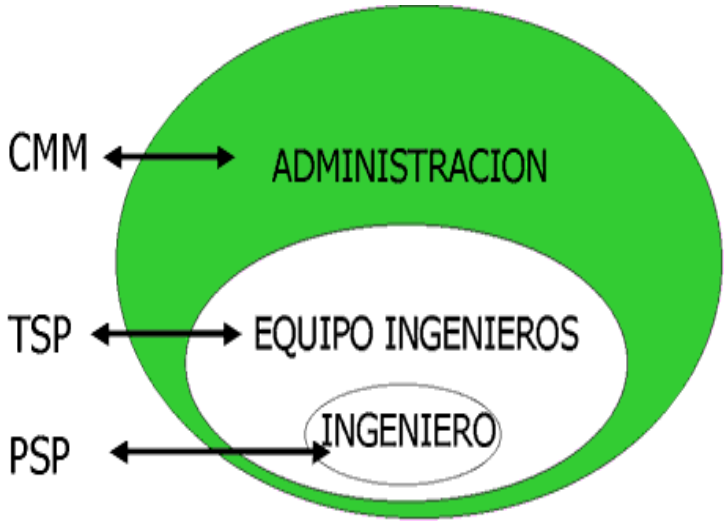
Figure 1. The CMM Levels and Corresponding Key Process Areas.



Team Software Process (TSP)

¿Qué es el TSP?

- Es una metodología para dirigir el trabajo de mejora y desarrollo de software además de establecer un entorno donde el trabajo efectivo de equipo sea normal y natural.



- Antes que los ingenieros de software puedan participar en el TSP, se requiere que ya hayan aprendido sobre el Personal Software Process (Proceso de desarrollo personal), tal que el TSP pueda funcionar de manera adecuada.

Objetivo del TSP

- Construir un equipo “autodirigido” para el proyecto, que se organice para producir software de alta calidad.

Maximizar la calidad del SW

y

Minimizar los costos

Según Humphrey, los objetivos del TSP son:

- Formar equipos autodirigidos que planeen y den seguimiento a su trabajo, que establezcan metas y que sean dueños de sus procesos y planes. Éstos pueden ser equipos de productos integrados (EPI) constituidos por 3 a 20 ingenieros.

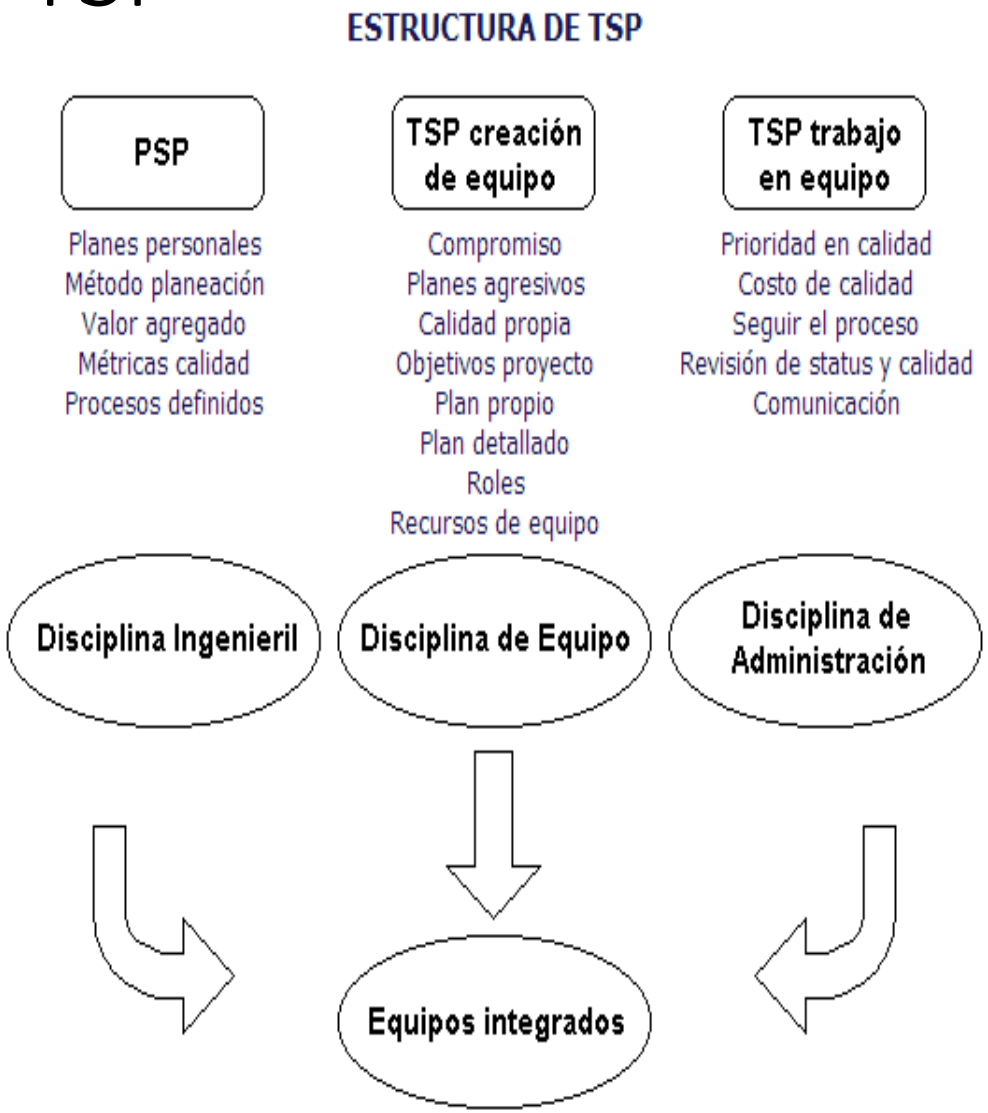
- Mostrar a los gerentes cómo dirigir y motivar a sus equipos y cómo ayudarlos a mantener un rendimiento máximo.

- Acelerar la mejora del proceso del software, haciendo del modelo de madurez de la capacidad, CMM, nivel 5, el comportamiento normal y esperado.

- Brindar a las organizaciones muy maduras una guía para la mejora.

- Facilitar la enseñanza universitaria de aptitudes de equipo con grado industrial.

Estructura del TSP



Grupos y equipos autodirigidos

- Los **grupos autodirigidos** son equipos a los cuales se les plantea una meta o problema a resolver y ellos son autónomos en la forma de resolverlo.

- Un **equipo autodirigido** tiene la comprensión consistente de sus metas y objetivos generales; define el papel y responsabilidad de cada miembro del equipo; da seguimiento cuantitativo a los datos del proyecto (sobre la productividad y calidad); identifica un proceso de equipo que sea apropiado para el proyecto y una estrategia para implementarlo; define estándares locales aplicables al trabajo de ingeniería de software del equipo; evalúa en forma continua el riesgo y reacciona en consecuencia; y da seguimiento, administra y reporta el estado del proyecto.

Equipos autodirigidos

- El TSP reconoce que los mejores equipos de software son los autodirigidos. Los miembros del equipo establecen los objetivos del proyecto, adaptan el proceso para que cubra las necesidades, controlan la programación de actividades del proyecto y, con la medida y análisis de las mediciones efectuadas, trabajan de manera continua en la mejora del enfoque de ingeniería de software que tiene el equipo.

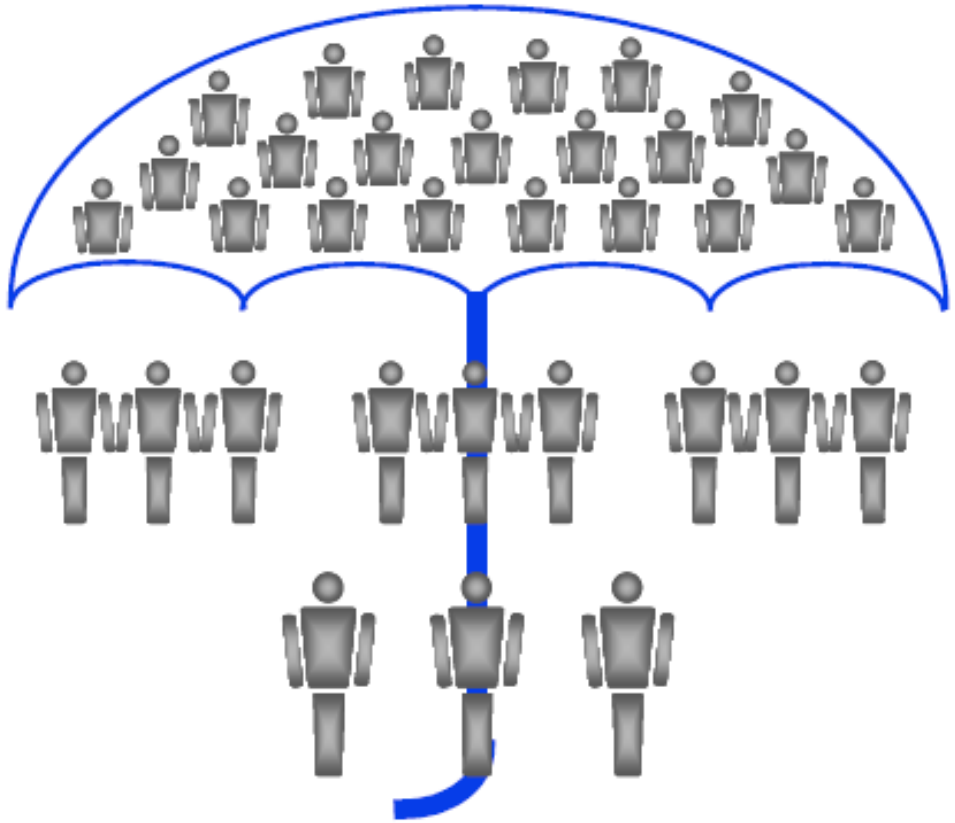
Actividades del TSP

- El PES define las siguientes actividades estructurales:
  - **Inicio del proyecto,**
  - **Diseño de alto nivel,**
  - **Implementación,**
  - **Integración y pruebas,**
  - **Post mórtem.**
- Estas actividades permiten que el equipo planee, diseñe y construya software en forma disciplinada, al mismo tiempo que mide cuantitativamente el proceso y el producto.
- La etapa **post mórtem** es el escenario de las mejoras del proceso.

Etapas del ciclo de vida del TSP

- Lanzamiento
- Estrategia
- Planteamiento
- Requerimiento
- Diseño
- Implementación
- Pruebas

Integración de procesos



CMM - Improves organization's capability; management focus.

TSP - Improves team performance; team and product focus.

PSP - Improves individual skills and discipline; personal focus.