EVOLUCIÓN DE LA FABRICACIÓN DE SOFTWARE: HACIA LA CALIDAD

Santander, 12 de julio 2010

Mario Piattini Velthuis

Universidad de Castilla-La Mancha

- **INTRODUCCIÓN**
- **DÉCADA DE LOS 50**
- **DÉCADA DE LOS 60**
- **DÉCADA DE LOS 70**
- **DÉCADA DE LOS 80**
- **DÉCADA DE LOS 90**
- **DÉCADA DE LOS 2000**
- **DÉCADA DE LOS 2010**
- **CONCLUSIONES**

- LA INGENIERÍA DEL SOFTWARE HA HECHO GRANDES AVANCES
- LENG. DE PROG. MÁS SOFISTICADOS
- PROCESOS MÁS MADUROS
- APLICACIONES MÁS COMPLEJAS

PERO . . .

- MENOR MADUREZ RESPECTO A OTRAS ING.
- DIFERENCIAS EN SATISFACCIÓN USUARIO

Our civilization runs on software

Bjarne Stroustrup

The Standish Group

	1994	1996	1998	2000	2002	2004	2006	2009
Successful	16%	27%	26%	28%	34%	29%	35%	32%
Challenged	53%	33%	46%	49%	51%	53%	46%	44%
Failed	31%	40%	28%	23%	15%	18%	19%	24%

A summary of evidence on software project cancellation rates*

Study, year, and location	Cancellation/abandonment rate (%)
Standish Group, 1994, US	31
Standish Group, 1996, US	40
Standish Group, 1998, US	28
Jones,[8] 1998, US (systems projects)	14
Jones,[8] 1998, US (military projects)	19
Jones,[8] 1998, US (other projects)	> 24
Standish Group, 2000, US	23
Standish Group, 2002, US	15
Computer Weekly,[9] 2003, UK	9
UJ,[10] 2003, South Africa	22
Standish Group, 2004, US	18
Standish Group, 2006, US	19

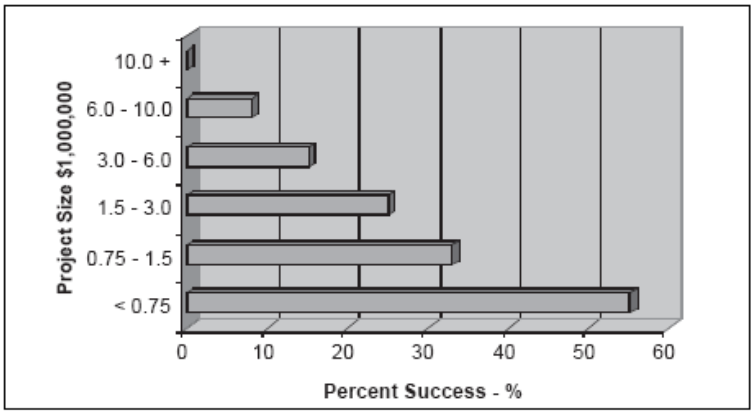
*The Standish Group data comes from various reports.[1–7]

El Emam, K. y Koru, A.G. 2008. A Replicated Survey of IT Software Project Failures. IEEE Software Volume: 25, Issue: 5

Why Big Software Projects Fail: The 12 Key Questions

Watts S. Humphrey
The Software Engineering Institute

In spite of the improvements in software project management over the last several years, software projects still fail distressingly often, and the largest projects fail most often. This article explores the reasons for these failures and reviews the questions to consider in improving your organization's performance with large-scale software projects. Not surprisingly, considering these same questions will help you improve almost any large or small project with substantial software content. The principal questions concern why large software projects are hard to manage, the kinds of management systems needed, and the actions required to implement such systems. In closing, the author cites the experiences of projects that have used the methods described and cites sources for further information on introducing the required practices.



Communications of the ACM

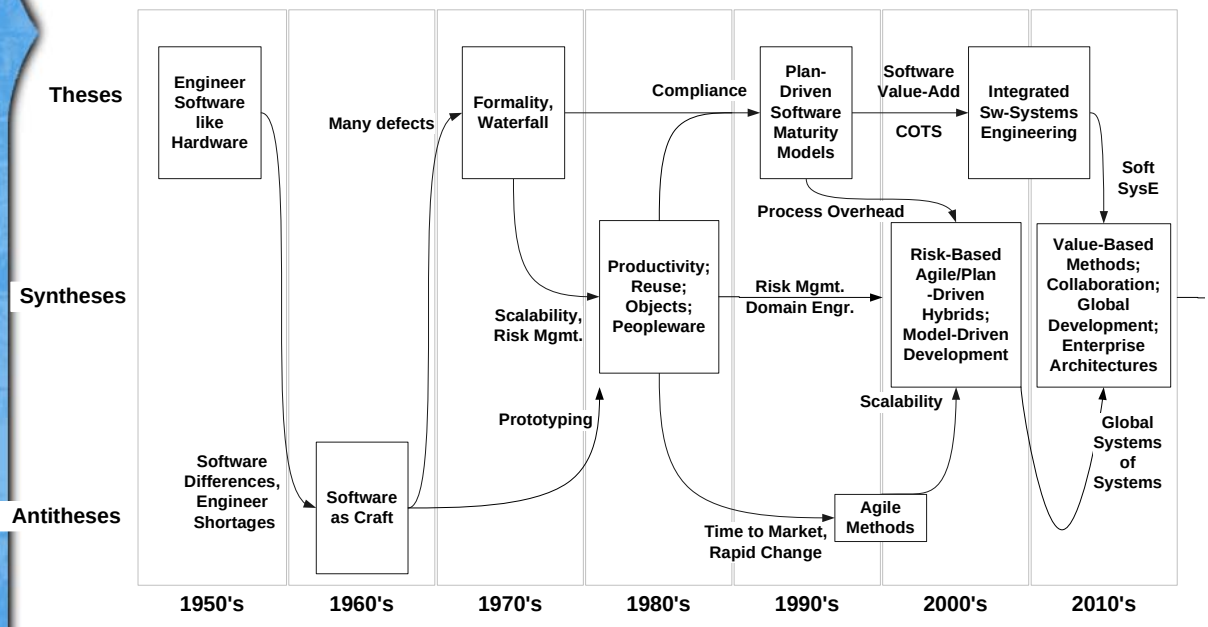
THE ONE-MINUTE RISK ASSESSMENT TOOL

An analysis of risks in software development, using data from senior IT managers, produced surprising results. Our one-minute assessment tool applies those results to assessing the risks of specific projects.

Of the $2.5 trillion spent on IT during 1997–2001, nearly $1 trillion was wagered on underperforming projects [3]. A large number of underperforming projects ultimately fail, costing U.S. companies more than $75 billion each year [8]. While some events cannot be predicted or controlled, many of the risks that repeatedly plague software projects can be assessed and managed.

How do software project managers walk the fine line between calculated risks that can lead to innovative business solutions and outright gambling that can lead to career-ending projects? Existing software project risk frameworks do not provide the kind of that managers often complain about the most. This article examines the relative importance of six key drivers of software project risk and introduces a one-minute risk assessment tool that can be applied to improve software practice.

Adaptado de Boehm (2006)



- INTRODUCCIÓN
- DÉCADA DE LOS 50
- DÉCADA DE LOS 60
- DÉCADA DE LOS 70
- DÉCADA DE LOS 80
- DÉCADA DE LOS 90
- DÉCADA DE LOS 2000
- DÉCADA DE LOS 2010
- CONCLUSIONES

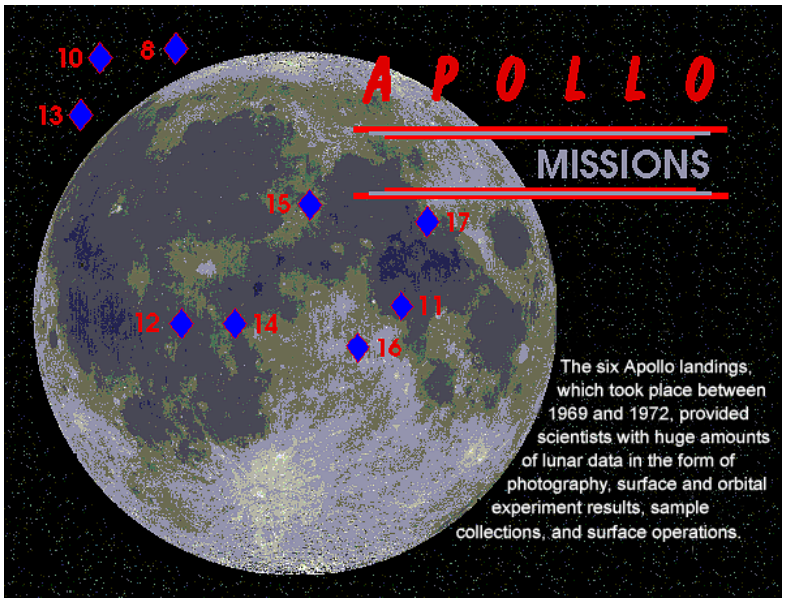
- El software se desarrolla como el hardware
- Coste del hardware muy superior
- Software para Defensa
- Mismos ingenieros para hard/soft

ACM Computer Machinery

IEEE Computer Society

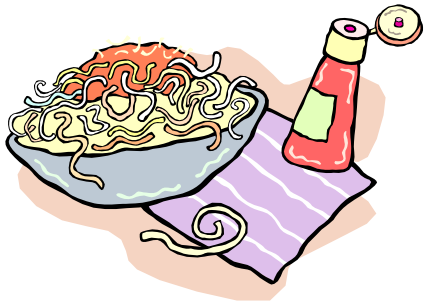
- INTRODUCCIÓN
- DÉCADA DE LOS 50
- **DÉCADA DE LOS 60**
- DÉCADA DE LOS 70
- DÉCADA DE LOS 80
- DÉCADA DE LOS 90
- DÉCADA DE LOS 2000
- DÉCADA DE LOS 2010
- CONCLUSIONES

- Lenguajes de alto nivel (COBOL, FORTRAN)
- Éxitos como OS/360, Apolo de la NASA



Pressman (2005):

- **El software se desarrolla, no se fabrica en un sentido clásico**
- **El software no se "estropea", pero se deteriora**
- **La mayoría del software se construye a medida**



- NASA/IEEE Software Engineering Workshop (1966)

The NATO Software Engineering Conferences (1968/1969)

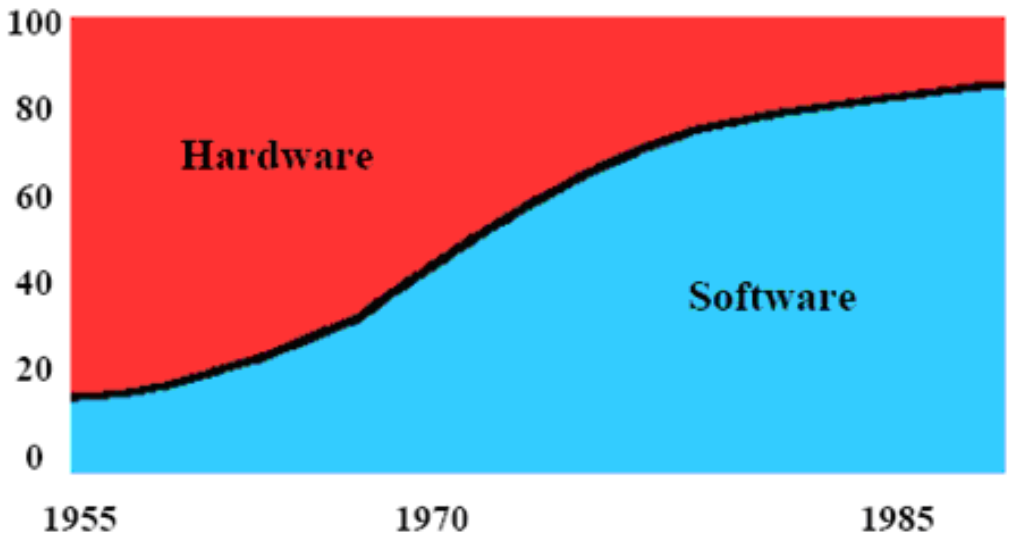
- **Dijkstra (1968)** *"Go To Statement Considered Harmful"*
- **McIllroy (1968) "reutilización/componentes sw"**
- **Bemer (1969)** *"parece que tenemos pocos entornos específicos (instalaciones de fábrica) para la producción económica de programas…. Una fábrica proporciona energía, espacio de trabajo, distribución del trabajo, controles financieros, etc. Por lo que una* ***fábrica de software*** *debería ser un entorno de programación residente en y controlado por un ordenador".*
- **Hitachi Software Works (1969):**
  - *Mejora de la productividad y fiabilidad por medio de la estandarización y control de procesos*
  - *Transformación del software de un servicio desestructurado a un producto con un nivel de calidad garantizado.*

ISACA® comenzó en 1967, cuando un pequeño grupo de personas con trabajos similares (controles de auditoría en los sistemas computarizados que se estaban haciendo cada vez más críticos para las operaciones de sus organizaciones respectivas) se sentaron a discutir la necesidad de tener una fuente centralizada de información y guía en dicho campo.

En 1969, el grupo se formalizó, incorporándose bajo el nombre de *EDP Auditors Association* (Asociación de Auditores de Procesamiento Electrónico de Datos).

http://www.isacamadrid.es/default/NoticiasDescripcion.asp?ID=151

- INTRODUCCIÓN
- DÉCADA DE LOS 50
- DÉCADA DE LOS 60
- **DÉCADA DE LOS 70**
- DÉCADA DE LOS 80
- DÉCADA DE LOS 90
- DÉCADA DE LOS 2000
- DÉCADA DE LOS 2010
- CONCLUSIONES



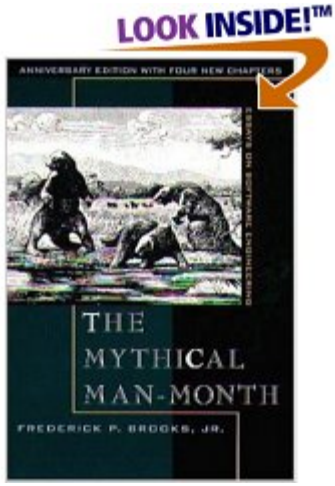
”No hay ninguna razón para que un individuo tenga un ordenador en su casa”

Ken Olson, Presidente de DEC, en 1977

- **Royce (1970) "ciclo de vida en cascada"**
- **Parnas (1972) "information hiding"/módulo**
- **Métodos estructurados**

AÑO	METODOLOGÍA
1968	Conceptos sobre la programación estructurada de DIJKSTRA, WARNIER y JACKSON
1974	Técnicas de programación estructurada de WARNIER y JACKSON
1975	Diseño estructurado de MYERS, YOURDON y CONSTANTINE
1976	Modelo E/R de CHEN
1977	Análisis estructurado GANE y SARSON
1978	Análisis estructurado: DEMARCO y WEINBERG MERISE
1981	SSADM Information Engineering

- **Fábricas de software** ***(Systems Development Corporation en 1975, NEC en 1976, Toshiba en 1977, Fujitsu en 1979 y 1983, Hitachi en 1985, NTT en 1985, Mitsubishi en 1987, …)***
- **Aspectos psicológicos** ***(Weinberg (1971): Psychology of Computer Programming; Brooks (1975): Mythical Man Month)***
- **Métodos formales**

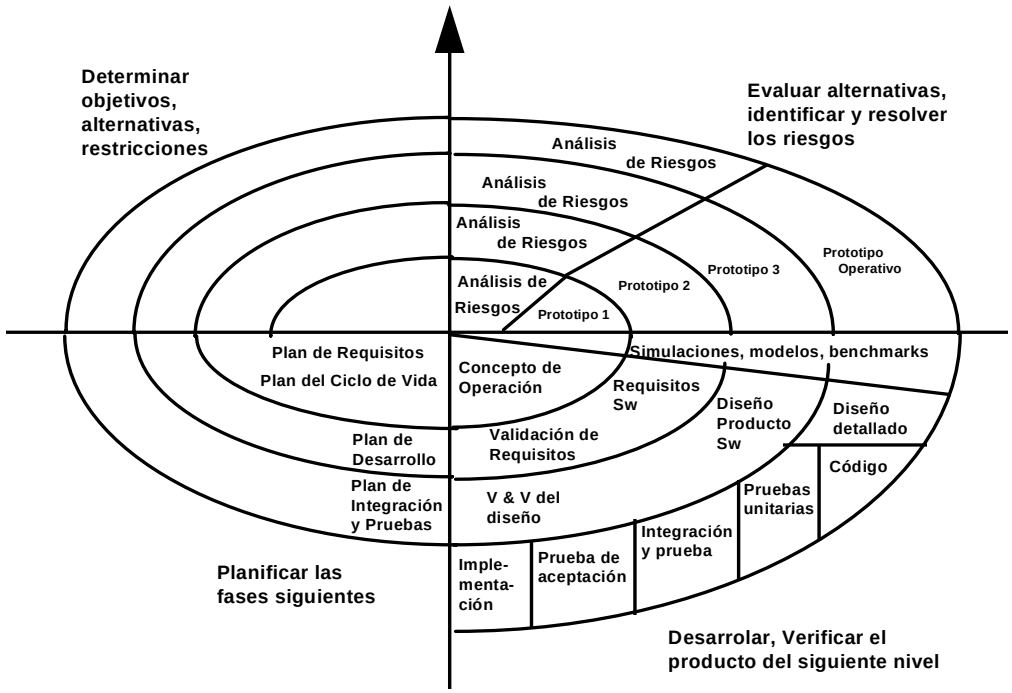


- INTRODUCCIÓN
- DÉCADA DE LOS 50
- DÉCADA DE LOS 60
- DÉCADA DE LOS 70
- **DÉCADA DE LOS 80**
- DÉCADA DE LOS 90
- DÉCADA DE LOS 2000
- DÉCADA DE LOS 2010
- CONCLUSIONES

DÉCADA DE LOS 80

- **Mc Cracken y Jackson (1982)** *"Life Cycle Concept Considered Harmful".*
- **Boehm (1986)** "*A Spiral Model of Software Development and Enhancement".*
- **Osterweil (1987)** "*Software Processes are Software Too".*
- **ISO 9000 (1987)**
- **Humphrey (1989)** *"Software Capability Maturity Model" (SW-CMM) del CMU Software Engineering Institute (SEI).*
- **Nueva tecnología:** SGBD, L4G, Sistemas expertos, Programación visual, Herramientas CASE/IPSE/PSEE, Lenguaje Ada, Orientación a objetos, etc.

MODELO EN ESPIRAL



PROCESO SOFTWARE

"*Conjunto de actividades, métodos, prácticas y transformaciones que la gente usa para desarrollar y mantener software y los productos de trabajo asociados (planes de proyecto, diseño de documentos, código, pruebas y manuales de usuario)"* (SEI, 1995).

"*Proceso o conjunto de procesos usados por una organización o proyecto, para planificar, gestionar, ejecutar, monitorizar, controlar y mejorar sus actividades software relacionadas*" (ISO, 1998).

"*Conjunto coherente de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, empaquetar y mantener un producto software*" (Fuggeta, 2000).

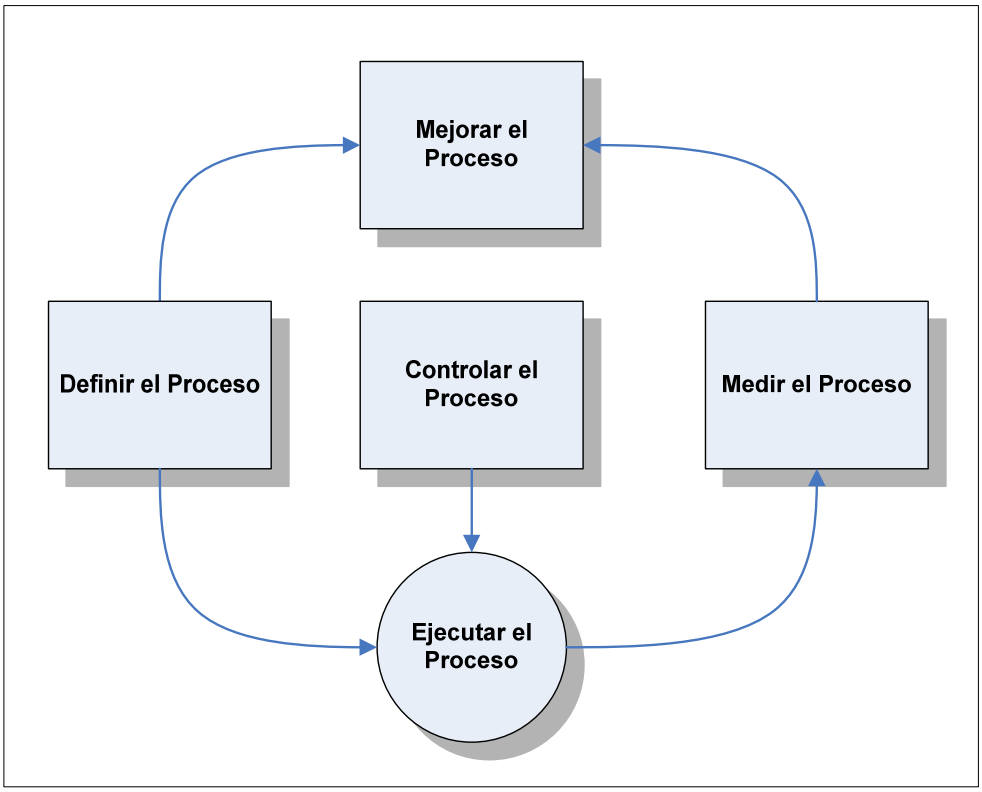
"*El proceso software define cómo se organiza, gestiona, mide, soporta y mejora el desarrollo, independientemente de las técnicas y métodos usados*" (Derniame et *al.*, 1999).

PROCESO SOFTWARE

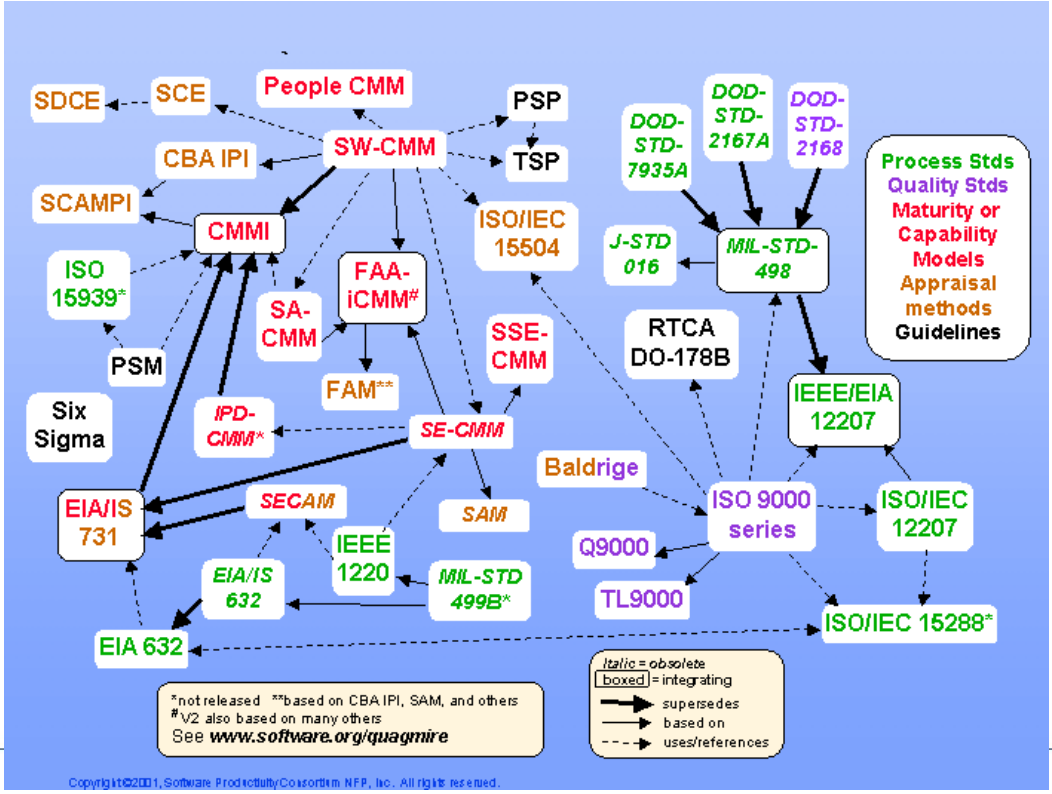
Naturaleza especial del proceso software

(Derniame et *al*., 1999)

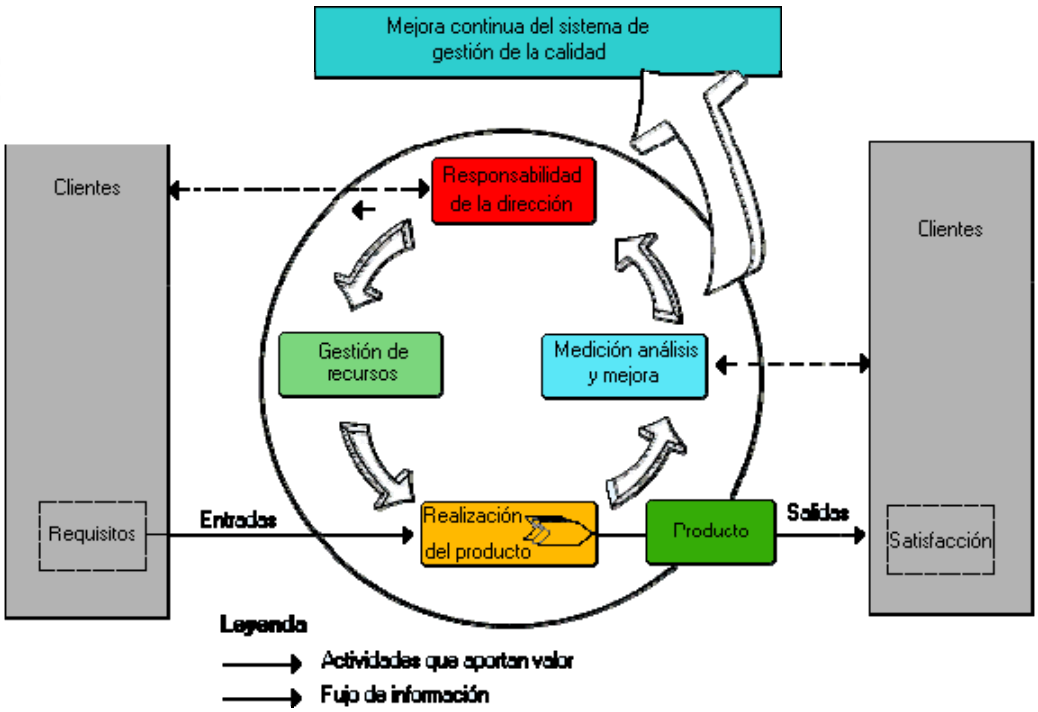
- Es complejo
- No es un proceso de producción típico
- Tampoco es un proceso de ingeniería "pura"
- No es (completamente) un proceso creativo
- Está basado en descubrimientos que dependen de la comunicación, coordinación y cooperación dentro de marcos de trabajo predefinidos



MODELOS DE CALIDAD Y MADUREZ



ISO 9000



SW-CMM

Nivel	Características	Resultados
Inicial	- Ausencia de gestión de proyectos. - El proceso de software es cambiante e irregular: - Los planes, estimaciones y calidad son impredecibles. - El rendimiento depende de la capacidad individual de los miembros del grupo. - Se establecen programas de formación del personal de desarrollo y mantenimiento.	Productividad y calidad escasa. Riesgo máximo
Repetible	- Los procesos de software son estables y repetibles. - La organización establece políticas de gerencia de proyectos y procesos. - La planificación se basa en proyectos similares. - Existen estándares definidos y exigidos. - El proceso se enmarca en un sistema de gerencia de proyectos basado en experiencias pasadas.	Productividad y calidad baja. Riesgo alto.

SW-CMM

Nivel	Características	Resultados
Definido	-Los procesos son definidos: estandarizados, documentados e institucionalizados. - Los procesos de ingeniería y gerencia son estables y se integran en uno sólo. - Existe un entendimiento común de los procesos, funciones y responsabilidades. - La organización mantiene un grupo dedicado a la definición, mejoramiento y difusión del proceso de Ingeniería de Software.	Productividad y calidad media. Riesgo medio.
Gestionado	- Los procesos son medibles o cuantificables - La productividad y la calidad se miden y registran para cada proyecto de la organización. - Se fijan metas cuantitativas de la calidad del software. -Mediante el uso de métricas de software, se crea una base cuantitativa para la evaluación y estimación en proyectos futuros.	Productividad y calidad alta. Riesgo mínimo.
Optimizando	- Los procesos se mejoran continuamente. - La organización busca lograr el nivel máximo de capacidad. - Se incorporan nuevas tecnologías y métodos para mejorar los procesos.	Productividad y calidad total. Riesgo nulo.

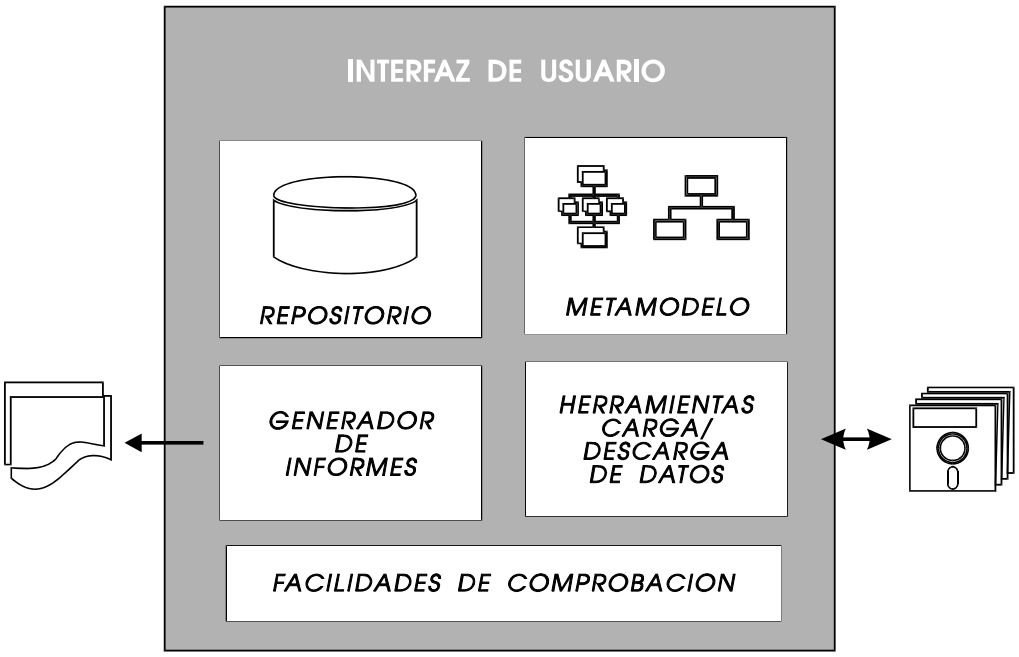
SGBD (De Miguel y Piattini, 1993)

1960	Primeros productos de bases de datos (DBOM, IMS, IDS, Total, IDMS, ...) Estándares Codasyl
1970	Modelo Relacional Prototipos SGBDR Trabajos teóricos relacionales Los tres niveles de la arquitectura (ANSI y Codasyl) Modelo E/R Primeros productos relacionales en el mercado
1980	Difusión de productos relacionales Bases de datos distribuidas **Estándares SQL** (ANSI, ISO) Manifiesto sobre Bases de Datos Orientadas a Objetos

SGBD (Piattini et al., 2006)

1990	Manifiesto sobre la tercera generación de Bases de Datos Arquitectura Cliente/Servidor (en dos capas) Primeros productos de Bases de Datos Objetos Modelos de Referencia (ISO/ANSI) SQL 92 Consorcio ODMG (Estándares OO) Almacenes de Datos SQL: 1999 (anteriormente, SQL3)
2000	Arquitectura Cliente/Servidor en tres capas Modelo Objeto-Relacional Bases de Datos multimedia Bases de Datos móviles SQL/MM Bases de datos XML SQL: 2003 Bases de datos grid

CASE (Piattini y Daryanani, 1995)



CASE (Piattini y Daryanani, 1995)

HERRAMIENTAS DE GESTION

- PLANIFICACION
- DIMENSIONAMIENTO
- SEGUIMIENTO

HERRAMIENTAS TECNICAS

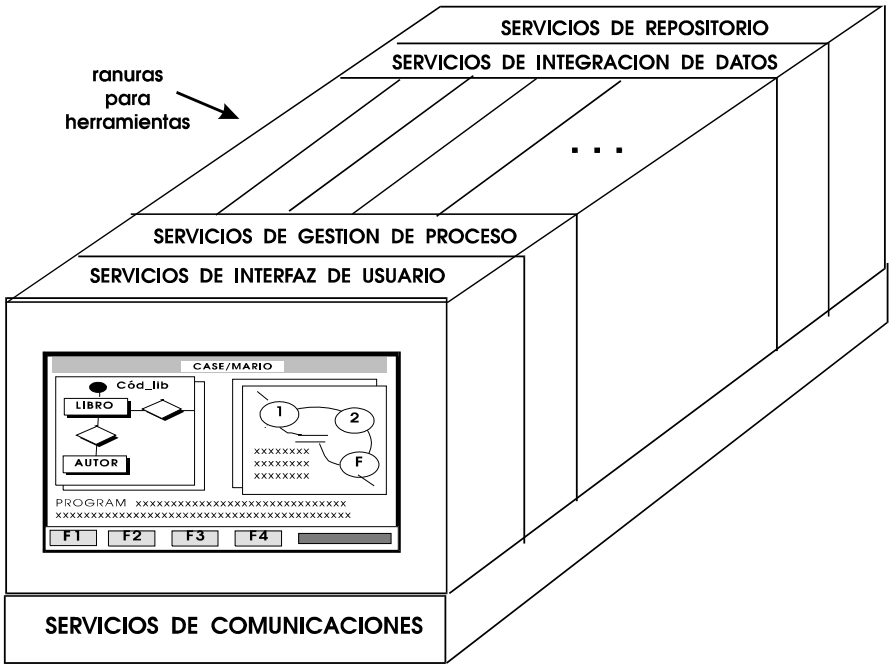
ANALISIS	DISEÑO	IMPLEMENTACION	PRUEBA	MANTENIMIENTO
CASE FRONTAL	CASE DORSAL	GENERADORES DE CODIGO	HERRAMIENTAS DE PRUEBA	HERRAMIENTAS DE MANTENIMIENTO
CASE INTEGRADO		LENGUAJES DE 4ª GENERACION		

HERRAMIENTAS DE SOPORTE

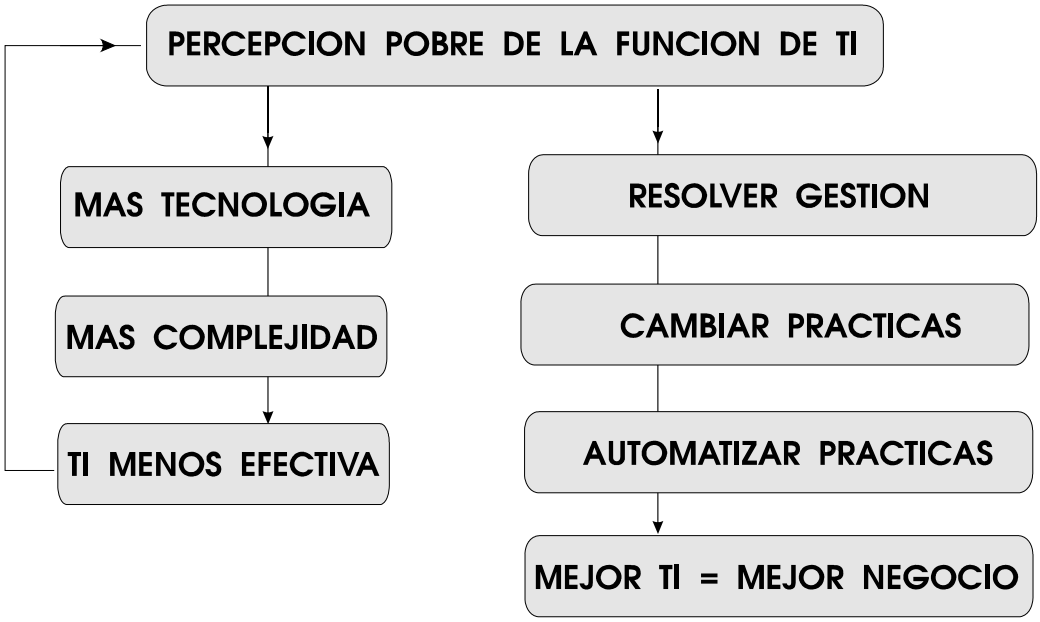
- SISTEMA DE REPOSITORIO/DICCIONARIO
- CONTROL DE CONFIGURACION
- SERVICIOS DE SEGURIDAD

CASE (Piattini y Daryanani, 1995)

CASE (Piattini y Daryanani, 1995)



CASE (Piattini y Daryanani, 1995)



ORIENTACIÓN A OBJETOS

Lenguajes

- Simula (1966)
- Smalltalk (1976)
- C++, Objective-C (1986)
- Eiffel (1988)
- Java (1990)
- C# (2001)

ORIENTACIÓN A OBJETOS

1986: "Object-Oriented Programming Workshop", por IBM en Yorktown Heights y la "First International Conference on Object-Oriented Programming Systems, Languages and Applications- OOPSLA"

- **Metodologías**
  - **BOOCH (1983) y (1986)**
  - **GOOD (General Object-Oriented Design)**
  - **SEIDEWITZ y STARK (1986)**
  - **BUHR (1984) y (1991)**
  - **EVB (1985)**
  - **HOOD (Hierarchical Object Oriented Design)**
  - **ESA (1989a) y (1989b)**
  - **SHLAER y MELLOR (1988) y (1990)**

- **INTRODUCCIÓN**
- **DÉCADA DE LOS 50**
- **DÉCADA DE LOS 60**
- **DÉCADA DE LOS 70**
- **DÉCADA DE LOS 80**
- **DÉCADA DE LOS 90**
- **DÉCADA DE LOS 2000**
- **DÉCADA DE LOS 2010**
- **CONCLUSIONES**

- Consolidación de la orientación a objetos
- Énfasis en el time-to-market (Ingeniería concurrente, gestión de riesgos)
- Reutilización
- Interacción-Persona-Computador
- Desarrollo de software libre
- Problemas del año 2000 y el Euro
- Consolidación de modelos y estándares

ORIENTACIÓN A OBJETOS

Metodologías

- BOOCH (1991)
- OMT, RUMBAUGH et al. (1991)
- Objectory/OOSE, JACOBSON et al. (1992)
- SOMA, GRAHAM (1993)

OMG OBJECT MANAGEMENT GROUP

UNIFIED MODELING LANGUAGE™

1997 UML1.0 y 1.1
1998 UML 1.4
. . .
2005 UML 2
. . .
2010 UML 2.3

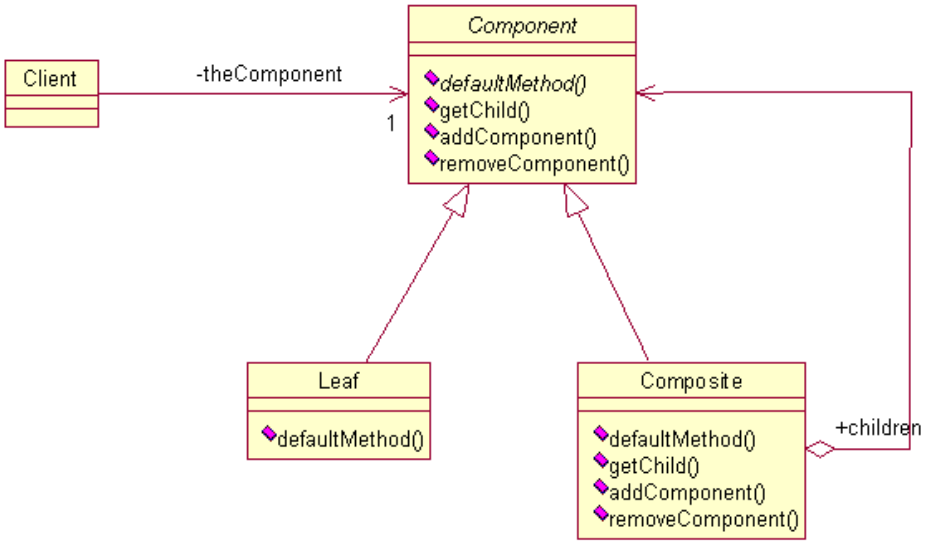
ORIENTACIÓN A OBJETOS

Garzás y Piattini (2006)

ORIENTACIÓN A OBJETOS

- Gamma et al. (1995) *Design patterns: Elements of Reusable Object Oriented Software*
- Buschmann et al. (1996) *A System of Patterns: Pattern-Oriented Software Architecture*
- Fowler (1996) *Analysis Patterns: Reusable Object Models*
- Riel (1996) *Object-Oriented Design Heuristics*
- Larman (1997) *Applying UML and Patterns*
- Rising (1998) *The Patterns Handbook*
- Fowler (2000) *Refactoring improving the design of existing code*
- . . .

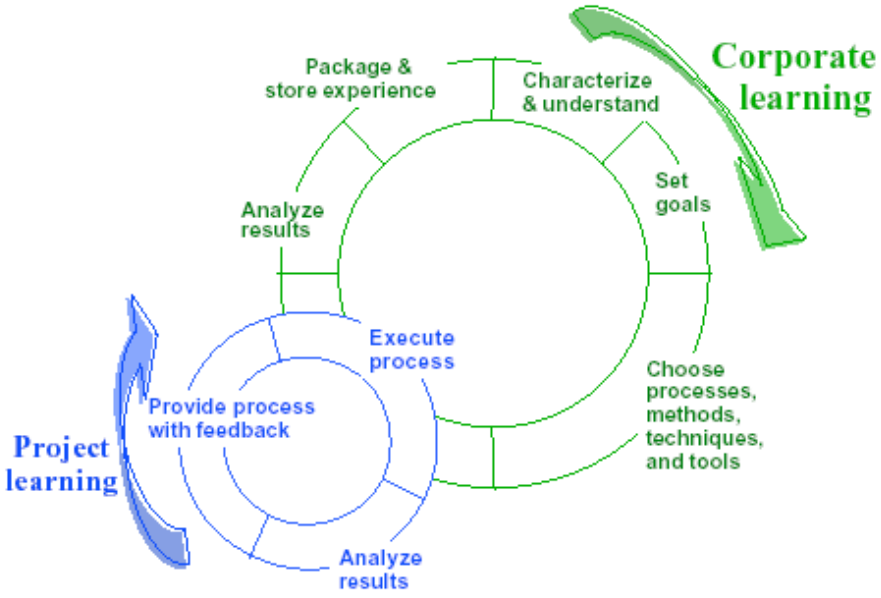
ORIENTACIÓN A OBJETOS



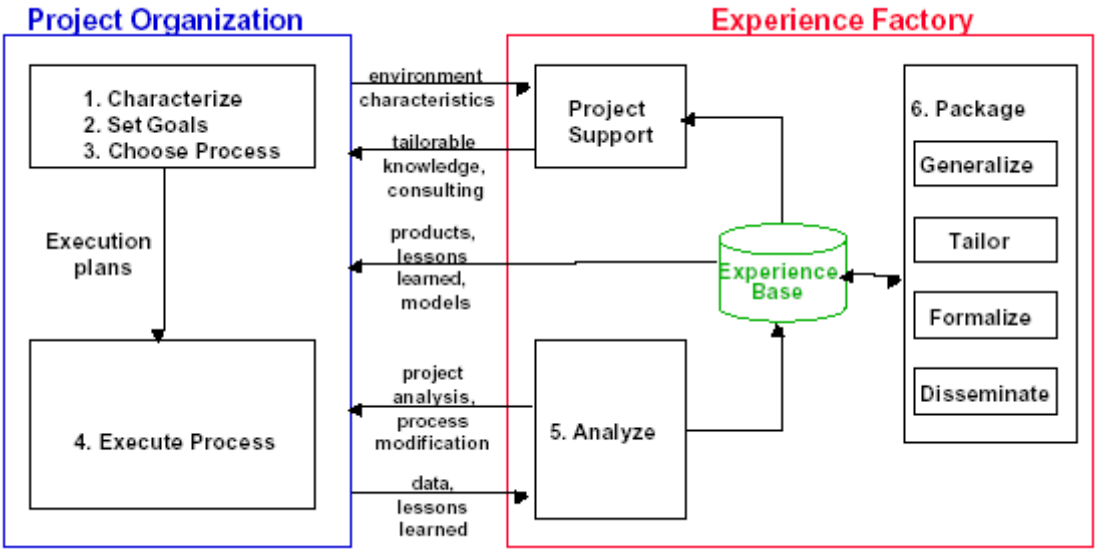
REUTILIZACIÓN

QIP (Quality Improvement Paradigm)

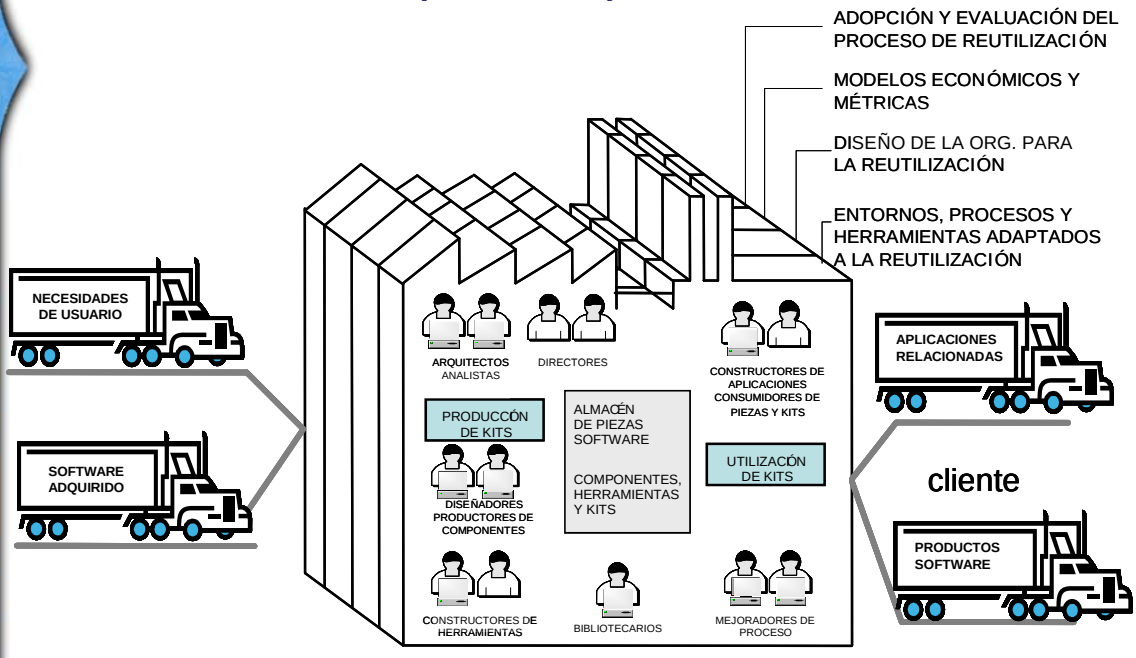
(Basili y Caldiera, 1995)



REUTILIZACIÓN



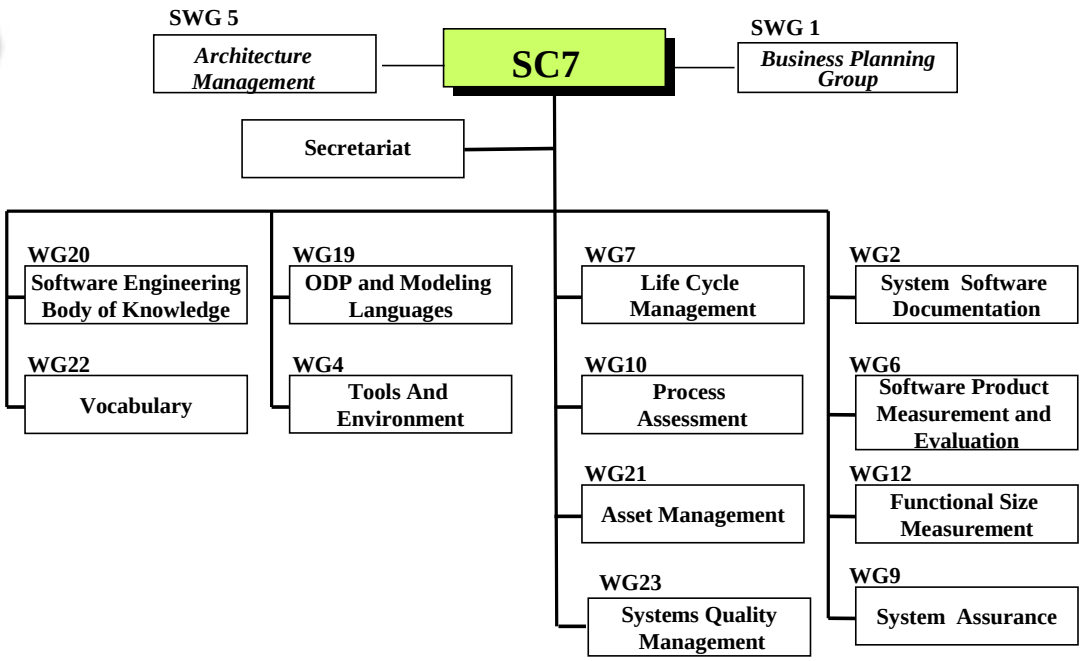
REUTILIZACIÓN
(Griss, 1993)



AÑO 2000 Y EURO

- Mantenimiento de software
- Problemas de evolución
- Mejora de las técnicas de prueba
- Fábricas de renovación de software
- Difusión del *outsourcing*

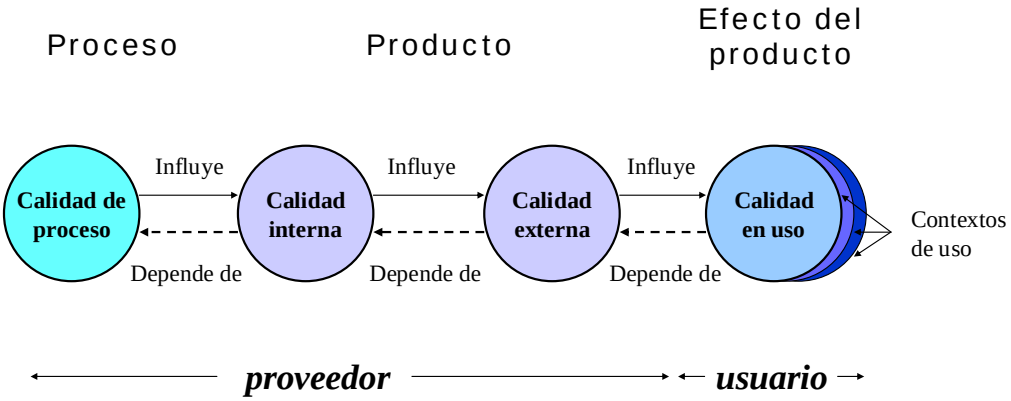
MODELOS Y ESTÁNDARES



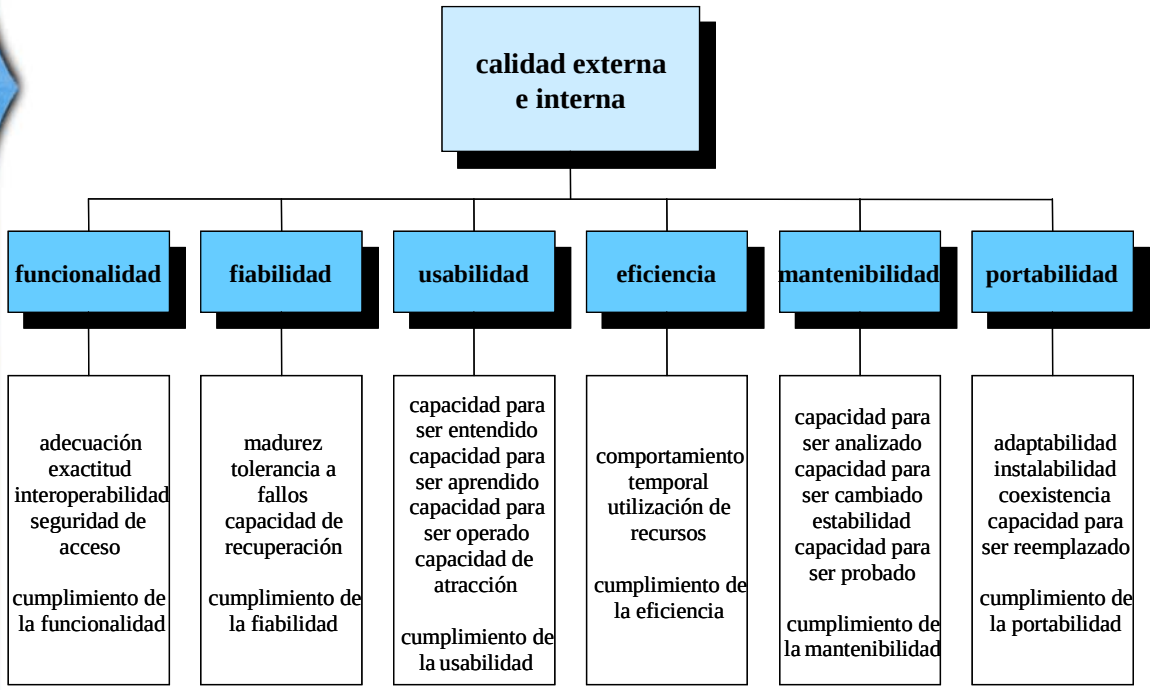
MODELOS Y ESTÁNDARES

- ISO 9000-3 (1997) -> ISO 90003 (2004)

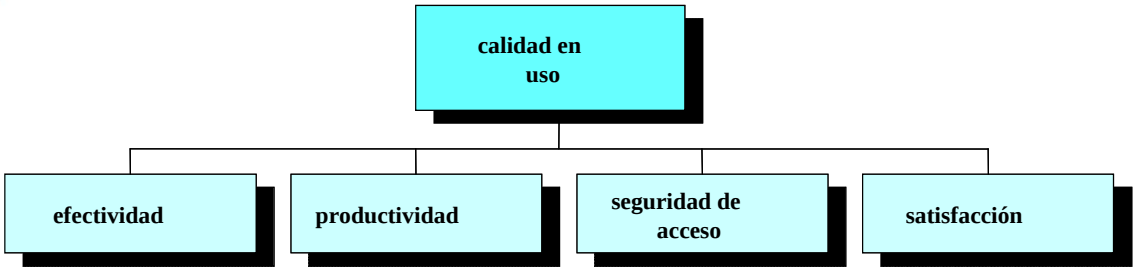
- ISO 9126 (1994) -> ISO 9126 (2001)



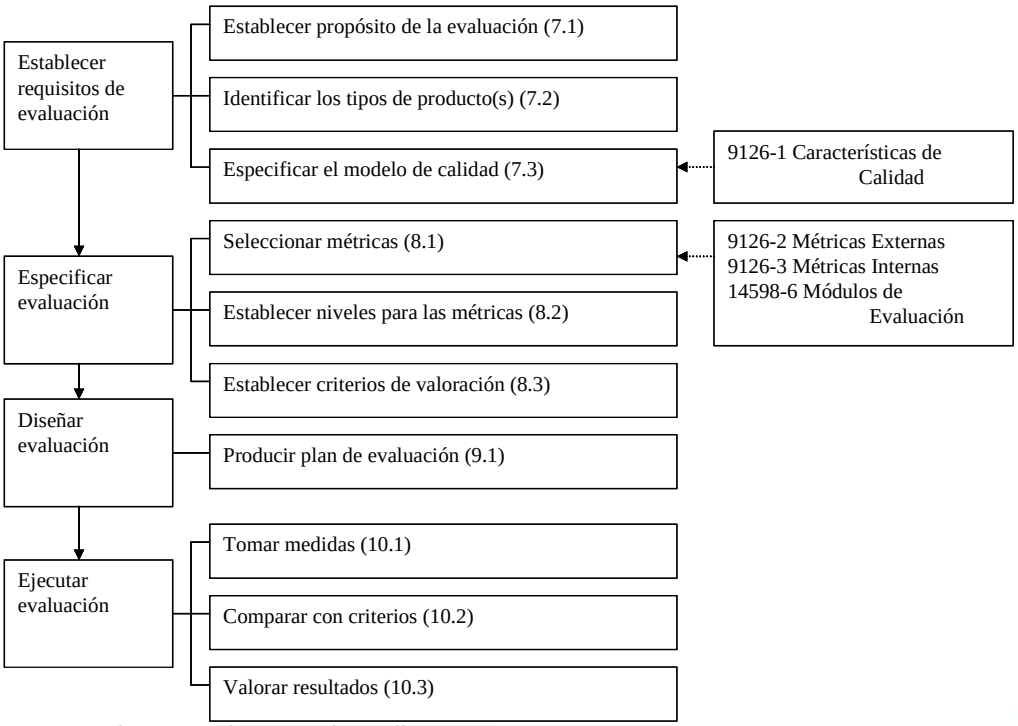
ISO 9126



ISO 9126



ISO 14598 (1999-2001)



IS 12207 (1995, 2002, 2004, 2008)

PROCESOS PRINCIPALES

- ADQUISICIÓN
- SUMINISTRO
- DESARROLLO
- EXPLOTACIÓN
- MANTENIMIENTO

PROC. ORGANIZACIONALES

- GESTIÓN
- INFRAESTRUCTURA
- MEJORA
- RECURSOS HUMANOS
- GESTIÓN DE ACTIVOS
- GEST. PROG. REUTILIZACIÓN
- INGENIERÍA DE DOMINIO

PROCESOS DE SOPORTE

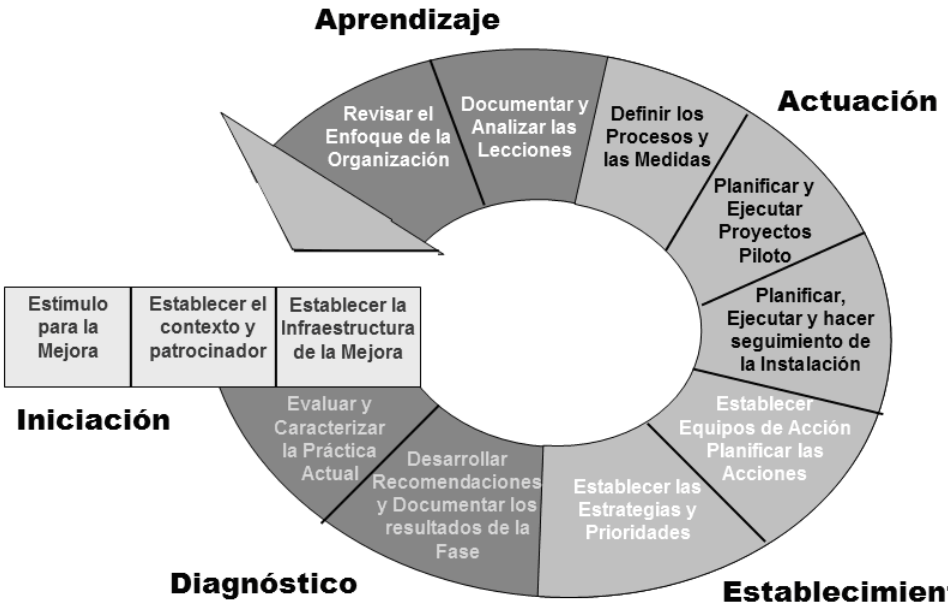
- DOCUMENTACIÓN
- GESTIÓN DE CONFIGURACIÓN
- ASEGURAMIENTO DE CALIDAD
- VERIFICACIÓN
- VALIDACIÓN
- REVISIÓN CONJUNTA
- AUDITORÍA
- RESOLUCIÓN DE PROBLEMAS
- USABILIDAD
- EVALUACIÓN DE PRODUCTO

PROCESO DE ADAPTACIÓN

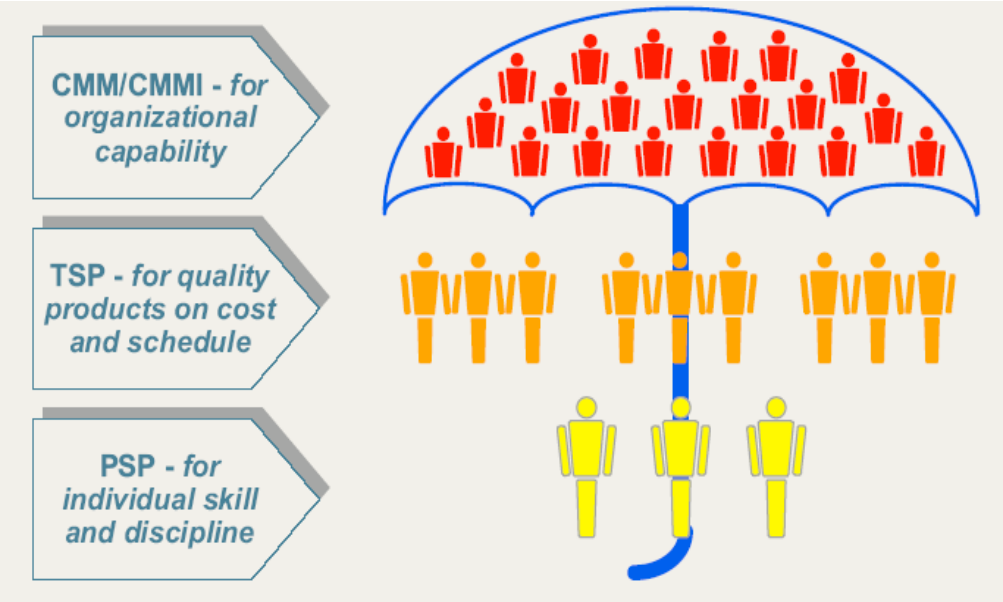
MODELOS Y ESTÁNDARES

MODELO IDEAL

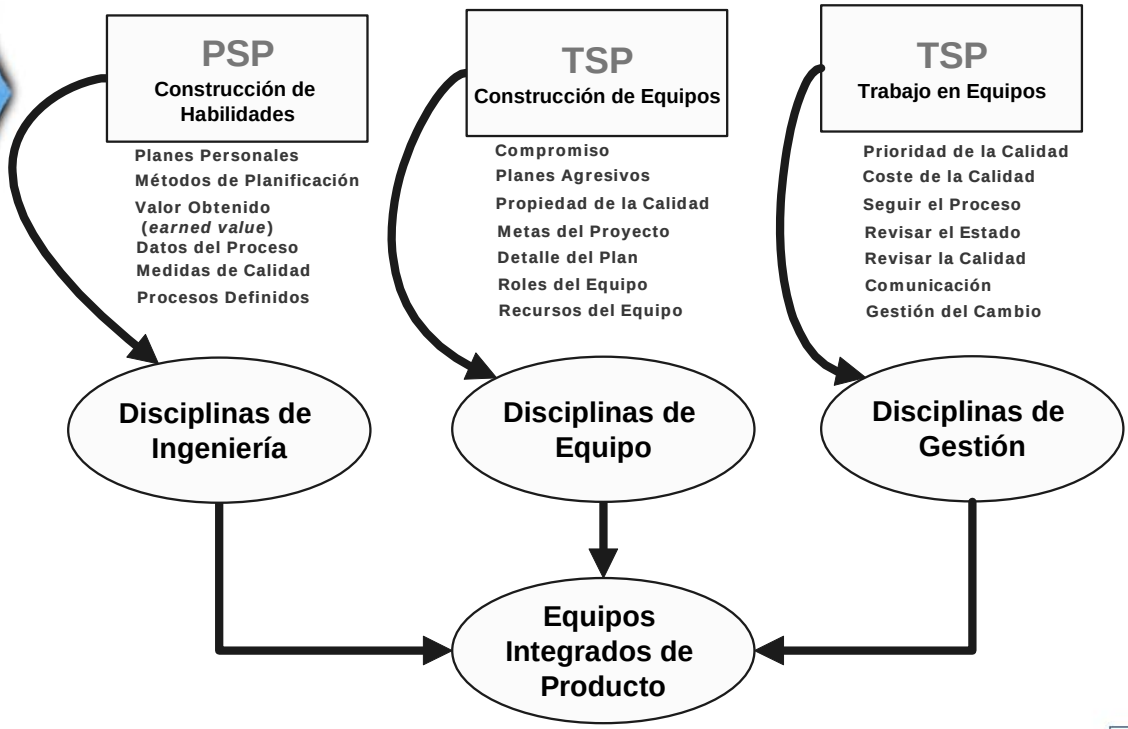
(Peterson, 1995)



MODELOS Y ESTÁNDARES



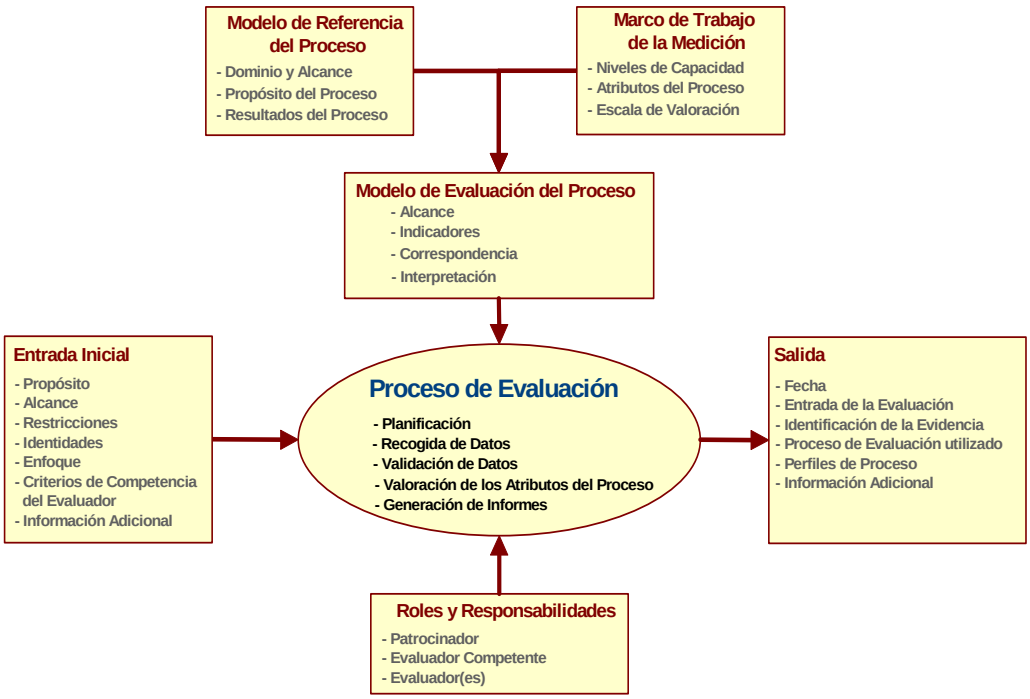
MODELOS Y ESTÁNDARES



MODELOS Y ESTÁNDARES
ISO 15504 (1998) -> ISO 15504 (2005-2010)

PARTES DE LA NORMA ISO/IEC 15504	CONTENIDO
1. Conceptos y Vocabulario	Proporciona una introducción general a los conceptos de la evaluación de los procesos y un glosario de términos relacionados.
2. Realización de la Evaluación	Establece los requisitos mínimos necesarios para realizar una evaluación que garantice la consistencia y repetibilidad de las valoraciones. Los requisitos ayudan a asegurar que la valoración de salida es consistente y proporciona la evidencia necesaria para corroborar los resultados y verificar su conformidad con los requisitos.
3. Guía para la Realización de la Evaluación	Proporciona una guía para interpretar los requisitos a la hora de realizar una evaluación.
4. Guía sobre el Uso para la Mejora del proceso y la Determinación de la Capacidad del Proceso	Identifica la Evaluación del proceso como una actividad que puede ser realizada como parte de una iniciativa de mejora de procesos o como parte de un enfoque de determinación de la capacidad. El propósito de la mejora de los procesos es mejorar de forma continua la eficiencia y efectividad de la organización. El objetivo de la determinación de la capacidad es identificar las fortalezas, debilidades y riesgos de los procesos seleccionados respecto a un requisito particular especificado a través de los procesos utilizados y de su alineamiento con las necesidades de negocio.
5. Un Ejemplo de Modelo de Evaluación de Procesos (en preparación)	Contiene un ejemplo de un modelo para realizar la evaluación de los procesos basados en el modelo de referencia de procesos definido en el estándar ISO/IEC 12207. Una evaluación se lleva a cabo utilizando un modelo de evaluación de procesos relacionado con uno o más modelos de referencia de procesos.

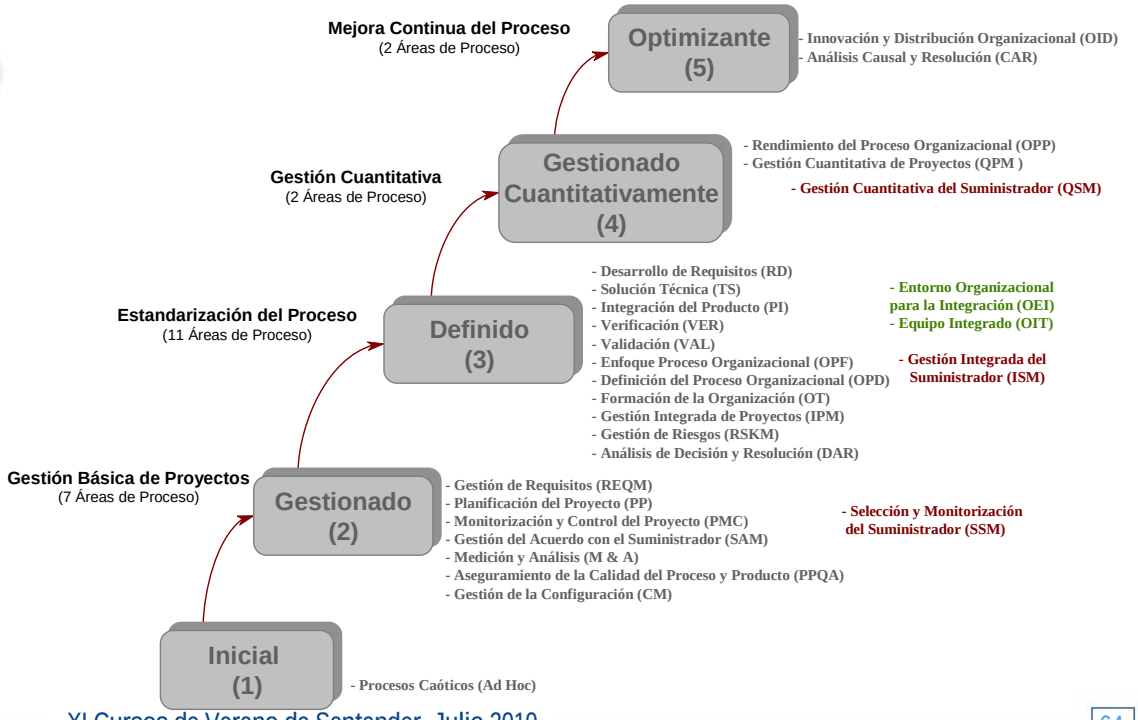
MODELOS Y ESTÁNDARES
ISO 15504 (1998) -> ISO 15504 (2005)



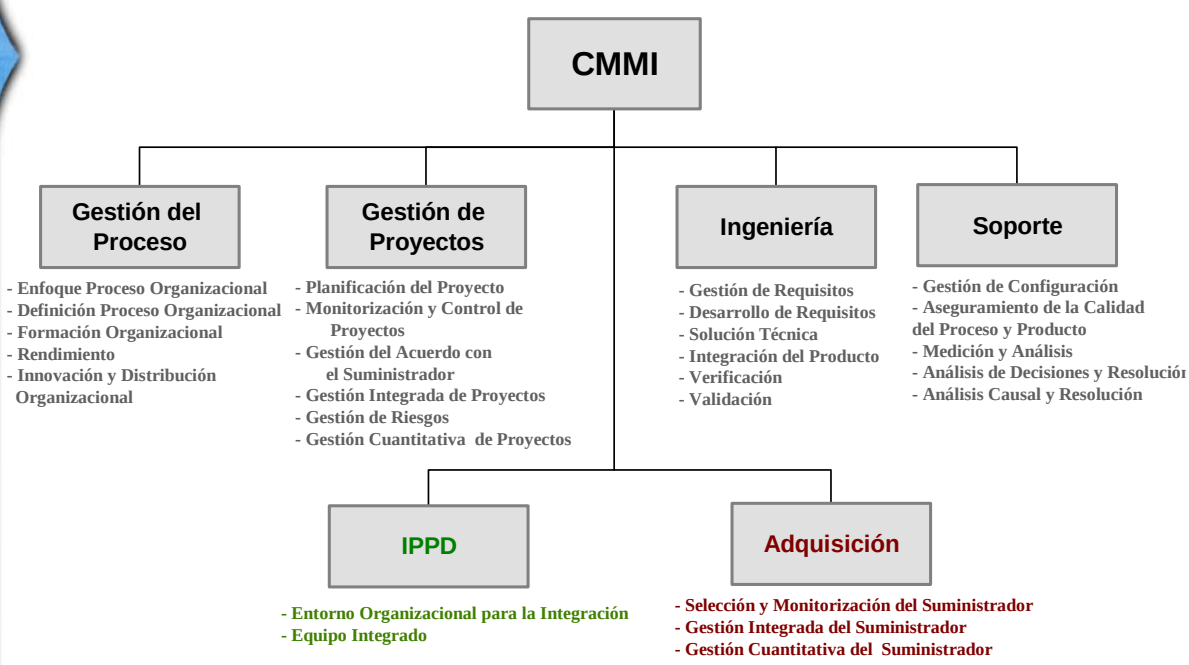
MODELOS Y ESTÁNDARES
CMMI

- Eliminar inconsistencias
- Reducir duplicaciones.
- Incrementar la claridad y comprensión
- Proporcionar terminología común
- Proporcionar estilos consistentes
- Establecer reglas de construcción uniformes
- Mantener componentes comunes
- Asegurar la consistencia con ISO 15504

MODELOS Y ESTÁNDARES
CMMI



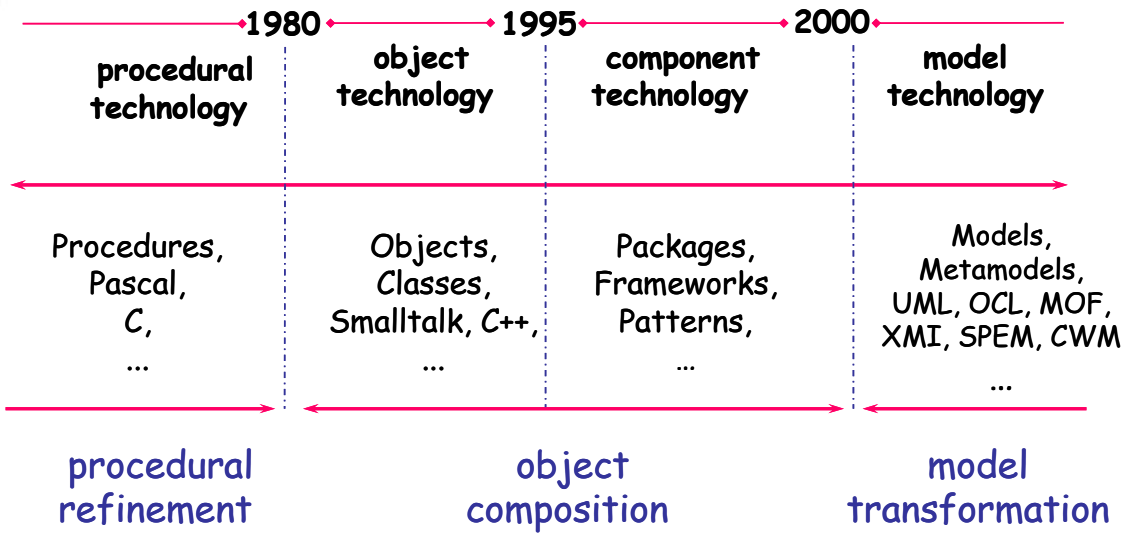
MODELOS Y ESTÁNDARES
CMMI



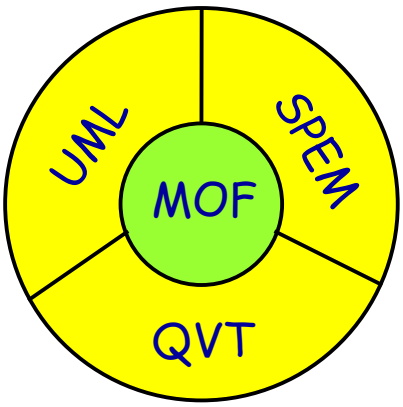
- INTRODUCCIÓN
- DÉCADA DE LOS 50
- DÉCADA DE LOS 60
- DÉCADA DE LOS 70
- DÉCADA DE LOS 80
- DÉCADA DE LOS 90
- **DÉCADA DE LOS 2000**
- DÉCADA DE LOS 2010
- CONCLUSIONES

- DESARROLLO DIRIGIDO POR MODELOS
- MÉTODOS ÁGILES/HÍBRIDOS
- ARQUITECTURA ORIENTADA A SERVICIOS
- LÍNEAS DE PRODUCTOS
- DESARROLLO GLOBAL DE SOFTWARE
- ING. DEL SW. BASADA EN VALOR

BEZIVIN (2006)

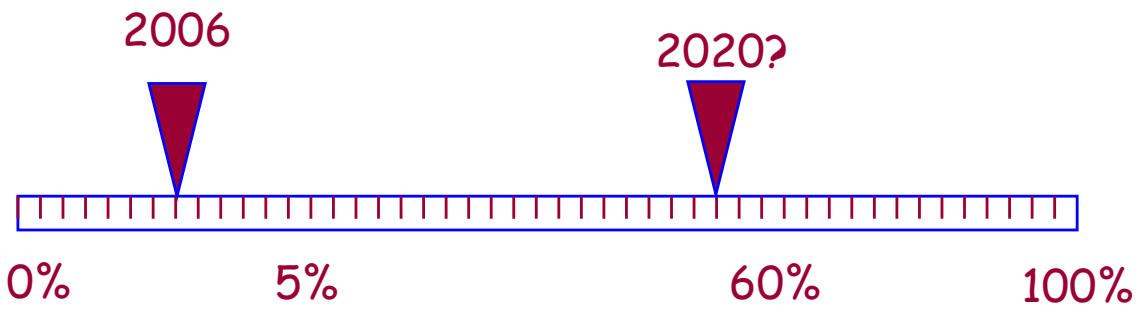


- MDA es MDD usando estándares OMG
  - MOF
    - Meta Object Facility
  - UML
    - Unified Modeling Language
  - OCL
    - Object Constraint Language
  - XMI
    - Metadata Interchange
  - MOF QVT
    - Query/View/Transformation
  - SPEM
    - Software Process Engineering Metamodel



BEZIVIN (2006)

Grado de cumplimiento de las promesas de MDE



MÉTODOS ÁGILES

- eXtreme Programming, Beck (1996)
- DSDM, Stapleton (1997)
- Scrum, Schwaber y Beedle (2001)
- Cristal, Cockburn (2001)
- . . .

Manifesto for Agile Software Development
(2001)

http://www.agile-spain.com

VALORES ÁGILES

- Valorar más a los individuos y su interacción que a los procesos y las herramientas

- Valorar más el software que funciona que la documentación exhaustiva

- Valorar más la colaboración con el cliente que la negociación contractual

- Valorar más la respuesta al cambio que el seguimiento de un plan

PRINCIPIOS ÁGILES

- Nuestra mayor prioridad es satisfacer al cliente a través de la entrega temprana y continua de software con valor.
- Aceptamos requisitos cambiantes, incluso en etapas avanzadas.
- Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos software frecuentemente, con una periodicidad desde un par de semanas a un par de meses, con preferencia por los periodos más cortos posibles.
- Los responsables de negocio y los desarrolladores deben trabajar juntos diariamente a lo largo del proyecto.
- Construimos proyectos con profesionales motivados. Dándoles el entorno y soporte que necesitan, y confiando en ellos para que realicen el trabajo.
- El método más eficiente y efectivo de comunicar la información a un equipo de desarrollo y entre los miembros del mismo es la conversación cara a cara.

PRINCIPIOS ÁGILES

- Software que funciona es la principal medida de progreso.
- Los procesos ágiles promueven el desarrollo sostenible.
- Patrocinadores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y los buenos diseños mejoran la agilidad.
- Simplicidad, el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños surgen de equipos que se autoorganizan.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo, entonces mejora y ajusta su comportamiento de acuerdo a sus conclusiones.

PRÁCTICAS XP

http://www.programacionextrema.org

Retroalimentación a escala fina

- DesarrolloGuiadoPorPruebas (TestDrivenDevelopment)
- JuegoPlanificacion
- OnsiteCustomer
- ProgramacionEnPares

Proceso contínuo en lugar de por lotes

- IntegracionContinua
- RefactorarSinPiedad
- LiberacionPequena

Entendimiento compartido

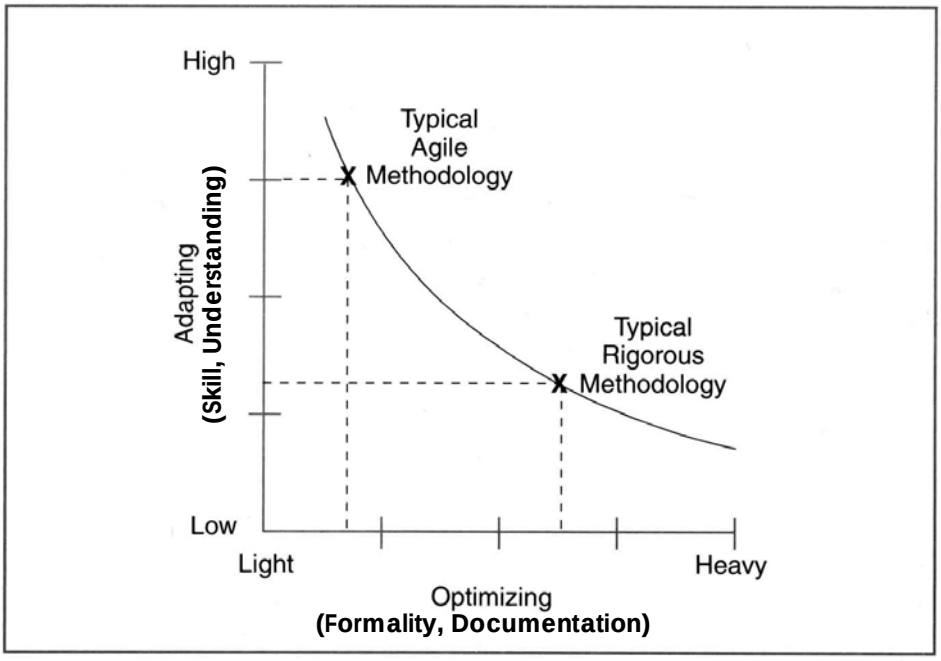
- DisenoSimple
- SystemMetaphor
- PropiedadColectivaCodigo
- ConvencionesCodigo

Bienestar del programador

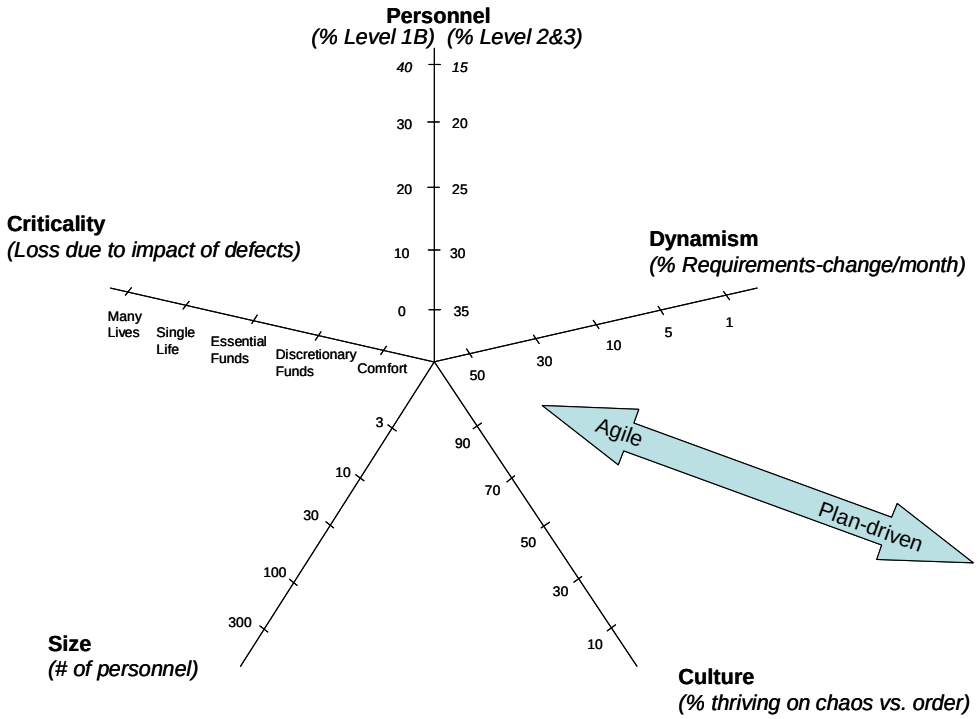
- PasoSostenible (SemanaCuarentaHoras)

MÉTODOS HÍBRIDOS

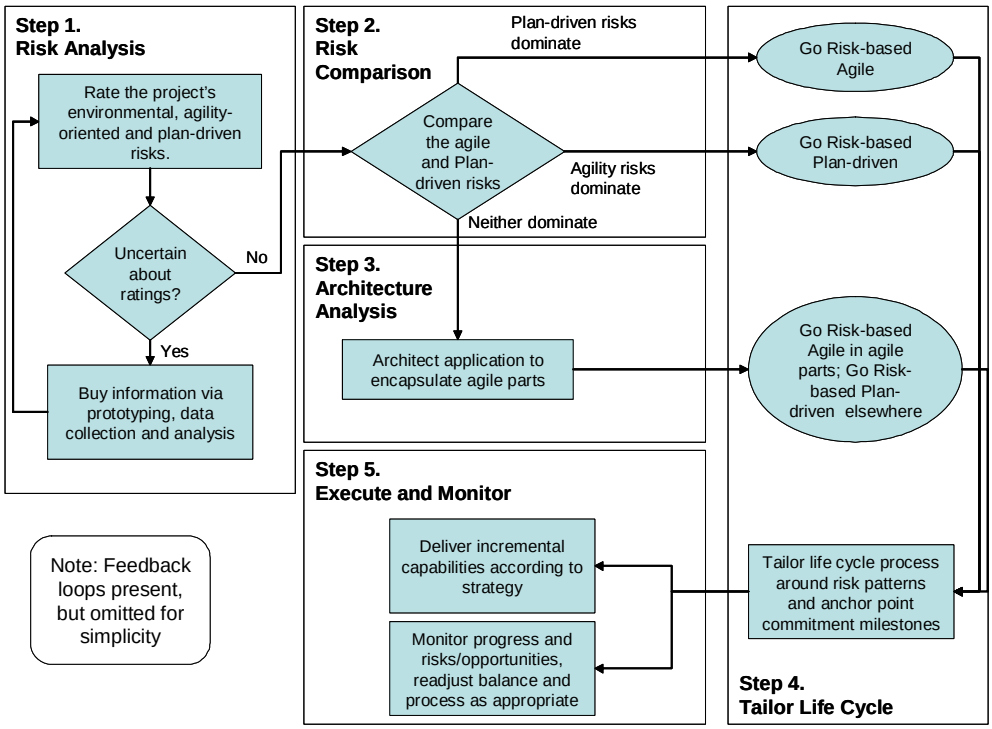
Boehm (2005)



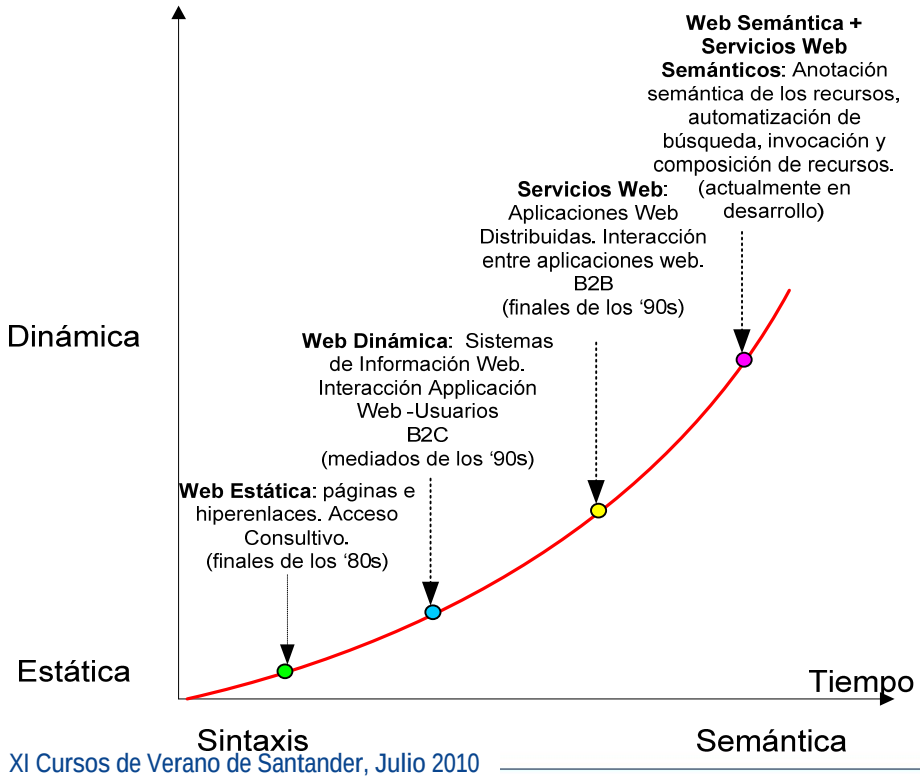
MÉTODOS HÍBRIDOS



MÉTODOS HÍBRIDOS



ARQUITECTURA ORIENTADA A SERVICIOS

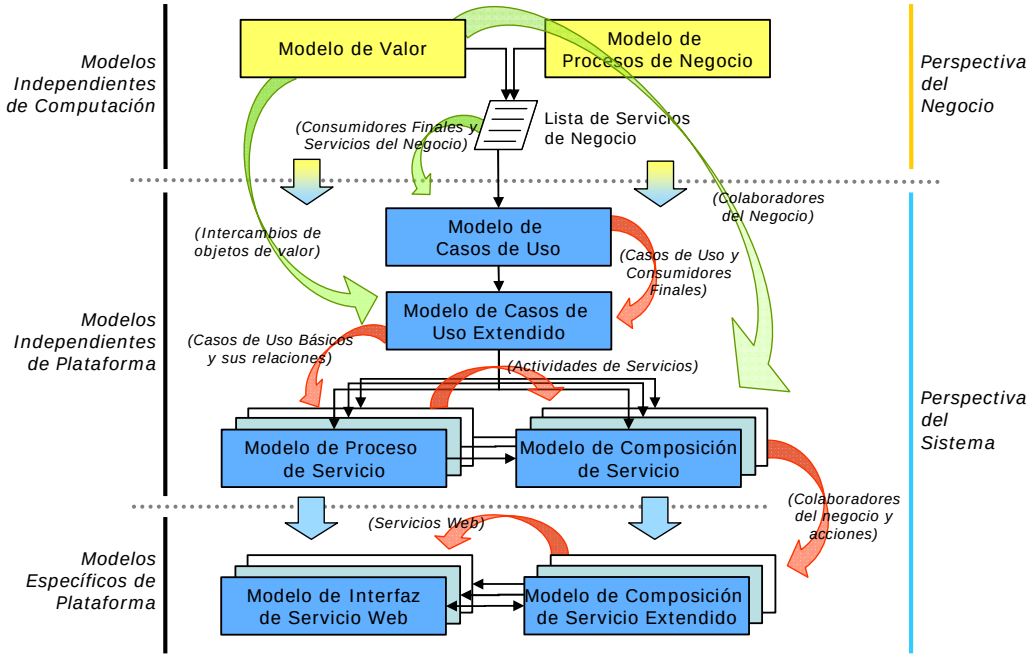


ARQUITECTURA ORIENTADA A SERVICIOS

Necesidades		Tecnología asociada	
		Nombre	Descripción
Intercambio de Información	Lenguaje estándar de comunicación	XML (eXtensible Markup Language)	Lenguaje extensible de etiquetas utilizado para el intercambio de información.
	Protocolo para el intercambio de mensajes	SOAP (Simple Object Access Protocol)	Protocolo de intercambio de mensajes.
Implementación de servicios Web	Descripción de servicios Web	WSDL (Web Services Description Language)	Lenguaje basado en XML para la descripción de la interfase de un servicios Web
	Descubrimiento y Localización de los servicios Web	UDDI (Universal Description, Discovery, and Integration)	Repositorio en el que se puede almacenar y consultar diversos servicios Web disponibles

ARQUITECTURA ORIENTADA A SERVICIOS

DE CASTRO ET AL. (2007)



LÍNEAS DE PRODUCTOS

(Clemens y Northrop, 2002)

“ un conjunto de sistemas software, que comparten un conjunto común de características (*features*), las cuales satisfacen las necesidades específicas de un dominio o segmento particular de mercado, y que se desarrollan a partir de un sistema común de activos base (*core assets*) de una manera preestablecida”.

LÍNEAS DE PRODUCTOS

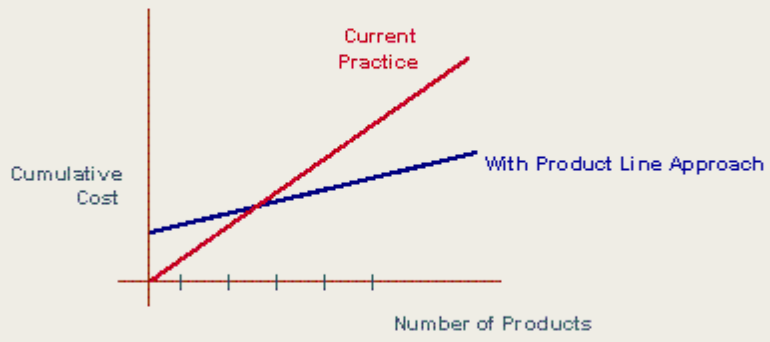
(Clemens y Northrop, 2002)

- Conseguir ganancias de productividad a gran escala
- Mejorar el *time-to-market*
- Mantener la presencia en el mercado
- Sostener un crecimiento sin precedentes
- Mejorar la calidad del producto
- Aumentar la satisfacción del usuario
- Lograr objetivos de reutilización
- Permitir la personalización masiva
- Compensar la falta de capacidad de contratar ingenieros software

LÍNEAS DE PRODUCTOS

Carnegie Mellon
Software Engineering Institute

Economics of Product Lines



Weiss, D.M. & and Lai, C.T.R..
Software Product-Line Engineering:
A Family-Based Software Development Process
Reading, MA: Addison-Wesley, 1999.

LÍNEAS DE PRODUCTOS

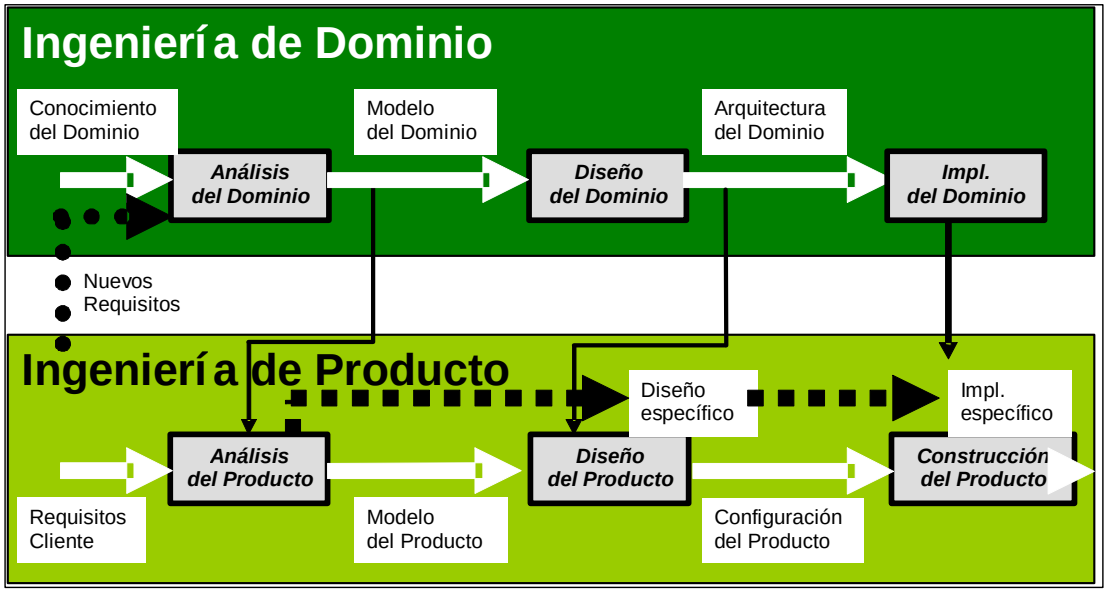
(Van der Linden, 2002)

Project topics

	Business	Architecture	Process	Organization
ARES		Dealing with variation Architecture description Resource management qualities	Recovery from legacy systems	
Praise		Domain-specific architectures Family architecture Development tools	Family development practices Variability and commonality Traceability between assets Architectural decisions	
Esaps	Scoping	Domain analysis Aspect analysis Family requirements Architecture glossary Commonality and variability Reference architecture Platform and components	Architecture assessment Architecture recovery Domain analysis Aspect analysis Family development process frameworks Requirements modeling and traceability Change management Evolution support Variant configuration and derivation	Platform and component development
Café	Business and market analysis Scoping Family development transition and adoption	Requirements engineering Heterogeneous platforms COTS use Design for quality Development tool support Test modeling Validation	Asset management Traceability Change management and impact analysis Family transition and adoption Configuration and version management Product derivation Family evolution Test strategy and methodology Validation	Asset management Validation and testing Product-line transition and adoption Change management Configuration and version management Product derivation

LÍNEAS DE PRODUCTOS

DÍAZ Y TRUJILLO (2007)

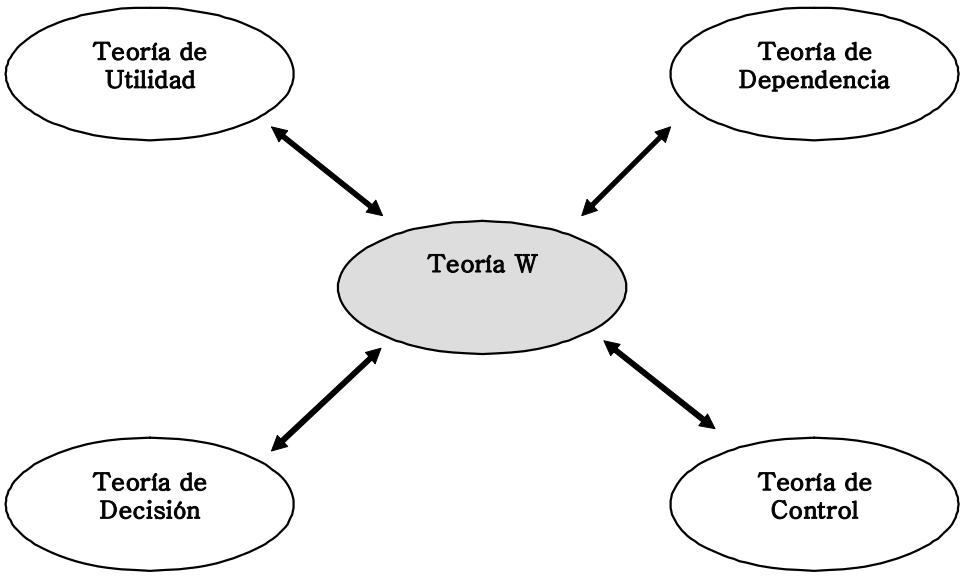


INGENIERÍA DE SW BASADA EN VALOR

(Biffl et al., 2005)

- Análisis del beneficio-realización.
- Extracción y conciliación del valor de cada implicado.
- Análisis del caso de negocio.
- Gestión continua de riesgos y oportunidades.
- Ingeniería del software y de sistemas de manera concurrente.
- Monitorización y control basado en valor.
- El cambio como oportunidad.

INGENIERÍA DE SW BASADA EN VALOR



INGENIERÍA DE SW BASADA EN VALOR

(Boehm, 2005)

- Ingeniería de requisitos basada en valor
- Arquitectura, diseño y desarrollo basado en valor
- Validación y verificación basada en valor
- Control y planificación basada en valor
- Gestión de riesgos basada en valor
- Gestión de la calidad basada en valor
- Gestión de personal basada en valor

DESARROLLO GLOBAL DE SOFTWARE

(Piattini et al., 2007)

	Misma Organización	Diferente Organización
Mismo lugar	Desarrollo co-localizado	Desarrollo co-localizado con subcontratación
Mismo país	Desarrollo distribuido (DSD)	Desarrollo distribuido (DSD) con subcontratación
Otro país	Deslocalización Desarrollo global (GSD)	Subcontratación deslocalizada Desarrollo global (GSD)

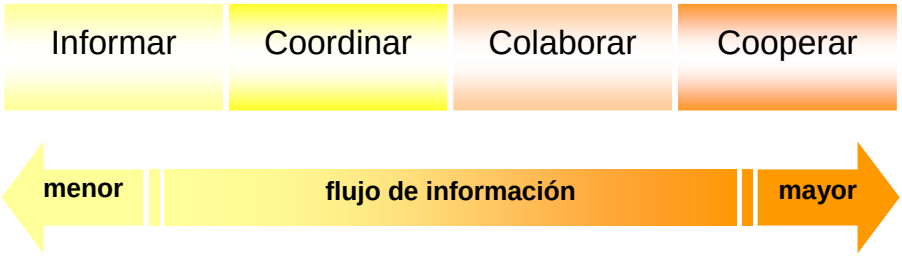
DESARROLLO GLOBAL DE SOFTWARE

(Piattini et al., 2007)

BENEFICIOS

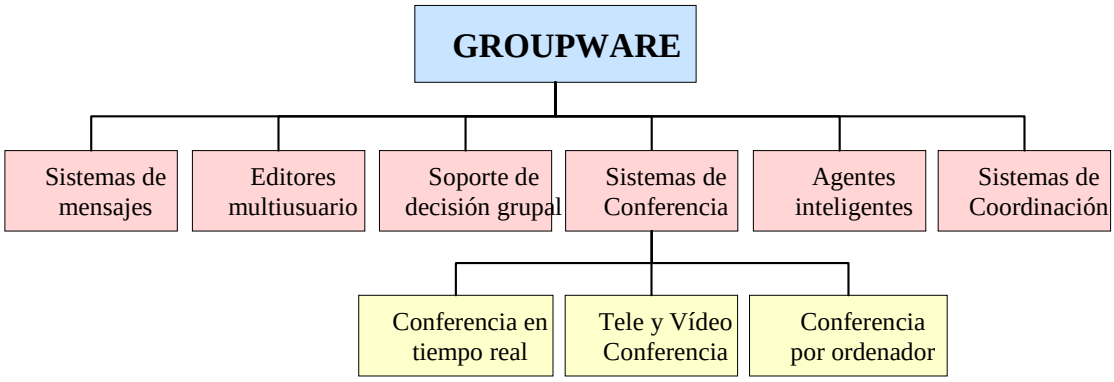
- Aprovechar la diferencia horaria entre los distintos sitios para lograr jornadas laborales más largas y conseguir mayor productividad
- Minimizar los costes de desarrollo
- Localizar a los desarrolladores más cerca del cliente
- Diversidad de experiencias, conocimiento técnico y destrezas de los stakeholders distribuidos

DESARROLLO GLOBAL DE SOFTWARE



	Mismo Momento	Diferentes Momentos
Mismo lugar	Interacción cara a cara	Interacción asincrónica
Distinto Lugar	Interacción sincrónica distribuida	Interacción asincrónica distribuida

DESARROLLO GLOBAL DE SOFTWARE



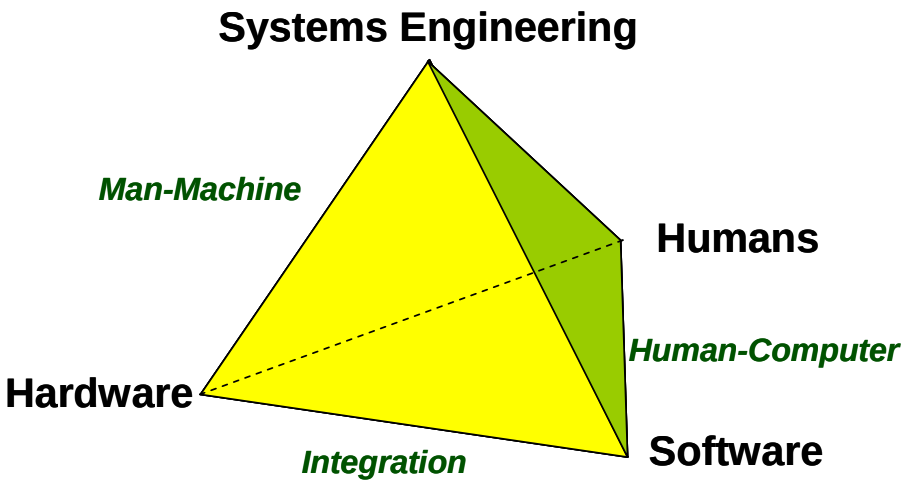
- INTRODUCCIÓN
- DÉCADA DE LOS 50
- DÉCADA DE LOS 60
- DÉCADA DE LOS 70
- DÉCADA DE LOS 80
- DÉCADA DE LOS 90
- DÉCADA DE LOS 2000
- **DÉCADA DE LOS 2010**
- CONCLUSIONES

- **SISTEMAS DE SISTEMAS**
- **COMPLECIÓN COMPUTACIONAL**
- **AUTONOMÍA**
- **BIOCOMPUTACIÓN**

SISTEMAS DE SISTEMAS SOFTWARE (SoS)

CARACTERÍSTICAS	RANGO VALORES
Tamaño	10-100 MLOC
Interfaces externas	30-300
Proveedores “coopetitivos”	20-200
Jerarquía de proveedores	6-12 niveles
Grupos de coordinación	20-200

BOEHM (2006)



- **COMPLECIÓN COMPUTACIONAL**
  - *Smart dust, smart paint, …*
  - Redes de sensores
- **AUTONOMÍA**
  - Agentes inteligentes cooperativos
  - Software reconfigurable
  - Aprendizaje automático
  - Robots a escala nano
- **BIOCOMPUTACIÓN**

"Pienso que hay mercado en el mundo como para unos cinco ordenadores". Thomas J. Watson, Presidente de IBM, 1948.

"*Los Macintosh usan un dispositivo apuntador llamado "ratón". No hay razón alguna para que la gente quiera usar esas cosas*". John C. Dvorak, 1984

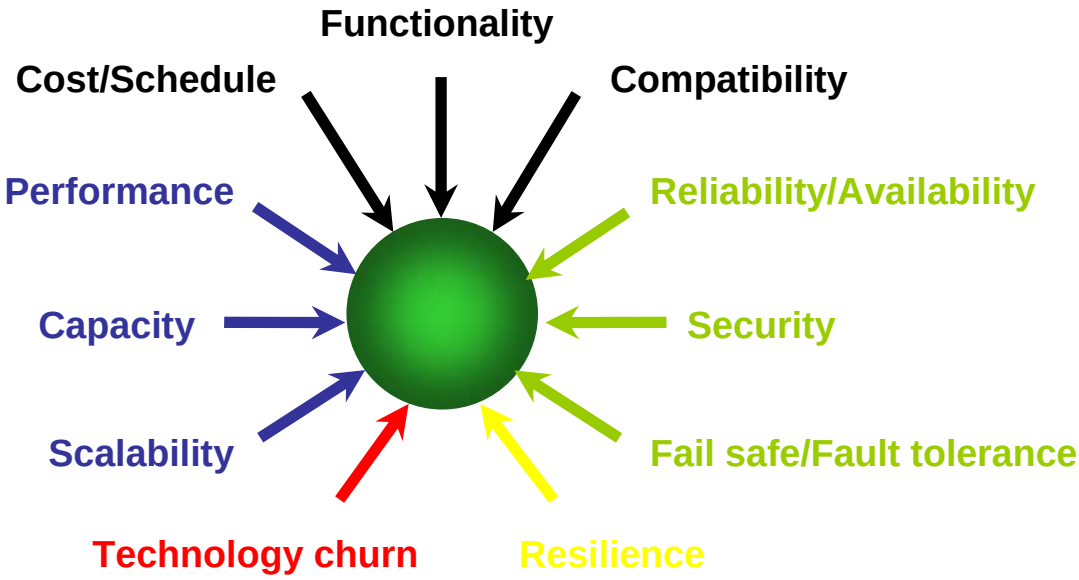
"*El problema de los virus es pasajero. En un par de años estará resuelto*". John McAfee, 1988

"*En dos años el problema del spam se habrá resuelto*". Bill Gates, 2004

- INTRODUCCIÓN
- DÉCADA DE LOS 50
- DÉCADA DE LOS 60
- DÉCADA DE LOS 70
- DÉCADA DE LOS 80
- DÉCADA DE LOS 90
- DÉCADA DE LOS 2000
- DÉCADA DE LOS 2010
- CONCLUSIONES

CONCLUSIONES

- *"El desarrollo de software ha sido, es y probablemente será fundamentalmente difícil".* Booch (2007)



CONCLUSIONES

La "Lemmingeniería del Software" (Davis, 1993) ha creado confusión y decepción en muchos usuarios y profesionales del software.

- DEMANDA CRECIENTE DE CERTIFICACIONES PARA PROCESOS Y PRODUCTOS
- LA CALIDAD SE ENCUENTRA COMO TEMA PRINCIPAL DE LAS AGENDAS ESTRATÉGICAS DE INVESTIGACIÓN
- NECESIDAD DE REDUCIR LA BRECHA ENTRE TEORÍA Y PRÁCTICA
- FORMACIÓN DE PROFESIONALES
- RELEVANTE PAPEL QUE PUEDE JUGAR ESPAÑA EN LA FABRICACIÓN DE SOFTWARE

EVOLUCIÓN DE LA FABRICACIÓN DE SOFTWARE: HACIA LA CALIDAD

"El arte nunca progresa, evoluciona"
Raúl Soldi

"Las tecnologías de desarrollo de software progresan, la Ingeniería del Software evoluciona"
Mario Piattini