

# **CALIDAD EN EL DESARROLLO DE SOFTWARE**

**Guillermo Pantaleo**



# **CALIDAD EN EL DESARROLLO DE SOFTWARE**

---

**Guillermo Pantaleo**



# **CALIDAD EN EL DESARROLLO DE SOFTWARE**

---

**Guillermo Pantaleo**



Pantaleo, Guillermo  
Calidad en el desarrollo de software. - 1a ed. - Buenos Aires  
Alfaomega Grupo Editor Argentino, 2011.  
208 p. ; 23x17 cm.  
ISBN 978-987-1609-23-9  
1. Informática. 2. Programación. I. Título  
CDD 005.3

Calidad en el desarrollo de software  
Pantaleo, Guillermo  
Queda prohibida la reproducción total o parcial de esta obra, su tratamiento informático y/o la transmisión por cualquier otra forma o medio sin autorización escrita de Alfaomega Grupo Editor Argentino S.A.

Revisión de estilo: Romina Yael Ryzenberg  
Diagramación: Melina S. Daffunchio

Internet: <http://www.alfaomega.com.mx>

Todos los derechos reservados © 2011, por Alfaomega Grupo Editor Argentino S.A.  
Paraguay 1307, PB, oficina 11

ISBN: 978-987-1609-23-9

Queda hecho el depósito que prevé la ley 11723

**NOTA IMPORTANTE:** la información contenida en esta obra tiene un fin exclusivamente didáctico y, por lo tanto, no está previsto su aprovechamiento a nivel profesional o industrial. Las indicaciones técnicas y programas incluidos han sido elaborados con gran cuidado por el autor y reproducidos bajo estrictas normas de control.

Alfaomega Grupo Editor Argentino S.A. no será jurídicamente responsable por: errores u omisiones; daños y perjuicios que se pudieran atribuir al uso de la información comprendida en este libro, ni por la utilización indebida que pudiera dársele.

Los nombres comerciales que aparecen en este libro son marcas registradas de sus propietarios y se mencionan únicamente con fines didácticos, por lo que Alfaomega Grupo Editor Argentino S.A. no asume ninguna responsabilidad por el uso que se dé a esta información, ya que no infringe ningún derecho de registro de marca. Los datos de ejemplos y pantallas son ficticios, a no ser que se especifique lo contrario. Los hipervínculos a los que se hace referencia no necesariamente son administrados por la editorial, por lo que no somos responsables de sus contenidos o de su disponibilidad en línea.

Empresas del grupo:  
Argentina: Alfaomega Grupo Editor Argentino, S.A.  
Paraguay 1307 P.B. "11", Buenos Aires, Argentina, C.P. 1057  
Tel.: (54-11) 4811-7183 / 0887 - E-mail: [ventas@alfaomegaedaitor.com.ar](mailto:ventas@alfaomegaedaitor.com.ar)

México: Alfaomega Grupo Editor, S.A. de C.V.  
Pitágoras 1139, Col. Del Valle, México, D.F., México, C.P. 03100  
Tel.: (52-55) 5575-5022 - Fax: (52-55) 5575-2420 / 2490. Sin costo: 01-800-020-4396  
E-mail: [atencionalcliente@alfaomega.com.mx](mailto:atencionalcliente@alfaomega.com.mx)

Colombia: Alfaomega Colombiana S.A.  
Carrera 15 No. 64 A 29, Bogotá, Colombia  
PBX (57-1) 2100122 - Fax: (57-1) 6068648 - E-mail: [scliente@alfaomega.com.co](mailto:scliente@alfaomega.com.co)

Chile: Alfaomega Grupo Editor, S.A.  
Dr. La Sierra 1437, Providencia, Santiago, Chile  
Tel.: (56-2) 235-4248 - Fax: (56-2) 235-5786 - E-mail: [agechile@alfaomega.cl](mailto:agechile@alfaomega.cl)

## **Agradecimientos**

*Gracias a mi familia por apoyarme siempre en mi postura de trabajar sólo en los proyectos que me gustan, los que me hacen crecer como profesional y que permiten expresarme con honestidad.*

*Gracias a mis alumnos por obligarme a mantenerme siempre actualizado y despierto para responder consultas e inquietudes de futuros profesionales que no se conforman con respuestas superficiales.*

*Gracias a mis colegas, clientes y empleadores porque de ellos aprendí lo que se debe y a veces lo que no se debe hacer en las más diversas situaciones.*

*Gracias a mi amigo Juan D. Rogers por las innumerables y acaloradas charlas, de allí aprendí los conceptos de administración del cambio en los procesos de mejoras.*

*Gracias a la Ingeniera Patricia Forradellas por el tiempo compartido en it-Mentor, empresa que fundamos juntos en la que analizamos y dicutimos muchos de los conceptos expuestos en este libro.*

*Gracias a Guillermo Rugillo, Diego Montaldo y Patricia Forradellas por las revisiones de los borradores y los muchos comentarios útiles que contribuyeron a que las ideas expuestas en este texto tengan más claridad y hayan sido expresadas de forma más inteligente.*

*Gracias a mis alumnos Maia Naftali, Israel Diego Lalo y Germán Castignani por el aporte de los casos de estudio de empresas a mejorar presentados en este libro.*

*Gracias a los responsables de la editorial por la confianza dispensada y la apuesta a esta publicación.*

*Gracias al editor ya que con sus revisiones y análisis hizo de mis escritos un libro.*



## **El autor**

### **INGENIERO GUILLERMO PANTALEO**

El autor, Guillermo Pantaleo, es ingeniero en Telecomunicaciones recibido en la Universidad Nacional de La Plata, Argentina. Su experiencia de treinta años en el desarrollo de software comenzó en el Instituto de Ingeniería Biomédica de la UBA (Universidad de Buenos Aires) haciendo docencia e investigación en procesamiento digital de señales. Participó en numerosos seminarios, congresos y es autor de publicaciones científicas nacionales e internacionales. Fue miembro del Comité Organizador de la Primera y Segunda Reunión de Trabajo en Procesamiento de la Información e integró el Centro de Informática, Computadoras y Procesamiento de la Información de la UBA donde trabajó en el desarrollo de un sistema autónomo de procesamiento de señales de perfiles de pozos petroleros.

Como programador independiente desarrolló software para el mercado de instrumentación y telefonía. Fue contratado por empresas nacionales e internacionales como programador senior y formador de recursos humanos. Participó en proyectos de desarrollo en las áreas de telecomunicaciones y finanzas.

Como líder de proyectos y miembro del SEPG (Software Engineering Process Group) participó en el proceso de certificación CMM3 de un software factory.

En la actualidad, como socio fundador de it-Mentor, empresa dedicada a la capacitación y acompañamiento para el desarrollo de software, diseña y desarrolla procesos de mejoras orientados a actualizaciones tecnológicas y a certificaciones CMMI e ITIL. Se desempeña como docente en cursos de actualización y especialización en temas relacionados a procesos de desarrollo, relevamiento e ingeniería de requerimientos así como análisis y diseño de software. Es profesor de las materias: Técnicas de Diseño, Arquitectura de Software y Calidad en el Desarrollo de Sistemas en la Facultad de Ingeniería de la UBA (Universidad de Buenos Aires).



## Prólogo

La economía moderna está completamente permeada por sistemas de software. Una proporción cada vez mayor de los productos industriales contiene algún elemento electrónico gobernado por un programa de complejidad cada vez mayor: desde los sistemas administrativos, de infraestructura, pasando por vehículos, dispositivos de consumo y hasta los artefactos para el hogar. El papel del software en toda la gama de bienes y servicios de la economía crece sin detenerse, y con este crecimiento viene aparejada la importancia de la calidad del software, la que también se brindará a todos esos bienes y servicios en los que se encuentra instalado. Muchos usuarios de computadoras reconocerán la experiencia de la pantalla azul, con un mensaje críptico que señala un error incomprensible, seguida de la pérdida de su trabajo. Cada vez más artefactos y procesos están expuestos a circunstancias parecidas. Si ocurre en un sistema de control de un automóvil, por ejemplo, el resultado puede ser que este detenga su marcha o, peor, que tome una marcha incontrolable. En resumidas cuentas, debido al papel creciente del software las demandas sobre su calidad también han aumentado.

La complejidad del software y del proceso de su producción agrega urgencia a este tema porque los factores que inciden en su calidad son muchos y muy variados. El software generalmente se produce en el contexto de una organización poblada de especialistas en toda una gama de actividades técnicas, administrativas y comerciales. La especialización de los participantes a menudo desafía la intuición y convierte a la gestión de calidad de software en una tarea muy difícil sin procedimientos fácilmente convertibles en rutinas de acción. Tal como se desprende claramente de esta obra, hay una combinación de factores técnicos y no técnicos que afectan la calidad del software que no está comprendida en la descripción de tareas de un grupo especialista en particular. Por este motivo, no se puede agrupar la mayor parte de la responsabilidad del control de calidad a una división de la organización, ya que está diseminada y difundida por toda la operación sin límites claros en las responsabilidades de sus distintos miembros.

El desafío de liderazgo en la gestión de organizaciones productoras de software es muy complejo pues muchos proyectos de gran envergadura tienen suficientes diferencias entre sí que demandan una reorientación de la organización para encarar cada uno si desea mantener los más altos niveles de calidad en forma sostenida. Requiere no sólo vigilancia constante para lograrlo, sino también una capacidad de

aprendizaje y una adaptación continua para superar y aprender de los inevitables errores que ocurren en proyectos de gran complejidad.

Una manera clásica de enfrentar la complejidad de una tarea es aplicar la división del trabajo. Se trata de dividirla en pedazos más pequeños, supuestamente más simples, y delegar cada parte a distintos ejecutores. Si bien se aplican principios de la división del trabajo en la producción de software, como son los sistemas modulares, el diseño apropiado de arquitecturas de sistemas, entre otros, la producción de software no admite una compartamentalización muy profunda de la gestión de calidad. Un principio conocido como la “ley de Conway” en la producción de software señala que la dinámica de comunicación de los equipos entre los que se dividieron las tareas del sistema se trasladará casi directamente a la operación con sus beneficios y sus fallas. Lo más importante es destacar que muy a menudo el fenómeno pasa completamente desapercibido por los diseñadores, gestores y desarrolladores.

En este contexto frecuentemente se generan situaciones conflictivas dentro de las organizaciones productoras de software pues la cadena de interdependencias de los sistemas propaga los efectos de los problemas en forma impredecible y sus fuentes son difíciles de aislar cuando provienen de combinaciones de factores distribuidos. Aun cuando el diagnóstico de la operación de una organización sea recibido en forma consensuada, la implementación de un proceso de mejoras necesario para elevar los niveles de calidad del software producido se vive como una profunda crisis organizacional que requiere de una altísima destreza de gestión. Las mejoras de proceso casi siempre conllevan cambios en la organización. La gestión de cambio organizacional es la misión más difícil para la cual la gran mayoría de los gestores no está preparada.

Toda la economía moderna en su orientación reciente al valor del conocimiento en todas las actividades ha acrecentado la necesidad de la atención a los detalles para mantener la competitividad de los productos en mercados cada vez más exigentes. El caso del software no sólo no es una excepción sino que está en el centro de esta dinámica económica global. Como dijimos antes, de la calidad del software depende la calidad de una gama virtualmente omnipresente de otros productos y procesos. A esto se agrega la dificultad especial del problema de la calidad de software y no puede evitarse la conclusión de su importancia crítica.

Esta obra de mi colega y amigo, el Ingeniero Guillermo Pantaleo, es una contribución clave en este contexto. Sin duda tendrá un efecto educativo para estudiantes, principiantes y experimentados, pero también un efecto indirecto económico: es el único camino hacia un negocio de software con rentabilidad sustentable.

Juan D. Rogers  
Georgia Institute of Technology

## Prefacio

Este libro fue concebido como una guía de estudio para los estudiantes de las disciplinas Ingeniería Informática y Análisis de Sistemas, y una referencia de consulta para los profesionales que desarrollan sus actividades desempeñando roles asociados a tareas de gestión y técnicas en proyectos de desarrollo de software. Los estudiantes encontrarán un medio de relacionar conocimientos adquiridos a veces como entidades aisladas cuando en realidad forman parte de un todo en cuyas relaciones se fundamenta su comprensión. Los profesionales encontrarán respuesta a muchos de los problemas que se les presentan a diario, un análisis de estos y una propuesta de solución. El objetivo es cubrir aspectos del desarrollo de software que son claves y en los cuales se debe trabajar bien a efectos de garantizar la calidad de los productos generados en los proyectos de desarrollo.

Este manual fue escrito pensando en los obstáculos que se presentan a la hora de llevar adelante un proyecto de desarrollo. Busca las causas de dichos problemas y guía al lector en la búsqueda de las soluciones a los problemas mencionados a partir de la formación de criterios elaborados en función de la experiencia que como desarrollador de software recogí a lo largo de 30 años en los escenarios más diversos. Al igual que las materias a mi cargo en la Facultad, el sesgo que presenta este libro es aportar soluciones a los diferentes problemas a partir del análisis de cómo estos se presentan. Por esta razón cuando se presentan los temas asumo que el lector los conoce por haber estudiado y trabajado. También que, como a todos nosotros, se le presentan dificultades para garantizar la calidad de los productos de sus proyectos mientras desarrolla las tareas de su especialidad. Para enfocarnos en este escenario, este no podría ser un libro en el cual teoricemos acerca del significado de algunas de las palabras claves del tema, sino que analizaremos cada tema en un contexto teórico introductorio y estableceremos criterios y un conjunto de buenas prácticas como solución. Hay excelentes libros que tratan los conceptos teóricos, los que recomendaré a lo largo de los distintos capítulos. También hay muy buenas referencias para cada uno de los temas de gestión y técnicos que asumo, como dije, conocidos por los lectores y que referenciaré en todos los párrafos donde trate los temas relacionados.

Las fuentes de los problemas claves y críticos que atentan contra la calidad en el desarrollo de software se encuentran en el trabajo realizado con los requerimientos, en la gestión de los proyectos, en la forma de trabajo utilizada para realizar el diseño, la codificación y las pruebas y en la estructura y dinámica de la organización. Por lo tanto nos ocuparemos de ellos poniendo énfasis en las cuestiones esenciales que hacen al deterioro de la calidad. Analizaremos los aspectos de cada uno de estos temas y cuáles son los puntos críticos a tener en cuenta.

No trataremos cómo escribir casos de uso, sino para qué escribirlos, quién los escribe, para qué serán utilizados y de qué manera. Sí trataremos acerca de los atributos de calidad que los casos de uso deben contar.

No trataremos cada uno de los temas asociados a la gestión de los proyectos, sino aquellos en los que los líderes fallan y después de analizar estos errores elaboraremos una guía acerca de cómo conducir los diferentes aspectos del desarrollo. Trataremos la falta de estrategias para la gestión de proyectos y recomendaremos formas de resolver esta falencia.

Presentaremos una forma de trabajo para llevar adelante las tareas técnicas de los proyectos, conocida como Integración Continua, la cual es propuesta como la solución al problema de contar en cada momento del proyecto con un producto que muestre con precisión y realismo lo realizado hasta entonces.

Todas y cada una de las soluciones propuestas necesitan del contexto apropiado para ser llevadas adelante. Por esta razón analizamos las empresas cuyo negocio es el desarrollo de software desde la perspectiva de una organización que debe generar las condiciones para las soluciones propuestas. En estos temas hemos incluido aspectos relacionados a la mejora de procesos, liderazgo y conocimiento organizacional.

Como puede verse de esta presentación resumida, éste no es un libro dedicado solo a los roles de gestión, tampoco está orientado a los roles técnicos. Este libro está dirigido a todos los involucrados en el proceso de desarrollo de software que participan en un proyecto llevando adelante las diferentes tareas. Mucha gente no entiende un punto fundamental, éste es un negocio en el cual se debe trabajar en forma interdisciplinaria. Cuesta mucho esfuerzo que la gente lo entienda y mucho más que se mantenga convencida a lo largo del tiempo. Por esta razón incluimos una sección donde analizamos la forma de trabajo para mantener este equilibrio inestable. Analizamos metodologías conducidas por los planes y ágiles, los aspectos positivos y negativos de las mismas y el impacto de su selección.

Este es un libro dedicado a todos los roles, programadores, analistas, miembros de QA (Quality Assurance), gerentes y clientes. Puede parecer demasiado ambicioso este objetivo, sin embargo los mayores problemas que deterioran la calidad en este negocio se deben a que estos roles consideran que poseen actividades tan específicas que no pueden compartir con el resto y que por ello se hace casi necesario trabajar cada uno en su hábitat. Esta falta de comunicación y ausencia de criterios comunes son las que se busca resolver a partir de una visión unificada y compartida. Los temas serán presentados de la forma que lo fueron hecho a lo largo de años de docencia en diferentes materias relacionadas al desarrollo de software y son el resultado de mi presentación como docente y del análisis, discusión y revisión. Además son el producto de conclusiones elaboradas con mis alumnos y de conclusiones elaboradas a partir de mi experiencia y están presentadas de manera abierta y sin prejuicio ninguno. Es seguro que los promotores de modelos de tipo CMMI nos critiquen por falta de formalidad y los adeptos a las metodologías ágiles lo hagan porque nos vean burocráticos. Bienvenidas todas las opiniones, nos ayudarán a aprender y quizás a mejorar nuestros razonamientos para futuras exposiciones.

El autor

## Contenido

<b>Agradecimientos .....</b>	5	mejoras .....	50
<b>Autor .....</b>	7	<b>3.2   Trabajando en los cambios .....</b>	53
<b>Prólogo .....</b>	9	3.2.1 Forma de trabajo.....	53
<b>Prefacio.....</b>	11	3.2.1.1 <i>Modelo ideal</i> .....	54
<hr/>			
<b>Capítulo 1 - Calidad en el software</b>			
<b>1.1   Introducción .....</b>	19	3.2.1.2 <i>Modelo eoalg</i> .....	55
<b>1.2   Calidad en el software.....</b>	19	3.2.2 Dos fenómenos espontáneos....	55
1.2.1 Evolución histórica .....	19	3.2.2.1 <i>Desconcierto</i> .....	55
1.2.2 Lecciones no aprendidas .....	23	3.2.2.2 <i>Procesos virtuales</i> .....	57
<b>1.3  Calidad versus velocidad de desarrollo.</b> .....	24	<b>3.3   Aspectos y factores del proceso de mejora .....</b>	57
1.3.1 Estándares.....	24	3.3.1 Direcciones del cambio organizacional en sus múltiples dimensiones.....	58
1.3.2 Creatividad.....	24	3.3.2 Aspecto socio-cultural de la gestión del cambio.....	58
1.3.3 Madurez .....	25	3.3.3 Factores críticos y de riesgo .....	59
<b>1.4   Modelos de calidad de software</b> .....	25	3.3.4 Factores generales de éxito .....	60
1.4.1 Surgimiento y evolución .....	25	3.3.5 Factores de éxito en pymes .....	62
1.4.2 Modelos .....	26	3.3.6 Factores adicionales .....	62
1.4.3 Certificaciones y evaluaciones .....	32	3.3.7 Recomendaciones .....	63
<b>1.5  Conclusión.....</b>	32	<b>3.4   Gestión del cambio .....</b>	63
<hr/>			
<b>Capítulo 2 - Causas que deterioran la calidad en el software</b>			
<b>2.1   Introducción .....</b>	35	3.4.1 Oposición al cambio.....	63
2.1.1 Definiciones .....	35	3.4.1.1 <i>Razones de resistencia al cambio</i> .....	63
2.1.1.1 <i>Calidad de producto</i> .....	35	3.4.1.2 <i>Rescatar lo positivo de la resistencia</i> .....	64
2.1.1.2 <i>Calidad de proceso</i> .....	35	3.4.2 <i>Influencia de la cultura organizacional</i> .....	65
<b>2.2   Causas que deterioran la calidad..</b>	36	3.4.2.1 <i>Tipos de cultura organizacional</i> .....	65
<b>2.3   Aspectos sobre los que trabajar para mejorar la calidad.....</b>	38	3.4.3 Comportamientos generadores de conflictos y tensiones en la implementación de procesos de mejora de los roles participantes. .....	66
<b>2.4   Forma de tratamiento de los temas ..</b>	40	3.4.4 Estrategia .....	69
<hr/>			
<b>Capítulo 3 - Trabajo con la organización - Mejora de procesos</b>			
<b>3.1   Visión del cambio .....</b>	43	3.4.5 Tácticas .....	70
3.1.1 Análisis de casos .....	43	3.4.5.1 <i>Diferentes tácticas para el trabajo con los miembros jerárquicos y con los demás miembros</i> .....	70
3.1.2 Primeros pasos en un proceso de		3.4.5.2 <i>Diferentes formas de comunicación con áreas jerárquicas y racionales, y canal de comunicación entre ellas</i> .....	72
		3.4.5.3 <i>Compartir lugar físico</i>	

con los miembros de las áreas .....	72	4.5   Validación y verificación.....	98
3.4.5.4 Foco en temas puntuales y en grupos reducidos con intereses comunes .....	72	4.5.1 Validación .....	98
3.4.5 Liderazgo .....	72	4.5.2 Verificación .....	98
<b>3.5   Respuesta a los casos de estudio</b> .....	73	<b>4.6   Administración de cambios a los requerimientos</b> .....	100
Caso 1 .....	73	4.6.1 Problema .....	100
Caso 2 .....	74	4.6.2 Alternativas de solución .....	102
Caso 3 .....	74	4.6.2.1 Nota para desarrolladores ágiles .....	102
<b>3.6   Conclusión</b> .....	75	<b>4.7   Conclusión</b> .....	102
<hr/> <b>Capítulo 4: Trabajo con Requerimientos</b>			
<b>4.1   Importancia de los requerimiento</b> .....	77	<b>5.1   Proyectos</b> .....	105
4.1.1 El rol de analista.....	78	5.1.1 Planes y planificación .....	105
4.1.1.1 Definición.....	78	5.1.2 Cascada versus iteraciones .....	106
4.1.2 ¿Qué son los requerimientos?.....	78	5.1.2.1 La dinámica de las iteraciones .....	107
4.1.3 ¿Para qué sirven?.....	78	5.1.2.2 Las vistas de los roles .....	109
4.1.4 ¿Cuál es el impacto en un proyecto de desarrollo de software?.....	78	5.1.3 Planificación de iteraciones .....	111
<b>4.2   Tareas asociadas a los requerimientos</b> .....	82	5.1.3.1 Medidas de estabilidad .....	112
4.2.1 Foco .....	83	5.1.4 Fases, actividades, objetivos .....	114
4.2.2 Nivel.....	84	5.1.5 Cuestiones a tener en cuenta y algunas recomendaciones.....	116
4.2.3 Vista .....	84	5.1.5.1 A tener en cuenta .....	116
<b>4.3   Estrategia y tácticas en el trabajo con requerimientos</b> .....	86	5.1.5.2 Recomendaciones .....	117
4.3.1 Estrategia .....	86	5.1.6 Condiciones de contexto .....	117
4.3.2 Tácticas .....	88	<b>5.2   Planificación de proyectos</b> .....	118
4.3.2.1 Especificación de requerimientos de software y sus atributos de calidad.....	88	5.2.1 Estrategia .....	120
4.3.2.2 Especificación de casos de uso .....	89	5.2.1.1 Un caso demostrativo.....	120
<b>4.4   Análisis de requerimientos</b> .....	91	5.2.2 Construcción de una estrategia.....	122
4.4.1 No confundir dominio y negocio con diseño .....	91	5.2.2.1 Visión.....	122
4.4.1.1 Nota para desarrolladores ágiles .....	95	5.2.2.2 Objetivos .....	122
4.4.1.2 Nota a los analistas de sistemas .....	96	5.2.2.3 Prioridades .....	123
4.4.2 Paquetes .....	96	5.2.2.4 Riesgos .....	123
4.4.2.1 Alternativas de selección .....		5.2.2.5 Estimaciones .....	124

<i>de proyectos para cubrir sus responsabilidades .....</i>	130	6.3.1.2 Principios.....	150	
5.3.1.2 <i>Qué cosas no debe hacer un líder de proyectos .....</i>	131	6.3.2 Infraestructura .....	151	
5.3.2 Actividades.....	131	6.3.3 Resultados .....	153	
5.3.3 Puntos de observación.....	132	<b>6.4   Revisiones de diseño y Código.....</b>	154	
5.3.4 Fotos versus película.....	133	6.4.1 Revisiones .....	154	
5.3.4.1 <i>Tratamiento de una decena de temas .....</i>	133	6.4.1.1 Objetivos .....	155	
5.3.4.2 <i>No seguimiento de los temas tratados .....</i>	133	6.4.1.2 Beneficios .....	155	
5.3.5 Escalamiento .....	133	6.4.1.3 Métricas guía .....	156	
5.3.6 Acciones.....	134	6.4.1.4 Indicadores .....	157	
5.3.7 Métricas .....	134	6.4.1.5 Polimétrica de complejidad.....	158	
<b>5.4  Conclusión .....</b>	137	<b>6.5   Conclusiones .....</b>	158	
<b>Capítulo 6: Trabajo con la Implementación - Diseño Codificación y Pruebas</b>		<b>6.6   Herramientas.....</b>	158	
<b>6.1   Diseño, codificación y pruebas.....</b>	139	<hr/>		
6.1.1 Problemas .....	140	<b>Capítulo 7: Trabajo con Modelos de Desarrollo - CMMI</b>		
6.1.1.2 <i>Proceso de diseño .....</i>	140	<b>7.1   Modelos de referencia.....</b>	161	
6.1.1.3 <i>Coordinación de la construcción .....</i>	140	7.1.1 CMMI .....	162	
6.1.1.4 Pruebas .....	140	7.1.2 Relación entre áreas de proceso....	167	
<b>6.2   Pruebas de software .....</b>	141	7.1.3 Desmistificando el modelo .....	169	
6.2.1 Trabajo con el repositorio .....	145	7.1.3.1 <i>Por qué CMMI .....</i>	169	
6.2.2 Test sistemáticos y automáticos..	146	7.1.3.2 <i>Con quién trabajar .....</i>	169	
6.2.3 Cómo adoptar la nueva forma de trabajo .....	146	7.1.3.3 <i>Cómo es el proceso de mejoras con CMMI .....</i>	171	
6.2.3.1 <i>Obstáculos para automatizar las pruebas .....</i>	147	7.1.3.4 <i>Qué recursos se necesitan .....</i>	175	
6.2.3.2 <i>Qué debería automatizarse ... .....</i>	148	7.1.3.5 <i>Cómo es la evaluación con CMMI (SCAMPI) .....</i>	177	
6.2.3.3 <i>Qué no debería automatizarse .....</i>	148	<b>7.2  Mejora de procesos utilizando el modelo CMMI .....</b>	181	
6.2.3.4 <i>Estrategia para comenzar la automatización .....</i>	148	7.2.1 Estrategia general .....	182	
<b>6.3   Integración continua .....</b>	149	7.2.1.1 <i>Políticas y Procesos .....</i>	183	
6.3.1 Forma de trabajo .....	149	7.2.1.2 <i>Interpretación y mapeo de objetivos y tareas .....</i>	184	
6.3.1.1 <i>Pasos .....</i>	149	7.2.3 Institucionalización .....	187	
		7.2.3.1 <i>Relación entre áreas de proceso y objetivos genéricos .....</i>	187	
		<b>7.3  Modelos y metodologías.....</b>	189	
		7.3.1 Metodologías y modelos .....	189	
		7.3.2 CMMI y metodologías .....	190	
		<b>7.4  Madurez .....</b>	191	

<b>7.5  Conclusiones .....</b>	192
--------------------------------	-----

---

## Apéndice

---

<b>Apéndice - Ejemplos de Activos ...</b>	195
---	-----

Especificación de requerimientos de software (ers) .....	195
--	-----

Modelo de especificación de casos de uso.....	197
---	-----

Ejemplo de priorización de requerimientos .....	199
---	-----

Modelo de informe de avance .....	202
-----------------------------------	-----

Modelo de procedimiento .....	203
-------------------------------	-----

Procedimiento de trabajo con código compartido (cc) en ambiente de lc ..	204
--	-----

Descripción .....	204
-------------------	-----

Forma de trabajo .....	204
------------------------	-----

Condiciones de entrada .....	205
------------------------------	-----

Entradas .....	205
----------------	-----

Roles .....	205
-------------	-----

Activos .....	205
---------------	-----

Pasos de la actividad .....	206
-----------------------------	-----

Salida .....	206
--------------	-----

Condiciones de salida .....	207
-----------------------------	-----

Métricas .....	207
----------------	-----

Verificación y validación .....	207
---------------------------------	-----



---

## **Capítulo 1: Calidad en el software**

---

<b>1.1   Introducción .....</b>	19
<b>1.2   Calidad en el software .....</b>	19
1.2.1 Evolución histórica .....	19
1.2.2 Lecciones no aprendidas .....	23
<b>1.3  Calidad versus velocidad de desarrollo .....</b>	24
1.3.1 Estándares .....	24
1.3.2 Creatividad.....	24
1.3.3 Madurez .....	25
<b>1.4   Modelos de calidad de software.....</b>	25
1.4.1 Surgimiento y evolución .....	25
1.4.2 Modelos .....	26
1.4.3 Certificaciones y evaluaciones .....	32
<b>1.5  Conclusión .....</b>	32

# 1

## CALIDAD EN EL SOFTWARE

### **1.1 | INTRODUCCIÓN**

---

El objetivo de este capítulo es presentar el concepto de calidad asociado al desarrollo de software. Lo hacemos desde una perspectiva histórica porque pensamos que la mejor forma de aprenderlo es a partir de los sucesos que lo afectaron a lo largo del tiempo. En definitiva, observaremos de qué manera y frente a qué acontecimientos se fue dando forma al concepto de calidad en el software como lo conocemos al día de hoy.

### **1.2 | CALIDAD EN EL SOFTWARE**

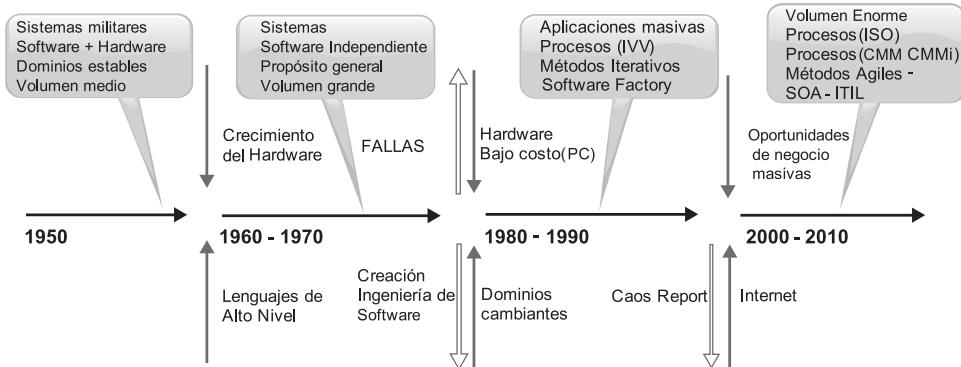
---

#### **1.2.1 EVOLUCIÓN HISTÓRICA**

Para esta presentación utilizaremos un modelo simplificado, como el que se muestra en la figura 1.1. Narraremos los sucesos más importantes y los hitos que constituyeron un cambio en la evolución que nos ocupa. Para aquellos que deseen leer acerca de esta secuencia de hechos en forma detallada y analizados con mayor profundidad pueden consultar el libro de Michael Cusumano<sup>1</sup>.

---

1| Cusumano, Michael. *The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad*. Reed Business Information - Reed Elsevier Inc. 2004.



**Fig. 1-1:** Esquema de la evolución histórica del desarrollo de software y del concepto de calidad asociado

En la figura anterior se puede ver que nuestra historia comienza en la década del 50. Después de la Segunda Guerra Mundial los importantes avances en el desarrollo de software tuvieron lugar en los Estados Unidos de América (EE. UU.) y se generaron en el ámbito de la industria militar. Controles de tiro y de navegación aérea fueron las áreas en las cuales se trabajó con mayor dedicación y recursos. En aquella etapa de la evolución del software las aplicaciones eran desarrolladas para un hardware dedicado, para sistemas que contaban al software como una de sus partes. La bondad (calidad) asociada a estos sistemas se lograba con pruebas exhaustivas una vez terminado de construir. Entre los participantes se pueden mencionar Digital Equipment Corporation (DEC) e IBM. No existen registros públicos acerca de las fallas detectadas y los errores cometidos. Una de las características salientes de estos sistemas era la estabilidad de los requerimientos, lo que los diferencia de manera importante de los desarrollos de hoy día.

Con los avances en la tecnología electrónica (hardware) y la aparición de lenguajes de alto nivel se estableció una nueva tendencia en el desarrollo, se comenzaron a producir sistemas no militares e independientes del hardware. Entre estos tipos de sistemas podemos citar a los de reservación de pasajes para aerolíneas (SABRE) como uno de los primeros y más importantes desarrollos comerciales.

En la década siguiente, del 60, los avances fueron muy importantes y estuvieron conducidos por grandes inversiones en universidades y en la industria, y orientados a producir sistemas de propósito general. Estos avances en las plataformas y lenguajes hicieron que estos nuevos desarrollos crecieran en sofisticación. Ya para esta época empresas de todas partes del mundo desarrollaban productos que buscaban la compatibilidad con IBM. Se trabajaba, en Inglaterra y Japón, en empresas como RCA, GE, Fujitsu y Hitachi. A su vez vieron la luz otros sistemas propietarios ofrecidos por Honeywell, Bull y Nec.

Además, en EE. UU. ya había cincuenta empresas grandes abocadas al desarrollo de software y unas tres mil pequeñas. En esta etapa, los sistemas resolvían problemas

en las áreas de explotación de petróleo, compañías de seguros y ventas en grandes mercados minoristas.

Entre 1963 y 1966 ocurrió un suceso que marcó un punto de cambio en esta evolución. Este evento surgió a partir de los inconvenientes de sobre presupuesto y tiempo adicional necesarios en la terminación del proyecto de desarrollo del sistema operativo OS/360 de IBM. Este fue uno de los mayores proyectos de la época y generó un alerta en el sentido de la necesidad de contar con métodos de desarrollo que garantizaran la calidad de los productos de software. Como resultado de esto, el gerente del proyecto, Frederick Phillips Brooks Jr., publicó un libro, que hasta el día de hoy es un best seller<sup>2</sup>, donde volcó la experiencia recogida en el proyecto. Estos acontecimientos fueron nombrados como la “crisis del software”.

Pero esta alerta generó, además de la aparición del libro citado, otro hito importante, el convencimiento de la necesidad y los primeros esfuerzos en la creación de una nueva disciplina llamada Ingeniería de Software (1980). En estos años ya se trabajaba en la modalidad de software factory en Japón, como Cusumano indica: “con buenos resultados en lo referente a la calidad de sus productos aunque a entender de algunos autores con capacidad de innovación limitada”.

En 1987 se creó el SEI (Software Engineering Institute) a instancias del Department of Defense de EE. UU. (DoD). En este instituto trabajaron personas como W. Humphrey<sup>3</sup>, miembro de IBM y mentor junto a otros del modelo Capability Maturity

Por ese entonces estaba claro que el desarrollo de software consistía en administrar el proceso de construir un producto o un sistema que cubriera las necesidades de los usuarios, probarlo, instalarlo en el ambiente productivo, mantenerlo y hacerlo evolucionar con los cambios del negocio. Por lo tanto un aspecto de la calidad vinculado al software consistía en llevar adelante este proceso de forma que permitiera al proyecto asociado terminar en el tiempo planificado y dentro del presupuesto asignado.

Model (CMM). Estos fueron los primeros pasos en la dirección de dar forma al concepto de calidad tal cual lo conocemos.

En relación con esto, en el capítulo cuatro nos ocuparemos del trabajo con los requerimientos vinculados a las necesidades mencionadas. En el capítulo cinco analizaremos los aspectos claves de la administración de los proyectos de desarrollo y en el capítulo seis presentaremos una propuesta de solución al problema de las pruebas utilizando integración continua.

Mientras esto sucedía, la tecnología seguía avanzando y se contaba con plataformas de bajo volumen y bajo costo que ofrecían la posibilidad de desarrollar software como una oportunidad de negocio a gran escala. Entonces apareció la Personal Computer (PC) como consecuencia del avance alcanzado en el desarrollo

2 | Brooks, Frederick. *The Mythical Man-Month: Essays on Software Engineering*. Addison Wesley. 1975.

3 | Humphrey, W. S. *Managing the Software Process*. Reading, MA: Addison-Wesley. 1989.

de microprocesadores. También asomaba al mundo una empresa llamada Microsoft, productora del sistema operativo para estas plataformas llamado DOS.

Por aquellos días volvió a trabajar a su país, EE. UU., W. Demming, quien había ayudado a dar sustento estadístico a la tradición de trabajo para generar productos de calidad en el Japón<sup>4</sup>. En ese entonces acuñó una frase que se hizo famosa y que de alguna manera sustancia la esencia del concepto de calidad:

“The quality of a product is directly related to the quality of the process used

A partir de esta aseveración, en una organización se definió como mejora de procesos en el software (SPI-Software Process Improvement) al conjunto de tareas llevadas adelante con el objetivo de generar productos de mejor calidad a partir de la revisión y adaptación de sus procesos y la incorporación de nuevos.

to create it<sup>5</sup>”, lo que significa que “la calidad de un producto está directamente relacionada al proceso utilizado para crearlo”.

Estos temas los trataremos en el capítulo tres, en el que analizaremos las mejoras y el impacto en las organizaciones.

En el capítulo siete utilizaremos el modelo CMMi (ver Modelos de calidad de software en este capítulo).

También deseamos enfatizar un aspecto clave de la instalación en las organizaciones del concepto de calidad a partir de otra referencia a W. Demming, quien en una conferencia dijo: “If Japan can... Why Can’t We?” (Poppendiek, op. cit.), cuya traducción es: “Si los japoneses pueden... ¿Por qué no podemos nosotros?”. En aquel reportaje explicó de qué forma el respeto de las organizaciones por sus miembros constituía la clave en la asimilación de procesos que generaban productos de excelente calidad.

En la década de 1990, el crecimiento de los sistemas se acentuó, con protagonistas como Microsoft, ya convertida en líder mundial, Netscape, Oracle y otros. Se desarrollaron sistemas como el operativo NT, uno de los más importantes del momento. Además se consolidaron las metodologías de desarrollo de tipo iterativas las cuales van suplantando a las conocidas como cascada. Aparecieron algunas metodologías llamadas ágiles y el concepto de integración continua, y también se sigue trabajando en IVV (Independent Verification Validation), temas que trataremos en el capítulo seis. Estas formas de trabajo tienen una fuerte influencia en la calidad del software. En 1994 fue publicado el informe del Standish Group llamado Caos Report<sup>6</sup>, el cual analizaremos en profundidad en el próximo capítulo, donde se relevaron, para miles de proyectos de desarrollo, las causas que atentan contra la calidad del software. También debemos mencionar el nacimiento y crecimiento del software libre como uno de los impulsores de importantes avances en esta época.

Nuestra historia arriba entonces a los años 2000-2010 donde el escenario

4 | Deming, W. E. *Out of the Crisis*. Cambridge. MA: MIT Center for Advanced Engineering Study. 1982.

5 | Poppendiek, Mary & Tom. *Implementing Lean Software Development*. Addison Wesley. 2007.

6 | Caos Report. Standish Group. [www.standishgroup.com](http://www.standishgroup.com). 1994. 2004.

establecido para el desarrollo de software está determinado por un hardware cada vez más poderoso, software de última generación (lenguajes orientados a objetos, lenguajes deductivos, interpretados, intermedios multiplataformas, arquitecturas orientadas a servicios), modelos de desarrollo (CMMI, ITIL) y metodologías ágiles (XP, Scrum). Arribamos a este estado de cosas luego de que en 1995 se implementara

Este escenario es propicio para destacar el otro aspecto vinculado a la calidad del software, el relacionado a los atributos de calidad de los productos generados. En los sistemas enterprise de hoy algunos de los más representativos son confiabilidad, usabilidad, seguridad, disponibilidad, escalabilidad, mantenibilidad y tiempo de salida al mercado<sup>7</sup>.

Internet a nivel global, lo cual generó la utilización masiva de productos y servicios de software que condujo a la aparición de innumerables oportunidades de negocio.

La degradación de estos atributos de calidad harán que la empresa pierda credibilidad, con las consecuencias que ello implica en un mercado transparente.

A mediados de la década del 90 se destacaron desarrollos como los navegadores de Netscape y Microsoft, en los cuales se ensayaron las mejores prácticas de la época. Esto hizo que actualmente existan cientos de miles de empresas dedicadas a este negocio, con una competencia feroz, lo que condiciona el desarrollo a una dinámica sin precedentes. Esta velocidad creciente impuesta por el mercado de productos de software tiene un impacto importantísimo en la calidad de los productos y servicios ofrecidos. Es de notar que estos cambios en la evolución de esta industria hicieron que la preparación de los desarrolladores de hoy día sea muy distinta a la que tenían aquellos programadores de sistemas integrados a un hardware dedicado y con requerimientos estables de los años cincuenta.

### 1.2.2 LECCIONES NO APRENDIDAS

De la sección anterior se desprende que si bien la Ingeniería de Software es una disciplina relativamente joven, durante su corta vida ha atravesado diferentes etapas de las cuales aprendió la lección: la calidad es un valor en sí mismo y no un gasto que las empresas deben realizar para que su negocio prospere. Sin embargo, esta ingeniería es como una dama de la cual todos alguna vez se enamoran pero que muy pocos están dispuestos a desposar. Digo esto porque es muy común encontrarse con proyectos que generan productos de dudosa calidad debido a que los responsables decidieron no realizar tareas de revisión de diseño y código o determinadas pruebas porque dilatan el tiempo de salida al mercado, cuestan dinero y además no habrá diferencia entre uno y otro producto cuando el proyecto termine. En mi opinión estas lecciones no aprendidas se deben a la falta de madurez de conceptos aprendidos de forma cruenta y en escenarios continuamente cambiantes. En este punto, haremos referencia otra vez a W. Demming, quien en su libro *Out of the Crisis*, para definir su visión del concepto de calidad, en uno de sus puntos, expresó: “Provide for the long range needs of the company; don´t focus on short term profitability. The goal

7 | Offutt, Jeff. *Quality Attributes of Web Software Applications*. IEEE Software. April, 2002.

*is to stay in business and provide jobs*", lo que significa que es mejor "proveer para las necesidades de largo plazo de las empresas y no focalizarse en las ganancias inmediatas; el objetivo es perdurar en el negocio generando trabajo".

Si bien estas definiciones son conocidas desde hace mucho tiempo y además compartidas por la comunidad informática, un porcentaje importante de las empresas continúan funcionando de manera exactamente opuesta, es decir acortando cada vez más los presupuestos orientados a garantizar la calidad y priorizando los objetivos de cortísimo plazo. Las consecuencias de percibir la calidad como un gasto del cual se puede prescindir genera el estado de situación en relación con el desarrollo de software que analizaremos en el próximo capítulo, en el que mostraremos diversos casos en los cuales se relegaron la validación y verificación de los sistemas.

Una causa de falta de sistematización y automatización que contribuye a la baja calidad es el no acuerdo entre los participantes de los grupos de desarrollo acerca de las mejores prácticas a aplicar en los proyectos. Este es un síntoma observado con frecuencia en la comunidad infomática, en la que el gerente tiene una opinión, el líder técnico otra y el líder del proyecto una tercera. En general este fenómeno se produce cuando hay miembros que tienen una visión diferente acerca de si lo que se hace en la empresa es investigación, desarrollo o ambas cosas. Un error común en los programadores es creer que están haciendo investigación cuando en realidad la mayor parte de las empresas ligadas al negocio del software hacen desarrollo.

Es verdad que los tiempos han cambiado y hoy la calidad parece tener otro valor que en los días de Demming. Sin embargo pensamos que sus afirmaciones siguen vigentes.

## **1.3 | CALIDAD VERSUS VELOCIDAD DE DESARROLLO**

---

### **1.3.1 ESTÁNDARES**

Para hacer que los proyectos de desarrollo fueran predecibles se necesitó establecer estándares para las diferentes tareas que se realizaban a lo largo de su ciclo de vida. Así fue que todo aquello que permitiera ser reutilizado fue adoptado como un elemento de valor por ser predecible y además bajar los costos. De este modo se generaron activos y procedimientos para su construcción en disciplinas como el trabajo con requerimientos, diseño y código. Estos estándares fueron propuestos como una forma de garantizar la calidad de procesos y productos.

### **1.3.2 CREATIVIDAD**

Con los procedimientos y activos estándares agrupados en nuevas metodologías y modelos de referencia se comenzó a transitar por la etapa ya descripta que se dio a partir de la instalación de Internet como plataforma de comunicación y oferta y consumo de servicios. Por ese motivo los tiempos de salida al mercado y de estabilidad de los requerimientos se acortaron de una manera impensada. Por esta razón se comenzó a cuestionar los estándares, ya que estos proponían una forma de trabajo

cuya estabilidad contrastaba con la dinámica de la nueva etapa. Así fue que nacieron la metodologías ágiles (Mary and Tom Poppendiek), a partir de que la comunidad comenzó a ponderar la creatividad y velocidad de adaptación a los nuevos escenarios por sobre el control y la predictibilidad en los proyectos. Este fenómeno de estabilidad de procesos versus creatividad ya se había percibido décadas atrás entre las software factory de Japón y la industria del software norteamericana, cuando se analizó la calidad de excelencia de los productos orientales a expensas de la ausencia de nuevos productos, los cuales eran ofrecidos por las empresas de EE.UU. que trabajaban en un ambiente mucho menos controlado (Michael Cusumano).

### 1.3.3 MADUREZ

Como muchas veces a lo largo de la historia de esta industria, hace algunos años se instaló una idea de moda. Esta vez consiste en pensar que cuanto menos estructuradas sean las empresas, mejor estarán posicionadas para enfrentar cambios y ganar nuevos negocios. Sin embargo, como Cosummano muestra en su libro, también están en condiciones mucho más vulnerables para perder todo lo realizado en solo unos meses. La razón de esto la explica W. Demming en su frase acerca de cuál es el objetivo de las empresas. Actualmente, el desafío con respecto a garantizar la calidad en las organizaciones dedicadas al desarrollo de software es organizarse para que sus proyectos sean ordenados y predecibles; aunque deben a su vez tener la capacidad de dejar de lado estos procesos rápidamente para reorganizarse adaptándose a los cambios. Para esto es muy importante lograr madurez en estos procesos de manera de seguirlos cómodamente para que no se constituyan en una carga que aumente los costos e impida contar con agilidad para cambiar. Nos dedicaremos a tratar este tema en relación con los modelos de referencia como CMMi en el capítulo siete.

## 1.4 | MODELOS DE CALIDAD DE SOFTWARE

### 1.4.1 SURGIMIENTO Y EVOLUCIÓN

Como mencionamos en la sección de historia, con el objetivo de establecer formas estándares en las prácticas y activos de desarrollo aparecieron a lo largo de los años distintas normas y modelos de referencia. Los llamamos así porque constituyen la referencia al momento de implementar una mejora de procesos en una organización con el objetivo de aumentar la calidad de procesos y productos generados. Un fin para el cual se ha utilizado estos modelos ha sido la calificación, clasificación y comparación de empresas por parte de los compradores de productos de software. El ejemplo más notorio, ya citado, es el del DoD y el modelo CMM, y después CMMi. Básicamente, con ellos se buscó establecer estándares de organizaciones. La utilización inteligente de estos modelos contribuyó a mejorar los productos de software, mientras que el mal uso generó frustraciones y despilfarro de recursos, como analizaremos en el capítulo siete. En la década de 1990 el modelo CMM (Capability Maturity Model) se convirtió en el estándar de hecho a nivel global. Más tarde, en la década de 2000, fue reemplazado

por su versión mejorada CMMi (Capability Maturity Model Integration).

### 1.4.2 MODELOS

A continuación presentamos un listado de normas y modelos citando en forma resumida las características salientes. Aquellos lectores interesados en alguno de ellos podrá encontrar también una referencia de dónde se puede encontrar información detallada. En el capítulo siete desarrollaremos en detalle el modelo CMMi por considerarlo el más focalizado al desarrollo de software. Los otros modelos mencionados si bien son más abarcativos, porque incluyen procesos relacionados a aspectos administrativos y de formación de recursos humanos, no tratan en detalle aspectos del desarrollo que se desean enfatizar por considerarlos críticos y claves. El modelo ITIL (Information Technology Infrastructure Library), como ya citamos, presenta entre sus componentes importantes el software como parte integrante de los servicios de IT (Information Technology).

#### ■ **CMM (Capability Maturity Model), 1993.**

- **Organización: SEI (Software Engineering Institute)**

- **Información: <http://www.sei.cmu.edu>**

- **Estructura**

- Niveles

- Inicial*

- Repetible*

- Definido*

- Gestionado*

- Optimizado*

- Áreas de proceso*

#### ■ **CMMi (Capability Maturity Model Integration), 2001.**

- **Organización: SEI (Software Engineering Institute)**

- **Información: <http://www.sei.cmu.edu>**

- **Estructura**

- Vistas por niveles y capacidades

- Áreas de proceso

- Grupos de áreas de procesos

- Ingeniería (7 áreas de procesos)*

Proyectos (5 áreas de procesos)

Organizativas (6 áreas de procesos)

### I TSP (Team Software Process)

- **Organización: SEI (Software Engineering Institute)**
- **Información: <http://www.sei.cmu.edu/library/abstracts/books/201731134.cfm>**

### I ISO/IEC 15504 – Information Technology – Process Assessment, 1998 – 2006.

- **Organización: ISO (International Standard Organization)**

- **Información: <http://www.isospice.com/>**

- **Características salientes**

- Establece un marco y los requisitos para cualquier proceso de evaluación de procesos y proporciona requisitos para los modelos de evaluación a ser utilizados.
- Proporciona también requisitos para cualquier modelo de evaluación de organizaciones.
- Proporciona guías para la definición de las competencias de un evaluador de procesos.
- Actualmente tiene 10 partes: 1-7 completadas y 8-10 en fase de desarrollo.
- Comprende: evaluación y mejora de procesos, determinación de capacidad.
- Proporciona en su parte 5 un modelo de evaluación de procesos para los procesos de ciclo de vida del software definidos en el estándar ISO/IEC 12207 que define los procesos del ciclo de vida del desarrollo, mantenimiento y operación de los sistemas de software.

### I ISO/IEC 12207 Information Technology / Software Life Cycle Processes, 1995-2008.

- **Organización: ISO (International Standard Organization)**

- **Información: <http://www.iso.org/iso/home.htm>**

- **Estructura**

- Procesos principales

Adquisición

Suministro

Desarrollo

- Operación*
- Mantenimiento*
- Procesos de soporte
  - Documentación*
  - Gestión de la configuración*
  - Aseguramiento de calidad*
  - Verificación*
  - Validación*
  - Revisión conjunta*
  - Auditoría*
  - Resolución de problemas*
- Procesos de la organización
  - Gestión*
  - Infraestructura*
  - Mejora*
  - Recursos Humanos*

### ■ **PSP (Personal Software Process), 1995.**

- **Organización: W. Humphrey**
- **Información: <http://www.sei.cmu.edu/tsp/>**
- **Estructura**

#### **NIVELES**

- Nivel 2 - inicial:
  - Seguimiento y control de proyectos.*
  - Planeación de los proyectos.*
- Nivel 3 - repetible:
  - Revisión entre colegas.*
  - Ingeniería del producto de software.*
  - Manejo integrado del software.*
  - Definición del proceso de software.*
  - Foco del proceso de software.*
- Nivel 4 - Definido:

*Control de calidad.*

*Administración cuantitativa del proyecto.*

- Nivel 5 - Controlado:

*Administración de los cambios del proceso.*

*Administración del cambio tecnológico.*

*Prevención de defectos.*

## FASES

- PSP0: proceso base, registro de tiempos, resgistro de errores, estándar de tipo de errores. [Proceso personal de arranque.]
- PSP0.1: estándar de codificación, medición de tamaño, propuesta de mejoramiento del proceso(PIP). [Proceso personal de arranque.]
- PSP1: estimación del tiempo, reporte de pruebas. [Proceso personal de administración.]
- PSP1.1: planeación de actividades, planeación de tiempos. [Proceso personal de administración.]
- PSP2: revisión de codificación, revisión del diseño. [Proceso personal de calidad.]
- PSP2.1: formatos de diseño. [Proceso personal de calidad.]
- PSP3: desarrollo en ciclos. [Proceso cíclico.]

## ■ ITIL (Information Technology Infrastructure Library), 1989 - 2007.

- **Organización: United Kingdom's Office of Government Commerce (OGC)**
- **Información: <http://www.itil-officialsite.com/home/>**
- **Estructura**

### ÁREAS DE APLICACIÓN (VERSIÓN 1)

- Administración de Servicios
  1. Service Support
  2. Service Delivery
- Guías Operacionales
  3. ICT Infrastructure Management
  4. Security Management
  5. The Business Perspective

- 6. Application Management
- 7. Software Asset Management
- Guías de Implementación
  - 8. Planning to Implement Service Management
  - 9. ITIL Small-Scale Implementation

### **FASES (VERSIÓN 3)**

- 1. ITIL Service Strategy
- 2. ITIL Service Design
- 3. ITIL Service Transition
- 4. ITIL Service Operation
- 5. ITIL Continual Service Improvement

Con el tiempo y por diferentes razones surgieron otros modelos inspirados en los anteriores que buscaron cubrir necesidades puntuales. Como por ejemplo el MOPROSOFT (Modelo de Procesos para la Industria del Software), concebido en México, con el objetivo de ser aplicado a las pequeñas y medianas empresas dedicadas al desarrollo de software. Está inspirado en los niveles 2 y 3 del modelo CMM y en las normas ISO/IEC 15504.

Algunos de los criterios con los que fue elaborado este modelo son los siguientes:

- Procesos estratificados según el organigrama típico de este tipo de organizaciones (gerencia alta, gerencia media y operación).
- La gerencia alta tiene como actividades centrales la de definir las estrategias de la organización y promover las mejoras continuas.
- La gerencia media es la encargada de proveer los recursos para los proyectos y monitorear el cumplimiento de las planificaciones orientadas a cumplir con los objetivos estratégicos.
- La operación es la encargada de llevar adelante los proyectos.
- Los procesos deben ser definidos de tal forma que mantengan entre ellos una relación integradora.
- El proceso de administración de proyectos es atómico.
- La ingeniería de productos es desarrollada con el soporte de otros procesos orientados a garantizar y controlar la calidad de los mismos (verificación, validación, documentación y control de la documentación).
- Se le asigna especial atención a la administración de los recursos que apor-

tan al conocimiento de la organización: productos generados por proyectos, mediciones, documentación de procesos y datos relevados de su implementación en los proyectos así como lecciones aprendidas.

## I MOPROSOFT (Modelo de Procesos para la Industria del Software), 2005.

- **Organización:** Asociación Mexicana para la Calidad en Ingeniería de Software
- **Información:** <http://www.comunidadmoprossoft.org.mx/>
- **Estructura**

### CATEGORÍA ALTA DIRECCIÓN (DIR)

Comprende la definición de las políticas, los objetivos de negocio y las estrategias para su logro incorporando la información de la evolución de la organización a efectos de promover la mejora continua.

- Gestión de Negocio: establecer la visión y los objetivos de negocio que fijen el contexto para las actividades de la gerencia media y los proyectos.

### CATEGORÍA GERENCIA (GER)

Comprende la gestión de los procesos, proyectos y recursos de la organización de manera alineada con las políticas definidas por la gerencia alta. Informa a la gerencia alta acerca de la evolución de la organización.

- Gestión de Procesos: promover la definición e implementación de los procesos de la organización a partir de las necesidades expresadas por el plan estratégico. Coordinar y colaborar en la mejora de los procesos definidos.
- Gestión de Proyectos: Asegurar el desarrollo de los proyectos de manera alineada a los objetivos de la organización expresados en su plan estratégico.
- Gestión de Recursos: Administrar los recursos de la organización proveyendo a los proyectos. Promover y facilitar la administración del capital de conocimiento de la organización.

Las actividades de este proceso son soportadas por tres subprocessos:

- Recursos Humanos y ambiente de trabajo
- Bienes, servicios e infraestructura
- Conocimiento de la organización

### CATEGORÍA OPERACIÓN (OPE)

Comprende las prácticas de gestión y técnicas de los proyectos de desarrollo.

- Administración de proyectos específicos: estimar, planificar y monitorear el desarrollo de los proyectos de desarrollo y la generación de los productos esperados.
- Desarrollo y mantenimiento de software: desarrollo de las actividades del ciclo de vida de productos nuevos o modificados (análisis, diseño, construcción, integración y pruebas) alineadas con los requerimientos del proyecto.

También surgieron organizaciones que orientaron sus actividades en torno a los modelos de calidad mencionados, como es el caso de European Software Institute (ESI-Tecnalia). Este es un centro tecnológico privado creado en 1993 por la Comisión Europea con el apoyo del gobierno Vasco y de empresas europeas líderes en el ámbito de las Tecnologías de la Información y la Comunicación. La principal actividad del ESI-Tecnalia es ayudar a la industria del software en sus objetivos de elaborar productos de software de buena calidad. ESI ofrece servicios de auditoría, consultoría, formación y certificación, así como soporte tecnológico (<http://www.esi.es/>).

#### **1.4.3 CERTIFICACIONES Y EVALUACIONES**

Los modelos y normas mencionados en la sección anterior, además de proponer objetivos y un conjunto de buenas prácticas para lograrlos, proveen una forma de evaluación que en el caso de ser exitosa acredita una certificación. Estas evaluaciones han ido cambiando con el tiempo y hoy en día es necesario que las empresas sean reevaluadas con cierta periodicidad para que conserven su calificación, como ocurre con el modelo CMMI versión 1.2. En general estas evaluaciones se realizan después de una mejora de procesos donde la organización se somete a una evaluación inicial que le permite obtener una medida del estado en que se encuentra. A partir de estas observaciones y de una planificación dependiente de adónde se desea arribar (nivel CMMI, por ejemplo) se trabaja hasta alcanzar una mejora que le permita ser evaluada exitosamente y ser calificada o certificada como una organización de tal o cual nivel. En el capítulo siete trataremos el proceso evaluatorio del modelo CMMI.

---

### **1.5 | CONCLUSIÓN**

Entendemos por desarrollo de software la administración del proceso de construir un producto o un sistema que cubra las necesidades de los usuarios, probarlo, instalarlo en el ambiente productivo, mantenerlo y hacerlo evolucionar con los cambios del negocio. Por lo tanto un aspecto de la calidad vinculado al software consiste en llevar adelante este proceso de la mejor forma que permita al proyecto asociado terminar en el tiempo planificado y dentro del presupuesto asignado.

Mejora de procesos en el software (SPI-Software Process Improvement) en una organización es el conjunto de tareas llevadas adelante con el objetivo de generar productos de mejor calidad a partir de la revisión y adaptación de sus procesos y la incorporación de nuevos.

Otro aspecto vinculado a la calidad del software es el relacionado a los atributos de calidad de los productos generados. En los sistemas enterprise de hoy día algunos de los más representativos son confiabilidad, usabilidad, seguridad, disponibilidad, escalabilidad, mantenibilidad y tiempo de salida al mercado.

La calidad es un valor en sí mismo y no un gasto que las empresas deben realizar para que su negocio prospere.

Establecer estándares para las diferentes tareas que se realizan a lo largo de su ciclo de vida permite la reutilización como un elemento de valor por ser predecible y además permita bajar los costos. Estos estándares fueron propuestos como una forma de garantizar la calidad a procesos y productos.

El desafío de hoy día con respecto a garantizar la calidad en las organizaciones dedicadas al desarrollo de software es organizarse para que sus proyectos sean ordenados y predecibles; aunque deben a su vez tener la capacidad de dejar de lado estos procesos rápidamente para reorganizarse adaptándose a los cambios. Para esto es muy importante lograr madurez en estos procesos de manera de seguirlos cómodamente para que no se constituyan en una carga que aumente los costos e impida contar con agilidad para cambiar.

En el próximo capítulo analizaremos las causas que deterioran la calidad en el desarrollo de software y sustentaremos el enfoque con el que trataremos los diferentes temas, motivados en resolver las causas de este deterioro.

---

**Capítulo 2: Causas que deterioran la calidad en el software**

---

<b>2.1   Introducción .....</b>	35
2.1.1 Definiciones .....	35
2.1.1.1 <i>Calidad de producto</i> .....	35
2.1.1.2 <i>Calidad de proceso</i> .....	35
<b>2.2   Causas que deterioran la calidad.....</b>	36
<b>2.3   Aspectos sobre los que trabajar para mejorar la calidad.....</b>	38
<b>2.4   Forma de tratamiento de los temas .....</b>	40

# 2

## CAUSAS QUE DETERIORAN LA CALIDAD EN EL SOFTWARE

### 2.1 | INTRODUCCIÓN

---

#### 2.1.1 | DEFINICIONES

La aseveraciones que siguen, derivadas del análisis histórico que hicimos en el primer capítulo, son aceptadas como verdades absolutas en el ambiente del desarrollo de software. Sin embargo muchas veces son ignoradas o dejadas de lado y eso genera varias consecuencias, las que analizaremos en las próximas secciones.

##### 2.1.1.1 Calidad de producto

Un producto es de buena calidad si le sirve a quien lo adquiere y si este lo usa para realizar las tareas para lo que fue concebido.

Practicar el control de calidad consiste en realizar las acciones necesarias para que ese producto cumpla con los atributos de calidad y con las prestaciones que lo califican.

##### 2.1.1.2 Calidad de proceso

Un proceso malo, mal concebido e implementado generará productos de mala calidad.

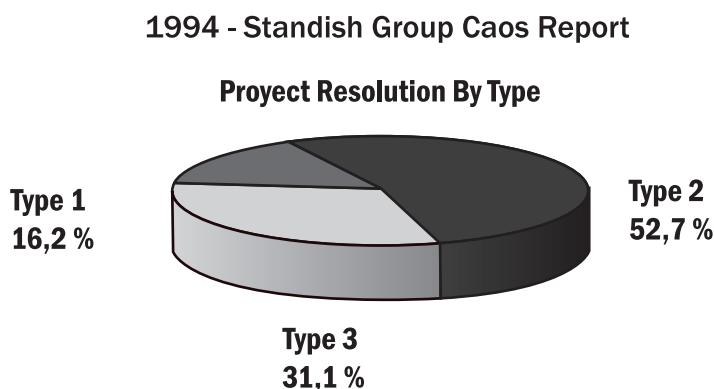
Un proceso bueno, bien concebido e implementado generará la mayor cantidad de las veces productos de buena calidad.

## 2.2 | CAUSAS QUE DETERIORAN LA CALIDAD

---

Para fundamentar la propuesta que haré en este libro podría utilizar diferentes ejemplos que me permitirían promover una forma de trabajo que mejore la calidad de los productos de software generados en los proyectos de desarrollo. Existe un informe, el cual ya mencionamos en el capítulo anterior, que se conoce como el Caos Report. Después de pensarlo, he decidido correr el riesgo y utilizarlo ya que es probable que sea conocido por muchos de los lectores. Aquel que no lo conozca puede acceder a él por Internet y sacar alguna conclusión previa antes de leer la de este libro.

A modo de brevísimo resumen diré que este trabajo fue una recopilación realizada sobre miles de proyectos de desarrollo, cuyos resultados pueden condensarse en la figura 2.1.



**Fig. 2.1** - Distribución de proyectos del Caos Report del año 1994.

En esta figura los proyectos clasificados como de tipo 1 son aquellos que se terminaron según la planificación original. Los de tipo 2 son los terminados fuera de tiempo y con un sobre presupuesto que en algunos casos llegó al 189%, y los de tipo 3 son los proyectos que fueron cancelados en algún momento de su ciclo de vida.

Para completar este resumen, la encuesta realizada a participantes de aquellos proyectos acerca de cuáles eran los factores más importantes para el éxito de un proyecto de desarrollo arrojó el siguiente listado, el que está ordenado según la valoración resultante (tabla 2.1).

SUCCESS CRITERIA	POINTS
1. User Involvement	19
2. Executive Management Support	16
3. Clear Statement of Requirements	15
4. Proper Planning	11
5. Realistic Expectations	10
6. Smaller Project Milestones	9
7. Competent Staff	8
8. Ownership	6
9. Clear Vision & Objectives	3
10. Hard-Working, Focused Staff	3
<b>TOTAL</b>	<b>100</b>

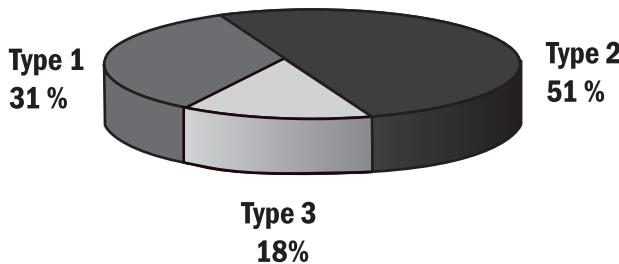
**Tabla 2.1**

Podemos leer este listado como el resultado de las lecciones aprendidas, razón por la cual se pueden ver a estos factores como los que no fueron bien administrados y condujeron a los proyectos a ser clasificados según los tipos anteriores.

Este informe se repitió diez años después y los resultados de esta segunda versión se presentan en la figura 2.2. El listado de factores resultó similar.

### 2004 – Standish Group Caos Report

#### Chaos Report 2004

**Fig. 2.2** - Distribución de proyectos del Caos Report del año 2004.

Si comparamos las dos figuras podemos arriesgar alguna conclusión. La cantidad de proyectos del tipo 1, exitosos, prácticamente se duplicó. La cantidad de proyectos de tipo 2, terminados tarde y más caros, permaneció aproximadamente constante, y la cantidad de proyectos de tipo 3, cancelados, se redujo a la mitad.

Ahora extraeremos una conclusión de estos resultados. Yes que si nos proponemos garantizar la calidad de los productos generados, por supuesto por proyectos de tipo 1, debemos abocarnos a trabajar en los aspectos, como los números nos muestran, que son las causas del deterioro de la calidad. Por lo tanto trabajaremos a lo largo de este libro haciendo propuestas para:

1. Lograr el involucramiento en el proyecto de parte de los clientes y usuarios
2. Lograr el compromiso de nuestra propia organización (somos los desarrolladores) con el proyecto
3. Lograr un claro entendimiento de los requerimientos
4. Lograr una planificación realista
5. Trabajaremos en forma iterativa
6. Evaluando objetivos
7. Capacitando a nuestros miembros
8. Motivando a todos los involucrados
9. Haciendo visibles los objetivos
10. Enfocándonos en generar valor agregado con cada entrega

Si así lo hacemos habremos revertido las causas por las cuales los proyectos fracasan. Esta conclusión coincide con nuestra experiencia, razón por la cual hemos finalmente decidido usar estos informes.

Entonces, trataremos los temas que permitieron, después de una decisión simple y de mucho trabajo, a las empresas que desarrollaban proyectos de tipo 2 ó 3 generar proyectos de tipo 1.

Me pregunto cuál es el motivo por el que hay empresas que sabiéndose generadoras de proyectos de tipo 2 no trabajan para cambiar.

¿Será que el negocio del desarrollo de software consiste en desarrollar proyectos de este tipo?

## **2.3 | ASPECTOS QUE HAY QUE TRABAJAR PARA MEJORAR LA CALIDAD**

En la figura 2.3 se muestran los componentes sobre los que trabajaremos y las relaciones entre ellos. En las relaciones se indican los aspectos claves sobre los que enfocaremos nuestras propuestas.

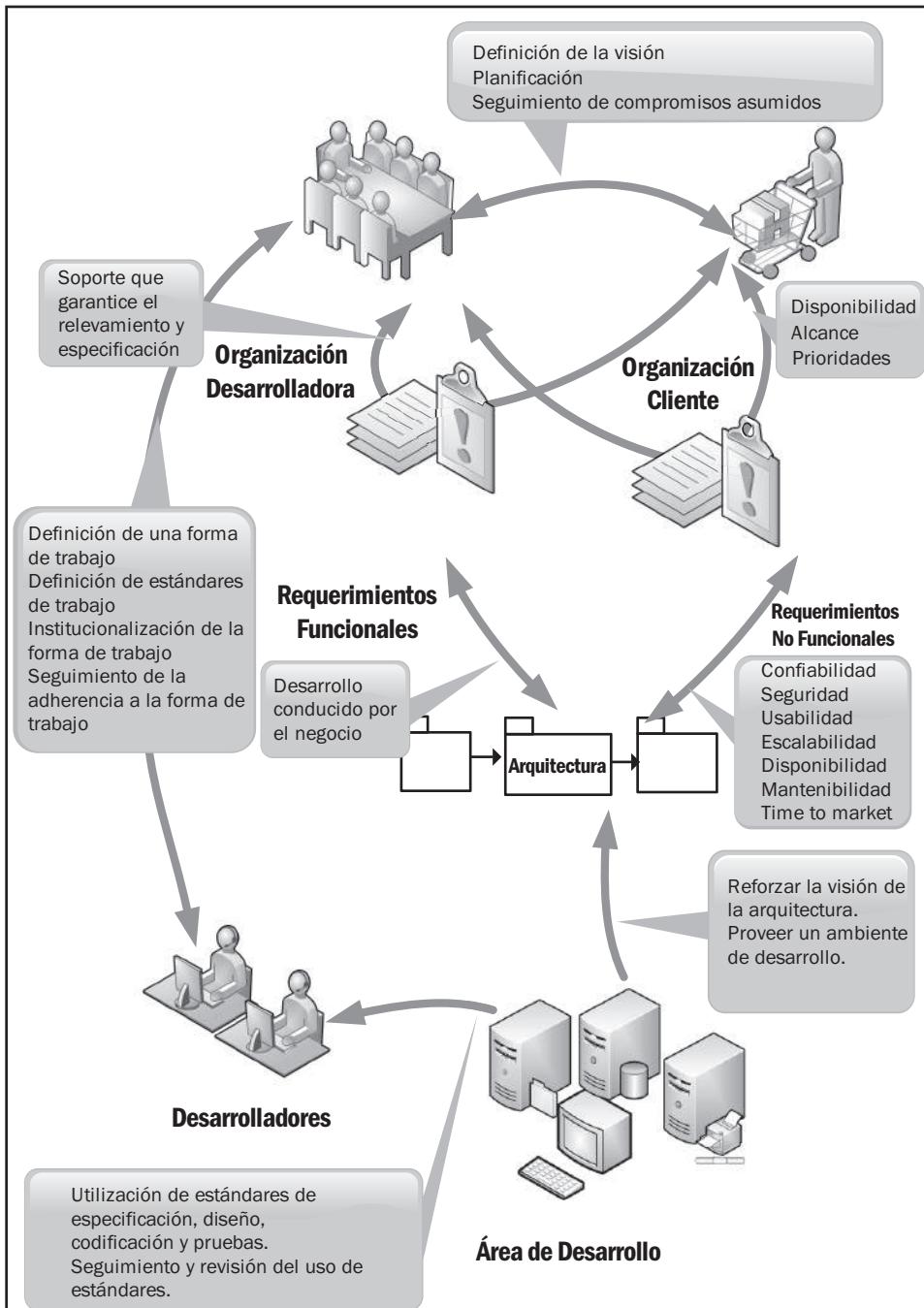


Fig. 2.3 - Componentes del negocio del desarrollo de software y sus relaciones.

Los ítems listados en la sección anterior serán cubiertos por propuestas de mejoras presentadas en los siguientes capítulos y apéndices:

- Trabajo con la organización - Mejora de Procesos: cubriremos los ítems 1, 2
- Trabajo con requerimientos: cubriremos el ítem 3
- Trabajo con la gestión de proyectos: cubriremos los ítems 1, 2, 4, 5 y 6
- Trabajo con la implementación - diseño, codificación y pruebas: cubriremos el ítem 10
- Trabajo con modelos de desarrollo - CMMi: cubriremos los ítems 1 y 2
- Metodologías: cubriremos los ítems 5, 6, 8 y 9
- Capacitación: cubriremos el ítem 7

## 2.4 | FORMA DE TRATAMIENTO DE LOS TEMAS

---

Steve McConnell<sup>8</sup> propone que a efectos de mejorar las prácticas de una organización desarrolladora de software se debe trabajar sobre los errores comunes, las tareas básicas de gestión y las tareas básicas técnicas. Afirma que la forma de lograr mejoras es comenzando por las bases; es decir, mejorar las actividades básicas diarias, las realizadas por los diferentes roles en su participación en los proyectos. En forma coincidente proponemos capacitar a los miembros de los grupos de desarrollo en la realización de sus actividades. Asumimos que esta capacitación se ha llevado a cabo y los involucrados en los proyectos ya están aplicando lo aprendido. Por lo tanto en este libro nos abocaremos a revisar estas actividades y enfocarnos en aspectos claves y críticos para la calidad de los procesos y productos de software. Para ello nos guiarímos por nuestra experiencia observando proyectos y detectando aquellas facetas que son más proclives a deteriorar dicha calidad. Cada propuesta que hagamos estará orientada a la mejora en el sentido de evitar las causas o minimizar los efectos del deterioro mencionado.

---

8| Mc Connell, Steve. *Software Project Survival Guide: How to Be Sure Your First Important Project Isn't Your Last*. Microsoft Press. 1997.



---

## **Capítulo 3: Trabajo con la Organización - Mejora de Procesos**

---

<b>3.1   Visión del cambio .....</b>	38
3.1.1 Análisis de casos .....	38
3.1.2 Primeros pasos en un proceso de mejoras .....	46
<b>3.2   Trabajando en los cambios .....</b>	53
3.2.1 Forma de trabajo.....	53
3.2.1.1 <i>Modelo ideal</i> .....	54
3.2.1.2 <i>Modelo eoalg</i> .....	55
3.2.2 Dos fenómenos espontáneos .....	55
3.2.2.2 <i>Desconcierto</i> .....	55
3.2.2.3 Procesos virtuales .....	57
<b>3.3   Aspectos y factores del proceso de mejora .....</b>	57
3.3.1 Direcciones del cambio organizacional en sus múltiples dimensiones.....	58
3.3.2 Aspecto socio-cultural de la gestión del cambio.....	58
3.3.3 Factores críticos y de riesgo .....	59
3.3.4 Factores generales de éxito .....	60
3.3.5 Factores de éxito en pymes .....	62
3.3.6 Factores adicionales .....	62
3.3.7 Recomendaciones .....	63
<b>3.4   Gestión del cambio .....</b>	63
3.4.1 Oposición al cambio.....	63
3.4.1.1 <i>Razones de resistencia al cambio</i> .....	63
3.4.1.2 <i>Rescatar lo positivo de la resistencia</i> .....	64
3.4.2 Influencia de la cultura organizacional .....	65
3.4.2.1 <i>Tipos de cultura organizacional</i> .....	65
3.4.3 Comportamientos generadores de conflictos y tensiones en la implementación de procesos de mejora de los roles participantes. ....	66
3.4.4 Estrategia .....	69
3.4.5 Tácticas .....	70
3.4.5.1 <i>Diferentes tácticas para el trabajo con los miembros jerárquicos y con los demás miembros</i> .....	70
3.4.5.2 <i>Diferentes formas de comunicación con áreas jerárquicas y racionales, y canal de comunicación entre ellas</i> .....	72
3.4.5.3 <i>Compartir lugar físico con los miembros de las áreas</i> .....	72
3.4.5.4 <i>Foco en temas puntuales y en grupos reducidos con intereses comunes</i> ....	72
3.4.6 Liderazgo .....	72
<b>3.5   Respuesta a los casos de estudio .....</b>	73
Caso 1 .....	73
Caso 2 .....	74
Caso 3 .....	74
<b>3.6   Conclusión.....</b>	75

# 3

## TRABAJO CON LA ORGANIZACIÓN - MEJORA DE PROCESOS

### **3.1 | VISIÓN DEL CAMBIO**

---

En el capítulo anterior hemos analizado cuáles son las causas del fracaso de los proyectos de desarrollo de software. En este analizaremos y haremos algunas propuestas acerca de cómo cambiar distintos aspectos dentro de una organización, con el objetivo de mejorar la calidad de los productos que en ella se generan. Cualquier cambio, obviamente, será en la dirección de una mejora por lo que llamaremos a este tipo de proyectos Mejora de Procesos. Cualquier cambio generará otros laterales que deberán gestionarse para evitar consecuencias no deseadas. Este capítulo nos explicará qué podemos esperar de los cambios realizados y cómo debemos administrar estos sucesos; cuáles son las cuestiones a tener en cuenta a la hora de planificar los cambios, quiénes podrán ayudarnos, cómo debemos comunicar todas las novedades; cómo preparar una organización para el proceso de mejoras. Tratamos estos temas primero ya que deseamos crear el contexto apropiado en el cual instalar los cambios asociados a la mejora de procesos conociendo los efectos que estos generarán y habiendo desarrollado un criterio acerca de cómo administrarlos.

#### **3.1.1 ANÁLISIS DE CASOS**

Les propuse a mis alumnos que en tres hojas de papel describieran: 1) el contexto de alguna organización que conocieran; 2) enumeren los problemas presentes; y 3) propongan acciones para mejorar la calidad de los productos generados por estas empresas a partir de la resolución de los problemas listados.

Debo aclarar que mis alumnos cursan el quinto año de la carrera de Ingeniería en Informática y que un altísimo porcentaje trabaja desarrollando software y su experiencia es de lo más variada, yendo de algunos meses a años.

A continuación se transcriben tres de los casos seleccionados para hacer el análisis propuesto sobre ellos y así utilizarlos como casos testigos.

---

### CASO 1 – “TECH WEB SOLUCIONES INFORMÁTICAS”

*En una cueva del microcentro porteño funciona “Tech Web Soluciones Informáticas”, un emprendimiento realizado a fines de los 90 cuando la Web empezaba a asomar.*

*Juan Carlos Sabelotodo, un programador Clipper certificado pero sin estudios terciarios, es el dueño y la máxima autoridad de la empresa.*

*La compañía se dedica al desarrollo de sitios Web corporativos, orientando su cartera de clientes a pequeñas y medianas empresas. Realizan sitios con bases de datos, animaciones de flash y además hacen el mantenimiento periódico cobrando un abono.*

*El personal productivo está compuesto por dos diseñadores gráficos, tres desarrolladores junior, un estudiante que hace de dba y a veces desarrolla, y un consultor del área de Marketing llamado Joaquín.*

*Los proyectos de Tech Web generalmente son desarrollos “cortos” (alrededor de dos meses). Cuando el señor Sabelotodo sale a pasear en yate y se hace de una amistad, enseguida arregla para una visita. El consultor se baña en Armani e impregna con sus encantos al Nuevo Cliente.*

*En una rueda de ruleta, se comprometen con un plazo de entrega casi siempre ridículo y recién después se sientan a hablar para ver lo que realmente hay que hacer.*

*Los desarrolladores reciben a diario servilletas de bares de Puerto Madero, con leyendas ilegibles del estilo: “Botones lindos, logo grande, un buscador como Google, que cambie de color cuando se hace click. Algo parecido a Facebook pero en negro”.*

*El primero que se desocupa agarra el intento de toma de requerimientos y comienza a armar la página como le parece. Cuando termina, la sube a un servidor interno y le muestra a Joaquín el trabajo para que le pase al cliente la url.*

*A veces ocurre que se acumulan las servilletas y el desarrollador no sabe cuál tomar primero. Es por eso que Juan Carlos, como lo sabe todo, les obsequió un pinche de pizzería que permite invertir la base y sacar siempre la primera servilleta que llega.*

*Las puestas en producción se hacen luego de que el consultor da la orden. Si algo no llega a funcionar y el cliente llama a Juan Carlos para quejarse, es el desarrollador el responsable y encargado de corregirlo y volverlo a subir.*

*Como la especificación de requerimientos es poco clara y los desarrolladores no tienen contacto con el cliente, se cometan asunciones que generan un retrabajo continuo hasta que la versión sale a producción.*

*A eso se le suman los eternos arreglos y cambios que solicitan los clientes abonados.*

*Bajo ese caótico contexto arriban nuevos proyectos. Al tratarse de sitios Web, hay un gran componente del sitio que sale de diseño y otro que se reutiliza. Los desarrolladores más añejos conocen en qué clientes está el código y con eso ahorran tiempo. Pero cuando el cliente solicita algo novedoso, el proyecto siempre está bajo fuego antes, durante y después del despliegue.*

*Los empleados están bajo constante presión y elevado volumen de trabajo. Siendo la mayoría junior en busca de experiencia, muchas veces abandonan los proyectos de un día para el otro porque les ofrecen un trabajo mejor. En una semana consiguen un nuevo junior, pero la capacitación le resta tiempo al equipo y tienen más aún para hacer.*

*El asunto de la rotación encontró una solución bastante fácil: el dueño regala gadgets baratos al programador estrella del mes, cocina croissants los lunes y les amasa pasta italiana los viernes. Es un fanático de la cocina y el equipo lo adora pese a todo por su costado “paternal”.*

*Juan Carlos quiere crecer y tomar proyectos mayores, pero sabe que con la estructura que tiene está estancado y no puede progresar.*

### Caso 2 – “Shurlatsoft”

*Shurlatsoft es una empresa dedicada por completo al mercado del software libre. Su principal mercado está centrado en clientes franceses y los productos que vende son distribuciones de Linux customizadas, es decir, con el software que el cliente elija.*

*Las distribuciones con las que trabaja la empresa son: Fedora 8, Fedora 9, Fedora 10, Centos y Debian. En el futuro se piensan agregar más distribuciones para customizar. La empresa también comercializa máquinas virtuales customizadas que corren bajo el VMware server 1 y 2 y son migrables al hipervisor ESX. El hyper-v de Microsoft no está soportado actualmente, pero más adelante se trabajará en soportar dicho servidor de máquinas virtuales.*

*Las imágenes customizadas de distribuciones de linux o de máquinas virtuales se entregan en formato DVD a los clientes. La infraestructura de la empresa está integrada por oficinas ubicadas en Buenos Aires, Argentina,*

y en Francia, con un plantel de menos de 20 personas actualmente (en Argentina hay solo dos personas fijas). Pero a medida que la misma crezca se irá incorporando más personal. Además, posee computadoras de última generación con software libre instalado en ellas con lo que se ahorra en licencias que se deberían adquirir en el caso del software privativo.

La empresa realiza desarrollos propios en forma de herramientas que sirven para automatizar el proceso de generación de las imágenes y también para realizar la interfaz web que los clientes utilizarán para comprar sus productos. Se utiliza algo de software privativo para el desarrollo de la interfaz web pero es mínimo. Las principales fuentes de ingresos de la empresa son la venta de las distribuciones/máquinas virtuales customizadas a los clientes y el ofrecimiento de soporte a los mismos sobre los productos adquiridos.

Las principales fuentes de gastos serían recursos calificados, gastos administrativos y hardware, ya que al utilizar software libre no se gastaría en licencias. Como la mayoría del software que utiliza la empresa para crear sus productos, sus desarrollos y para funcionar es libre sus proveedores son repositorios de Internet de donde el software se puede bajar libremente sin gastar un centavo. Para diferenciarse de sus competidores la empresa planea desarrollar productos novedosos de forma que los clientes puedan elegir el perfil de distribución o máquina virtual que necesiten para sus respectivas empresas (los distintos perfiles incluirían distintos paquetes de software que serían agregados a la imagen final o se quitarían de acuerdo a la elección del cliente). En el caso de las máquinas virtuales el cliente podrá elegir el hardware virtual que estas tendrán. Todos estos productos se ofrecerán a un costo competitivo y en un tiempo aceptable. Un problema que tiene esta empresa es que no tiene bien definido un calendario para los proyectos que está encarando por lo cual los mismos se pueden llegar a atrasar o, en el caso extremo, no completarse. También se detectó que las estimaciones realizadas por los empleados en general están colchoneadas (sobreestimadas).

Otro problema detectado es que no se efectuan tantas pruebas sobre las herramientas desarrolladas para automatizar la generación de distribuciones y muchas veces fallan lo cual ocasiona retrabajo para corregir los errores detectados. Tampoco se utiliza ninguna metodología para realizar los desarrollos.

Como es una empresa muy nueva se cuenta con poco personal, y aunque los empleados están altamente capacitados también están sobrecargados de trabajo lo cual ocasiona que las personas se desgasten y rindan menos. De momento no se puede contratar más personas hasta que se lance la beta del sistema web que sirve para que los clientes puedan armar sus distribuciones customizadas y empiecen a comprar los productos que la empresa vende.

Una parte del negocio de la empresa, como se mencionó anteriormente, es la venta de máquinas virtuales customizadas. Se necesitan realizar pruebas de migración de las VMware sever version 1 y 2 al ESX y ESXi pero actualmente no se cuenta con una máquina instalada con dichos sistemas y el único servidor ESX disponible se encuentra en Francia por lo que las

pruebas de migración se están realizando con imágenes de máquinas virtuales de pequeño tamaño (lo más pequeño que se puede crear son 500 Mb), y aun siendo imágenes chiquitas tardan mucho en instalarse en el server de Francia por lo que se pierde mucho tiempo en eso.

La instalación eléctrica de la oficina de Argentina no está terminada y hay muchos cables sueltos por el piso lo cual ocasiona que a veces los empleados desconecten accidentalmente las máquinas en las que trabajan y pierdan, en ocasiones, información importante. También podrían ocasionar caídas y lesiones a los empleados. Este tema debe ser atendido urgentemente.

Actualmente en las oficinas de Argentina no se cuenta con un ambiente que sea exclusivamente para pruebas, solo hay un ambiente de desarrollo y el ambiente de producción está únicamente en las oficinas de Francia.

La empresa no utiliza ninguna técnica de administración de proyectos (WBS, indicadores de control, critical chain, etc.). Solo utiliza control de versiones para sus desarrollos mediante SVN y la comunicación se maneja mediante skype y mail.

Los empleados no cuentan con un WiKi en donde figuren las tareas y errores más comunes y cómo solucionarlos, lo cual acarrearía una pérdida de tiempo a los nuevos empleados que se incorporen ya que habrá que explicarles cómo resolver los errores anteriormente mencionados y como realizar las tareas.

---

### Caso 3 – “Power Light”

El presente caso de estudio corresponde a la empresa EMPRESA S.A. en la que he desarrollado mi actividad profesional en el área comercial dentro del sector de la energía, la industria y la infraestructura.

EMPRESA es una empresa del sector PyME, con sede en Buenos Aires y proyección hacia los principales países de Sudamérica. Su principal área de negocio corresponde a la señalización y al control inteligente del tránsito urbano e interurbano. EMPRESA posee un historial como empresa de la mano de una multinacional alemana, quien cedió su participación accionaria a un grupo inversor de origen colombiano. Desde la constitución de su nueva estructura de gerenciamiento en 2003, EMPRESA ha puesto principal atención en la mejora de la calidad de sus procesos, certificando en 2005 la norma ISO 9001-2000 en Sistemas de Gestión de la Calidad, en 2006 la norma ISO 14001-2004 en Sistemas de Gestión medioambiental y en 2007 la norma OHSAS 18001:1999 en Sistemas de Salud y Seguridad Laboral.

Desde su escisión como unidad de negocios de la multinacional alemana, EMPRESA ha tenido un crecimiento destacable en su volumen de negocio, en la diversidad de productos y servicios y en su cartera de clientes públicos

y privados. Su facturación se ha incrementado desde los 6 millones de pesos en 2003 a los actuales 80 millones de pesos proyectados para el ejercicio en curso.

EMPRESA ha dejado de ser una simple empresa suministradora de sistemas de tránsito, alumbrado público y sus servicios de mantenimiento preventivo y correctivo asociados, para pasar a ser un proveedor de soluciones de alto contenido tecnológico, que superan el área de incumbencia tradicional. De este modo, EMPRESA posee negocios en el sector de la infraestructura civil urbana, centros de monitoreo de la seguridad urbana por video vigilancia, instalación y mantenimiento de subestaciones transformadoras, sistemas de automatización de edificios y negocios de marketing y comunicación visual, entre otros.

Este crecimiento del volumen de negocio asociado a la diversificación de la actividad de EMPRESA ha traído aparejado una complejización de sus procesos, de modo que se ha debido realizar un estudio profundo sobre los mismos para asegurar su calidad. Este aseguramiento está avalado por las certificaciones obtenidas ante los distintos organismos de certificación, pero esta situación no garantiza totalmente la calidad total de sus operaciones, la que debe ser mantenida y monitoreada de manera constante.

En la actualidad existen diversos problemas a nivel organizacional que de cierto modo limitan la calidad de EMPRESA. En este caso, presentaré tres problemáticas concretas que corresponden a áreas de la empresa bien diferenciadas.

**Área de comercialización:** En sus orígenes EMPRESA actuaba en el mercado como proveedora de productos y servicios de un nicho bien determinado: la señalización luminosa y el alumbrado público. Hoy en día la empresa ha diversificado su oferta de productos y servicios de modo que se requieren diferentes perfiles comerciales con diferentes estrategias para atacar cada nuevo sector. La empresa no ha trabajado sobre el perfil de su fuerza de ventas, que sigue teniendo el mismo perfil desde sus orígenes. Por otro lado, la diversidad de productos y servicios ha llevado a una diversidad de clientes. Esto se diferencia de la situación que caracterizaba al pasado, en el que la empresa era dependiente principalmente de un solo cliente. Esto ha ocasionado un aumento en la complejidad del manejo de clientes, que anteriormente no estaba asistida por ningún sistema o herramienta informática.

**Área de desarrollo de proyectos:** La misma problemática presentada anteriormente también tiene efectos en el área de desarrollo de proyectos, responsable de la ejecución de los proyectos comercializados por la empresa. La diversificación de la empresa hacia negocios no tradicionales ha traído como consecuencia una serie de limitaciones en la ejecución de proyectos debido a la especialización tecnológica de los mismos. El sector de desarrollo de proyectos tiene problemas para organizar y ejecutar proyectos que salen de las líneas tradicionales de negocio.

*Los nuevos proyectos, que generalmente son de menor duración y valor económico que aquellos tradicionales, requieren de una dinámica superior. Los tiempos de reacción para estos no son tan dilatados como en el caso de los tradicionales. Si no se posee el expertise requerido, la ejecución de estos proyectos toma una complejidad elevada dentro de la empresa, dificultando el aseguramiento de altos niveles de calidad.*

*Área de administración y finanzas: Como se ha comentado en la sección xxx, la empresa ha aumentado su facturación en al menos un orden de magnitud. Esta situación ha traído aparejado un aumento en el número de transacciones contables y una complejización del control de las mismas. El área de administración y finanzas ha sufrido numerosas transformaciones desde los orígenes de EMPRESA, pero aún no se ha encontrado la solución óptima que garantice una operación estable y sostenible en el tiempo de dicho sector. Desde la escisión de la multinacional alemana, en donde se utilizaba el aplicativo SAP para la gestión administrativa y financiera de la empresa, por motivos de costos de licenciamiento, EMPRESA decidió pasar a un aplicativo contable de bajo costo. Actualmente la empresa opera con el mismo aplicativo de gestión. Con esta herramienta existen numerosas deficiencias y limitaciones que afectan a la calidad de la toma de decisiones en EMPRESA. La información no puede ser procesada y analizada del modo que se requiere, debiendo duplicar la misma en otros aplicativos (EXCEL, por ejemplo) para su análisis, lo que requiere de tiempos prolongados debido a la migración de los datos y al tratamiento manual de los mismos.*

Seguramente las propuestas de mis alumnos a la solución de los problemas que detectaron son razonables y más allá de alguna variante es fácil estar de acuerdo con ellos. Sin embargo ninguno de ellos reparó en las dificultades que cada cambio para lograr la mejora buscada seguramente conlleva.

En este capítulo nos ocuparemos de describir y fundamentar las fuentes de estas dificultades y haremos propuestas para sobrelevarlas.

Para el caso 1, mi alumna propuso instrumentar una base de conocimiento para capitalizar errores y aciertos de los proyectos, y la documentación de los proyectos en marcha que permita a los desarrolladores aprovechar la experiencia pasada y tener visibilidad del contexto de los proyectos en el cual trabajan. Esta alternativa de mejora va en contra de la cultura de la empresa y el manejo personal que el dueño y el marketing hacen de los proyectos. El principal obstáculo será la oposición de ellos argumentando falta de recursos y confidencialidad de la información. Otra acción propuesta es la administración ordenada de requerimientos por un analista, a cambio del manejo personal y discrecional de la actualidad. Las personas que actualmente realizan estas tareas sentirán que su área de trabajo está siendo invadida y se opondrán al cambio. También se propone la planificación de los proyectos, la que actualmente lleva adelante el dueño. Aunque en realidad no lo hace nadie ya que se trabaja sin planificación. El dueño se ocupa del contacto con el cliente y acuerda compromisos que luego delega a

los desarrolladores. Si bien esta persona delega tareas, las mismas son estrictamente técnicas. El intento de introducir una planificación hará que la gestión personal sea desplazada y el dueño no lo permitirá.

¿Cómo se hace entonces para implementar las mejoras propuestas? A lo largo de este capítulo contestaremos esta pregunta.

Para resolver los problemas del caso 2, mi alumno sugiere instalar una forma de planificación que permita trabajar con estimaciones de alcance y esfuerzo para ordenar los proyectos. Sin embargo esta empresa tiene rasgos de ser altamente tecnológica y con ningún apego a los procesos. Parece ser un emprendimiento instalado en el país a efectos de aprovechar buenos recursos humanos locales a bajo costo. La implementación de procesos va en contra de su cultura que se basa en recursos expertos en tecnología sin necesidad de una forma de trabajo. Por esta misma razón la inversión en la realización de pruebas a los productos también será rechazada, al igual que la mejora de la infraestructura. Aunque con el tiempo esto último será implementado porque generará problemas visibles con los cuales no podrán sobrevivir. Esta empresa es la típica outsourcing orientada a maximizar las ganancias a cualquier costo.

¿Es posible implementar mejoras en esta organización?

En una organización como la del caso 3 existen problemas que impactan directamente en la transferencia de los productos a los clientes y en la gestión de la administración comercial. La plataforma de gestión integral propuesta es un proyecto de embargadura que requiere muchos recursos materiales y humanos, y además tiempo para su implementación y aprendizaje para su uso. La decisión de este proyecto depende, en gran medida, del costo beneficio que perciban los responsables de la empresa. El cambio de un sistema de estas características implica una modificación en los procesos de la organización que seguramente por ya haberlo cambiado en una oportunidad se conoce. La capacitación propuesta a los miembros de los grupos de proyecto podrá implementarse siempre que los responsables no la consideren como un gasto sino como una inversión.

En esta organización más grande que las anteriores ¿cuáles son los primeros pasos a dar en el camino hacia una mejora?

Cuando hallamos terminado este capítulo daremos algunas pautas dedicadas a los tres casos presentados.

### **3.1.2 PRIMEROS PASOS EN UN PROCESO DE MEJORAS**

Seguiremos a la teoría de las restricciones en la formulación de los pasos a dar en un proceso de mejoras. Sin embargo antes de trabajar en dicho proceso debe tomarse la decisión de hacerlo. Para que esto suceda son variadas las causas por las cuales los responsables de las organizaciones deciden mejorar y acudir a alguien que los ayude en este proyecto. ¿Cuáles son estas causas?

- Trabajar de una mejor manera para generar un ambiente de trabajo más valorado por sus miembros y captar así a los mejores profesionales al convertir a la empresa en un lugar deseado.
- Mejorar los procesos de desarrollo para mejorar la calidad de los productos generados y hacer a la empresa más competitiva, y como consecuencia de esto ganar una porción mayor del mercado.
- Ser beneficiados por incentivos legales instituidos por políticas públicas y lograr beneficios impositivos que favorezcan las finanzas de la organización.

Estas son algunas de las motivaciones por las cuales una organización decidiría dejar de ser una empresa con proyectos tipo 2 ó 3 y convertirse en una con proyectos de tipo 1.

*Nuestra visión de esta cuestión es la siguiente:*

- Se trata de entender al negocio del desarrollo de software como una actividad en la cual la categorización de los participantes depende de la forma de organización y trabajo.
- Los negocios a los cuales una empresa puede acceder, al momento de ser escrito este libro, están fuertemente condicionados por la calidad de sus procesos, evaluada por algún organismo y según algún modelo de referencia.
- Un mismo proyecto podrá ser llevado adelante con un menor costo cuanto mejores y más maduros sean los procesos con los que cuente la organización.
- Los recursos que no se invierten en mejoras se gastan en trabajos de reparación de errores para disminuir las fallas de los sistemas.

Cualquiera sea la causa del evento que dispare la realización de la mejora de procesos, comenzaremos trabajando de la forma que se muestra en la tabla 3.1. En ella aparecen las preguntas que nos haremos y el orden en que nos las haremos. Si no nos respondemos cualquiera de ellas no podremos avanzar a la próxima.

Preguntas	Objetivos	Nivel de resistencia
¿Qué hay que cambiar?	<ul style="list-style-type: none"> <li>Evaluación de la situación, descripción de la realidad actual, identificación de los problemas principales y supuestos que los sustentan.</li> <li>Diagnóstico, análisis sistémico.</li> </ul>	1. Falta de consenso del problema a resolver.
¿Hacia dónde debe conducir el cambio?	<ul style="list-style-type: none"> <li>Bosquejar visión y solución, describir estrategia.</li> <li>Tácticas y buenas prácticas.</li> </ul>	2. Falta de consenso sobre la dirección que se le debe dar a la solución. 3. Falta de consenso en cuanto a que la solución planteada produce los resultados esperados. 4. Preocupación acerca de que la solución generará efectos no deseados.
¿Cómo implementar el cambio?	<ul style="list-style-type: none"> <li>Desarrollo de tácticas y planes detallados.</li> <li>Sincronización de esfuerzos.</li> <li>Planear, asignar responsabilidades y liderazgo.</li> </ul>	5. Falta de una clara visión del camino y los obstáculos en el logro de la solución. 6. Falta de confianza, temores y dudas acerca del plan elaborado para la solución.

**Tabla 3.1** - Primeros pasos del proceso de mejoras

Es importante saber dónde buscar las respuestas y cómo responder a estas preguntas.

Para respondernos la primera debemos descubrir y listar los problemas de la organización, de los cuales surgirán los aspectos a cambiar. Luego será posible acordar estos cambios con los miembros. Los problemas no son difíciles de detectar ya que su manifestación es visible a los que los padecen y sus consecuencias afectan directamente al negocio. Un síntoma generalizado es mezclar estos con sus causas. Pero en esta primera instancia solo nos interesa determinar los problemas sin causas ni responsables. Por lo tanto acordaremos un listado de inconvenientes que nos permitan direccionar los cambios que a continuación implementaremos. Es importante manifestar explícitamente que no se buscan responsables, para que sea posible un análisis objetivo y despersonalizado.

Cuando se logre contar con el listado de problemas, analizaremos las causas y avanzaremos en un listado de alternativas orientadas a cambiar diferentes aspectos en la empresa considerados causas de estos. Es importante generar un espacio en el cual podamos tener en cuenta la opinión de los diferentes miembros de la organización. Las opiniones que por distintas razones no aporten a nuestro listado, en el futuro se constituirán en obstáculos y resistencia a la mejora de procesos.

Es importante remarcar que durante este proceso los mentores tendrán un comportamiento de facilitadores del trabajo de los miembros. Coordinarán las tareas interpretando los acontecimientos.

Este rol será muy importante en esta etapa ya que determinar los cambios necesarios es una tarea ardua y difícil. Todo el tiempo se deben confrontar propuestas distintas y contrapuestas. Este análisis y su balance en todos los casos deben ser claramente fundamentados a partir de sus ventajas y desventajas.

Una vez que se acuerden los problemas y los cambios a generar, habremos pasado las dos etapas de mayor dificultad. Esto es así porque su logro depende, como dijimos, de acuerdos entre miembros de una organización entre los cuales siempre existen intereses opuestos. A diferencia de la próxima etapa, que depende de la planificación técnica y de los recursos con que se cuenta o no para el desarrollo del proyecto asociado a la mejora de procesos.

Si estas preguntas no son resueltas de una manera franca y efectiva tanto por las personas que implementarán el cambio como por las que se verán directamente implicadas, el cambio propuesto será muy difícil de lograr.

## 3.2 | TRABAJANDO EN LOS CAMBIOS

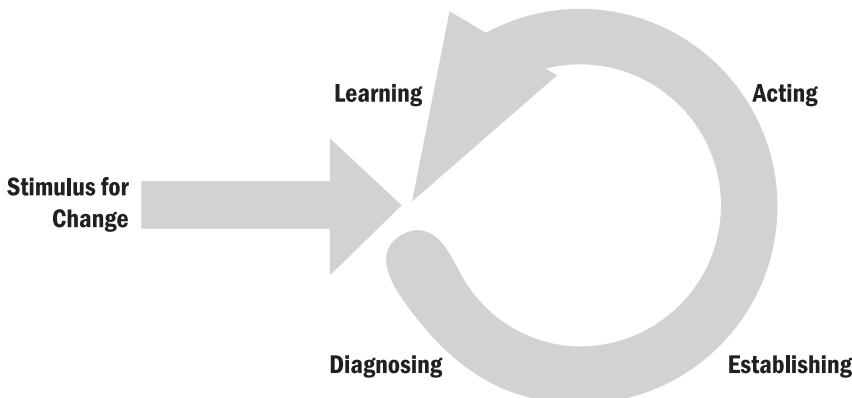
### 3.2.1 FORMA DE TRABAJO

A la hora de trabajar en las mejoras, un estándar de hecho es el modelo IDEAL. Hemos participado en numerosos proyectos en los cuales se desarrollaban las tareas asociadas a las mejoras aplicando este modelo a las diferentes áreas de procesos del modelo CMMI, por ejemplo, como veremos en el capítulo siete. Con esta presentación queremos revisar y cambiar la actitud y forma de trabajo en la utilización de este modelo.

### 3.2.1.1 Modelo IDEAL

*IDEAL : Initiating, Diagnosing, Establishing, Acting, Learning*

En la figura 3.1 se muestra el ciclo de vida de este modelo utilizado con frecuencia en las mejoras de procesos (imagen adaptada del sitio del SEI - Software Engineering Institute).



**Fig. 3.1** – Ciclo de vida del modelo IDEAL

Se trata de trabajar en forma iterativa sobre cada aspecto a mejorar, como lo indica este ciclo. Luego del gap análisis y de la planificación del proyecto se acostumbra utilizarlo como forma de organización de los planes pilotos en las áreas a cambiar. El desarrollo de estos últimos nos permite probar el funcionamiento de los cambios propuestos antes de institucionalizarlos y, por lo general, se llevan adelante en uno de los proyectos de desarrollo en marcha y es la oportunidad de validar las definiciones realizadas en la fase de definiciones y diseño de los cambios. Sin embargo muchas veces este ciclo finaliza como se muestra en la siguiente secuencia de pasos:

1. Inicio
2. Análisis y presentación de resultados
3. Planificación
4. Diseño
5. Implementación
6. Búsqueda de los culpables del fracaso del proceso de mejoras

Con frecuencia este modelo se utiliza como un patrón en los procesos de mejora sin la participación en las fases de planificación y de diseño de todos los involucrados. Es un error que muchas organizaciones cometan. Con el tiempo la fase de Aprendizaje (Learning) se convierte en una búsqueda de los culpables del fracaso en la implementación de los cambios. La no participación de todos los involucrados se debe generalmente a que se tiene la sensación de que no es posible la participación

de todos los miembros, cuando en realidad no se sabe cómo hacer para que todos participen y encausar esa participación. Esta es la causa más frecuente de fracaso en la implementación de cambios y lo que causa la posterior búsqueda de los culpables.

### 3.2.1.2 Modelo EOALG

#### *EOALG: Escuchar y Observar A La Gente*

El modelo IDEAL bien aplicado da muy buenos resultados, sin embargo debe ser tenido en cuenta como una referencia, un marco de trabajo, pero sabiendo que dentro de este marco se debe trabajar con flexibilidad, creatividad y sin miedo a equivocarse<sup>10</sup>.

Por la razón ya expuesta proponemos dar un nuevo sentido a las etapas del modelo utilizando la secuencia de pasos que sigue:

1. Presentar
2. Escuchar y observar
3. Compartir
4. Escuchar y observar
5. Inducir
6. Escuchar y observar
7. Guiar
8. Escuchar y observar
9. Evaluar
10. Escuchar y observar
11. Acordar y cambiar
12. Comunicar

En base a esto y reforzando el primer paso a dar en un proceso de mejoras, según la primera sección, es fundamental la participación de todos los involucrados y la comunicación de todo lo que suceda en el proceso de mejoras.

### 3.2.2 DOS FENÓMENOS ESPONTÁNEOS

A medida que el proceso de mejoras se institucionaliza y avanza en su desarrollo se generan dos fenómenos que si no son bien administrados pueden atentar contra los cambios planificados. A continuación analizamos ambos.

#### 3.2.2.1 Desconcierto

¿Qué ocurre en el tiempo que va desde la decisión de cambiar hasta que el

10 | Margaret J. Wheatley & Myron, Kellner-Rogersx, Bringing Life to Organizational Change, Journal for Strategic Performance Measurement, <http://www.margaretwheatley.com/>, April/May. 2008.

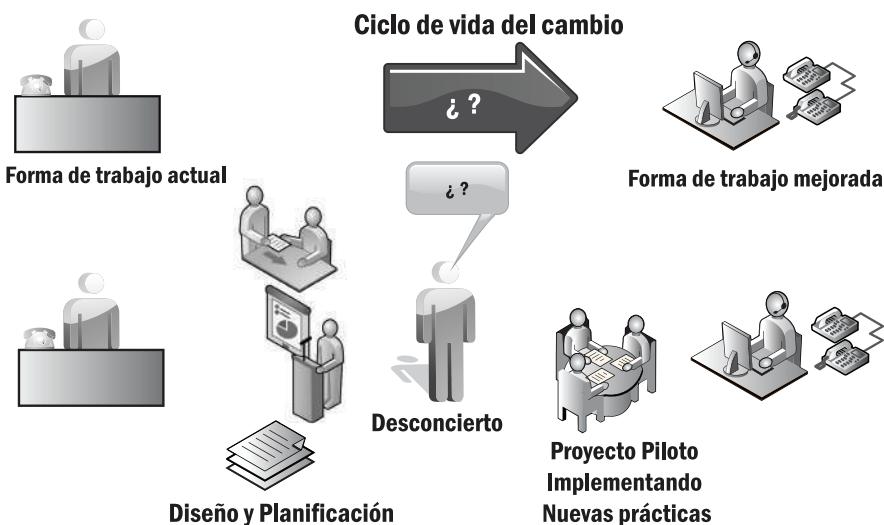
cambio se hace efectivo? Esto se muestra en la parte superior de la figura 3.2. Todo proceso de cambios conlleva una capacitación acerca de las buenas prácticas a incorporar en las actividades básicas que, como ya expresamos, siguiendo a Steve McConnell, son la base de las mejoras. En este proceso, que va desde la forma original de trabajo a la forma mejorada, ocurre uno de los fenómenos que llamamos Desconcierto. Este fenómeno es analizado genéricamente por Tom DeMarco y Timothy Lister, en *Peopleware — Productive Projects and Teams*, 1987.

Los miembros fueron capacitados en la nueva forma pero no tienen experiencia con ella, no tienen madurez, se sienten inseguros, necesitan convencerse de que lo que se les propone es superior a lo anterior.

Esta etapa de desconcierto es el punto de inflexión en el cambio. Si los miembros de la organización encuentran en los mentores el apoyo necesario, terminan reemplazando la forma de trabajo. En cambio si la duda y la inseguridad superan al apoyo y acompañamiento, entonces gana el desconcierto y se vuelve a la forma original, mala pero segura.

En la parte inferior de la figura 3.2 se muestra, a modo de ejemplo, el fenómeno mencionado.

También se muestra que una forma efectiva de acompañamiento es la realización de proyectos pilotos, que luego de analizar los resultados obtenidos y los ajustes necesarios permiten institucionalizar los cambios a partir de esta primera experiencia.



**Fig. 3.2** - Proceso de cambio en tareas que conduce a una etapa intermedia de desconcierto

### 3.2.2.2 Procesos virtuales

El segundo fenómeno se denomina Proceso virtual. Esto se genera a partir de la confusión que muchas veces produce la dilatación en la implementación de los cambios planificados. Existen dos y solo dos formas de trabajo en una organización. La real (original) implementada y la propuesta (a adoptar como consecuencia del cambio). Sin embargo cuando se sobreanaliza y sobrehabla acerca de los cambios sin efectivamente implementarlos, aparece una tercera forma de trabajo que es una mezcla de las dos anteriores. Esta solo existe en la mente de los miembros de la organización. Este fenómeno es también perjudicial para la mejora de procesos, cuyo objetivo debe ser acortar la distancia entre los procesos mostrados según algún modelo tomado como referencia y el realmente implementado. En la figura 3.3 se esquematiza el fenómeno mencionado.

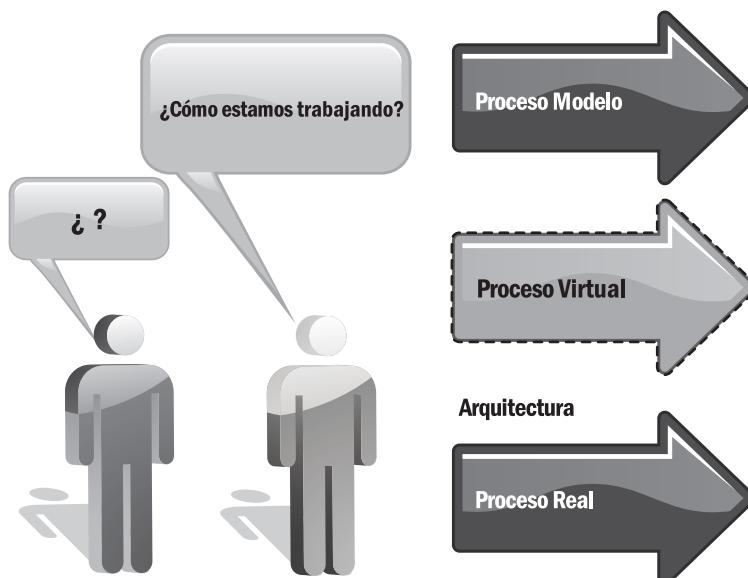


Fig. 3.3 - Los procesos virtuales como deformación de la mejora de procesos

Este fenómeno tácito no es consistente con los principios de las mejoras de procesos, en especial con aquel que postula ir de lo tácito a lo explícito. Explícito significa visible, capaz de ser medido, con el objetivo de ser mejorado. Estos principios se presentan en la siguiente sección.

## 3.3 | ASPECTOS Y FACTORES DEL PROCESO DE MEJORAS

Como habrán observado, a lo largo de todo el libro utilizamos como sinónimos mejoras y cambio, ya que toda mejora implica un cambio y siempre el cambio será para mejorar. Pero ¿cuáles son las direcciones de mejora de estos cambios?

### 3.3.1 DIRECCIONES DEL CAMBIO ORGANIZACIONAL EN SUS MÚLTIPLES DIMENSIONES

Los principios básicos de un proyecto de mejora de procesos son los siguientes:

1. Ir de lo tácito a lo explícito
2. Ir de lo informal a lo formal
3. Ir de lo episódico a lo sustentable
4. Ir de lo individual a lo organizacional
5. Ir de lo intuitivo a lo empírico
6. Ir de lo cualitativo a lo cuantitativo
7. Ir de lo táctico a lo estratégico

*Estos principios buscan evitar:*

1. Conocimiento privado, particular, de personas, en lugar de conocimiento organizacional.
2. La existencia de habilidades solo en manos de quienes por diferentes circunstancias las poseen en lugar de buenas prácticas adquiridas a partir de capacitación y acompañamiento.
3. Buenos resultados logrados por la participación de tal o cual miembro en los proyectos o por las características del proyecto en lugar de formas de trabajo generadoras de estos resultados.
4. Capacitación personal en lugar de capacitación en el entorno de trabajo compartido por los proyectos con los diferentes roles.
5. Decisiones tomadas en función de la experiencia individual en lugar de decisiones basadas en la experiencia capitalizada con un grupo de desarrollo.
6. Calificar los proyectos y sus problemas en función de semejanzas evaluadas informalmente en lugar de métricas sistemáticas obtenidas a lo largo del tiempo.
7. Planificar para hoy y mañana sin saber cuáles son las alternativas a más largo plazo en lugar de planificar a más largo plazo y revisar repitiendo el ejercicio de planificación en forma continua.

### 3.3.2 ASPECTO SOCIO-CULTURAL DE LA GESTIÓN DEL CAMBIO

Entre los objetivos de la gestión del cambio podemos enumerar los más importantes:

- Incrementar la capacidad de los empleados y de la organización.
- Asegurar que la calidad sea un atributo organizacional.

- Alinear los intereses de los individuos con los de la organización.
- Retener recursos humanos y conocimiento críticos para la organización.

Se puede ver que estos objetivos indican que la mejora de procesos expresada a partir de los ellos implica mucho más que un cambio en las actividades diarias. Ignorar este aspecto del proceso de cambio implica la implementación de cambios que no perdurarán en el tiempo o, en otros casos, el fracaso del proyecto. Estos aspectos pueden requerir cambios culturales y de procedimientos muy profundos. Por esta razón importan no sólo estos objetivos sino los pasos para llegar a ellos.

Por eso, la gestión del cambio debe ser abarcativa y ocuparse de administrar todos estos aspectos.

### 3.3.3 FACTORES CRÍTICOS Y DE RIESGO

Algunos factores críticos a tener en cuenta en la planificación de un proceso de mejoras son:

1. Historia de procesos previos de mejora
2. Recursos financieros disponibles
3. Recursos humanos dedicados al proceso
4. Capacidad de ingeniería de software
5. Disponibilidad de soporte técnico
6. Obligaciones contractuales
7. Alcance de la intervención
8. Cultura de la organización
9. Estándares de la industria y la organización
10. Comprensión y apoyo de todos los niveles de gerencia y desarrollo.
11. Presión política de la corporación
12. Objetivos de negocio
13. Visión

Son importantes porque:

1. Procesos previos fracasados demandarán mayor esfuerzo.
2. Recursos no suficientes demandarán mayor esfuerzo de los miembros o la no factibilidad del proyecto.

3. Sin la dedicación del sector de recursos humanos, que también son miembros de la empresa, no es posible implementar las mejoras.
4. Cuánto menor sea el conocimiento de los miembros, mayor será la capacitación necesaria.
5. Sin soporte técnico para las tareas del proyecto no es posible su desarrollo.
6. Condicionan la factibilidad de realización de pilotos en los proyectos.
7. Debe ser claramente establecido el alcance del proyecto para no generar confusión.
8. La cultura debe ser tenida en cuenta ya que condiciona el proceso de cambio.
9. Los estándares deben ser tenidos en cuenta y pueden condicionar el proyecto.
10. Sin el apoyo gerencial se aconseja no comenzar el proyecto.
11. La presión de la corporación hay que tenerla en cuenta y contrarrestarla con planificaciones realistas, nunca ceder ante ella.
12. Los objetivos del negocio deben ser tenidos en cuenta siempre, se derivan de la visión.
13. En el punto de partida, respecto de la visión, debe responderse la pregunta: ¿por qué? Para descubrirla y darle forma.

### **3.3.4 FACTORES GENERALES DE ÉXITO**

Entre los factores que la mayor parte de las veces garantizan el éxito se pueden enumerar:

1. Establecer expectativas realistas.
2. Asegurar el apoyo de los cuadros gerenciales.
3. Identificar las necesidades gerenciales, sus expectativas y la comprensión del problema.
4. Crear un Grupo de Ingeniería de Procesos de Software (con un “campeón” del cambio).
5. Comenzar mejoras poco después de una evaluación.
6. Capacitar a todos los usuarios de los nuevos procesos.
7. Gestión de la dimensión humana del proceso de mejoras.
8. Selección cuidadosa de proyectos piloto.

9. Hacer auditorías de proceso durante la implementación.
10. Hacer encuestas de efectividad de los equipos de trabajo.
11. Comenzar el proceso de mejoras con los ajustes gerenciales más altos (dar el ejemplo).
12. Tener acompañamiento de expertos en cambio organizacional.
13. Mantener relación cercana entre mejoras y objetivos de negocios.
14. Adoptar un vocabulario compartido.

*Porque:*

1. Evita planificaciones fuera del alcance de la organización.
2. Dota al proyecto de compromiso y soporte.
3. Facilita el desarrollo y evita conflictos por afectación de intereses.
4. En este ámbito es donde se generan los cambios.
5. Evita la generación de procesos virtuales, como se describió en la sección anterior.
6. Es esencial como parte del proceso de comunicación.
7. Evita el rechazo generado por los cambios, como veremos en una próxima sección.
8. Para no afectar proyectos críticos y convertir a los pilotos en responsables de los retrasos y fallas de los proyectos.
9. Para corregir desviaciones de la planificación y mantenerlo vivo.
10. Para generar realimentación y conocer la opinión de todos los involucrados.
11. Es parte de la generación de un ambiente propicio de confianza en el que se apoyarán los mentores a la hora de acompañar a los miembros.
12. Las empresas en general no cuentan con recursos especializados en los temas necesarios por una cuestión de costos.
13. Para lograr también la aprobación y conformidad de la gerencia de marketing y comercial al ver cumplidos sus objetivos.
14. Para generar un ambiente en el cual el proceso de cambio tenga identidad.

### 3.3.5 FACTORES DE ÉXITO EN PYMES

A la hora de tener en cuenta el tamaño de la organización deberemos considerar los siguientes factores:

1. Adaptar el marco del proceso de mejoras a las características de la empresa
2. Facilitar el aprendizaje horizontal con la experiencia de otras pequeñas empresas
3. Necesidad de acompañamiento, indispensable dado el número reducido de personal
4. Claridad en el financiamiento de todo el proceso

*Porque:*

1. Una planificación no acorde hará fracasar al proyecto
2. Es una forma de potenciar recursos y aprovechar esfuerzos
3. Para no afectar los procesos de la organización por falta de personal
4. Para contar con una clara visión de hasta dónde se podrá trabajar

### 3.3.6 FACTORES ADICIONALES

Si bien en los aspectos y factores analizados y en los principios listados se enfatizan características de sustentabilidad, organizacional, estratégico y previsibilidad, es interesante e importante mencionar algunas alertas:

- Altos grados de madurez significan altos grados de automatismo
- Contextos cambiantes pueden requerir la habilidad para salirse del “libreto”
- Si la misión cambia, los objetivos de niveles inferiores pueden quedar sin cumplir
- El problema de la “irrationalidad” de lo bien aprendido
- Hay que evitar la pérdida de la capacidad de reacción

En mercados de negocios cambiantes, como lo es el de desarrollo de software, es muy importante contar con la capacidad de adaptación a cambios vertiginosos. Los factores mencionados buscan establecer un alerta cuyo fin es la pérdida de respuesta a los cambios. Esto se logra entendiendo al proceso de cambio como continuo y no como un camino ascendente que lleva a una meseta cuyo nivel es el destino final.

### 3.3.7 RECOMENDACIONES

*De todo lo que se expuso podemos extraer algunas conclusiones:*

- No buscar una certificación como un fin en sí sino como consecuencia del proceso de mejoras
- Compromiso total con la objetividad, la racionalidad y el aprendizaje
- Adecuar el proceso de mejoras y eventual certificación a las características de la empresa (tamaño, cultura, estructura, tipo de negocio)
- Encuadrar el proceso de mejoras dentro de la estrategia global de negocios
- Acompañamiento como apoyo a la gestión de cambio y riesgo incremental

## 3.4 | GESTIÓN DEL CAMBIO

En esta sección analizaremos y propondremos soluciones a los problemas generados por los cambios introducidos en la organización por la mejora de procesos. Una visión positivista de las mejoras ignora estos fenómenos. Pero yo no puedo porque fueron una parte importante de TODOS los proyectos en los que participé y debo ser consistente con los objetivos del libro. Sin embargo debo decir que muchas organizaciones prefieren pasar por encima estos aspectos por distintos intereses. Las empresas a mejorar, porque su cultura positivista les impide hacerlo; las consultoras, porque agregar un ingrediente como este, que puede atentar contra el éxito del proyecto, es considerado una mala estrategia. En todos los casos, cuando las consecuencias por ignorarlos se hacen presentes, casi siempre es demasiado tarde.

### 3.4.1 OPOSICIÓN AL CAMBIO

#### 3.4.1.1 Razones de resistencia al cambio

Todo cambio genera rechazos de parte de las personas afectadas. Las personas somos seres que naturalmente nos acomodamos a situaciones, no estamos naturalmente preparados para vivir en contextos que cambian todo el tiempo. Necesitamos la tranquilidad y la seguridad de lo previsible y establecido.

A continuación se enumeran las causas más frecuentes de resistencia a los cambios:

- El riesgo de cambiar es mayor que el de permanecer en la misma situación que provocó el análisis de la alternativa del cambio.
- Las personas están relacionadas a otras que permanecen identificadas con el *status quo*
- Hay personas que no tienen ningún rol en el nuevo modelo
- Las personas temen la pérdida de competencia ante los cambios

- Las personas se sienten sobrecargadas y abrumadas por nuevas tareas
- Hay personas que poseen un sano escepticismo y desean estar seguras antes de cambiar
- Las personas temen a agendas no comunicadas detrás del cambio que las perjudique
- Las personas sienten que los cambios los afectarán en su identidad
- Las personas presienten pérdidas de estatus y/o calidad de vida
- Las personas no coinciden con los cambios y piensan que son una mala idea

Estas causas están siempre presentes en mayor o menor medida y requieren de una adecuada administración para que no se conviertan en la causa del fracaso del proyecto. ¿Cómo gestionar esta problemática? A continuación damos una recomendación que nos ha resultado de mucha utilidad en varios casos y en algunos de ellos considero que fue la única estrategia de cambio posible.

### **3.4.1.2 Rescatar lo positivo de la resistencia**

Siempre hay conductas, deseos, opiniones y visiones comunes que se expresan paralelamente a los comportamientos de oposición al cambio. Llamamos a estos “emergentes”. Toda esta corriente de pensamiento y opinión puede o no contener aspectos positivos. En la mayoría de los casos los tiene, por lo tanto es necesario interpretarlos a efectos de entenderlos y determinar la relación con el proceso de cambio. La estrategia es encausar estos emergentes a favor de los cambios (ver Margaret J. Wheatley & Myron, Kellner-Rogers, *Bringing Life to Organizational Change*, Journal for Strategic Performance Measurement, <http://www.margaretwheatley.com/>, April/May 1998).

#### *Emergentes*

1. Las organizaciones están compuestas por personas, por lo tanto son organizaciones vivas.
2. Los seres vivos no cambian, reaccionan.
3. Detectar los emergentes en estas reacciones e interpretar su visión.
4. Alinear los intereses emergentes con los del proceso de cambio.

La razón de que los emergentes constituyan una herramienta de gestión del cambio está dada por lo siguiente:

- Las personas en general desean y disfrutan de sentirse útiles.
- Las personas necesitan ser reconocidas por su trabajo.
- Las personas gustan de aprender y aplicar buenas prácticas.
- Las personas necesitan sentirse parte de la organización en la que trabajan.

Los personas trabajan donde trabajan por gusto profesional aunque no comparten algunas políticas de la organización.

El trabajo con estos aspectos está directamente relacionado a la aplicación del modelo EOALG (Escuchar y Observar A La Gente) ya presentado. Este modelo es la herramienta para la detección y capitalización de los emergentes a partir de la participación en el proyecto de los miembros de la organización.

### 3.4.2 INFLUENCIA DE LA CULTURA ORGANIZACIONAL

Una definición de cultura organizacional, de las varias que existen, es:

“La cultura de una organización es una estructura cognitiva y social. Sus miembros comparten creencias y entendimientos, basados en valores comunes y un cuerpo de conocimiento compartido que guía sus acciones”<sup>11</sup>.

En esta sección daremos algunos ejemplos para mostrar la influencia de la cultura en el proceso de mejoras.

#### 3.4.2.1 Tipos de cultura organizacional

En la tabla 3.2 se muestran los tipos de cultura y las características de relevancia desde el punto de vista de la mejora de procesos. A continuación daremos ejemplos de ellos.

Dimensiones	Jerárquica	Racionalista	Consensual	Desarrollo Humano
Estructura	Reglas y rutinas formales	Basada en la capacitación	Basada en la colaboración grupal	Basada en la colaboración grupal
Claves	Estabilidad y control	Productividad y eficiencia	Cohesión y moral	Flexibilidad, adaptabilidad, centrado en la gente
Objetivos	Cumplimiento de procedimientos	Seguimiento de objetivos	Mantenimiento el grupo	Crecimiento y desarrollo de los miembros
Comportamiento ante cambios	Resistencia, statu quo	Abierta a cambios por objetivos	Abierta a cambios	Cambios naturales y parte del desarrollo

**Tabla 3.2** - Tipos de cultura organizacional y sus rasgos salientes

**Jerárquica:** Para aclarar la definición mencionamos a las organizaciones militares como ejemplo, aunque estén fuera del interés de este trabajo. Dentro del tipo de organizaciones vinculadas al desarrollo de software podemos citar a las áreas de organizaciones con mezcla de tareas operativas y de desarrollo. Esta no es una buena

11 | Morgan, G. *Images of Organization*, 2nd. ed., Newbury Park, CA: Sage. 1997

forma de organización, sin embargo es frecuente encontrar estas combinaciones siempre con pésimos resultados. En estos casos se deberá gestionar el cambio ya que un rasgo saliente de este tipo de cultura es su oposición al mismo.

**Racionalista:** El ejemplo típico es la software factory, con procesos adquiridos que siguen un modelo de tipo CMMI. Un aspecto saliente de esta cultura es el foco en los objetivos sin considerar otros aspectos. Por esta razón se deberá trabajar en la gestión del aspecto humano del cambio para que no se convierta en la causa del fracaso del proyecto.

**Consensual:** Organizaciones no gubernamentales sin fines de lucros. Es un tipo de cultura por lo general ajena a nuestro interés.

**Desarrollo Humano:** Comunidades científicas, para las cuales es válida también la consideración anterior.

Es común observar escenarios contradictorios cuando dos áreas de una organización poseen rasgos de distinto tipo de cultura y deben comunicarse para llevar adelante su trabajo. Un ejemplo de esto es un Banco en el que coexisten áreas de desarrollo que proveen productos de software y áreas operativas de administración de la infraestructura de IT. Debe gestionarse este aspecto puntual y encontrar la forma de comunicación adecuada, de lo contrario el conflicto impedirá el desarrollo de proyectos comunes.

Los conflictos generados por la introducción en el contexto de la empresa de nuevas variables son muy diversos y se manifiestan de distintas maneras. En la próxima sección analizaremos este tema.

### **3.4.3 COMPORTAMIENTOS GENERADORES DE CONFLICTOS Y TENSIONES EN LA IMPLEMENTACIÓN DE PROCESOS DE MEJORA DE LOS ROLES PARTICIPANTES**

Hemos elegido desarrollar este tema a partir de los roles porque son los protagonistas de los sucesos acontecidos a lo largo del proyecto de mejoras.

A continuación detallamos los roles típicos de organizaciones dedicadas al desarrollo de software, a los que les hemos sumado el de los mentores que acompañan a la organización en el proceso:

- CEO
- Gerencias y Responsables de QA(Quality Management)
- Miembros desarrolladores
- Departamento comercial
- Recursos humanos
- Consultores / Mentores

Cada uno de ellos contribuye de diferente forma a la generación de conflictos.

En el siguiente cuadro mostramos las causas y las consecuencias.

Rol	Causas	Nivel de resistencia
<b>CEO</b>	<p>Adopción de la certificación como objetivo exclusivo en sí mismo.</p> <p>Doble mensaje según audiencia (idealiza las mejoras hacia afuera, enfatiza el pragmatismo de la certificación hacia adentro).</p> <p>Falta de comprensión del concepto “Madurez” y de sus consecuencias.</p> <p>Evaluación inadecuada de la aptitud del personal para ajustarse a nuevos roles priorizando el mantenimiento de privilegios según la administración anterior.</p> <p>Incomprensión de los nuevos roles derivados del proceso de mejoras.</p> <p>Falta de objetivos claros de negocio.</p> <p>Administración personalizada y subjetiva de la organización.</p>	<p>Conflictos personales por actitudes no adecuadas.</p> <p>No compromiso con el proceso de mejoras.</p> <p>No entendimiento y no cumplimiento del proceso.</p> <p>Distorsión de la cadena de responsabilidades.</p> <p>Caos en los proyectos.</p> <p>Trabajo no realizado de acuerdo a conocimiento y principios.</p>
<b>Gerentes</b>	<p>Indecisión frente a los nuevos escenarios de trabajo.</p> <p>Incapacidad de adaptarse a cambios técnicos y gerenciales por falta de preparación profesional.</p> <p>Mala comunicación por falta de adaptación a los nuevos roles en la organización.</p> <p>Centralización de tareas, por falta de confianza, ante los nuevas escenarios.</p> <p>Incapacidad para resolver conflictos creados por los cambios que se producen.</p> <p>Concentración de poder en gerencia de QA que resultan ser “dueños” virtuales del proceso de mejoras.</p>	<p>Se frena la evolución y dinámica del proceso de mejoras.</p> <p>Se erosionan las pautas de respeto mutuo y se genera desconfianza, inseguridad, sensación de caos, pérdida de rumbo.</p> <p>Imagen de falta de confianza.</p> <p>Sensación de que QA no hace trabajo objetivo y la instalación del viejo mito de Desarrollo versus QA.</p> <p>Generación de cambios en el proceso por parte de QA sin definición previa ni acuerdo por el PEG (Process Engineering Group, CMMi).</p>

<b>Desarrolladores</b>	Falta de conocimiento y experiencia en trabajo en grupos.  Los project leaders no se convierten en los representantes del proceso en los proyectos.  Falta de conocimiento de procesos.  Confusión ante los canales de comunicación cambiantes.  Falta de experiencia en la relación con clientes.  Falta de profesionalismo.	Conflictos personales por actitudes no adecuadas.  No compromiso con el proceso de mejoras.  No entendimiento y no cumplimiento del proceso.  Distorsión de la cadena de responsabilidades.  Caos en los proyectos.  Trabajo no realizado de acuerdo a conocimiento y principios.
<b>Agentes comerciales</b>	Desconocimiento del proceso de desarrollo.  Desconocimiento del producto a vender.  Falta de compromiso con la adquisición de estos conocimientos.  Falta de profesionalismo al no respetar estimaciones.  Oposición ante cambios en el proceso de preventa y relación con clientes.	Pérdida de posibles proyectos.  Mala relación profesional y personal con miembros desarrolladores.  Incompetencia para vender.  Traba en el proceso de preventa.  Doble imagen ante clientes.
<b>Recursos Humanos</b>	Incomprensión del proceso de mejoras.  Desconocimiento de la formación previa necesaria.  Desconocimiento de la capacitación futura necesaria.	Incumplimiento del rol de contención de los miembros ante los cambios.  Errores en la contratación de personal.  Mala comunicación.  Imposibilidad para estructurar planes de carrera.
<b>Consultores/ Mentores</b>	Transmisión de conocimientos incompleta por falta de sistematización del conocimiento tácito. No abundancia de casos de éxito en el mercado.	Genera sensación de pérdida de tiempo en los miembros de la organización.  Genera sensación de que los objetivos no serán logrados.

**Tabla 3.3** - Los roles del negocio de desarrollo de software como generadores de conflictos ante la mejora de procesos

Cada una de estas fuentes de conflictos se deben tratar generando las condiciones para que no prosperen. La forma de hacerlo es detectarlas y atacarlas individualmente. Sin embargo la estrategia general es generar buenos canales de comunicación para que a nadie le falte información en la organización en lo referente al proceso de mejoras, crear escenarios de participación para que todos los afectados puedan intervenir y establecer un programa de capacitación continua para que haya un conocimiento acabado de cada uno de los aspectos referidos al proyecto.

La conclusión que extraemos de este análisis es que es necesario y muy importante la gestión del cambio. Para ello se propone un programa de gestión de cambios organizacionales generados por la mejora de procesos basado en los siguientes principios<sup>12 13</sup>:

- Acompañamiento, trabajo compartido diario
- Aporte de contenidos, qué hacer y cómo hacerlo
- Gestión de cambios organizacionales, compatibilidad de cambios, nuevos mecanismos y estilos de comunicación, aprendizaje organizacional

#### 3.4.4 ESTRATEGIA

La siguiente estrategia es general y debería ser adaptada a cada caso. Sin embargo incluye los ingredientes esenciales que hemos fundamentado en las secciones anteriores. La estrategia para un proyecto de mejoras exitoso debe incluir las siguientes acciones:

- Acordar visión y objetivos con la autoridad, ya que sin autoridad es imposible dar directivas y que la gente realice su trabajo siguiéndolas.
- Crear canales de comunicación y comunicar, ya que el obstáculo mayor es a veces la falta de información más que la oposición al cambio real.
- Detectar liderazgo y alinear con el cambio<sup>14</sup>, ya que necesitamos de referentes internos a la organización que guíen y generen confianza en los involucrados en los cambios.
- Convertir la reacción en Emergentes y utilizarla en la Planificación, ya que la gente siempre está interesada en realizar un trabajo de mejor calidad por el cual sentirse reconocida.
- Siempre detectando y acordando cuáles son los problemas a resolver, qué cambiar para resolverlos y cómo implementar dichos cambios.

12 | Ing. Pantaleo, Guillermo. Los gerentes como los políticos deben ser los agentes del cambio, PhD. Juan D. Rogers. [http://www.it-mentor.com.ar/noticias/it-news/200803/it-mentor\\_news.htm](http://www.it-mentor.com.ar/noticias/it-news/200803/it-mentor_news.htm). Marzo, 2008.

13 | Ing. Forradellas, Patricia, Ing. Guillermo Pantaleo. El modelo CMM/CMMi. Garantizando el éxito del proceso de mejoras en las organizaciones superando los conflictos y tensiones generados por su implementación, <http://www.it-mentor.com.ar/>. (Ver página de noticias.)

14 | Michael Fullan. *Leading in a Culture of Change*. 2003.

### 3.4.5 TÁCTICAS

Las tácticas comunes a seguir para el desarrollo de las diferentes tareas son:

- Detectar agentes de cambio y trabajar con ellos como agentes multiplicadores
- Seleccionar temas (tareas) puntuales, NO todo a la vez. Trabajar en forma incremental
- Trabajar con la gente en sus lugares de trabajo
- Rescatar lo bueno, bien hecho y conservarlo como parte de la mejora
- Fundamentar los cambios y mostrar resultados con frecuencia

Además de las anteriores existen algunas otras que requieren una explicación extendida, por eso las tratamos aparte.

#### 3.4.5.1 Diferentes tácticas para el trabajo con los miembros jerárquicos y con los demás miembros

Los rasgos que caracterizan la cultura organizacional pueden estructurarse en tres niveles, los cuales capturan aspectos distintos de estos rasgos. Es interesante notar el comportamiento dinámico de estos últimos contenidos en los niveles y cómo se manifiestan de forma diferente en los roles de una organización. De hecho este mecanismo funciona de manera distinta en los miembros que ocupan roles jerárquicos que en los demás miembros. El modelo del cual hablamos nos ayudará a explicar la variedad de comportamientos.

Los niveles mencionados se muestran en la figura 3.4.



Fig. 3.4 - Expresiones de la cultura organizacional

1. Los artefactos son visibles a través de las actividades diarias, son la parte visible de los miembros de una organización.
2. Los valores y creencias son los valores a los cuales la organización se adhiere y así esta dispone que determinadas acciones son juzgadas como ejemplares y otras como desechables.
3. Los supuestos básicos incluyen la visión más profunda que poseemos acerca de la realidad. Nuestras verdades fundamentales acerca de la gente y el mundo.

Existe una realimentación de los artefactos que contribuyen a la generación y adaptación de los supuestos básicos.

A partir de este modelo<sup>15</sup>, puede verse que debido a que no todos los niveles son observables, es difícil caracterizar a una cultura organizacional a partir de la interacción consus miembros. Sin embargo, cuando en situaciones de estrés, tensión y/o nerviosismo, la dinámica del comportamiento se ve alterada, se ponen de manifiesto inconsistencias entre los aspectos visibles (artefacto) y los ocultos (valores y supuestos).

Estos escenarios se generan a menudo en un proceso de mejoras cuando los mentores relevan los procesos activos y el conocimiento de la organización. Entonces los miembros jerárquicos responsables de la calidad sienten vergüenza por las falencias que presenta su trabajo profesional. En estos casos se simulan actitudes que entran en conflicto con los supuestos básicos de que el trabajo en equipo, ordenado y sistemáticamente definido es inútil y no aporta a los proyectos.

En cambio, los miembros no jerárquicos (no responsables del estado de cosas) actúan en estos escenarios con mayor transparencia ya que no se sienten responsables de las falencias expuestas.

Sin embargo, todos pertenecen a la misma organización, lo cual muestra que algunos modificaron sus rasgos visibles por la situación analizada. Los miembros jerárquicos se sienten en falta por ser los responsables del estado de la organización, por esta razón no rechazan y simulan el deseo de mejorar. En este caso el mecanismo de los rasgos culturales ha modificado su funcionamiento, el cual se observa por comparación con los otros miembros.

Los miembros jerárquicos se comportan como miembros de una organización racional, cuando en realidad pertenecen a una organización jerárquica. Elaboran y acuerdan planes que nunca siguen. No trabajan por objetivos y no se comprometen con los proyectos. En este contexto hay que analizar y planificar las acciones que mantengan todo el tiempo esta problemática visible.

Los miembros no jerárquicos se comportan como lo que son, por esta razón los responsables hacen que delegan responsabilidades en las personas de sus grupos pero estos no la asumen porque necesitan el control y el empuje diario típico de su cultura.

Como se observa, después de este análisis, a efectos de trabajar en el proyecto de mejoras, debemos utilizar distintas tácticas con unos y otros.

15 | Dooley, Jeff. Cultural Aspects of Systemic Change Management. <http://www.well.com/user/dooley/culture.pdf>

### **3.4.5.2 Diferentes formas de comunicación con áreas jerárquicas y racionales, y canal de comunicación entre ellas**

En las organizaciones existen áreas de tecnología muy operativas, como lo son las responsables de la infraestructura, las cuales tienen los rasgos típicos de organizaciones jerárquicas. Otras, en cambio, son más racionales tal como las de desarrollo de software. En cada una es recomendable reforzar los rasgos característicos más apropiados, sin embargo hay que planificar cuidadosamente la forma de comunicación entre estas áreas.

### **3.4.5.3 Compartir lugar físico con los miembros de las áreas**

Considero fundamental la integración de los miembros mentores y los de la organización a mejorar. Pero esta no se logrará si no se comparte mínimamente un mismo lugar de trabajo.

### **3.4.5.4 Foco en temas puntuales y en grupos reducidos con intereses comunes**

A efectos de obtener la colaboración de los miembros de la organización en la mejora de procesos es primordial lograr acuerdos y trabajar en los temas de interés. La táctica debe ser detectar estos temas con la participación protagónica de los miembros y focalizar el trabajo en alguno, y una vez logrado los objetivos de este, trabajar sobre uno nuevo.

En general los temas surgen como emergentes de los diferentes grupos de proyecto. Esta es la mejor estrategia para programar y planificar el proyecto, ya que permite lograr compromiso de los miembros. Además estos emergentes constituyen el mejor indicativo del estado de la organización.

## **3.4.6 LIDERAZGO**

Un asunto esencial a tener en cuenta: nunca te embarques en un proceso de mejoras sin el empuje y el compromiso de los dirigentes de la organización en la cual se llevará adelante el proyecto. A veces el liderazgo recae en los mismos dirigentes, otras no. En cualquier caso una condición indispensable es la participación de los líderes con el empuje de los dirigentes.

En la figura 3.5 se muestran los valores que hacen a una persona líder (Michael Fullan, 2003, op. cit.). Algunas definiciones que consideramos esenciales al momento de establecer el liderazgo son:

- Un líder no es aquel que enseña cómo se hacen determinadas tareas para lograr un objetivo usando su conocimiento, sino aquel que sin saberlo muestra el camino para lograrlo y motiva a los miembros a su cargo.
- Una organización necesita de líderes en todas sus áreas.
- Para que esto sea posible debe formar parte de la cultura de la organización la delegación de decisión (libertad) para ser usada con responsabilidad.
- Para que esto sea posible debe formar parte de la cultura de la organización

la asunción de responsabilidades, otorgadas por la delegación, en la toma de decisiones.

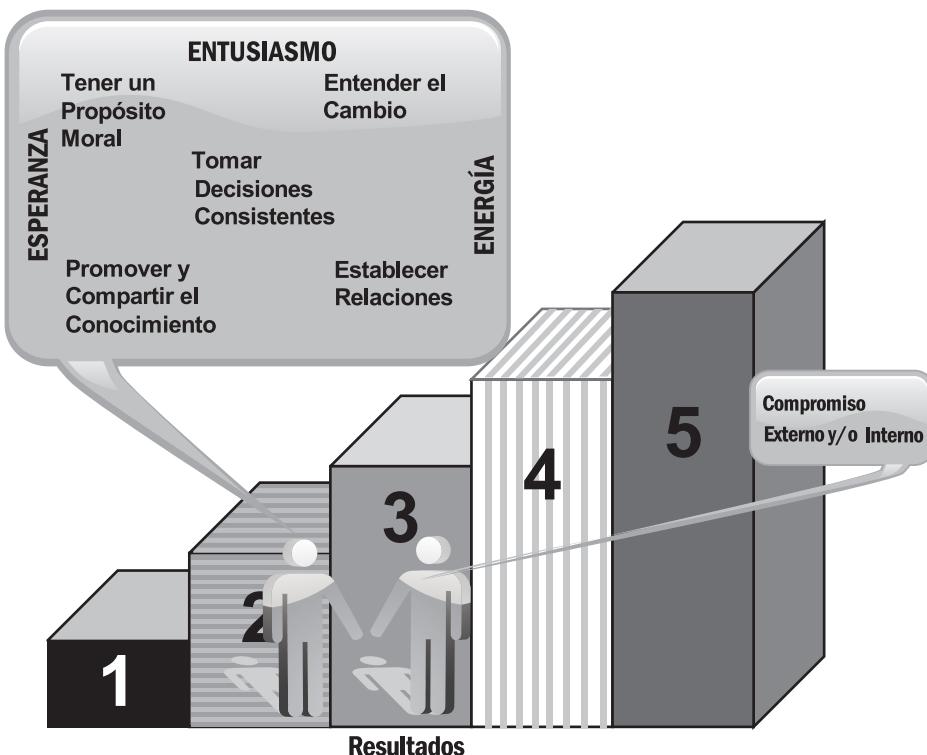


Fig. 3.5 - Condiciones de liderazgo

### 3.5 | RESPUESTA A LOS CASOS DE ESTUDIO

Propondremos la implementación de mejoras a los casos planteados al comienzo del capítulo siguiendo la guía de los tres pasos, la estrategia y tácticas planteadas en las secciones anteriores.

#### Caso 1

Está claro que en una empresa pequeña como “Tech Web Soluciones Informáticas” y con un manejo personal por parte de su dueño, la decisión de implementar mejoras depende fuertemente de su convencimiento. Primero debemos identificar cuáles de las consecuencias de la forma de trabajo actual deterioran su participación de administrador de la relación

con los clientes. Algo que lo moleste, que atente contra su ego y su imagen de administrador dominador del negocio. Debemos acordar con él que este es un problema que lo afecta y convencerlo de que lo podremos solucionar. Paralelamente, haciendo uso de los emergentes, debemos vincularlo con el resto de los problemas ya conocidos y acordados por todos los miembros. La estrategia es generar un plan de mejoras motivado por la solución al problema que afecta al responsable de la empresa. Cambiar y mejorar utilizando buenas prácticas aplicadas a problemas acotados y mostrando resultados con frecuencia. La cultura de esta empresa es una mezcla de jerárquica (dueño-jefe) con conductas de acercamiento a la gente (rasgos consensuales). Se recomienda no introducir razonamientos racionales en extremo porque chocarán con la cultura actual y serán rechazados.

---

## CASO 2

Es muy posible que los problemas derivados del mal trabajo afecten la imagen de la empresa de líder tecnológica que sus responsables le han querido imponer. Es importante mostrar alternativas de soluciones posibles de implementar con pocos recursos. Tratar de implementar alguna mejora simple y mostrar sus resultados. La estrategia es lograr el “ok” para la implementación de mejoras sin recursos adicionales, en primera instancia, y dejar para después las mejoras que necesitan recursos importantes, para cuando se haya instalado el convencimiento de la necesidad y la factibilidad de mejorar. Utilizar herramientas free / open source y tiempos asignados a tareas de proyectos. Aprovechar el carácter racional de este tipo de organización para mostrar las contradicciones de los errores que se cometen con los objetivos visibles de convertirse en líderes en su ramo. No se debe encarar la mejora ignorando los objetivos de la empresa, que hacen a su cultura, la cual es maximizar las ganancias a partir de un menor gasto en recursos humanos.

---

## Caso 3

Este caso presenta la dificultad de las empresas más grandes, en las cuales es difícil llegar a quienes deben decidir acerca de la mejora. En la cultura de estas empresas las mejoras se piensan como grandes desembolsos y grandes cambios. Si la actual situación afecta a todos los miembros, deben alinearse las disponibilidades para las mejoras y hacer un balance de cómo con esos recursos se podrían resolver algunos de los problemas. Comenzar por las capacitaciones, las que pueden ser internas en el caso de contar con algún referente en los temas de interés. De lo contrario contratar cursos externos a desarrollar preferentemente en el ámbito de la empresa. La estrategia es mostrarles a los responsables que existen ganas y voluntad de mejora. La empresa paga los cursos y los empleados se quedan fuera

*de hora para tomarlos. En el caso de contratar apoyo externo, hacer una evaluación del tipo de organización que se necesita. Los proveedores de herramientas siempre aconsejarán su adquisición, por lo tanto se puede optar por alguien que proponga analizar los procesos de la organización para tomar decisiones. La ventaja de esta empresa es que aparentemente es de cultura racional, lo cual indicaría que si se tiene la oportunidad de exponer razones a las personas indicadas se podrán implementar las mejoras. La desventaja es que al ser grande los responsables a veces no asumen la autoridad que su rol les otorga, ingrediente esencial a la hora de implementar un proceso de mejoras.*

### **3.6 | CONCLUSIÓN**

---

Un proceso de cambio, es decir, de mejoras, no es posible sin el compromiso de los dirigentes de la organización, sin canales de comunicación bien establecidos, sin miembros de la organización con condiciones de liderazgo alineados al proceso y sin la canalización de los emergentes que expresan el deseo de la mayoría de sus miembros.

---

## **Capítulo 4: Trabajo con Requerimientos**

---

<b>4.1   Importancia de los requerimiento</b> .....	77
4.1.1 El rol de analista.....	78
4.1.1.1 Definición.....	78
4.1.2 ¿Qué son los requerimientos?.....	78
4.1.3 ¿Para qué sirven?.....	78
4.1.4 ¿Cuál es el impacto en un proyecto de desarrollo de software?.....	78
<b>4.2   Tareas asociadas a los requerimientos</b> .....	78
4.2.1 Foco .....	82
4.2.2 Nivel.....	83
4.2.3 Vista .....	84
<b>4.3   Estrategia y tácticas en el trabajo con requerimientos</b> .....	86
4.3.1 Estrategia .....	86
4.3.2 Tácticas .....	88
4.3.2.1 Especificación de requerimientos de software y sus atributos de calidad. ....	88
4.3.2.2 Especificación de casos de uso .....	89
<b>4.4   Análisis de requerimientos</b> .....	91
4.4.1 No confundir dominio y negocio con diseño .....	91
4.4.1.1 Nota para desarrolladores ágiles .....	95
4.4.1.2 Nota a los analistas de sistemas .....	96
4.4.2 Paquetes .....	96
4.4.2.1 Alternativas de selección .....	96
<b>4.5   Validación y verificación</b> .....	98
4.5.1 Validación .....	98
4.5.2 Verificación .....	100
<b>4.6   Administración de cambios a los requerimientos</b> .....	100
4.6.1 Problema .....	102
4.6.2 Alternativas de solución .....	102
4.6.2.1 Nota para desarrolladores ágiles .....	102
<b>4.7   Conclusión</b> .....	102

# 4

## TRABAJO CON REQUERIMIENTOS

### 4.1 | IMPORTANCIA DE LOS REQUERIMIENTOS

---

En el capítulo 2 hemos analizado cuáles son las causas del fracaso de los proyectos de desarrollo de software. En este, después de haber establecido las condiciones apropiadas en el contexto de la organización y su área de desarrollo, administrando las consecuencias generadas por los cambios, nos ocuparemos del trabajo con los requerimientos.

En una gran cantidad de proyectos de las más variadas empresas, el trabajo con los requerimientos está desprestigiado. Es visto como una actividad de segunda categoría, siendo el diseño y la programación las primeras actividades. Esta visión de las actividades relacionadas a los requerimientos produce una realimentación positiva en el sentido de que se le destinan menos recursos para la adquisición de herramientas y capacitaciones. También, aunque parezca increíble, se asignan menos tiempo a la concepción de los sistemas a desarrollar que a las otras tareas de los proyectos, con las consecuencias imaginables. Los fabricantes y vendedores de tecnología son en gran parte responsables de esta desvirtuación al promover la sensación de que si se domina tal o cual framework o herramienta basta para convertir a los proyectos en exitosos. La confusión llega a tal nivel que también los analistas son vistos como desarrolladores de segunda clase. En algunos grupos que he conocido, en los cuales se trabajaba utilizando alguna metodología ágil, ninguno de los desarrolladores desarrollaba el rol de analista, como si bastara con la cercanía del cliente para que los requerimientos del sistema a desarrollar se plasmaran solos, por generación espontánea, en un listado o diagrama de casos de uso. Solo atinaban a analizar history boards en targetas con escenarios no siempre entendidos y casi nunca sistematizados.

El trabajo con requerimientos necesita de un rol especializado que lo conduzca y ese es el analista.

## 4.1.1 EL ROL DE ANALISTA

### 4.1.1.1 Definición

El analista es el rol que en un grupo de desarrollo de software posee una doble interfaz, una hacia el negocio y otra hacia el seno del grupo de desarrollo.

**Interfaz de Negocio:** para implementar esta interfaz el analista debe estar capacitado en técnicas de relevamiento de requerimientos, UML (diagramas de actividad, estado y secuencia) y documentación de procesos de negocio.

**Interfaz de Desarrollo:** para implementar esta interfaz el analista debe estar capacitado para trasladar el producto de su trabajo con los usuarios y clientes al seno del grupo de desarrollo. Para esto debe saber UML (clases, casos de uso y paquetes), participar en la construcción de un modelo de dominio y debe tener conocimientos básicos de arquitectura de sistemas.

En consecuencia, necesitamos un profesional capaz y con muy buena formación, ya que el rol en cuestión es muy abarcativo.

En el mercado laboral argentino de la informática y, por consiguiente, en las empresas existe un concepto asociado a este rol que es viejo y obsoleto. Se piensa en él como aquel que releva requerimientos sin ninguna formación tecnológica. Esta visión está ligada a paradigmas superados y corresponde a la que se tenía hace dos o tres décadas. Debo decir que contribuyen a esta visión algunas carreras universitarias con planes no siempre debidamente actualizados.

Para comenzar entonces a tratar el tema de trabajo con los requerimientos comenzaremos respondiéndonos las siguientes preguntas básicas:

### 4.1.2 ¿QUÉ SON LOS REQUERIMIENTOS?

- Definen lo que un sistema permite hacer desde el punto de vista del usuario (*funcionales*)
- Definen condiciones de funcionamiento del sistema en el ambiente operacional (*no funcionales*)
- Definen las condiciones a cumplir durante el desarrollo (de proceso).

### 4.1.3 ¿PARA QUÉ SIRVEN?

- Definir el sistema a desarrollar
- Comunicar y acordar el alcance y las prioridades
- Planear el proyecto de desarrollo

### 4.1.4 ¿CUÁL ES EL IMPACTO EN UN PROYECTO DE DESARROLLO DE SOFTWARE?

- La calidad del trabajo con los requerimientos determina la buena o mala evolución y posterior terminación del proceso de desarrollo del proyecto

- El costo del proyecto puede verse influenciado en forma determinante por la calidad del trabajo con los requerimientos
- Las fallas más graves en el desarrollo de sistemas se deben a las generadas por errores en el trabajo con los requerimientos.

Para respaldar y justificar nuestras propuestas acerca de la importancia de los requerimientos en el desarrollo de software apelaremos a un par de publicaciones que nos ayudarán a fundamentar algunas de las aseveraciones anteriores. La primera es el informe del Standish Group<sup>16</sup> y el segundo es un artículo publicado hace ya algún tiempo llamado Requirements Development, Verification, and Validation Exhibited in Famous Failures<sup>17</sup>.

Los resultados del primero se mostraron en la tabla 2.1 del capítulo dos. Allí se pueden ver las causas por las cuales los proyectos presentados en dicho informe se comportaron de la forma mostrada. La tercera de las causas enumeradas en dicho informe fue “un mal entendimiento de los requerimientos”. ¿Qué significa un mal entendimiento? Significa lo siguiente:

- Falta de conocimiento del negocio por parte del cliente y de los usuarios
- Falta de comunicación del conocimiento que se tiene del negocio por parte del cliente y de los usuarios
- Falta de colaboración con los desarrolladores en el relevamiento y análisis por parte del cliente y de los usuarios
- Falta de planificación de las tareas de trabajo con los requerimientos por parte de los desarrolladores
- Falta de conocimiento de la forma de construir modelos de requerimientos que faciliten el posterior diseño por parte de los desarrolladores
- Falta de comprensión del concepto clave acerca de que el desarrollo debe siempre ser conducido por los requerimientos, por parte de los desarrolladores
- Falta de validación y verificación de los requerimientos por parte de los desarrolladores

En resumen, el no entendimiento de los requerimientos significa no conocer o reconocer alguno de los aspectos mencionados en el listado anterior.

Respecto de la validación y verificación, en el artículo que mencionamos más arriba se analizan sistemas famosos que han fallado y se los clasifica según la fuente de error, es decir, las falencias en la realización de dichas tareas. El modelo propuesto en este trabajo permite definir un criterio de clasificación y hacer una clasificación de los sistemas tratados. Se desea fundamentar, en referencia a este trabajo, por un lado, el rol fundamental del relevamiento y análisis; y por otro, de la validación y verificación de

16 | <http://www.standishgroup.com>

17 | Bahill 1, A. Terry, Steven J. Henderson. Requirements Development, Verification, and Validation Exhibited in Famous Failures. Systems Engineering archive. Vol. 8. N.o 1. Wiley Periodicals. Inc. 2005.

18 | Definiciones del modelo CMMI.

los requerimientos. Los sistemas de la tabla 4.1 fueron analizados en este artículo y se concluyó que tuvieron falencias en alguno de los siguientes aspectos<sup>18</sup>.

RD: desarrollo de requerimientos (validación de requerimientos)

VER: verificación (de requerimientos)

VAL: validación (de sistemas)

<b>Tareas mal realizadas en los sistemas famosos relevados</b>				
<b>Nombre del sistema</b>	<b>Año</b>	<b>RD</b>	<b>VER</b>	<b>VAL</b>
HMS Titanic: transatlántico que naufragó al chocar con un tempano de hielo mientras navegaba.	1912	NO	NO	SI
Tacoma Narrows Bridge: puente que se desplomó entrando en resonancia al ser atravesado por un fuerte viento.	1940	SÍ	SÍ	NO
Apollo 13: módulo lunar que se quedó sin sistema de ventilación mientras orbitaba la luna.	1970	SÍ	NO	SI
Concorde SST: aeronave de pasajeros pensada para volar por 40 años y que se dejó de utilizar mucho antes.	1976 - 2003	SÍ	SÍ	NO
IBM Pcj: primera computadora portable que no salió nunca al mercado.	1983	NO	SÍ	SÍ
Space Shuttle Challenger: transbordador espacial que explotó al despegar.	1986	SÍ	NO	NO
Chernobyl Nuclear Power Plant: planta atómica de generación de energía que se descontroló y esparció material atómico al exterior de su estructura.	1986	SÍ	SÍ	NO
A12 airplane: aeronave militar que nunca fue construida por falta de visión.	1980's	NO	NO	NO
Ariane 5 missile: cohete que explotó al perder el control cuando despegaba.	1996	SÍ	NO	NO
Mars Climate Orbiter: nave de exploración interplanetaria que se estrelló contra Marte cuando sobrevolaba el planeta.	1999	NO	NO	NO
Space Shuttle Columbia: transbordador espacial que explotó cuando reingresaba a la atmósfera de la Tierra.	2002	SÍ	NO	NO

**Tabla 4.1**

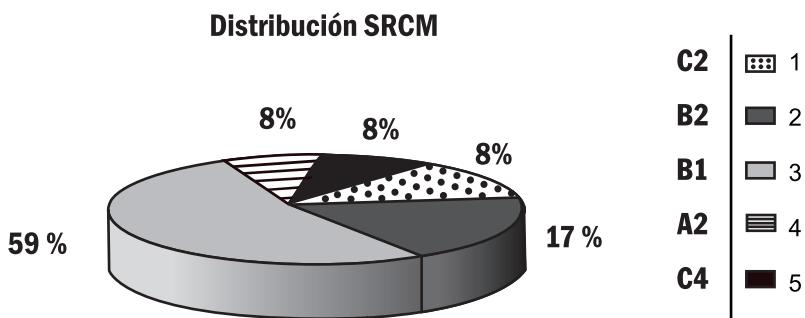
El NO indica que la tarea asociada no fue llevada adelante de la forma correcta o ni siquiera fue realizada. Los sistemas de la tabla anterior son solo algunos de los presentados en dicho artículo. Estos sistemas son conocidos por las consecuencias que generó su fracaso.

En la figura 4.1 se muestra la matriz de dos dimensiones que define el modelo SRCM (System Requirement Classification Model) propuesto por los autores. La idea del modelo es clasificar a los sistemas según alguna de las categorías expresadas por cada celda de la matriz, adaptada del artículo Requirements Development, Verification, and Validation Exhibited in Famous Failures.

Modelo de clasificación de Requerimientos y Sistemas (SRCM)			
Sistema verificado y validado	Sistema no verificado y no validado	Sin sistema	
A1	B1	C1	Requerimientos válidos.
A2	B2	C2	Requerimientos incorrectos, incompletos o inconsistentes.
A3	B3	C3	Sin requerimientos.
		C4	Requerimientos no factibles.

**Fig. 4.1** - Tipos de proyectos según el trabajo con sus requerimientos

Si hacemos el ejercicio de clasificar a los sistemas de la tabla anterior según este modelo, obtendremos el resultado que se muestra en la figura 4.2.



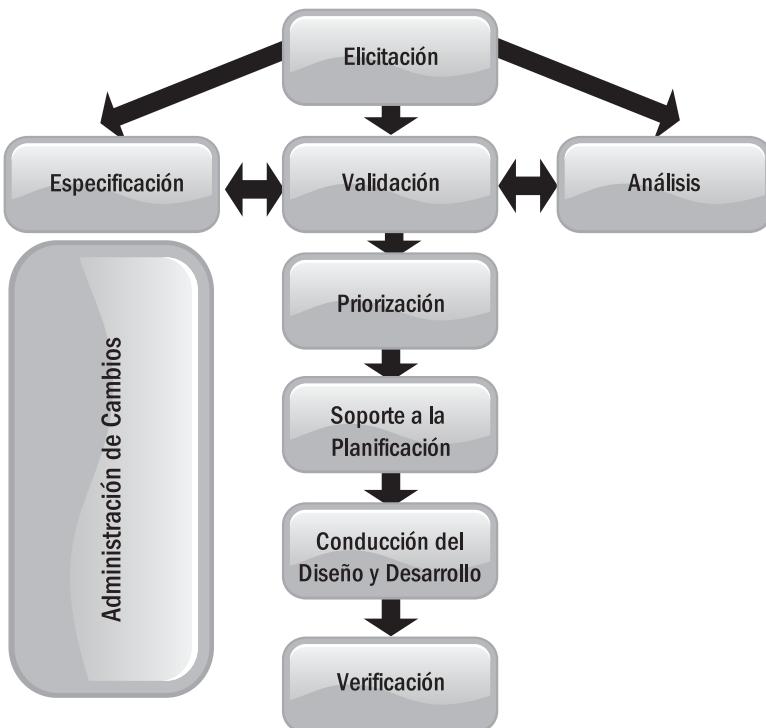
**Fig. 4.2** - Distribución de los sistemas famosos analizados según el modelo SRCM

Se observa con claridad que mayoritariamente los sistemas de tipo B1 con requerimientos válidos pero no validados y no verificados correctamente son los que dominan. De este ejemplo podemos concluir que a la hora de planificar el desarrollo de sistemas es de primerísima importancia invertir recursos en la ingeniería (relevamiento y modelado), en la validación y en la verificación de los requerimientos en los proyectos. Los requerimientos, como tantas veces se expresa, deben describir el sistema que los clientes y usuarios necesitan, por eso hay que validarlos. El trabajo con los requerimientos no se agota en el relevamiento y el listado resultante, los sistemas necesitan validar reglas que el negocio impone y que deben ser analizadas por los involucrados, comprendidas y asignadas a las entidades indicadas de un modelo que represente el comportamiento del negocio. Los requerimientos deben expresar las condiciones de funcionamiento del sistema en desarrollo y ser verificados, para comprobar que cubren todos los requerimientos y que el sistema que los implementa posee el comportamiento esperado en el ambiente operacional definido. Estos temas serán desarrollados en las próximas secciones de este capítulo.

## 4.2 | TAREAS ASOCIADAS A LOS REQUERIMIENTOS

En la figura 4.3 se muestran las tareas realizadas con los requerimientos. Algunas de ellas centradas en ellos mismos y otras para las cuales los requerimientos sirven como soporte, como lo son las estimaciones de alcance y esfuerzo y el establecimiento de prioridades en la planificación de los proyectos.

Estas tareas son llevadas adelante por los diferentes roles involucrados, poseen un nivel de granularidad distinto, son llevadas adelante en distintos momentos del ciclo de vida del proyecto y además generan productos diferentes.



**Fig. 4.3** - Tareas en el trabajo con los requerimientos

Es importante mencionar estas diferencias para ubicar a los productos generados como resultado del trabajo con los requerimientos en cada momento del ciclo de vida del proyecto y apreciar entonces el valor agregado por cada uno de ellos.

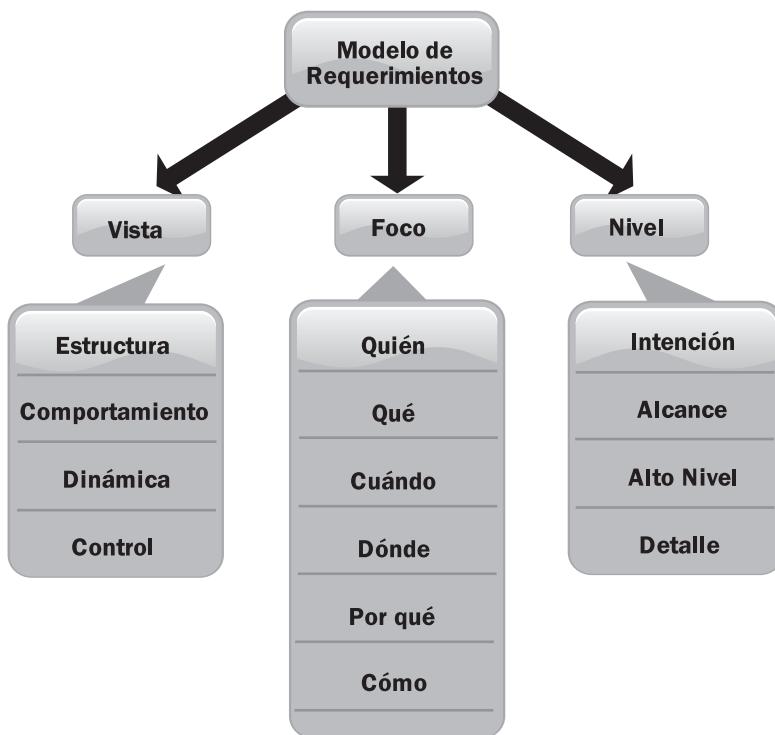
Estos distintos aspectos del modelo de requerimientos son muy bien presentados en Requirements by Collaboration: Workshops for Defining Needs<sup>18</sup>. En la figura 4.4, se presenta un esquema que muestra estos aspectos.

<sup>18</sup> Gottesdiener, Ellen. *Requirements by Collaboration: Workshops for Defining Needs*. Addison Wesley Professional. 2002.

### 4.2.1 FOCO

El enfoque desde el cual son modelados los requerimientos es un factor determinante. Este contexto abarcativo fue utilizado por Zachman<sup>19</sup> para ampliar el escenario en el cual se definen las arquitecturas de tipo enterprise y ha mostrado ser apropiado en otros campos de la tecnología. En el caso de los requerimientos son la guía para determinar:

- La visión del proyecto (por qué)
- La inclusión del vínculo de todos los involucrados en el negocio (quién)
- El alcance del sistema a desarrollar (qué)
- Las condiciones del ambiente operacional (dónde)
- El comportamiento funcional esperado (cómo)
- Los eventos del negocio (cuándo)



**Fig. 4.4** - Modelos de requerimientos Nivel

<sup>19</sup> | Zachman, J. A. *A framework for information systems architecture*, IBM System Journal. Vol. 26. 1987.

#### 4.2.2 NIVEL

El nivel determina el grado de granularidad o detalle con que son tratados cada uno de los aspectos sobre los que se trabaja. Depende fuertemente del momento del ciclo de vida:

- Preventa (intención)
- Inicio del proyecto (alcance)
- Concepción y planificación (alto nivel)
- Implementación (detalle)

#### 4.2.3 VISTA

Existen varias vistas de los requerimientos dependiendo de los modelos que se utilicen.

- Estructura (modelo de dominio)
- Dinámica (diagramas de secuencia)
- Comportamiento (diagramas de casos de uso)
- Control (reglas de negocio, modelo de análisis)

Cada uno de estos modelos implica un conocimiento menor o mayor del negocio a resolver con el sistema que se construirá. Por esta razón es también interesante vincular estas vistas con los niveles en que son de utilidad. En la tabla 4.2, adaptada de Requirements by Collaboration: Workshops for Defining Needs, se muestra esta relación siguiendo el modelo de Zachman. El contenido de cada una de las celdas de la matriz está constituida por los modelos de las vistas presentadas anteriormente. Una última relación que es interesante tener en cuenta es la que existe entre las vistas y el ciclo de vida del proyecto. Esta es muy similar a la que existe con los niveles. A medida que avanza el proyecto es mayor la información con la que se cuenta y por lo tanto son más detallados los modelos que pueden construirse.

Con la presentación de esta matriz buscamos vincular todas las tareas que se desarrollan y los modelos que se construyen en el trabajo con los requerimientos, y que iremos siguiendo a medida que avancemos en el desarrollo de los diferentes temas.

		Foco					
		Nivel					
		Qué	Cómo	Dónde	Quién	Cuándo	Por qué
Intención	Características	Listado de Procesos de Negocio	Diagramas de Contexto	Diagramas de actividades	Workflow (roles y eventos)	Listado de eventos	Visión Objetivos
Alcance	Listado de Requerimientos	Diagramas de Actividad	Ambiente Operativo, Requerimientos No Funcionales	Rol de Negocio	Cronograma de Negocio	Prioridades de Negocio	Intención
Alto Nivel	Listado Casos de Uso, Trazas	Especificación de Casos de Uso principales	Subsistema, Módulo	Actores Descripción	Precedencia de Casos de Uso	Prioridades del Proyecto, Reglas de Negocio	Alcance
Detalle	Diagrama Casos de Uso, Prototipos de Interfaces, Modelo de Dominio	Especificación de Casos de Uso, Prototipo de Navegación	Paquete	Actores Permisos Responsabilidades	Diagramas de Estados	Reglas de Negocio asignadas en el Modelo de Análisis	Alto Nivel
	Detalle						

Tabla 4.2 - Matriz de Zachman aplicada al trabajo con requerimientos

## 4.3 | ESTRATEGIA Y TÁCTICAS EN EL TRABAJO CON REQUERIMIENTOS

### 4.3.1 ESTRATEGIA

Como en casi todos los aspectos de un proyecto, el trabajo con los requerimientos depende de las características del proyecto y de la metodología con que se trabaje. Proponemos un modelo de trabajo que presenta a las diferentes tareas como un mini proceso cuya formalidad irá decreciendo con el volumen del proyecto. Este modelo nos permitirá cubrir proyectos grandes con una forma de Trabajo con Requerimientos tipo workshop donde participarán varios involucrados. Esta perspectiva se irá simplificando a medida que encaremos sistemas más simples, terminando así con la participación de un analista entrevistando a un usuario. Sin embargo, conservaremos algunas cuestiones esenciales que a nuestro entender deben ser tenidas en cuenta siempre y que hemos visto ausentes en muchos proyectos.

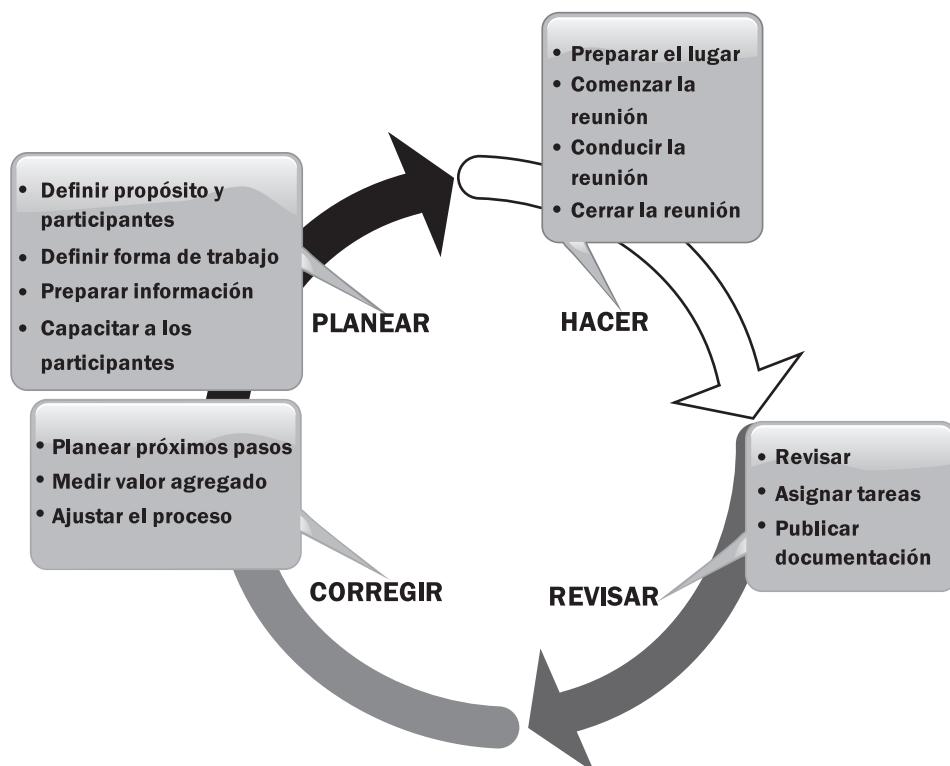


Fig. 4.5 - Ciclo de vida del trabajo con los requerimientos

En la figura 4.5 se muestra un ciclo de trabajo en el cual se nombran todas las

actividades que se llevarán adelante.

Este modelo propone planificar, ejecutar, revisar y corregir. Es común en proyectos medianos a chicos desarrollar el relevamiento de requerimientos y su análisis sin planificación alguna. También es común establecer reuniones sin un claro propósito a las cuales los usuarios acuden para hablar de sus necesidades y los analistas toman notas sin ninguna planificación previa. Esto impide a los usuarios preparar y ordenar sus conocimientos del negocio y a los analistas definir una estrategia de avanzar sobre los diferentes aspectos del problema a investigar. Este modelo de trabajo denominado workshop de colaboración como es presentado en Requirements by Collaboration: Workshops for Defining Needs es ideal para proyectos grandes y formales, pero además es fácil de adaptar para proyectos menores. En su fortaleza radica también su debilidad. Su fortaleza consiste en que organizar el trabajo de esta forma requiere de la participación de todos los involucrados; su debilidad en que para que esto sea posible se requiere una madurez importante de todas las partes.

Este modelo lo explicaremos enumerando algunos beneficios y ejemplificando cada una de sus partes con los escenarios que comúnmente se observan en los proyectos.

**Planear permite:** conocer el propósito de la reunión, los temas a tratar, conocer de antemano quiénes participarán, planificar la agenda de los involucrados que deberán asistir, preparar la documentación por parte de los usuarios y clientes, ordenar y planificar el desarrollo de las preguntas a hacer por parte de los analistas.

**Planear evita:** reuniones en las que los asistentes se enteran del motivo de esta cuando acuden a ellas, ausencias por falta de comunicación sin el tiempo previo necesario, postergaciones para próximas reuniones por falta de información, falta de documentación por desconocimiento de su necesidad, tertulias donde se tratan los más diversos temas y ninguno relacionado con el sistema a desarrollar.

**Ejecutar permite:** desarrollar la reunión de la manera planeada, en el lugar establecido, en los tiempos planificados, con las personas indicadas y definidas.

**Ejecutar evita:** no contar con una sala para el desarrollo, demoras hasta que se nos asigna una, esperas a asistentes que acuden a horarios diferentes, tratar temas dispersos con escaso valor agregado.

**Evaluar permite:** ordenar los resultados de las reuniones y publicarlas a todos los involucrados, dar consistencia a los productos generados, analizar y organizar el trabajo, contar con modelos relacionados (traceables), contar con modelos sobre los cuales basar los acuerdos.

**Evaluar evita:** generar sensación de que el esfuerzo anterior fue en vano, requerimientos personalizados y solo conocidos por los interesados, proyectos con escasa visibilidad, falta de acuerdo por falta de visibilidad.

**Actuar permite:** mejorar y aprender de la forma de trabajo, corregir y hacer ajustes en la dirección apropiada para el proyecto.

**Actuar evita:** repetir errores de organización, repetir errores de desarrollo, desacuerdos, ausencia de conducción y sensación de falta de gestión.

Deseamos hacer notar que no proponemos activos con formatos determinados

para ayudarnos en estas tareas debido a que no queremos que el lector caiga en la tentación de valorar el modelo anterior a partir de un documento. Dejamos que dicho documento lo elabore a medida de sus necesidades. Lo importante es rescatar el ejercicio de pensarlo, acordarlo y proponerlo al seno de la organización y el proyecto a partir de una forma de trabajo como la descripta.

También pensamos que la esencia de esta forma de trabajo trasciende la metodología elegida para llevar adelante el proyecto.

### 4.3.2 TÁCTICAS

#### 4.3.2.1 Especificación de requerimientos de software y sus atributos de calidad

Entendemos por especificación de requerimientos de software al listado de “el sistema debe...” derivado a partir del relevamiento de las necesidades de los usuarios. Estos requerimientos funcionales expresan la totalidad de la funcionalidad que el sistema brindará. Por otra parte las condiciones de funcionamiento en el ambiente operacional son expresadas por los requerimientos no funcionales.

Este listado de requerimientos fue dejado de lado, no utilizado por los seguidores de metodologías tipo UP (Unified Process Software Development) que se apoyaron en la utilización de los casos de uso como método no sólo de modelar sino de relevar requerimientos en las diferentes herramientas de tipo CASE que implementan UML (Unified Modelling Language).

Hemos observado errores de los más diversos cuando se trabaja de esa forma y con esos diagramas sin pasar previamente por un listado de requerimientos que cumplan con atributos tales como (Gottesdiener, Ellen; 2002, op. cit.):

- Completitud
- Correctitud
- No ambigüedad

Nuestra experiencia nos indica que para elaborar un diagrama de casos de uso es necesario hacer un ejercicio de elaboración en el cual generalmente se cometen errores. La elaboración de un listado de requerimientos es una tarea más simple y mantiene el foco en el “qué” debe satisfacer el sistema en desarrollo sin preocuparse por el “cómo”. Por esta razón recomendamos la utilización de un listado de requerimientos como paso previo a la elaboración de los casos de uso.

Otra razón por la que recomendamos dicho listado se debe a que los requerimientos deben ser acordados con todos los involucrados en un proyecto. A pesar de los esfuerzos marketineros de venta de los casos de uso por parte de los vendedores de herramientas, utilizando el argumento de que son muy fáciles de entender, los usuarios y clientes no los entienden y no desean verlos. Por eso es necesario comunicarse y acordar con ellos a partir de un listado bien construido de requerimientos. En los apéndices se presenta una plantilla de ERS (Especificación de Requerimientos de Software) como guía para la construcción de un listado bien escrito.

Un listado de requerimientos bien escrito debe, entre otras cosas, no presentar ambigüedad. Cuanto más detallado sea mejor escrito estará. Un error común que hemos observado en gran cantidad de proyectos es la escritura o especificación de los requerimientos expresando características del producto a construir más que centrándose en los requerimientos. Esto es consecuencia, en la mayor parte de los casos, de falta de información detallada y de expresiones concretas que describan a los requerimientos. Esa es la razón por la cual algunas ERS se parecen más a un prospecto o folleto de publicidad de un producto que a un listado de requerimientos, como se muestra en la siguiente tabla.

Características de producto	Requerimientos
El sistema permitirá la administración de clientes.	<p>El sistema permitirá la registración de clientes.</p> <p>Los operadores y supervisores podrán modificar los datos de los clientes.</p> <p>El sistema mantendrá un registro histórico de clientes.</p>
El sistema permitirá la administración de reclamos.	<p>El sistema permitirá generar solicitudes de reclamos.</p> <p>El sistema permitirá realizar el seguimiento de las solicitudes de reclamos según sus estados.</p> <p>Los supervisores podrán aceptar o rechazar las solicitudes de reclamos generando créditos en las cuentas de los clientes en los casos que corresponda.</p>

**Tabla 4.3** - Características de producto versus requerimientos de producto

#### 4.3.2.2 Especificación de casos de uso

Después de una década de la masificación en la adopción de los casos de uso como forma de modelar requerimientos, todavía hoy se siguen usando mal y de manera excesiva estos diagramas. La tentación creada por las herramientas CASE, que facilitan la creación de dichos artefactos, hace que muchos desarrolladores sólo construyan estos diagramas sin ninguna especificación textual de los casos de uso. Esta es la causa de la creación de casos de uso que cuando se analizan en más detalle no se sabe cuál es su razón de ser. Por estos motivos aconsejamos la especificación textual como medio de dar consistencia y precisión a estos diagramas.

Otro abuso y mal uso se debe a la utilización de relaciones entre casos de uso. En nuestra experiencia las relaciones útiles son las de inclusión (include) y extensión (extend); no así la de generalización (herencia o extensión o especialización). La razón es que si bien las herramientas facilitan su diagramación, es muy confusa su

20 | Frank Armour, Granville Miller. *Advanced Use Case Modeling*. Addison-Wesley. 2001.

21 | Kurt Bittner and Ian Spence. *Use Case Modeling*. Addison-Wesley. 2002.

especificación textual. Los mejores libros que tratan el tema de casos de uso son Advanced Use Case Modeling<sup>20</sup> y Use Case Modeling<sup>21</sup>.

Para nosotros es de mucha utilidad complementar la especificación con un prototipo de interfaces, de diseño lógico, que muestren los tipos de los campos manipulados, las condiciones de obligatoriedad, de modificabilidad, las reglas a validar y la relación con el modelo de análisis. En el apéndice A se presenta una plantilla guía de especificación de caso de uso en el que pueden verse todos los aspectos mencionados. La figura 4.6 muestra los artefactos propuestos que conforman la definición de cada caso de uso.

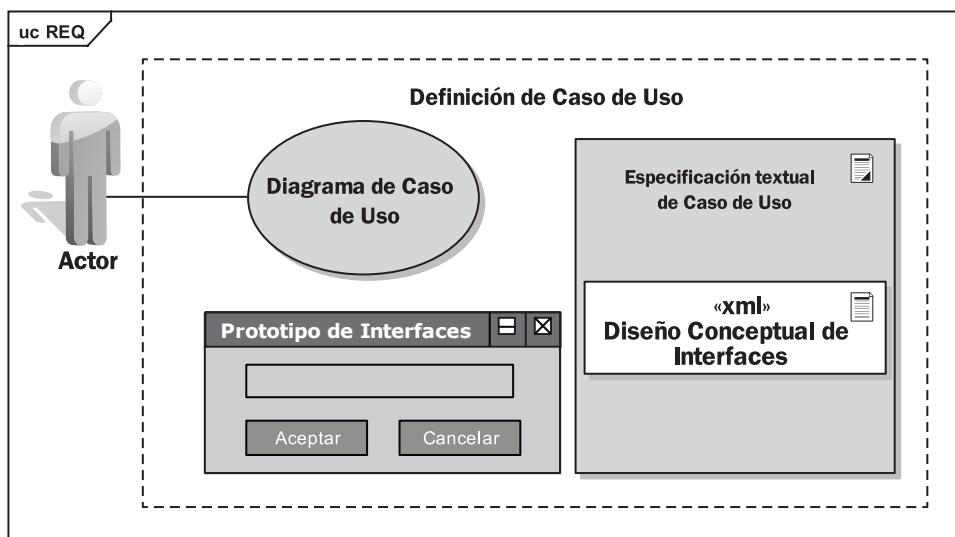


Fig. 4.6 - Componentes de la especificación de caso de uso

Cada vez que desarrollo una capacitación o presentación de este tema apelo a la siguiente frase: “esto no compila...”. La razón es que muchos desarrolladores esperan de los diagramas de UML en general y de los de casos de uso en particular más precisión de la que deberían. Por eso es muy importante el acuerdo entre las partes del grupo de desarrollo del significado de ciertas palabras en el momento de la construcción de estos diagramas. Es frecuente encontrar casos de uso nombrados como Administrar XXX. Administrar puede significar diferentes cosas para las distintas personas de un grupo de desarrollo. Por ejemplo, podría significar alta, baja y modificación; o solo alta; o todo lo anterior más procesar XXX.

Es muy importante darle claridad y consistencia a los diagramas de este tipo definiendo, al nivel del grupo, el significado de las palabras claves. También es importante hacerlo cuando se listaron los requerimientos, en aquel caso para todos los involucrados en el proyecto.

Todas estas definiciones hacen a la organización del grupo de desarrollo.

## 4.4 | ANÁLISIS DE REQUERIMIENTOS

### 4.4.1 NO CONFUNDIR DOMINIO Y NEGOCIO CON DISEÑO

En los últimos tres o cuatro años, ha habido una oleada de propuestas acerca de cómo hacer para que el desarrollo de los sistemas esté conducido por el negocio. Estas promociones van desde el excelente libro Domain Driven Design, de Eric Evans<sup>22</sup>, hasta el sustento de un nuevo paradigma tal como SOA (Service Oriented Architecture), cuya esencia se basa en cómo los servicios de IT (Information Technology) se funden con los del negocio para crear nuevos negocios. A gran parte de la comunidad informática internacional le pareció una novedad ya que en diferentes foros ambas fueron acogidas con gran atención y hasta recibieron premios. Como algunos de mis colegas, con experiencia de tres décadas en el desarrollo de software, me pregunto cuál es la innovación. Pensar las soluciones de software (programas) de esta manera fue una de las primeras cosas que nos enseñaron. Entonces uno se pregunta ¿cuál fue la razón por la que esta estrategia se dejó de lado con las consecuencias que enseguida analizaremos? Yo he encontrado una respuesta que me satisface: la razón fue la complejidad de la tecnología. Suena contradictorio pero no lo es. La complejidad de la tecnología de las últimas dos décadas ha hecho que los desarrolladores estén muy ocupados aprendiendo a utilizar tal o cual herramienta, tal o cual ambiente de desarrollo, tal o cual lenguaje, tal o cual framework; que se olvidaron del negocio. Me ha tocado revisar muchos proyectos de desarrollo y me ha sorprendido encontrar lo que Martin Fowler llama objetos anémicos. Una capa de objetos que solo cuentan con atributos y gets y sets, y ninguna otra operación. Sin embargo dichos objetos estaban mapeados con precisión a la base de datos y contaban con capacidad transaccional y seguridad, y etc. Los desarrolladores resuelven las cuestiones vinculadas a la infraestructura y dan poca importancia al negocio. La consecuencia de esta forma de trabajo son sistemas mal diseñados, no extensibles desde el punto de vista del negocio, con un comportamiento que no es el esperado. Este es el resultado de resolver aspectos tan diferentes como el negocio y la infraestructura en la misma actividad durante el desarrollo.

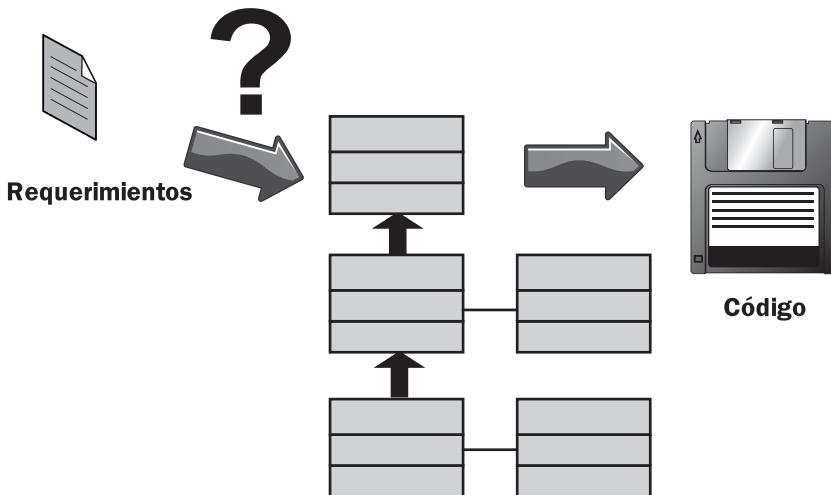
Pongamos un poco de claridad, en este punto tenemos claro qué son los requerimientos y todos sabemos qué es el código ejecutable. Ahora bien, ¿cuáles son los artefactos que generamos al realizar las actividades que nos llevan de uno al otro, como se indica en la figura 4.7?



**Fig. 4.7** - Qué artefactos nos llevan desde los requerimientos al código

22 | Evans, Eric. *Domain Driven Design*. Addison-Wesley. 2004.

También es un concepto conocido el diagrama de clases que conforman el diseño del código mostrado. Sin embargo nos sigue quedando una brecha entre este diseño y los requerimientos, como se ilustra en la figura 4.8.



**Fig. 4.8** - El diseño como artefacto ligado al código

Antes de continuar incorporaremos a este análisis los conceptos ya vistos en secciones anteriores que hacen a la especificación de los casos de uso como expresión de los requerimientos funcionales. Aún así no hemos llenado esta brecha porque los casos de uso describen el comportamiento de nuestra aplicación en desarrollo, es decir, su dinámica. Pero en el diagrama de clases ligado al código hay conceptos estáticos además de la dinámica mencionada. La pregunta es cómo surgieron estos conceptos. Deberíamos haber partido de algún modelo también estático para llegar a este diseño. Así es, por lo tanto debemos recorrer un camino paralelo al que realizamos con la especificación del comportamiento, pero en este caso manipulando aspectos que aporten la estructura del diseño resultante.

Este punto de partida consiste en pensar el problema a resolver en términos de conceptos extraídos del dominio de este. Estos conceptos primarios, de los cuales la mayor parte terminará siendo objeto de nuestro código, los relacionaremos en un modelo de clases que llamamos *Modelo de Dominio*. Estamos así trabajando, como Booch<sup>23</sup> describe, haciendo uso del mecanismo de descomposición. En este punto nuestros conceptos no tienen alcance preciso expresado en sus atributos. Este alcance lo obtendremos a partir de enriquecerlos con la información provista por el comportamiento (casos de uso, prototipo de interfaces y diseño conceptual de interfaces). Este ejercicio de refinamiento nos lleva del *Modelo de Dominio* al *Modelo*

23 | Booch, Grady. *Object - Oriented Analysis and Design with Applications*. 2.<sup>nd</sup> Edition. 1994.

de Negocio. Este último es lo que Eric Evans llama lenguaje ubicuo y lo construimos desde el dominio utilizando patrones de análisis. Estos patrones de análisis fueron catalogados de una forma didáctica y precisa por Jill Nicola<sup>24</sup> y Martin Fowler<sup>25</sup>. El Modelo de Negocio resultante no contiene ninguna información de la tecnología, sino que describe el negocio en forma precisa y completa, incorporando las reglas a los objetos del modelo. Los objetos resultantes no son para nada anémicos. A partir de su colaboración se modela el negocio del sistema en construcción.

Quiero ser claro y preciso al diferenciar el Modelo de Negocio del de Diseño, ya que las preocupaciones de los desarrolladores al construirlos son distintas.

### Modelo de Negocio

- Objetivo: entender en detalle el negocio y sus reglas
- Mecanismo utilizado: patrones de análisis

### Diseño

- Objetivo: implementar una solución al problema planteado en el análisis más las restricciones impuestas por los requerimientos no funcionales
- Mecanismo utilizado: patrones de diseño

El Modelo de Negocio resultante es la base sobre la cual trabajarán los desarrolladores para realizar su diseño, guiados ahora por otro objetivo y utilizando otros criterios tales como<sup>26</sup>: inversión en la cadena de dependencia, código clausurado ante cambios, principio de substitución de Liskov, segregación de interfaces, dependencias no cíclicas, etc. A partir de aquí los desarrolladores se valdrán de herramientas, como los patrones de diseño que describen Gamma<sup>27</sup> y Folwer<sup>28</sup>, y además llega el momento de tener en cuenta la tecnología. El proceso comentado y la generación de los artefactos resultante se muestran en las figuras 4.9, 4.10 y 4.11. Esta visión de ambos flujos de trabajo en lo que hace al proceso de desarrollo es similar a la presentada en ICONIX Process<sup>29</sup>, en el que se pone gran énfasis en los modelos de robustez. Es conveniente poner el acento en el modelo de dominio refinado con los patrones de análisis y las reglas de negocio, con lo cual se trabaja de manera independiente en los modelos del negocio y la aplicación.

24 | Jill Nicola (et al.). *Streamlined Object Modeling - Patterns, rules and implementations*. PHPTR. 2002.

25 | Fowler, Martin. *Analysis Patterns: Reusable Object Models*. Addison-Wesley Object Technology Series. 1994.

26 | Martin, Robert C. *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall. 2002.

27 | Gamma (et al.). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. 1994.

28 | Fowler, Martin. *Patterns of Enterprise Application Architecture*. Addison-Wesley Signature Series. 2002.

29 | ICONIX Process, <http://iconixprocess.com>.

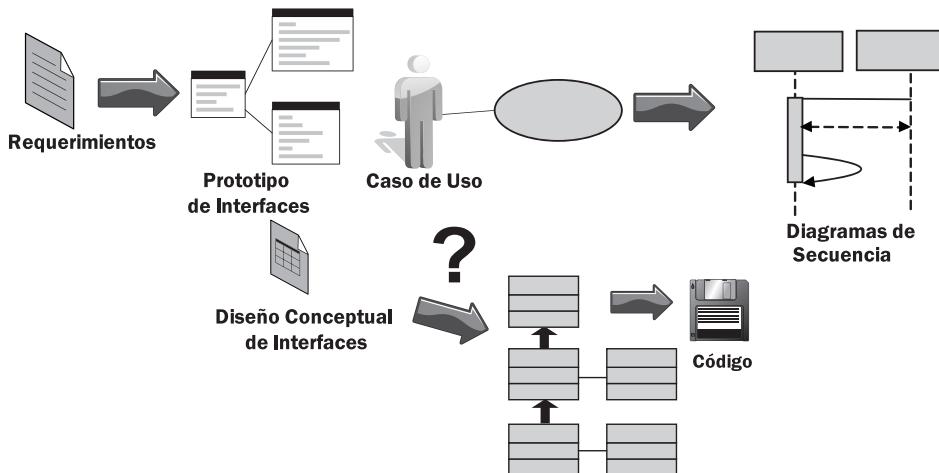


Fig. 4.9 - Cómo llegamos desde la especificación de los requerimientos de la aplicación al diseño

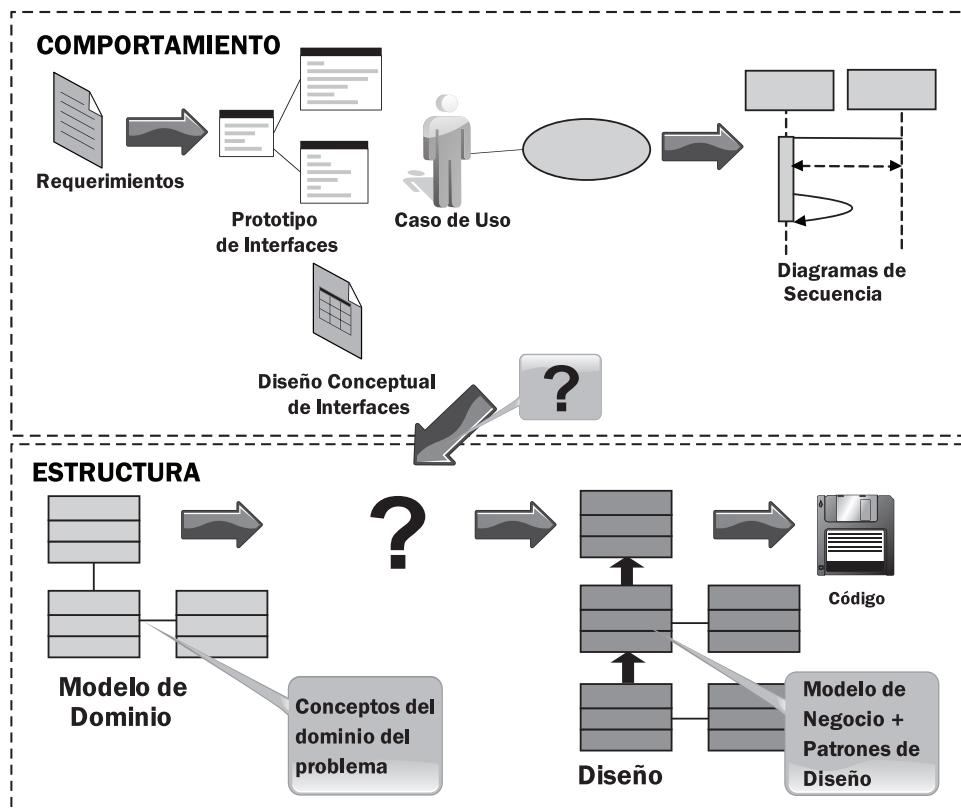


Fig. 4.10 - Trabajo paralelo con la estructura (Modelo de Dominio) y el comportamiento (Caso de Uso)

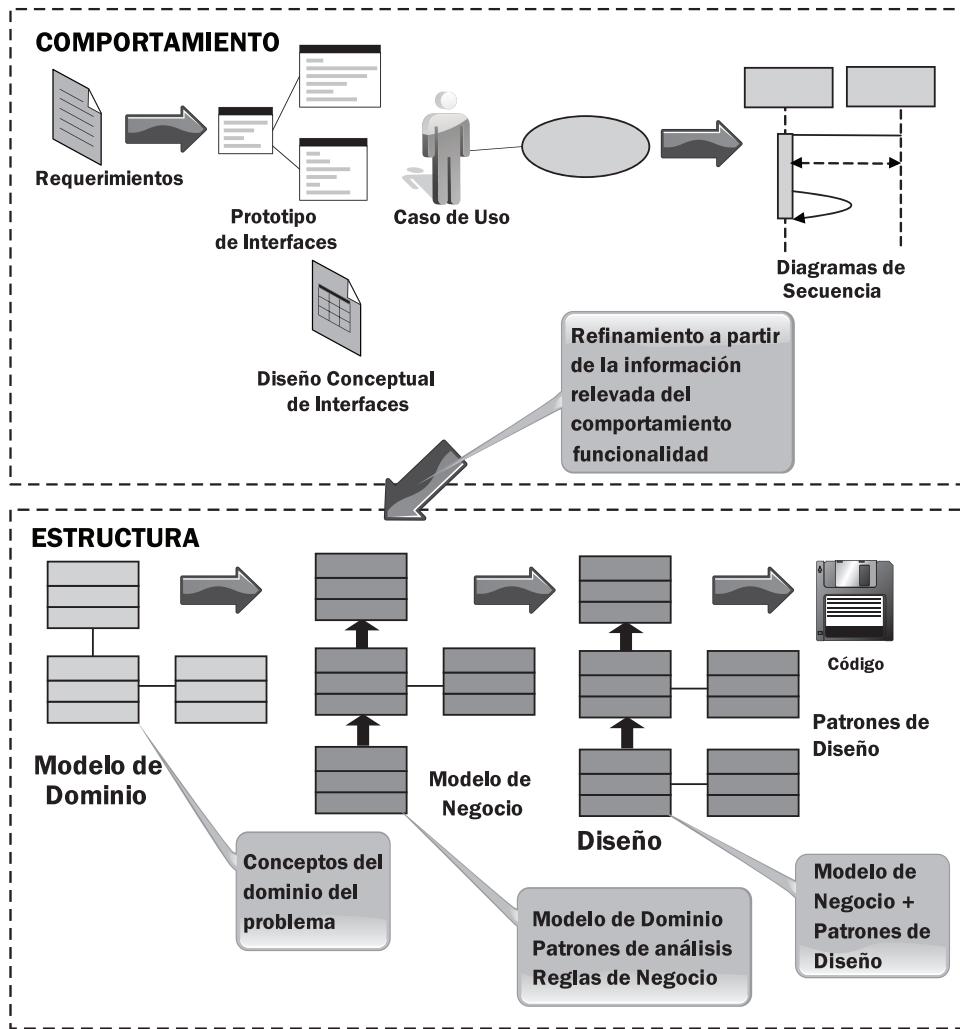


Fig. 4.11 - El Modelo de Negocio y de Diseño en el proceso de desarrollo de software

#### 4.4.1.1 Nota para desarrolladores ágiles

Deseamos notar con énfasis que lo propuesto no significa hacer para todos los proyectos los mismos diagramas, de la misma forma, como una receta preestablecida. Por eso proponemos que los lectores se queden con la esencia de la forma de trabajo propuesta y la adapten de una manera inteligente a su ámbito. Estamos convencidos de que la que acabamos de presentar es una estrategia de desarrollo que ayudará a quien la use a generar productos de muy buena calidad.

Cuando nombramos algunas de las vistas de un proyecto como Modelo no nos referimos a un documento, sino a los conceptos manipulados en ese momento del ciclo de vida y con el conocimiento que entonces tenemos de ellos, los cuales deberán aprenderse, relacionarse y documentarse de la manera que ustedes encuentren más apropiada.

#### **4.4.1.2 Nota a los analistas de sistemas**

Existe una gran tendencia a gastar casi todo el esfuerzo en el relevamiento, especificación y prueba de los requerimientos funcionales en detrimento de los no funcionales. Estos últimos caracterizan al ambiente operacional en el cual debe funcionar el sistema en desarrollo. Este rasgo distintivo del trabajo con requerimientos por parte de los analistas lo hemos observado en muchos proyectos. Ambos son igualmente importantes y a ambos se le debe asignar el esfuerzo necesario.

#### **4.4.2. PAQUETES**

El término paquete es utilizado en diferentes ámbitos del desarrollo de software con distintos significados, aquí lo usamos en un sentido lógico y amplio. Paquete es en principio un conjunto de casos de uso cohesivos con algún criterio de negocio. Cuando el proyecto avance, estos paquetes se convertirán en físicos, constituyendo los subsistemas que componen el sistema en desarrollo.

La elección de estos paquetes primarios es una tarea sumamente importante, ya que constituyen el primer esbozo de la arquitectura del sistema en desarrollo. Durante el proceso de desarrollo del proyecto estos paquetes son la unidad de release de cada iteración y cuando el proyecto está terminado e instalado constituyen la unidad de mantenimiento, como lo indican Fowler (op. cit., 1997) y Robert C. Martin<sup>30</sup>.

##### **4.4.2.1 Alternativas de selección**

¿Cómo seleccionamos los paquetes de un sistema?

La respuesta es simple e importante. Una forma de elección es en base a las clases más representativas del modelo de dominio. La idea es centrar los paquetes en estas entidades y refinar el conjunto resultante. Los vínculos entre estos paquetes están dados por la relación entre los casos de uso empaquetados en cada uno. Cuando realizamos el diseño aparecen nuevos que alojan aspectos de la tecnología.

Otra posibilidad es elegir los paquetes a partir de las áreas de negocio que el sistema en desarrollo resolverá. Un posible inconveniente con esta alternativa es que a veces resultan paquetes muy grandes que deben a su vez ser particionados. En la figura 4.12 se muestran los conceptos básicos del dominio de un negocio de transporte de cargas, el cual les factura a los clientes el servicio de acuerdo al itinerario seleccionado de un conjunto posible. En la figura 4.13 se muestran los paquetes resultantes seleccionados con el criterio de centralizarlos en los conceptos más representativos.

---

30 | Martin, Robert C. *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall. 2002.

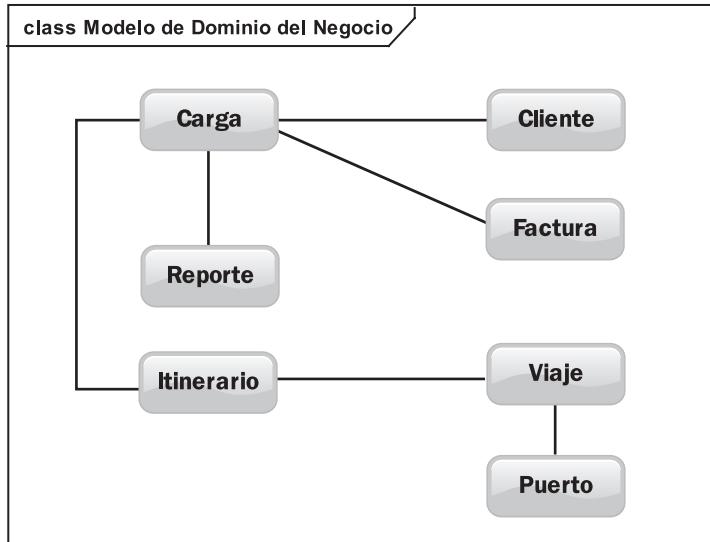


Fig. 4.12 - Conceptos extraídos del dominio del problema

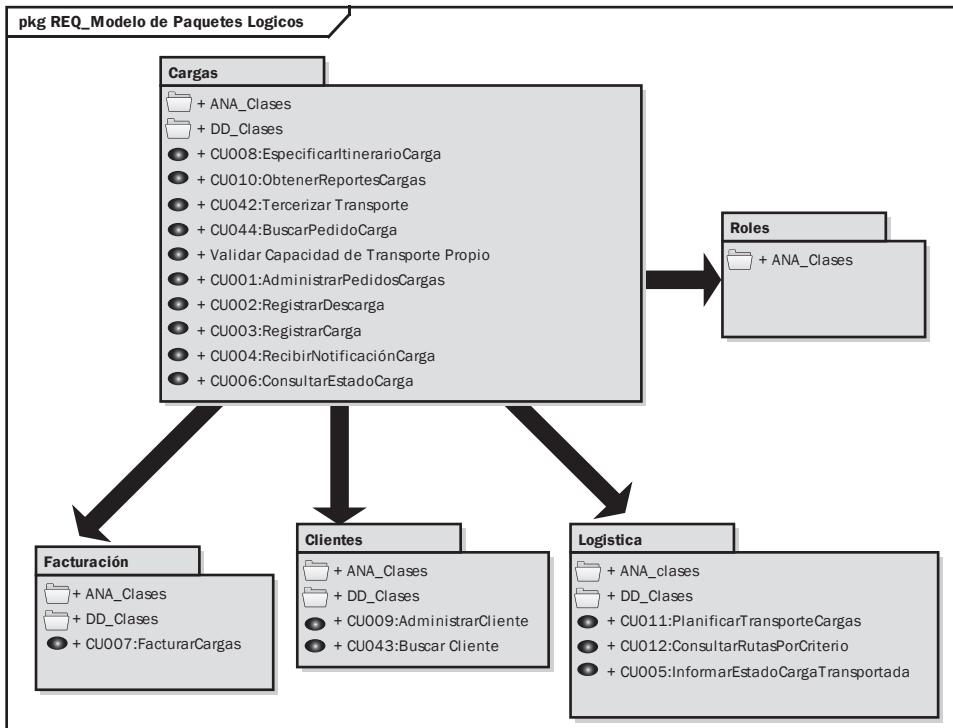


Fig. 4.13 - Diagrama de paquetes correspondiente al dominio de la figura anterior

## 4.5 | VALIDACIÓN Y VERIFICACIÓN

---

Además de las tareas de relevamiento y especificación, las tareas de validación y verificación fueron las que utilizamos para enfatizar la importancia del trabajo con los requerimientos en el inicio de este capítulo. A continuación presentamos nuestra propuesta de cómo llevarlas adelante a efectos de garantizar la calidad del software resultante.

### 4.5.1 VALIDACIÓN

Una forma directa de validación del comportamiento del Modelo de Negocio es su programación definitiva o prototipo y las pruebas ejecutadas sobre este que muestren que el comportamiento es el esperado. El objetivo de esta validación es mostrar la ejecución de las validaciones de las reglas de negocio y asegurarse de que esta dinámica es la solicitada y que tiene consistencia según las condiciones fijadas por las reglas.

Esta programación, por las mismas razones que dimos en la sección anterior, no debe asentarse en clases de diseño que cumplan con criterios de buen diseño. Por este motivo puede tratarse de un prototipo o código no óptimo que con modificaciones posteriores termine siendo el definitivo. La validación de este comportamiento es clave en modelos complejos.

Por otro lado, la validación del comportamiento de la aplicación (funcionalidad) con los usuarios utilizando los prototipos de interfaces que forman parte de la especificación de los casos de uso es una muy buena práctica que en nuestra experiencia nos ha dado muy buenos resultados.

Ambas validaciones son necesarias, ya que a partir de lo presentado en secciones anteriores quedó claro que Modelo de Negocio y Aplicación (casos de uso) son cosas diferentes. Uno expresa el área de negocio sobre el que nuestro sistema operará y la otra qué y cómo usamos ese negocio desde el nuevo sistema en desarrollo.

El resultado exitoso de estas validaciones nos da la certeza de que construiremos el sistema correcto.

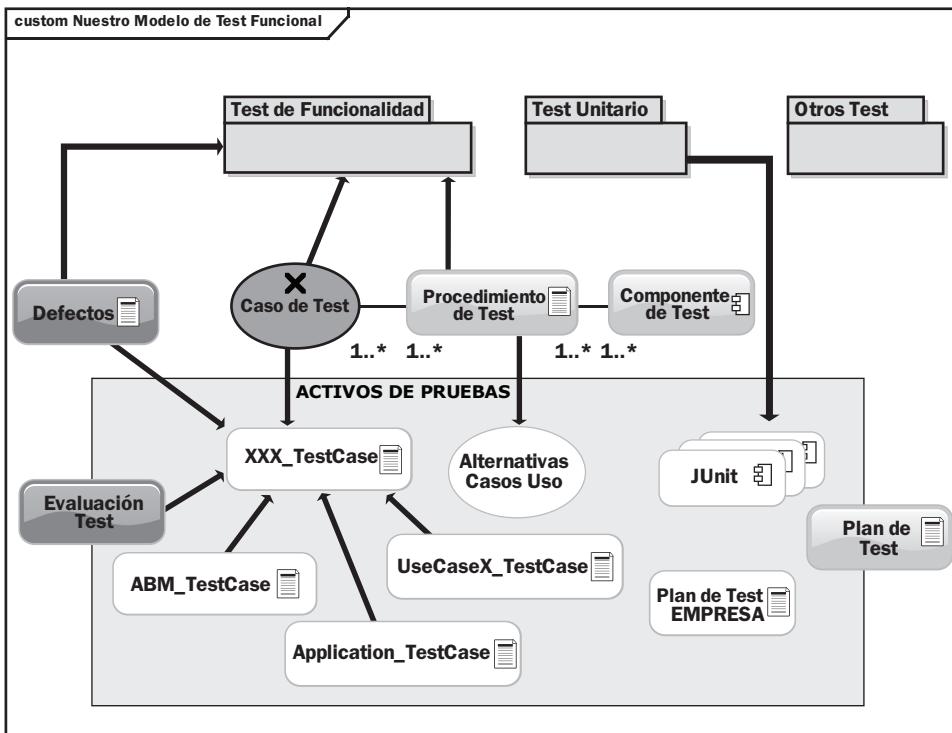
### 4.5.2 VERIFICACIÓN

La verificación de la corrección de la construcción del sistema la llevaremos a cabo diseñando y ejecutando pruebas (test) que nos darán la certeza de que hemos construido el sistema de manera correcta.

Me referiré aquí solamente a las pruebas funcionales, es decir a aquellas que prueban el comportamiento de la aplicación en términos de los casos de uso. El resto de las pruebas, otros tipos de pruebas, las trataremos en el capítulo seis.

Si nuestros casos de uso fueron validados y son los que debíamos construir, luego llega el momento de probar que fueron construidos correctamente. Por lo tanto a partir de los casos de uso derivaremos los casos de prueba mencionados en los

trabajos de Jim Heumann<sup>31</sup> y Peter Zielczynski, Director of Technology Solutions<sup>32</sup>. Nuestros procedimientos de pruebas están expresados en la especificación textual de los casos de uso. Además para cada uno de estos contamos con los diferentes caminos básicos y alternativos, cuyas combinaciones constituyen los distintos escenarios de prueba. Debemos seleccionar el conjunto de datos con los cuales probar cada caso y tendremos así los casos de prueba definidos. En la figura 4.14 se muestra un diagrama que ejemplifica la constitución de los casos de prueba incluyendo todas sus partes como se acaba de describir.



**Fig. 4.14** - Componentes del Modelo de Pruebas Funcionales<sup>33</sup>

Este modelo lo hemos utilizado en diferentes proyectos y muestra por cada caso de uso tres diferentes tipos de caso de prueba.

- Las altas, bajas y modificaciones generalizadas de forma tal que tanto la especificación de los casos de uso como las pruebas resulten reutilizables. Por esta razón aparece un activo llamado ABM\_TestCase, el cual constituye el caso de prueba para estos casos.

31| Heumann, Jim. *Generating Test Cases from Use Case*. Rational Software. 2001.

32| Traceability from Use Cases to Test Cases, Peter Zielczynski. Director of Technology Solutions. The A Consulting Team, Inc., <http://www.ibm.com/developerworks/rational/library/04/r-3217/index.html>.

33| Modelo construido en it-Mentor con la ingeniera Patricia Forradellas.

- Por cada interfaz de usuario de cada caso de uso diseñamos pruebas orientadas a probar la presentación. Se testean la presencia, la disposición y el orden de los diferentes controles (layout), el paginado de datos, etc. A este caso de prueba lo llamamos Application\_TestCase y se ejecuta para todas las pantallas de los casos de uso de la aplicación. Esta es otra muestra de la estrategia de reutilización aplicada a los requerimientos, ya que la consistencia dada a la aplicación nos permite plantear casos de prueba idénticos para todas sus vistas. Estos patrones de especificación, que se dan a partir de acuerdos básicos entre los desarrolladores, son fundamentales en la organización del ambiente, los que facilitan el desarrollo y generan productos de mejor calidad.
- Para aquellos casos de uso diferentes construimos el caso de prueba asociado y lo nombramos UseCaseX\_TestCase.

Los activos resultantes los implementamos en planillas electrónicas. Cada una de ellas posee lenguetas para las diferentes ejecuciones, columnas para registrar los resultados obtenidos y otros datos de interés.

Cuando tratemos otros temas relacionados a las pruebas de software en el contexto de xUnit e Integración Continua, volveremos sobre el concepto de verificación.

## 4.6 | ADMINISTRACIÓN DE CAMBIOS A LOS REQUERIMIENTOS

---

### 4.6.1 PROBLEMA

Hace casi tres décadas trabajábamos en proyectos de desarrollo de software orientados a resolver problemas científicos tecnológicos en la universidad utilizando una computadora Digital PDP-11. En esos tiempos trabajé en proyectos de investigación y desarrollo en áreas como ingeniería biomédica, exploración petrolera y telefonía. Los programas que construimos en su mayoría no eran interactivos, más bien procesaban datos y solo se les proveían algunos parámetros y a partir de sus salidas se elaboraban reportes. Los proyectos comenzaban y terminaban casi con los mismos requerimientos.

Con la irrupción de la PC en el mercado la variedad de áreas a las cuales fue posible proveer una solución de software creció casi exponencialmente y con esa misma velocidad decreció el tiempo en que los requerimientos permanecían estables, sin cambios. La posibilidad de contar con interfaces de usuario de gran variedad contribuyó a que los clientes tuvieran aproximaciones más tempranas a los productos en construcción y a partir de estas vistas solicitar cambiar sobre la marcha. Lo que hoy era válido, mañana ya no lo era. Tuvo que pasar mucho tiempo antes de que los procesos de desarrollo resolvieran con algo de coherencia la administración de estos cambios. Es más, aún hoy día en algún proceso de mejoras tengo que trabajar con proyectos de desarrollo en los que no se administran los cambios a los requerimientos y peor aún, debo darles una explicación detallada para fundamentar la necesidad de este proceso a los mismos responsables de los proyectos que sufren las consecuencias de no hacerlo.

Este proceso es simple en lo conceptual y complicado en su implementación. Se trata de acordar qué cambios se aceptan realizar, cuáles se postergarán, cuáles se rechazarán y por qué razón. Cada cambio propuesto o pedido en el proyecto pasa por diferentes estados de acuerdo a su ciclo de vida. Esta administración es en general llevada adelante con alguna herramienta de soporte que facilita, entre otras cosas, realizar el seguimiento de cada cambio según su estado, evaluación de su impacto y asignación de las tareas asociadas a este.

En la figura 4.15 se muestra un posible ciclo de vida de cambios a los requerimientos de un proyecto. Se muestran los diferentes estados por los que pasa una solicitud de cambio, las transiciones posibles y los involucrados en el proceso.

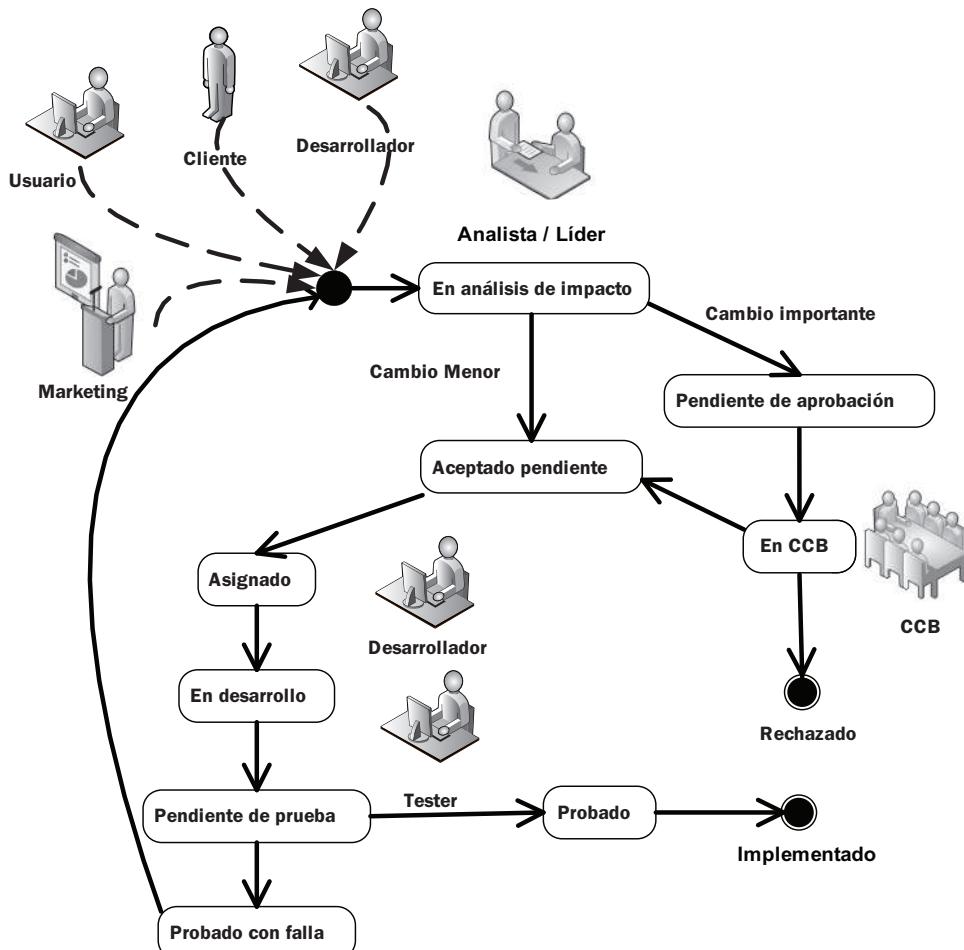


Fig. 4.15 - Flujo de trabajo del proceso de administración de cambios a los requerimientos

Un error común que he observado es la implementación de este proceso sin tener en cuenta el tamaño del proyecto, del grupo de desarrollo y de la cultura de la organización. Todas estas características influyen y deben respetarse con mayor rigor en proyectos grandes y organizaciones corporativas.

#### **4.6.2 ALTERNATIVAS DE SOLUCIÓN**

En organizaciones pequeñas y medianas el CCB (Change Control Board) estará integrado solo por el responsable del proyecto, el referente de parte del cliente y la persona a la cual se le asignó el rol de analista. La frecuencia de reuniones y análisis de los cambios será adaptada a la velocidad con que se generen los cambios. La idea es agruparlos para su tratamiento.

En organizaciones corporativas con proyectos grandes y numerosos involucrados, el CCB estará constituido por un referente de cada parte, el responsable del proyecto, analistas, líder técnico y gerentes. La frecuencia de análisis de los cambios será prefijada de antemano en quince a veinte días.

En todos los casos es necesario e imprescindible:

Documentar todos y cada uno de los análisis de impacto y los acuerdos de las reuniones del CCB.

La asistencia de involucrados con poder de decisión en la planificación del proyecto (tiempos, presupuesto, características del producto en desarrollo).

##### **4.6.2.1 Nota para desarrolladores ágiles**

Cuando hablamos de controlar los cambios no nos referimos a definir sistemas cuyos requerimientos permanecerán congelados y carentes del valor que proporciona la adaptación de las especificaciones a la evolución del negocio, de ninguna manera, sino a instrumentar un mecanismo que permita balancear los esfuerzos de desarrollo con las actividades necesarias para implementar los cambios surgidos. Lo hacemos convencidos de que los proyectos aparentemente flexibles, en los cuales se da cabida a todos los cambios solicitados, terminan desestabilizados y en consecuencia no aportan ningún valor al negocio.

---

## **4.7 | CONCLUSIÓN**

---

De lo expuesto se concluye que el trabajo con los requerimientos es mucho más que relevar y listar los “el sistema debe...” de todo proyecto de desarrollo.

Es de primera importancia planificar el trabajo con los requerimientos.

Es fundamental relevar los requerimientos desde los diferentes enfoques (Foco), construyendo los diferentes modelos (Vistas) y trabajando según el momento del ciclo de vida del proyecto (Nivel).

Es necesario que el listado cuente con determinadas propiedades (detallado, atómico, completo, correcto, no ambiguo) para no ser solo un listado de características.

Se debe, a partir de la especificación del comportamiento de la aplicación a desarrollar (casos de uso), definir cómo se implementarán estos requerimientos.

Es necesario contar con un modelo de dominio que siente las bases para un posterior diseño y que esencialmente describa al negocio.

En todos los casos, administrar los cambios surgidos durante el proceso de desarrollo.

Por último debemos validar y verificar todo este trabajo con los requerimientos.

---

## **Capítulo 5: Trabajo con la Gestión de Proyectos**

---

<b>5.1   Proyectos .....</b>	105
5.1.1 Planes y planificación .....	105
5.1.2 Cascada versus iteraciones .....	106
5.1.2.1 La dinámica de las iteraciones .....	107
5.1.2.2 Las vistas de los roles .....	109
5.1.3 Planificación de iteraciones .....	111
5.1.3.1 Medidas de estabilidad .....	112
5.1.4 Fases, actividades, objetivos .....	114
5.1.5 Cuestiones a tener en cuenta y algunas recomendaciones.....	116
5.1.5.1 A tener en cuenta .....	116
5.1.5.2 Recomendaciones .....	117
5.1.6 Condiciones de contexto .....	117
<b>5.2  Planificación de proyectos .....</b>	118
5.2.1 Estrategia .....	120
5.2.1.1 Un caso demostrativo.....	120
5.2.2 Construcción de una estrategia .....	122
5.2.2.1 Visión.....	122
5.2.2.2 Objetivos .....	122
5.2.2.3 Prioridades .....	123
5.2.2.4 Riesgos .....	123
5.2.2.5 Estimaciones .....	124
5.2.2.6 Estrategia.....	127
<b>5.3   Seguimiento del desarrollo de proyectos .....</b>	130
5.3.1 Roles.....	130
5.3.1.1 Qué cosas debe hacer un líder de proyectos para cubrir sus responsabilidades..	130
5.3.1.2 Qué cosas no debe hacer un líder de proyectos .....	131
5.3.2 Actividades .....	131
5.3.3 Puntos de observación.....	132
5.3.4 Fotos versus película.....	133
5.3.4.1 Tratamiento de una decena de temas .....	133
5.3.4.2 No seguimiento de los temas tratados .....	133
5.3.5 Escalamiento .....	133
5.3.6 Acciones.....	134
5.3.7 Métricas .....	134
<b>5.4  Conclusión .....</b>	137

# 5

## TRABAJO CON LA GESTIÓN DE PROYECTOS

### 5.1 | PROYECTOS

---

Las actividades que componen la gestión de los proyectos si son bien desarrolladas contribuirán a minimizar la influencia de las causas que atentan contra la degradación de la calidad de los productos de software. Estas actividades tendrán como objetivos lograr el involucramiento de la organización cliente, el compromiso de la organización desarrolladora, facilitar las condiciones para desarrollar un claro entendimiento de los requerimientos y administrar la construcción del software de una manera controlada, entre otros. En este capítulo nos dedicaremos a enfatizar cuáles son, a nuestro criterio, los errores que los líderes de proyecto cometemos a lo largo del ciclo de vida de nuestros proyectos y cuáles son las actividades con falencias que generan estos errores.

Asumimos que los lectores conocen las prácticas de la gestión al estilo del Project Management Institute (PMI)<sup>34</sup>. Sin embargo quiero decir que siempre fui crítico de las personas que hacen uso de estos conocimientos, que creen que con algunos números generados por estas prácticas la gestión de los proyectos es una actividad que consiste en una secuencia de tareas desarrolladas casi en forma automática.

Todo lo que analizaremos en estas secciones aplica a proyectos de desarrollo llevados adelante por un grupo de desarrolladores en el contexto de una organización. Están excluidos los emprendimientos personales, compartidos por un par de personas, así como los proyectos cuyo objeto de trabajo no es el desarrollo de software, para los cuales deberían hacerse consideraciones especiales.

#### 5.1.1 PLANES Y PLANIFICACIÓN

Como nativo de mi tierra, tengo una costumbre que he heredado de mis mayores que consiste en organizar comidas los fines de semana y en ocasiones como cumpleaños, graduaciones, casamientos y encuentros con amigos. Todos los domingos con regularidad, hace alrededor de treinta años, asamos carne. Teniendo en cuenta las cincuenta y dos

semanas de cada año, la cantidad de asados asciende a mil quinientos sesenta. Con el tiempo he reparado que, aun teniendo ya una gran experiencia en estas lides, antes de cada evento planificamos su realización.

Entre las cosas que tenemos en cuenta podríamos mencionar: quiénes asistirán, qué tipo de carne gustan comer, dónde nos reuniremos para asar y comer la carne, qué tipo de bebidas compraremos, cuánta ensalada y pan serán necesarios, cuál será el postre, quién se ocupará de realizar las compras, qué decisión tomaremos si llueve (suspensión o cambio de lugar), qué cantidad y tipo de leña necesitaremos.

No nos preguntamos por qué organizamos dicho evento, obviamente lo sabemos, sin embargo la respuesta a esta pregunta condiciona muchas de las respuestas a las preguntas anteriores. Con esto quiero decir que una reunión con amigos es diferente del cumpleaños de un familiar con invitados y aun de una reunión con compañeros de trabajo.

En general confeccionamos un listado con los productos que vamos a comprar y sus cantidades, con alguna indicación de proveedor, los utensilios que necesitaremos (vajilla, parrilla, mesas, sillas, etc.) y los nombres de los responsables de trasladar tal o cual cosa. Pero más allá de las listas, he llegado a la conclusión de que el éxito de nuestras reuniones es el ejercicio de planificación que llevamos adelante.

Salvando las distancias, un asado es en casi todos los sentidos similar a un proyecto de desarrollo: por qué haremos el evento (visión del proyecto), utencillos necesarios (infraestructura necesaria), cantidad de alimentos (alcance del proyecto), tipo de carne al gusto de todos (todos los involucrados), qué hacemos si llueve (administración de riesgos), horarios de comida y otras actividades recreativas (cronograma), etc. Por esta razón deseo rescatar la importancia del ejercicio de planificación por sobre los planes. Es una decisión de sentido común la confección de listas de ayuda memoria en nuestros asados y lo es también en los proyectos. Estas listas son solo un papelito con unas pocas líneas en reuniones de entrecasa con cuatro personas, pero son más importantes cuando el evento es mayor en número de asistentes y formalidad. Así lo son también los planes en los proyectos. Sin embargo, el ejercicio de planificación siempre debe ser desarrollado.

Así como en nuestros asados surgen imprevistos, también en los proyectos, y es este ejercicio continuo de planificación lo que nos permite resolver los diferentes problemas. Un producto que no es provisto a tiempo, utencillos faltantes, arribo de nuevos asistentes no tenidos en cuenta en nuestras listas, no asistencia de invitados, lluvia y otros pueden compararse en el escenario correspondiente a infraestructura no entregada a tiempo, cambios en el alcance, compromisos no cumplidos por las diferentes partes, etc. Siempre resolveremos estos imprevistos de una mejor manera a partir de un ejercicio de planificación continuo. Los planes son importantes, la planificación continua es imprescindible.

### 5.1.2 CASCADA VERSUS ITERACIONES

Una de las contribuciones más importantes de los lenguajes orientados a objetos ha sido el mecanismo de encapsulamiento. Este permite trabajar en forma paralela en diferentes módulos de un software, a partir de la definición de interfaces estables.

También es factible trabajar en forma secuencial en dichos módulos y al momento de construir el segundo no tener la necesidad de modificar el primero ya implementado. Esta posibilidad permite trasladar esta forma de trabajo a los entregables de un proyecto, los cuales podrán ser construidos de manera de una agregación al sistema en desarrollo. El sistema crece a medida que los diferentes módulos son diseñados en detalle, codificados y probados. Decimos que estos entregables están clausurados ante cambios, es decir, son extensibles sin necesidad de modificación. El trabajo de esta forma es conocido como iterativo y es la solución a diferentes problemas que se plantean cuando se trabaja en cascada. Este último nombre proviene de la realización de las diferentes tareas de un proyecto en forma secuencial, es decir una a continuación de la otra, con la escasa realimentación que permiten las herramientas procedurales, las cuales son vulnerables a las modificaciones por contar con un encapsulamiento mucho más pobre. ¿Por qué esta forma de trabajo tiene ventajas? ¿Cuáles son estas ventajas? ¿Cómo llevamos adelante esta forma de trabajo? ¿Cómo gestionamos un proyecto con esta forma de trabajo? ¿Qué aspectos de la planificación de un proyecto se ven alteradas por esta forma de trabajo? Estas preguntas son las que contestaremos en las siguientes secciones de este capítulo<sup>35</sup>.

### 5.1.2.1 La dinámica de las iteraciones

Apelaré a otro ejemplo no vinculado al software para exponer este tema. Supongamos que nos dedicamos al negocio de turismo y organizamos excursiones por nuestra ciudad para los visitantes. Cada viaje es organizado de acuerdo a la solicitud de los viajeros. Es otro tipo de proyectos. Sin embargo posee las características de interés para el aspecto a tratar. Al momento de la primera entrevista nuestros huéspedes nos piden visitar un conjunto de destinos (Catedral, Museo de Ciencias Naturales, Centro de la Ciudad, República de los Niños, Estadio Único de Fútbol, Zona de Destilería, Puerto, Pasaje Histórico del Fundador, Zoológico). Además nos piden realizar el recorrido sin prisa, ya que desean detenerse en los lugares visitados. También nos cuentan que disponen de solamente un día. Como en el caso de nuestros asados hacemos un ejercicio de planificación en el que realizamos las siguientes preguntas:

¿Por qué están interesados en realizar la excursión?

¿Por qué desean conocer cada uno de los lugares?

¿Cuántos viajeros harán el viaje?

¿Qué día?

¿Cómo prefieren trasladarse?

¿Desean tener un almuerzo formal o informal?

¿Cuánto dinero, como máximo, están dispuestos a gastar?

A partir del análisis de las respuestas concluimos que no será posible realizar la excursión en un solo día, sin embargo los turistas no cuentan con más tiempo.

35 | Jacobson, Ivar. Grady Booch & Jim Rumbaugh. *Unified Software Development Process*. Addison-Wesley. 1999.

También deducimos que el presupuesto que dijeron estar dispuestos a gastar es insuficiente. A la luz de estas conclusiones, ¿qué decidimos hacer?, ¿tenemos alternativas?

Una posibilidad es planificar una agenda de actividades cerrada y seguramente apretada con un itinerario fijo preestablecido con arriba a los diferentes lugares de acuerdo a su ubicación relativa, estadias limitadas en cada lugar, con compra de antemano de entradas, consumiciones y otros gastos. Es una alternativa riesgosa, cualquier imprevisto retrasará la excursión y quedarán sitios sin recorrer, consumiciones y entradas pagas sin utilizar y casi con seguridad la inconformidad de los viajeros.

Otra alternativa es planificar un itinerario abierto, visitar primero los lugares de acuerdo a una prioridad acordada en base al interés de los turistas. Compraremos las entradas y consumiciones en cada lugar al momento de necesitarlas y nos trasladaremos de un lugar a otro cuando los viajeros lo soliciten. De acuerdo a cómo gastemos el tiempo, les ofreceremos alternativas, comentando la forma en que se va desarrollando el paseo en cada momento.

Es seguro que no lograremos visitar todos los lugares, sin embargo también es altamente probable que la satisfacción de nuestros clientes será mayor que con la alternativa anterior. Habremos visitado los lugares de mayor interés y no habrán gastado en servicios que luego no tomaron. También habrán participado de las decisiones que fuimos tomado en torno al itinerario a seguir en cada momento. Es muy posible que nuestros viajeros regresen en el futuro a contratarnos una excursión a los lugares que no pudimos visitar.

Al igual que en los proyectos de desarrollo de software se trata de cambiar la situación en la cual finalizado el tiempo del proyecto no hemos cumplido con los compromisos asumidos, el cliente ha pagado por algo que no recibe y de lo que recibe, gran parte no es lo que quería.

Estas situaciones se presentan todo el tiempo en los proyectos de desarrollo de software. Estos se venden con precio, características del producto a desarrollar y planificación cerradas. El resultado de estos proyectos es el mismo que en la primera alternativa de los viajes de excursión.

*Retomando la alternativa iterativa, veamos cuáles son sus claves<sup>36 37</sup>:*

- Tecnología que permita hacer crecer el sistema en desarrollo con cada iteración de forma que el producto construido esté clausurado ante cambios.
- Acuerdo en la priorización de las características del producto a construir entre la organización desarrolladora y el cliente.
- Participación del cliente en los proyectos de desarrollo.

Cuando tratemos la planificación de las iteraciones nos extenderemos en estos puntos.

En la figura 5.1 se muestran la dinámica de las iteraciones según se analizó y las tareas fundamentales de cada una de ellas:

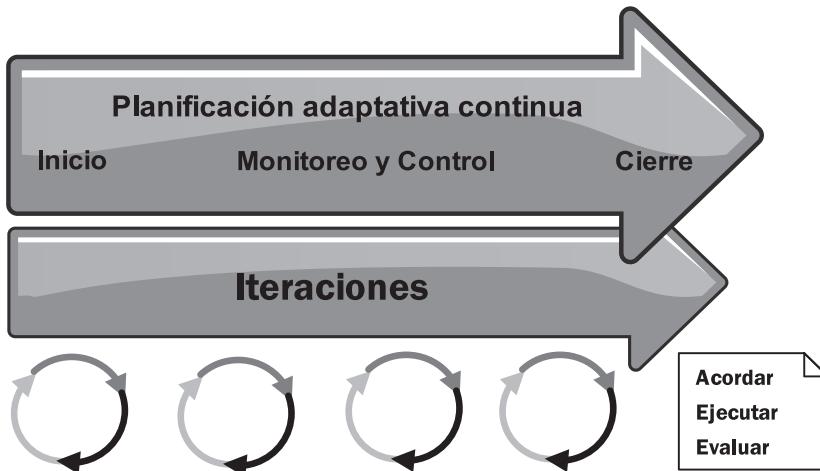
---

36 | Cokburn, Alistar. *Agile Software Development*. Addison-Wesley. 2001.

37 | Beck, Kent. *Extreme Programming Explained*. Addison-Wesley. 2000.

**Acordar:** para establecer prioridades, forma de trabajo y planificación.

**Ejecutar:** para construir el software según lo planificado.



**Evaluar:** para reforzar los aciertos, corregir los errores y replanificar.

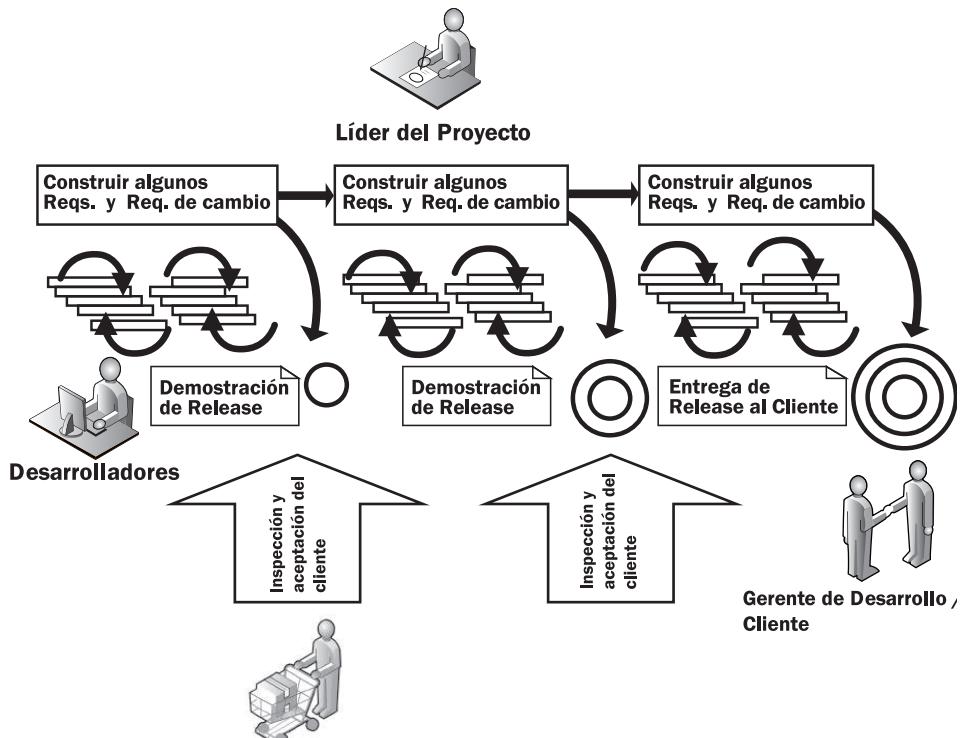
**Fig. 5.1** - Esquema mostrando la dinámica de los proyectos iterativos

### 5.1.2.2 Las vistas de los roles

Una de las características del desarrollo iterativo es el carácter más autónomo de los grupos de desarrollo y la delegación a sus miembros para la toma de decisiones. Por esta razón los roles involucrados tienen necesariamente diferentes vistas del proyecto a lo largo del ciclo de vida. En la figura 5.2, adaptada de Kurt Bittner y Ian Spence<sup>38</sup>, se muestran estas vistas. Los desarrolladores se concentran en cada iteración y su foco son los requerimientos a implementar en cada una de las iteraciones. El líder del proyecto tiene la visión de la secuencia de iteraciones y los requerimientos a implementar en cada iteración en base a las prioridades acordadas. Trabaja en todo momento administrando los riesgos, conducido por la estrategia fijada para el proyecto. La vista del cliente/usuario está dada por los entregables de cada iteración y su verificación con relación a los requerimientos. El cliente contribuye generando realimentación hacia el seno del grupo de desarrollo.

<sup>38</sup> Bittner, Kurt. Ian Spence. *Managing Iterative Software Development Projects*. Addison-Wesley. 2006.

### Las vistas de un proyecto iterativo según los roles



**Fig. 5.2** - Vistas de los distintos roles del proceso iterativo de desarrollo de software

El gerente de desarrollo observa el proyecto desde la vista compuesta por el seguimiento de los compromisos, la generación de entregables que muestran el avance y los acuerdos con el cliente. Esta visión es posible debido a la delegación de responsabilidades que realizó en el líder y a su vez este en los desarrolladores. Esta forma de trabajo es la necesaria y la única posible para que el proyecto evolucione en iteraciones acotadas en el tiempo (15 a 30 días); en este escenario las decisiones y los acuerdos deben ser rápidos a efectos de no convertirse en obstáculos para la evolución del proyecto. Es importante notar que esta forma de trabajo requiere delegación y asunción de responsabilidades por parte de todos los roles de ambas partes: organización desarrolladora y cliente. No hay lugar para gerentes que quieren conocer detalles de la implementación del producto ni para desarrolladores que no toman decisiones sin antes conocer la opinión de los gerentes. Se trata de generar un ambiente en el que todos los roles se apropien del proyecto y trabajen colaborando con el resto de los involucrados para lograr los objetivos planteados. De lo expuesto se deriva que es muy importante la actitud y aptitud de los miembros de las organizaciones. En una próxima sección trataremos temas de capacitación y nos ocuparemos de cómo lograr que los miembros de las organizaciones estén preparados para este tipo de funcionamiento.

### 5.1.3 PLANIFICACIÓN DE ITERACIONES

La planificación de las iteraciones de un proyecto comienza cuando hemos priorizado los requerimientos a nivel de paquetes. La pregunta es qué paquete construiremos primero. Luego de responder esta pregunta, la reformularemos y nos preguntaremos qué casos de uso del paquete construiremos primero. Los criterios a aplicar para la priorización en ambos casos son similares. Pero estos serán tratados en una próxima sección. Por el momento nos basta con el listado de casos de uso a construir con sus respectivas prioridades. Supongamos que los hemos clasificado en:

- 1. Esenciales:** son aquellos sin cuya funcionalidad no puede llevarse adelante el negocio.
- 2. Importantes:** son aquellos que aportan valor y complementan a la funcionalidad esencial de forma que el negocio puede llevarse adelante y además con el valor agregado por estos.
- 3. Menos importantes:** son aquellos que decoran la funcionalidad esencial pero que sin embargo el negocio puede llevarse adelante sin ellos. No aportan gran valor agregado.
- 4. Poco importantes:** son aquellos que suman valor accesorio a la realización del negocio.

En base a esta clasificación de las prioridades de cada requerimiento, podemos seleccionar una de dos alternativas para planificar las iteraciones.

#### Iteraciones de duración fija

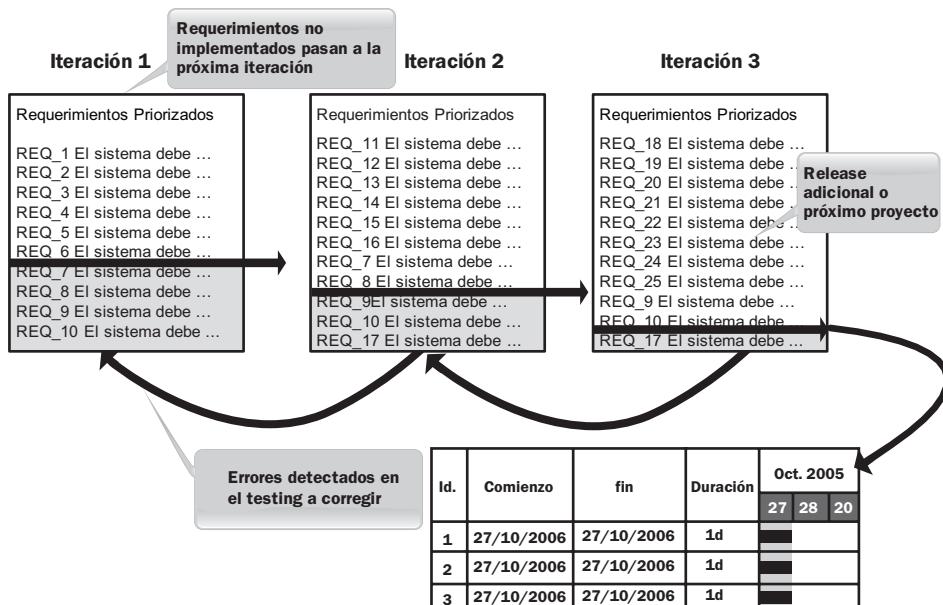


Fig. 5.3 - Requerimientos priorizados se implementan en cada iteración

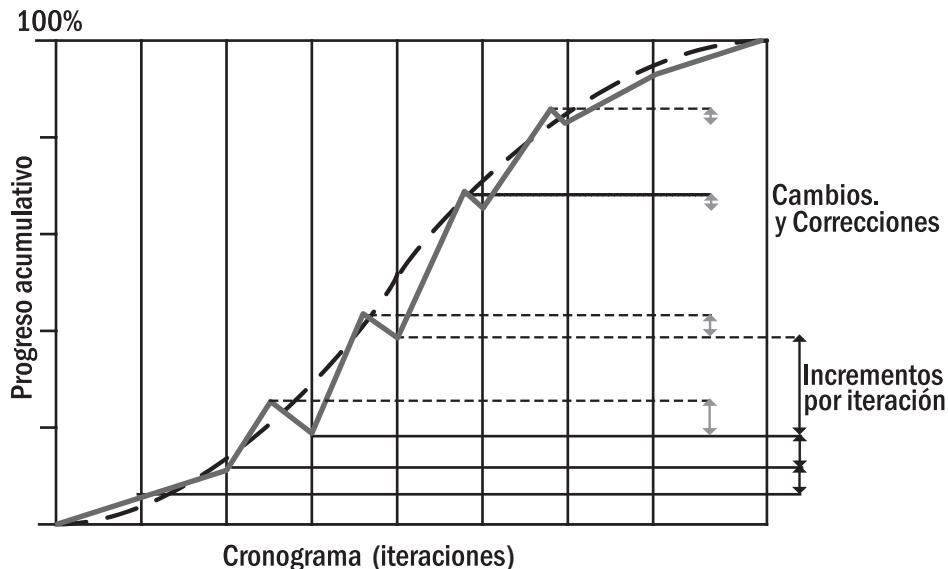
**Iteración de alcance fijo:** se trata de establecer una planificación que incluya la construcción de los componentes que implementan toda la funcionalidad de uno o más casos de uso a partir de una estimación de esfuerzo.

**Iteración de tiempo fijo:** se trata de prefijar la duración de la iteración y trabajar para maximizar los componentes construidos guiados por las prioridades acordadas. En la figura 5.3 se muestra en forma esquemática esta alternativa.

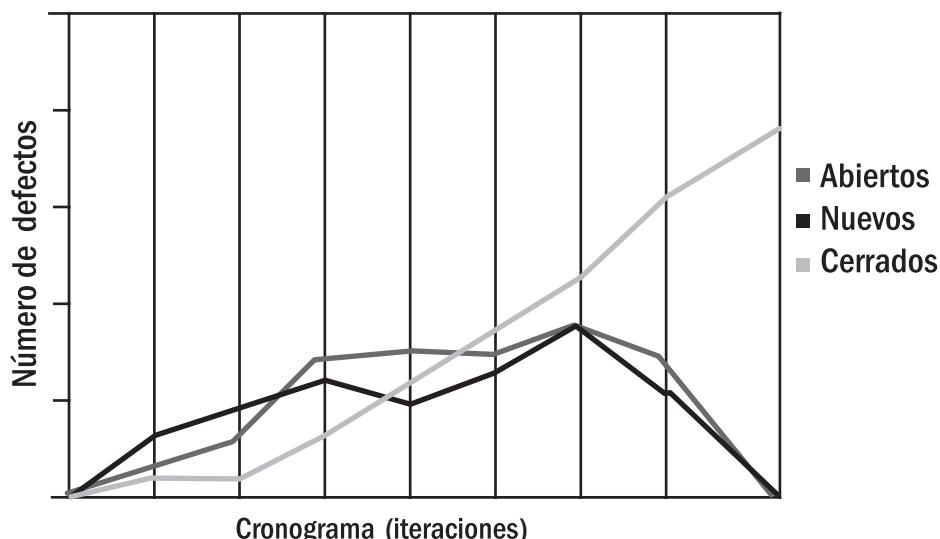
En mi opinión esta segunda alternativa es la mejor dado que permite generar, con seguridad, entregables con una frecuencia determinada. También ayuda a aprender la velocidad de desarrollo de los grupos. No existe un criterio establecido para la determinación de la duración, pudiendo probarse con 15 días para proyectos chicos y 30 días para proyectos grandes. Lo importante es respetar a rajatabla la duración y administrar los requerimientos que no se hayan podido construir en este tiempo de manera de trasladarlos a la siguiente iteración y volver a priorizarlos junto con los seleccionados para esta. El primer objetivo es terminar la construcción de todos los componentes. Si no es posible, entonces se terminarán los priorizados como esenciales e importantes y se reasignarán los menos importantes y poco importantes para la próxima. Si no es posible lograr estos objetivos, estamos frente a un caso de mala estimación de esfuerzo por optimismo o debido a algún riesgo que terminó convirtiéndose en problema. De cualquier forma, si este es el caso, la iteración ha fallado. Trataremos las acciones a seguir en la sección Seguimiento del desarrollo de proyectos. *Es importante aclarar que la alternativa propuesta implica un costo fijado, una planificación fijada (tiempo) y un alcance variable de acuerdo a las prioridades mencionadas para el proyecto. Esta forma debe ser claramente comunicada y explicada a la organización cliente.*

### 5.1.3.1 Medidas de estabilidad

Un aspecto importante es que en cada una de las iteraciones además de construir (refinar el análisis, diseñar, codificar y probar) cada componente deberemos corregir errores detectados en las pruebas. La mayor parte de las veces es necesario asignar tiempos en cada iteración relacionado con los de iteraciones anteriores. En las figuras 5.4 y 5.5 se muestran las evoluciones de dos indicadores que indican el grado de estabilidad de un proyecto de desarrollo iterativo. En la primera se observa la variación de los incrementos en la implementación de los distintos requerimientos a medida que el proyecto avanza y se desarrollan las iteraciones planificadas. Estos incrementos deben aumentar, llegar a su máximo al promediar el proyecto y luego decrecer a medida que el proyecto se acerca a su terminación. De la misma manera, en un proyecto estable los cambios y correcciones deben decrecer cuando el proyecto avanza, de lo contrario el proyecto no convergerá a un estado estable. En la segunda figura se muestra la evolución de los errores detectados y corregidos para un proyecto estable. Estos indicadores son fáciles de construir y sus medidas sencillas de tomar.



**Fig. 5.4** - Métricas mostrando la evolución de los incrementos del sistema en desarrollo y los errores y cambios pedidos a lo largo de las iteraciones



**Fig. 5.5** - Métricas mostrando la evolución de los defectos en un proyecto estable

Estos indicadores contribuirán a tomar acciones para corregir sesgos indeseados cuando los proyectos muestren signos de inestabilidad.

#### 5.1.4 FASES, ACTIVIDADES, OBJETIVOS

Independientemente de la metodología utilizada, un proyecto de desarrollo pasa por algunos estados que llamaremos fases, sin referirnos, de antemano, a ningún proceso determinado. Estos estados o fases son la concepción del sistema o producto a desarrollar, la elaboración o definición de la arquitectura que lo soportará, la construcción del software y la transferencia o instalación en el ambiente productivo.

Es muy importante entender claramente cuándo empiezan y cuándo terminan estas fases y qué significado tienen estos hitos o comienzos y finalizaciones.

Una deformación que he observado en muchos proyectos de las más diversas empresas de desarrollo es que los miembros del grupo “viven” estas fases a partir de las actividades asignadas a su rol. Por ejemplo, en los primeros días del proyecto se relevan requerimientos hasta que un cronograma dice que esa fase que estaba planificada debería terminar. Por lo tanto se deja de relevar requerimientos y se pasa a la próxima tarea. Este comportamiento seguramente fue adquirido a partir de la falsa visión que se tiene de las metodologías y sus fases, en algunos casos influenciada todavía por la organización en cascada de algunos proyectos. En general la gente asocia los momentos de los proyectos con las actividades que en ellas se llevan adelante. Las actividades son importantes, ya que si no las realizamos el proyecto no avanza, pero esta visión es falsa y conduce a errores a veces irreparables.

Lo esencial es que los miembros del grupo adquieran una comprensión del proceso de desarrollo a partir de los objetivos a cumplir en cada momento. Es fundamental esta visión, ya que se debe trabajar (desarrollar actividades) cada día en la dirección de estos objetivos. Podríamos enfatizar este análisis diciendo que se puede relevar requerimientos durante meses y no lograr achicar los riesgos de falta de conocimiento del negocio a partir de ellos. Se pueden hacer definiciones y diseñar la arquitectura del sistema por largo tiempo, sin lograr achicar los riesgos asociados a la tecnología en relación con el sistema en desarrollo.

Cuando estas cosas suceden, aunque un cronograma diga que estamos trabajando en el proyecto desarrollando tal o cual fase, aún nos encontramos en alguna anterior por no haber logrado los objetivos y por no haber alcanzado nunca el hito de terminación de la fase correspondiente. Se ha producido un desfasaje entre los planes y la realidad del proyecto.

*La realización de actividades en el proceso de desarrollo de un proyecto no evidencian ningún hito si no se lograron los objetivos.*

En la tabla 5.1 (adaptada de Kurt Bittner y Ian Spence, op. cit.) se presentan las fases mencionadas y los objetivos correspondientes que deberían buscarse cumplir en ellas. Pueden verse los riesgos que conducen las actividades para el logro de los objetivos y los hitos de cada una de las fases.

Concepción	Elaboración	Construcción	Transferencia
Foco del riesgo	Foco del riesgo	Foco del riesgo	Foco del riesgo
<ul style="list-style-type: none"> <li>Negocio</li> </ul>	<ul style="list-style-type: none"> <li>Arquitectura</li> </ul>	<ul style="list-style-type: none"> <li>Logística</li> </ul>	<ul style="list-style-type: none"> <li>Ambiente Productivo</li> </ul>
<b>Preguntas</b> <ul style="list-style-type: none"> <li>• ¿Estamos construyendo la cosa correcta para el cliente?</li> <li>• ¿Es la solución factible?</li> <li>• ¿Cuánto esfuerzo costará?</li> </ul>	<b>Preguntas</b> <ul style="list-style-type: none"> <li>• ¿Sabemos qué estamos construyendo?</li> <li>• ¿Sabemos cómo construirlo?</li> <li>• ¿Estamos de acuerdo en qué construir?</li> <li>• ¿Cuánto dinero costará?</li> <li>• ¿Cuáles son los riesgos técnicos?</li> <li>• ¿Podemos mitigarlos?</li> </ul>	<b>Preguntas</b> <ul style="list-style-type: none"> <li>• ¿Lo estamos construyendo?</li> <li>• ¿Estará terminado a tiempo?</li> <li>• ¿Es suficientemente bueno?</li> <li>• ¿Mantenemos nuestros supuestos y decisiones?</li> <li>• ¿Están los usuarios listos?</li> </ul>	<b>Preguntas</b> <ul style="list-style-type: none"> <li>• ¿Es aceptable?</li> <li>• ¿Está siendo usado?</li> <li>• ¿Hemos terminado</li> </ul>
<b>Artefactos claves</b> <ul style="list-style-type: none"> <li>Visión</li> <li>Riesgos</li> <li>Plan de proyecto</li> <li>Listado de casos de uso críticos</li> <li>Modelo de negocios</li> </ul>	<b>Artefactos claves</b> <ul style="list-style-type: none"> <li>Modelo de casos de uso</li> <li>Especificación de los casos de uso principales</li> <li>Especificación de la arquitectura</li> <li>Prototipo de arquitectura</li> <li>Prueba de arquitectura</li> </ul>	<b>Artefactos claves</b> <ul style="list-style-type: none"> <li>Especificación de casos de uso</li> <li>Diseño</li> <li>Código</li> <li>Pruebas</li> <li>Resultado de pruebas</li> <li>Material de capacitación y documentación de usuario</li> </ul>	<b>Artefactos claves</b> <ul style="list-style-type: none"> <li>Instaladores</li> <li>Convertidores de datos</li> <li>Listado de últimos defectos y resoluciones</li> </ul>
<b>Salidas</b> <ul style="list-style-type: none"> <li>Acuerdo de proyecto</li> </ul>	<b>Salidas</b> <ul style="list-style-type: none"> <li>Arquitectura probada</li> </ul>	<b>Salidas</b> <ul style="list-style-type: none"> <li>Solución probada, documentada y lista para instalar</li> </ul>	<b>Salidas</b> <ul style="list-style-type: none"> <li>Solución instalada y funcionando en forma productiva</li> </ul>
<b>Factibilidad Acordada</b>			
<b>Alternativa Probada</b>			
<b>Solución Disponible</b>			
<b>Proyecto Terminado</b>			

Tabla 5.1.- Fases de los proyectos con sus hitos y objetivos

Otra falsa visión de algunas personas es la que ofrece la documentación de los proyectos. Cuando esta no es utilizada para los fines para los que fue concebida, a veces se tiene la visión de que el proyecto está en un estado determinado por el solo hecho de haber construido, por ejemplo, una Especificación de Requerimientos (ERS) o un Plan de Proyecto (PP). La utilidad mayor de la documentación es comunicar. Una ERS o un PP elaborado no determina ningún hito ni el logro de objetivos si no están acordados con todos los involucrados.

Otra vez, si se comete este error, los planes del proyecto y su estado real estarán desfasados. Me ha tocado revisar proyectos en los cuales al día de hacerlo se trabajaba en las pruebas de código ya escrito cuando todavía no se había acordado el alcance a partir de la ERS. Esto es como participar en una competencia atlética por etapas y creer que se está en la segunda cuando en realidad no hemos alcanzado la meta de la primera, ya que cambiamos el rumbo antes de arribar. Creemos estar más cerca del final de la prueba porque corremos en la dirección correcta pero sin advertir que estamos descalificados y haciendo esfuerzos en vano ya que nunca finalizamos la primera de las etapas.

Un documento terminado no implica ningún logro en el proyecto ni evidencia ningún grado de avance si no está acordado y aprobado por todos los involucrados.

En la tabla 5.2 se comparan ambas visiones (adaptada de Kurt Bittner y Ian Spence, op. cit.).

<b>Vista desde los Objetivos y no de los Productos</b>		
<b>Hitos</b>	<b>Interpretación incorrecta</b>	<b>Interpretación correcta</b>
Fin de la concepción del sistema	Planificación completa	La factibilidad del proyecto está acordada
Fin de la definición de la arquitectura	Especificación completa	La alternativa propuesta ha sido probada
Fin de la construcción	Codificación completa	Una solución usable está disponible
Fin de la transferencia	Producto transferido	El proyecto está terminado

**Tabla 5.2** - Diferentes formas de interpretación de los hitos de un proyecto

En la próxima sección presentamos a modo de guía algunas recomendaciones a efectos de lograr una buena gestión de los proyectos de desarrollo sin que las falencias mencionadas se conviertan en verdaderos obstáculos.

## 5.1.5 CUESTIONES A TENER EN CUENTA Y ALGUNAS RECOMENDACIONES

### 5.1.5.1 A tener en cuenta

- Capacitar e inducir a todos los involucrados (miembros de la organización desarrolladora y de la organización cliente) acerca de la forma de trabajo en el proyecto para evitar falsas visiones.

- Asegurarse de que todos los involucrados comprenden los hitos y objetivos asociados a las fases del proyecto.
- Los riesgos condicionan el cumplimiento del Plan, el cual debe adaptarse a partir de un ejercicio continuo de planificación.
- Las pruebas y entregas frecuentes reducen los riesgos.
- La documentación es utilizada para comunicar (acordar).
- Las actividades realizadas únicamente con foco en los objetivos nos conducirán a su logro.

### 5.1.5.2 Recomendaciones

Estas recomendaciones están basadas en Los siete hábitos de la gestión exitosa de proyectos (Kurt Bittner, Ian Spence, op. cit.).

1. Encausar el trabajo de manera que la solución avance de a pequeños pasos que generen entregables.
2. Poner el foco en generar resultados sin temor a fallar.
3. Ejercitarse la adaptación de la planificación en forma continua.
4. Siempre con atención en los riesgos.
5. Siempre abierto y honesto (visibilidad).
6. Basarse en evaluaciones obtenidas a partir de mediciones objetivas y no subjetivas.
7. Focalizarse en la meta de entregar una solución funcionando.

### 5.1.6 CONDICIONES DE CONTEXTO

Cuando hace unos quince años leí en un artículo que una de las mayores dificultades en la implementación de un proceso de trabajo iterativo era la resistencia de las gerencias me sorprendí y pensé que el autor del trabajo exageraba la situación. Sin embargo, después de todo este tiempo, aún hoy cuando reviso proyectos compruebo que todavía es así. La razón de este fenómeno es que no forma parte de la cultura de las organizaciones la *delegación de responsabilidades* necesaria para el trabajo con grupos cohesivos y autónomos. Este comportamiento genera además otro comportamiento en los desarrolladores que se manifiesta como la falta de compromiso y la *no asunción de responsabilidades*. En mi opinión este último es consecuencia del primero.

Otra falencia de las organizaciones a la hora de implementar una forma iterativa de trabajo es la falta de *madurez para compartir un proyecto con la organización cliente*. Me refiero a la tan mal utilizada frase: “el cliente incluido en el proyecto”. Como en toda relación el comportamiento de la otra parte también se ve afectado. Existe, está instalada en el mercado, la postura de que la organización desarrolladora trabaja y es la responsable del proyecto, y que la organización cliente pide y paga, sin responsabilidad ninguna en el proyecto. Cuando en realidad el proyecto es de

ambas, es un *bien compartido* y ambas partes tienen *reponsabilidades diferentes*, pero responsabilidades al fin. Necesitamos al cliente trabajando en el proyecto para aportar su conocimiento del negocio, para priorizar requerimientos, para definir pruebas de aceptación pero por sobre todo para que participe de cada decisión y esté informado al día de la evolución de los trabajos que se hacen y se dejan de hacer. Esta visibilidad es el único medio de generar confianza para introducir cambios, replanificar, quitar o agregar trabajos adicionales, alterar el presupuesto y llevar al proyecto a buen fin; es decir, a generar un sistema o producto que agregue valor al negocio del cliente. Para esto el cliente debe romper con la postura de que “yo pago para que otros trabajen” y las organizaciones desarrolladoras deben madurar y compartir con sus clientes sus fortalezas y debilidades.

A modo de resumen listamos las que a nuestro entender son las condiciones que deben generarse para implementar una forma de trabajo iterativo, como indican Barry Boehm y Richard Turner<sup>39</sup>, y Mary Poppendieck, Tom Poppendieck<sup>40</sup>:

- Delegación de responsabilidades de los dirigentes de la organización desarrolladora hacia sus miembros desarrolladores.
- Actitud y aptitud de los miembros desarrolladores para asumir las responsabilidades delegadas.
- Madurez de la organización desarrolladora para compartir el proyecto con el cliente.
- Madurez de la organización cliente para colaborar trabajando en el proyecto con algunos de sus miembros.

La responsable de generar estas condiciones es la organización desarrolladora. Una vez que sus dirigentes delegan responsabilidad en sus miembros y adoptan una postura de transparencia (visibilidad) para con el proyecto, los líderes deben vender a la organización cliente la forma de trabajo y fundamentar todos y cada uno de los beneficios que esta forma de trabajo aporta.

## 5.2 | PLANIFICACIÓN DE PROYECTOS

---

Como mencionamos en la Introducción, la presentación detallada de los temas que tratamos no es el objetivo del libro, sino enfatizar los aspectos dentro de las actividades desarrolladas en un proyecto que presentan costados críticos que pueden degradar la calidad.

En los últimos años me ha tocado revisar muchos proyectos y planes de organizaciones, con las cuales trabajamos para ayudarlas a prepararse para una evaluación en algún nivel del modelo CMMi. Un rasgo común en las más organizadas

---

39 | Boehm, Barry. Richard Turner. *Balancing Agility & Discipline: A Guide for the Perplexed*. Addison Wesley. 2003.

40 | Poppendieck, Mary. Tom Poppendieck. *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley Professional. 2006.

fue encontrar sus planes elaborados según algún template, por ejemplo, tomados de la IEEE (Institute of Electrical and Electronics Engineering). Una de las secciones de este template se titula “Estrategia”. A mi entender es una sección clave ya que si encontramos en ella un contenido consistente, fundamentado y verosímil, es una muestra de que el responsable del proyecto llevó a cabo un buen ejercicio de planificación. Sin embargo en la gran mayoría de estos planes se presentaba como estrategia algo de la forma que se transcribe a continuación:

*“(...) se construirán los módulos que surjan del análisis de los requerimientos. Se comenzará por aquellos que no posean dependencias y presten servicios básicos al resto de la aplicación. Se definirá la arquitectura en la fase de elaboración y se establecerá el ambiente de desarrollo de forma que los desarrolladores utilicen estas definiciones como guía en toda la fase de construcción.*

*Se utilizarán unittest para las pruebas unitarias y se probará la funcionalidad al término de cada iteración de dicha fase (...)"*

La pregunta que yo me hacía se la traslado a ustedes: ¿y la estrategia?

En las secciones que siguen nos ocuparemos de elaborar una respuesta a esta pregunta. Para esto nos guiarímos por el gráfico de la figura 5.6 que será nuestro mapa para la construcción de la estrategia para nuestros proyectos.



**Fig. 5.6** - Esquema de los componentes y sus relaciones que contribuyen a la elaboración de una estrategia para la gestión de los proyectos

## 5.2.1 ESTRATEGIA

### 5.2.1.1 Un caso demostrativo

Antes de avanzar con los temas específicos necesarios para responder la pregunta anterior proponemos analizar el siguiente caso que nos ayudará a entender este asunto de la estrategia.

En la figura 5.7 se muestran imágenes de una competencia de natación en aguas abiertas. Los competidores deben recorrer el itinerario que comienza en la imagen inferior del centro y continua con la imagen de arriba. Cuando los competidores ingresan al mar deben hacerlo por una zona de escasa profundidad donde además hay piedras. Desde allí deben nadar hasta la punta que se ve en el horizonte. Al dar vuelta a la misma se encuentran con una corriente marina cuya dirección apunta mar adentro con bastante intensidad. El recorrido continua en la imagen del centro de arriba, en la cual se muestra que deben rodear la pequeña península pasando por una zona de aguas frías, hacer pie en la playita donde termina la primera etapa y luego recorrer el mismo camino de vuelta (segunda etapa) hasta terminar la prueba.

Hemos decidido realizar unas preguntas a un par de competidores para conocer cómo encararán la competencia.

#### Cuestionario realizado a los participantes antes de la largada:

##### PARTICIPANTE N° 1.

**Pregunta:** ¿Qué estrategia seguirás en la competencia?

**Respuesta:** Nadaré estilo libre (croll en mi caso), de esa forma espero lograr un muy buen tiempo.

##### PARTICIPANTE N° 2.

**Pregunta:** ¿Qué estrategia seguirás en la competencia?

**Respuesta:** En la zona de la largada nadaré estilo pecho, para no lastimarme con las piedras, hasta que haya suficiente profundidad para nadar scroll. A partir de allí nadaré scroll. Al llegar a la zona de corrientes, me dejaré arrastrar y nadaré relajado hasta que la fuerza de la corriente sea leve y pueda retomar mi ritmo. En la zona de aguas frías nadaré a ritmo constante y a esfuerzo medio. Cuando pase esa zona retomaré mi ritmo hasta llegar a la playa. En el camino de vuelta (segunda etapa) usaré la misma estrategia si la misma resulta adecuada.

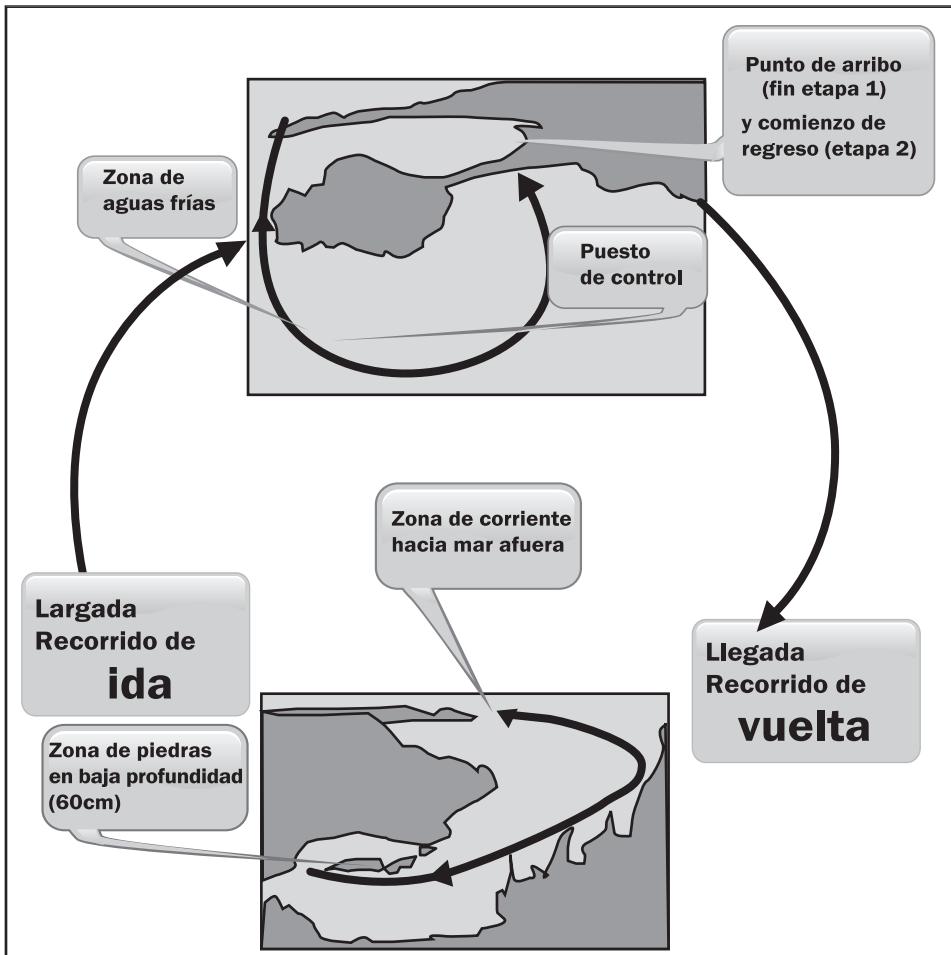


Fig. 5.7 - Metáfora para ejemplificar la elaboración de una estrategia

### Resultado de la competencia

El participante N° 2 arribó entre los diez primeros. Mientras que el participante N° 1 fue descalificado en el puesto de control después de haberse comprobado que tomó el camino de vuelta sin haber llegado a la playa. Hasta ese momento su tiempo de competencia excedía holgadamente al de los diez participantes mejor ubicados.

### Análisis de causas

Claramente existe una diferencia entre una metodología y una estrategia para llevar adelante la competencia. El participante N° 1 no tenía una estrategia, simplemente contaba con una metodología: nadar croll.

Al retomar el camino, sin cumplir con la consigna de pisar la playa, todo el esfuerzo del nado de vuelta fue desperdiciado. Siempre estuvo nadando en la primera etapa, no importa en qué dirección.

El participante N° 2 tenía una estrategia y, mala o buena, la cumplió y llegó a la meta. La misma la construyó teniendo en cuenta los objetivos y los obtáculos (riesgos) que se presentaban en el camino a la meta.

### 5.2.2 CONSTRUCCIÓN DE UNA ESTRATEGIA

Inspirados conceptualmente en la narración anterior y utilizando como guía la figura 5.6, intentaremos construir una estrategia. Para construirla será necesario repasar algunos temas fundamentales de la planificación de proyectos mencionados en la figura.

#### 5.2.2.1 Visión

¿Por qué construiremos este sistema / producto? ¿Para qué lo construiremos?

¿Alguna vez les hicieron esta pregunta a los clientes de un proyecto de desarrollo? Yo lo he hecho y una gran cantidad de veces me costó muchas horas de entrevistas y análisis, sin obtener una respuesta. Es más, muchos de los involucrados por parte del cliente no tenían una respuesta y vacilaban entre por qué es necesario y por qué la empresa lo dispuso. Es de fundamental responder claramente esta pregunta para elaborar una visión del proyecto. Esta debe provenir de los planes de negocio de la organización que encargará el desarrollo y debe ser conocida y compartida por todos los involucrados en el proyecto. Es tan importante que diría que no deberíamos involucrarnos en un proyecto si al cabo de algunos días de trabajo no estamos en condiciones de definir una visión. Es de esta importancia ya que de ella se derivan los objetivos para el proyecto; y como con cualquier emprendimiento en la vida, si no contamos con objetivos no sabremos hacia dónde conducirnos y cuáles deberían ser nuestras decisiones.

Algunas cuestiones que formarán parte de la visión de un proyecto son: la prestación de un servicio alineado a un plan de negocio, captar una porción del mercado en un segmento determinado, mejorar los procesos de negocio de una organización para lograr mayor eficiencia, contar con información para tomar decisiones, integrar áreas de negocio a partir de la unificación o integración de procesos, etc.

#### 5.2.2.2 Objetivos

A partir de una visión compartida trataremos de fijar los objetivos del proyecto. Estos pueden variar para un mismo desarrollo y esta variación se debe a la visión. No es lo mismo desarrollar el mejor producto que salir con el producto lo antes posible al mercado. No es lo mismo ordenar los procesos de una empresa replicando lo que se viene haciendo con varios sistemas pequeños y planillas de cálculo a construir un sistema repensando la forma de resolver los problemas actuales. Debemos fijar los objetivos del proyecto a partir de la visión. Distintos objetivos para un mismo proyecto determinarán diferentes prioridades y diferente esfuerzo en su construcción lo cual determinará una planificación diferente.

### 5.2.2.3 Prioridades

Como mencionamos en la sección Planificación de iteraciones, las prioridades de los requerimientos son una herramienta de utilidad a la hora de planificar las iteraciones del proyecto. Un mal síntoma es encontrar una especificación de requerimientos sin priorizar, donde todos son importantes. Esta situación evidencia una carencia de objetivos por falta de una visión claramente definida. Las prioridades las habíamos definido como esenciales, importantes, menos importantes y poco importantes. A cada requerimiento le asignaremos una de estas de acuerdo a los objetivos del proyecto. Como vemos cada uno de los aspectos que estamos tratando están fuertemente relacionados. Cuando se los observa como cosas independientes evidencia una falencia muy vista en muchos proyectos en los que en sus planes se presentan como ítems aislados. Esto es una muestra de que se enfatizan los planes frente al ejercicio de planificación. Si los objetivos fueron acordados estaremos en mejores condiciones de asignar prioridades. Esta asignación no es una tarea fácil por diferentes motivos. Recomiendo la lectura de Mike Cohn<sup>41</sup> para una presentación de las variantes en la asignación de prioridades. Es importante incluir en el análisis los siguientes criterios:

- **Costo de desarrollo y retorno de la inversión:** transferir funcionalidad que al ser usada comience a generar ingresos.
- **Valor agregado al negocio:** aportar funcionalidad que resuelva problemas.
- Deseo de contar con la funcionalidad por parte de usuarios/clientes: funcionalidad a incorporar según la visión de los diferentes involucrados por parte de la organización cliente.
- **Conocimiento del negocio adquirido:** por el grupo de desarrollo al implementar determinados requerimientos que faciliten el desarrollo de otros.
- **Riesgos de desarrollo:** balance entre agregar valor y reducir riesgos.

En el apéndice A se presenta un ejemplo de priorización que se condujo a partir de estos criterios.

### 5.2.2.4 Riesgos

Si definimos los riesgos de un proyecto como aquellas cuestiones que con cierta probabilidad se podrían convertir en obstáculos para lograr los objetivos, es importante comenzar a trabajar en forma temprana. En algunos libros de administración de proyectos (Steve McConnell, 1996<sup>42</sup> y Steve McConnell<sup>43</sup>, 1997 ) se presentan listados de riesgos que pueden ser utilizados como guía.

41| Cohn, Mike. *Agile Estimating & Planning*. Prentice hall. 2005.

42| McConnell, Steve. *Software Project Survival Guide*. Microsoft Press. 1996.

43| McConnell, Steve. *Desarrollo y gestión de Proyectos Informáticos*. Microsoft Press. 1997.

Sin embargo, no debe olvidarse que son una guía y que posiblemente no concuerde con los riesgos de su proyecto. El trabajo con riesgos es otra de las tareas en la cual es importante contar con cierta madurez para hacer visibles posibles fuentes de riesgo que por diferentes razones no son fáciles de explicitar.

Las fuentes de riesgos en general se deben a:

- **Requerimientos:** la complejidad y la falta de disponibilidad constituyen una fuente de riesgo común en muchos proyectos.
- **Tecnología:** la falta de experiencia en su uso o de madurez de alguna tecnología son fuente de riesgos.
- **Organización desarrolladora:** la falta de apoyo de la gerencia o de compromiso para con el proyecto según fue planificado al momento de competir para ganarlo son fuentes de riesgo frecuentes en los proyectos.
- **Organización cliente:** la falta de experiencia en el involucramiento en proyectos de desarrollo es una fuente frecuente de riesgo.
- **Grupo de desarrollo:** problemas generados por grupos de desarrollo recientemente formados para el proyecto, sin trabajos anteriores compartidos, son fuentes de riesgo comunes en los proyectos.

Los riesgos son importantes porque el líder conducirá la gestión del proyecto a partir de su administración, en el día a día, y además condicionarán la construcción de la estrategia a seguir para dicha gestión (Kurt Bittner, Ian Spence, 2006; Mike Cohn, 2005; McConnell, 1997).

### 5.2.2.5 Estimaciones

Las estimaciones de alcance y esfuerzo de un proyecto de desarrollo de software son un tema abierto acerca del cual los miembros de la comunidad informática continúan discutiendo y formando opiniones diversas. Recomiendo leer el libro de Steve McConnell<sup>44</sup> para contar con una visión abarcativa de los problemas relacionados. En mi experiencia, independientemente del método que se utilice para hacer estimaciones, se deben tener en cuenta los siguientes aspectos:

- Las estimaciones no son ni objetivos ni compromisos de negocio
- Las estimaciones se hacen con información reducida
- Las estimaciones como tal tienen límites de incertidumbre
- Las estimaciones deben ser presentadas como estimaciones
- Las estimaciones deben ser construidas por todos los involucrados
- Las estimaciones teóricas deben ser ajustadas a la realidad del proyecto
- Las estimaciones de organizaciones que conservan sus grupos de desarrollo formados y aprenden de sus errores son cada vez más precisas

---

44 | Cohn, Mike. *Agile Estimating & Planning*. Prentice hall. 2005.

- Las estimaciones cerradas de proyectos vendidos con alcance fijado, costo fijado y planificación (tiempo) fija conducen a proyectos fracasados.

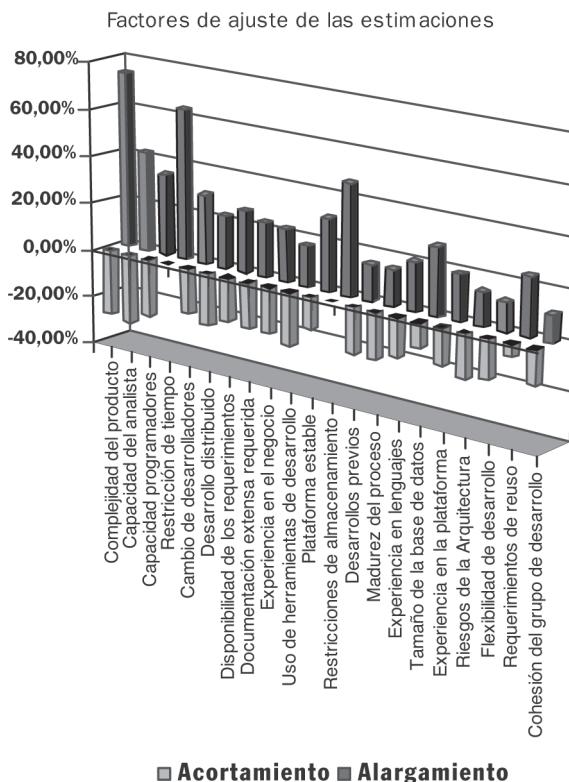
Las estimaciones realizadas para un proyecto trabajado en forma iterativa (alcance variable de acuerdo a prioridades, costo fijo y planificación fija) deben ser comunicadas y explicadas a la organización cliente como tal, de lo contrario se presentarán conflictos cuyo desenlace puede ser la cancelación del proyecto.

De todas estas recomendaciones nos extenderemos en un par de ellas.

### **Las estimaciones teóricas deben ser ajustadas a la realidad del proyecto**

Las estimaciones realizadas con la mayoría de los métodos dan como resultado una probabilidad de cumplir con los tiempos de desarrollo de un 50%. De allí que es costumbre agregar buffers temporales que amortiguen los posibles problemas y representen una probabilidad cercana al 90%. ¿Cuáles son las fuentes de estos posibles problemas?

En la figura 5.8 (adaptada de McConnell, 2006) se presenta la cuantificación de los posibles problemas generada a partir de un gran número de proyectos.

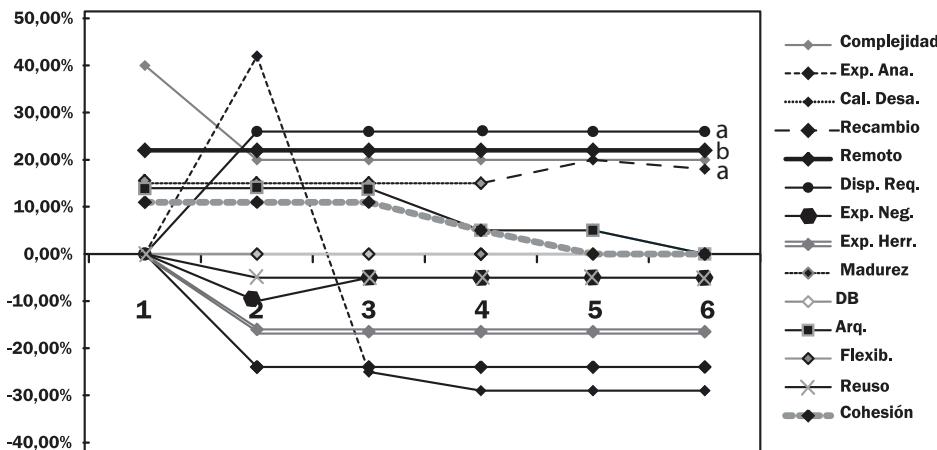


**Fig. 5.8** - Factores que influyen en las estimaciones de esfuerzo

Además de estos factores que influencian las estimaciones es importante mencionar que algunos lo hacen de manera opuesta a factores similares en proyectos de desarrollo en otras ingenierías. Se pone de manifiesto en el desarrollo de software que no funciona en proyectos de esta naturaleza la economía de escala. Es decir, cuánto más grande y mayor cantidad más complejo y por lo tanto más caro (McConnell, 2006, op. cit.).

Los factores que influyen de manera que las estimaciones al 50% serán holgadas (acortamiento de la planificación) deben conservarse a partir de acentuar la causa de dicho factor. En los factores que influyen de manera que las estimaciones al 50% serán insuficientes (alargamiento de la planificación) es donde debe focalizarse el trabajo para contrarestar dicha influencia. Contra esta podremos trabajar para minimizar su efecto. Contra la de otros, no. Sin embargo es importante que su impacto no crezca. En la figura 5.9 se muestra la evolución del impacto de los factores en un proyecto de desarrollo a lo largo de su ciclo de vida.

**Evolución de los factores de ajuste en el ciclo de vida de un proyecto**

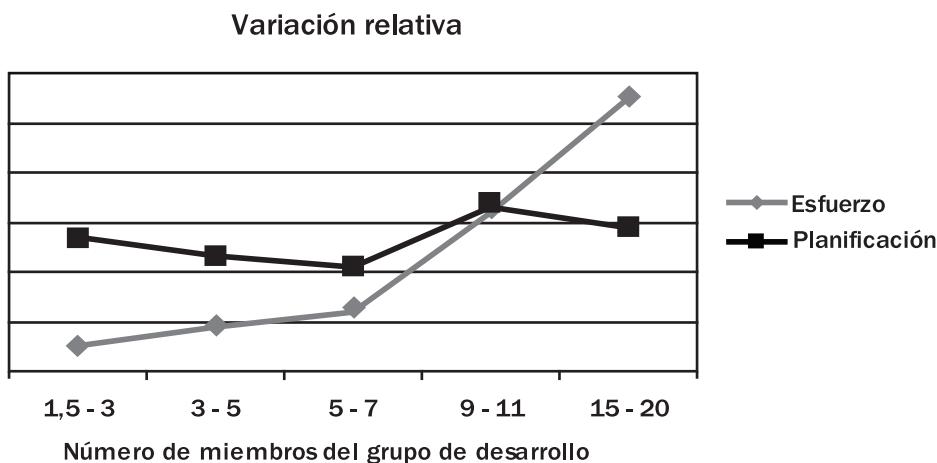


**Fig. 5.9** - Evolución de los factores a lo largo del ciclo de vida de un proyecto

Puede verse claramente que la mala influencia de algunos factores pudo reducirse (madurez del proceso, riesgos de la arquitectura, cohesión del grupo de desarrollo y experiencia del analista), que se capitalizó la buena influencia de unos (reuso, experiencia en el negocio, experiencia con herramientas y calidad de los desarrolladores) y que no se pudo mejorar el impacto negativo de otros (disponibilidad de los requerimientos, complejidad del negocio, rotación de los miembros del grupo y desarrollo remoto). Esta evolución es interesante de medir a efectos de afectar las reestimaciones realizadas a lo largo del proyecto.

### Las planificaciones no pueden acortarse sumando gente a los proyectos

Si bien es una problemática conocida, cuando se perciben retrasos en las planificaciones se sigue sugiriendo sumar desarrolladores a los proyectos. Una planificación no puede reducirse más allá de un 75% a partir del agregado de miembros al grupo de desarrollo. Existe una llamada “zona imposible” de evolución de los proyectos de desarrollo cuyo límite está dado por el número mencionado (McConnel, 2006, op. cit.). Este hecho se muestra en la figura 5.10 en la que se observa que por efecto de la mayor dificultad en gestionar el trabajo de un grupo grande, la planificación en lugar de reducirse aumenta cuando se sobrepasa en 5 a 7 la cantidad de miembros de un grupo de desarrollo.



**Fig. 5.10** - Variación del esfuerzo y la planificación con el número de miembros de un grupo de desarrollo

#### 5.2.2.6 Estrategia

Como se puede observar en la figura 5.6, que se encuentra al comienzo de la sección, hemos repasado todos los aspectos necesarios para la construcción de una estrategia.

Esta estrategia apunta al logro de los objetivos implementando los requerimientos priorizados y sorteando los obstáculos (riesgos) que se presenten. Para esto es necesario elaborar un cronograma de tareas en el que el esfuerzo estimado estará dispuesto según esta estrategia definida.

Como se desprende de este análisis, dista de ser una metodología. La metodología de desarrollo la conservamos a lo largo del ciclo de vida del proyecto al igual que la estrategia. Sin embargo haremos ajustes a las tácticas que utilicemos para la implementación de la estrategia en cada ejercicio de planificación. Estos ajustes dependerán de la evolución del proyecto en relación con los riesgos.

## Guía para el establecimiento y desarrollo de la estrategia a seguir

1. Entender los objetivos del proyecto a partir de la visión compartida
  - a. Perspectiva financiera
  - b. Perspectiva del cliente y los usuarios
  - c. Posibilidades de cambio y crecimiento para acompañar al negocio
  - d. Valores a sumar al negocio
2. Conocer los riesgos
  - a. Detectarlos
  - b. Analizar la gestión del proyecto en base al control de los mismos
3. Entender la solución y el alcance
  - a. Relevar requerimientos
  - b. Listar casos de uso
  - c. Priorizar casos de uso
4. Estimar el esfuerzo ajustado al proyecto
  - a. Casos de uso
  - b. Interfaces con sistemas externos
  - c. Reportes
5. Definir los procesos a seguir
  - a. Definir la metodología
  - b. Asignar roles a los involucrados
6. Crear un cronograma de tareas según la estrategia elaborada
  - a. Lista de tareas según el orden determinado por las prioridades
  - b. Asignación de tareas
  - c. Hitos
  - d. Relaciones entre tareas

## Ejemplo de estrategia

A continuación se presenta parte de la estrategia definida para un proyecto del cual proviene el siguiente fragmento de plan de proyecto. Corresponde a un desarrollo remoto por parte de una software factory recién fundada para un cliente sin experiencia en la compra de desarrollo de sistemas, para un negocio de venta de servicios de telefonía sin el soporte de un sistema integrado y en un contexto de negocio muy desordenado.

- Según se deriva de la visión, este proyecto busca sistematizar el funcionamiento de un negocio que actualmente es gestionado sin la ayuda de ningún sistema; es decir, el cliente no tiene el hábito de gestionar el negocio con un sistema. Se busca ordenar los procesos de negocio y planificar su evolución con información certeza provista por el sistema en desarrollo. Por esta razón existe el riesgo, en la implementación de algunos módulos críticos, que una vez acordados e implementados haya que realizar cambios en los productos construidos por cambios en la definición por parte del cliente. Por esta razón se trabajará primero en estos módulos (ejemplo: definición de productos a vender).
- Existe el riesgo de que la visión original del proyecto cambie de rumbo desde la perspectiva del cliente, sobre todo cuando puedan utilizar los primeros entregables. Por esta razón se trabajará en contacto permanente con ellos reforzando la visión de que el propósito del sistema es ordenar y racionalizar el negocio. Para esto se instalarán los binarios en una extranet cuando sean liberados y se le dará acceso al cliente y a sus usuarios para que se familiaricen con su uso. Se utilizará este mismo mecanismo para compartir la documentación del proyecto.
- Existe el riesgo de que los involucrados por parte de la gerencia del cliente, así como los de la gerencia de la empresa desarrolladora, no respeten los compromisos preacordados. Por esta razón se trabajará en mantener el involucramiento de todos con el proyecto a partir de un seguimiento que genere gran visibilidad.
- Existe un módulo de tarificación cuyos algoritmos no están claramente definidos al momento de construir este plan. Se trabajará en aclarar todos los aspectos de estos algoritmos. Se construirá este módulo ni bien se cuente con dichas definiciones.
- Parte de la información necesaria para facturar a los clientes proviene de sistemas externos. No se conoce la definición de los formatos en que dicha información es provista por algunos de estos sistemas. Se construirá un componente genérico que sea reutilizable para el procesamiento de los diversos tipos de archivos de los distintos clientes.
- El flujo de trabajo del negocio no es complejo, sin embargo se seleccionará una herramienta de administración de workflow y se definirá la forma de integrarla a la arquitectura. Si los detalles del negocio lo requieren, se decidirá su incorporación.
- La aplicación de los impuestos a los productos vendidos se acordó que se realizará utilizando un sistema externo que el cliente debe definir. Se diseñará el módulo de tarificación orientado a la integración de los servicios de ese sistema externo.
- Se trabajará en minimizar el impacto de incorporación del sistema a la empresa cliente haciendo que los usuarios se familiaricen con este durante el desarrollo a partir de la prueba de los distintos entregables. Para esto se asignarán horas al acompañamiento de los usuarios en estas tareas.

También se integrará al personal técnico que deberá hacerse cargo de la administración del sistema.

- Se fomentará una forma de trabajo de fuerte integración y cooperación del grupo de desarrollo con el objetivo de conformarlo y contrarestar la falta de experiencia y madurez de algunos de sus miembros, así como la falta de trabajos previos compartidos.

## 5.3 | SEGUIMIENTO DEL DESARROLLO DE PROYECTOS

---

### 5.3.1 ROLES

Uno de los aspectos claves en la planificación de un proyecto es la definición de la forma en que se llevará adelante el seguimiento del avance del proyecto. Estas definiciones formarán parte de nuestro plan, el cual será publicado a todos los involucrados. Cada uno de los involucrados debe conocer de antemano cuál es el rol que desempeña, de esta forma tendrá claro cuáles serán sus responsabilidades. Para evitar que los proyectos se desvíen es fundamental un seguimiento cercano, es decir, mantener comunicados a todos los involucrados o lo que es lo mismo propiciar una gran visibilidad.

En estas cuestiones el líder del proyecto tiene una responsabilidad importantísima porque es el coordinador de las tareas del proyecto. Si tuviera que resumir sus responsabilidades diría que son:

1. Generar en tiempo y forma los eventos planificados
2. Facilitar los acuerdos necesarios
3. Tomar decisiones

#### 5.3.1.1 Qué cosas debe hacer un líder de proyectos para cubrir sus responsabilidades

- Mantener cohesionado al grupo de desarrollo
- Atender las necesidades personales de los miembros
- Facilitar las tareas de la planificación generando las condiciones para que puedan desarrollarse de la mejor manera
- Tomar decisiones apoyado en los referentes técnicos del grupo
- Mantenerse informado y actualizado en la tecnología que se utiliza en el proyecto
- Tomar decisiones cuando no se logran acuerdos
- Mantener motivado y comprometido a los usuarios y clientes acordando en todo momento
- Escalar los problemas que no puede resolver

### 5.3.1.2 Qué cosas no debe hacer un líder de proyectos

- No debe competir en cuestiones tecnológicas con los referentes técnicos del proyecto.
- No debe ignorar la tecnología que se utiliza en el proyecto
- No debe desatender las cuestiones del proyecto por otras ligadas a carreras corporativas.
- No debe olvidar que los miembros del grupo de desarrollo son lo más importante mientras dure el proyecto.
- No debe olvidar que el objetivo primero del proyecto es entregar un producto funcionando.
- No debe temer detener el proyecto cuando algunos desacuerdos indican que la factibilidad de este se ve cuestionada.

Si debo realizar una autocrítica a alguna de mis participaciones en proyectos como líder es no haber detenido un proyecto a tiempo. El hecho de parar un proyecto a tiempo evita frustraciones mayores para todos los involucrados y pérdidas innecesarias tanto para los clientes como para la propia organización. Los factores que en general influyen en esta decisión son el temor a perder el trabajo, a ser visto como incompetente para desarrollar esta actividad y cumplir con los objetivos, y tener que comunicar una decisión poco grata al grupo de desarrollo, algo que siempre genera frustración. Se requiere una gran madurez para tomar estas decisiones, tanto de parte de quien las toma como de las organizaciones involucradas en el proyecto.

La peor decisión cuando se generan problemas en la evolución de un proyecto es buscar realizar otras tareas que eviten esos obstáculos y seguir adelante. Este tipo de decisiones dilatan situaciones que tarde o temprano sucederán. Mary y Tom Poppendieck (op. cit.) promueven esta postura en su metodología derivada de la forma Toyota de desarrollo de productos aplicada al desarrollo de software y que ellos llaman Clean Software Development.

### 5.3.2 ACTIVIDADES

Las actividades críticas a la hora de realizar el seguimiento de un proyecto de desarrollo son:

1. Acordar las prioridades de los requerimientos a implementar en cada iteración con todos los involucrados.
2. Analizar los riesgos y problemas, y acordar acciones orientadas a solucionarlos con todos los involucrados.
3. Tomar métricas que muestren la estabilidad del proyecto y publicarlas a todos los involucrados.
4. Analizar los informes generados por el servidor de integración continua y conducir las pruebas y revisiones de los productos generados.

5. Analizar y evitar errores en cada iteración aprendiendo de la anterior.
6. Con cada iteración mostrar tendencias de la evolución del proyecto a todos los involucrados.

### 5.3.3 PUNTOS DE OBSERVACIÓN

Es interesante fijar puntos de observación en el cronograma del proyecto orientados a informar el estado y avance y tomar decisiones al respecto. Estos puntos de observación son informados en algún reporte y analizados con todos los involucrados. Suelen coincidir con el comienzo y terminación de fases e iteraciones y sirven para replanificar y evaluar la evolución del proyecto. Es importante no perder de vista lo siguiente:

- Las Fases **no** terminan si **no** se alcanzan los objetivos asociados a los hitos de terminación
- Cuando se trabaja con iteraciones de duración fija, estas terminan en la fecha prefijada
- Una Fase exitosa (objetivos alcanzados) puede significar la cancelación del proyecto
- Una iteración exitosa siempre implica la continuación del proyecto o su terminación **ok**
- Una iteración fallida puede implicar la cancelación o extensión del proyecto

Es costumbre relacionar estos puntos de observación con los siguientes momentos del proyecto:

#### **Fin de la concepción**

Analizar la factibilidad del proyecto y adaptar la planificación. Es un ejemplo de una fase exitosa que puede mostrar la conveniencia de la cancelación del proyecto por la imposibilidad de llevarlo adelante en las condiciones establecidas al momento del inicio.

#### **Fin de la definición de la arquitectura**

Analizar la factibilidad de la tecnología propuesta una vez probada la arquitectura, la que soportará la funcionalidad requerida. Se evalua la minimización de los riesgos impuestos por la tecnología.

#### **Fin de cada iteración durante la construcción**

Evaluar lo construido y lo que resta por implementar. Se evalúan los requerimientos priorizados y cómo estos se vienen implementando en cada iteración. Cómo se cumple con los objetivos del proyecto.

#### **Fin del proyecto**

Analizar las pruebas de aceptación según la planificación del proyecto. Se verifican los requerimientos y cómo el sistema desarrollado cubre e implementa los mismos.

### 5.3.4 FOTOS VERSUS PELÍCULA

El error más común en el seguimiento de un proyecto es realizar reuniones de este tipo y generar en los involucrados la sensación de que con esto se cumple con su gestión. Ya criticamos esta forma de gestión en la sección Fases, iteraciones y objetivos, en la que se confunde logro de objetivos con la realización de actividades.

Hace un par de años me tocó participar como mentor en un proceso de mejoras en una organización grande, en la cual convivían áreas de infraestructura informática, seguridad informática y desarrollo de proyectos informáticos. Las gerencias respectivas se reunían semanalmente para realizar el seguimiento. Estas reuniones duraban tres o cuatro horas y en ellas se trataban una decena de temas. Un anotador tomaba nota de lo tratado y se elaboraba una minuta que luego se repartía a los participantes. En la reunión de la semana siguiente se trataba otra decena de temas diferentes y se procedía de igual modo; y así en la reunión próxima. Pero esta vez se volvían a tratar algunos que ya se habían tratado en la preanterior, como si fuese la primera vez. En esta anécdota están presentes los dos errores a los que quiero referirme.

#### 5.3.4.1 Tratamiento de una decena de temas

Cuando alguien no quiere resolver los problemas de una organización los trata de resolver todos juntos y de una vez. Esta manifestación la pude observar en muchas empresas y proyectos. Es necesario priorizar los temas, dedicarles el tiempo necesario, planificar su solución asignando responsables para ello y definir fechas tentativas de revisión. La mayor parte de las veces las empresas no cuentan con recursos para resolverlos en conjunto, por lo tanto la mejor estrategia es tratar de resolverlos según su prioridad y de a unos pocos.

#### 5.3.4.2 No seguimiento de los temas tratados

Cada vez que se organiza una reunión para seguir un proyecto es imprescindible la revisión de lo tratado en la reunión anterior antes de analizar las novedades. Si realizamos un informe de cada reunión, estos deben estar “atados”. De esta forma lograremos contar con una película y no con fotos de algunos momentos del proyecto.

En los apéndices se muestra un template de un informe que en una de sus secciones incluye el vínculo con los anteriores correspondientes a reuniones precedentes. Esta forma de informar reuniones de seguimiento ayuda a tomar métricas que muestran cómo se realiza dicho seguimiento. Al encontrar en más de un informe un mismo problema detectamos su falta de seguimiento, pues si hubiese sido resuelto hubiese desaparecido de los informes.

### 5.3.5 ESCALAMIENTO

Entiendo por escalamiento los mecanismos orientados a comunicar problemas que determinado rol no puede resolver porque estos se producen en situaciones que exceden a sus responsabilidades. Esto es: fue establecido en la organización que determinados problemas deben ser tratados por un rol con mayor responsabilidad.

Es tan importante que los líderes de proyecto escalen los problemas de los proyectos en tiempo y forma como que los gerentes de desarrollo, por ejemplo, tomen las acciones necesarias para resolverlos. Para que esto ocurra deben darse dos condiciones:

### **Roles predefinidos y asumidos por cada uno de los miembros de la organización**

Cuando los gerentes no asumen su rol con responsabilidad y coherencia, los líderes no cuentan con un referente de confianza en quién descargar decisiones complicadas que exceden su rol. He presenciado este tipo de problemática en empresas de organización horizontal donde todos los miembros tienen acceso a los más diversos aspectos. A veces, estas organizaciones confunden a sus miembros, generan una sensación de falta de liderazgo y cuando se necesita que alguien asuma la responsabilidad de tomar decisiones difíciles no se cuenta con tal persona, con las consecuencias que esto implica.

### **Gerentes muy bien informados acerca de los proyectos**

Esta condición es responsabilidad tanto de los líderes como de los gerentes. Líderes que no generan una alta visibilidad en sus proyectos contarán con gerentes desinformados. Gerentes con poco compromiso con los proyectos, en general, tomarán malas decisiones por falta de información cuando deban implementar acciones orientadas a la resolución de problemas.

#### **5.3.6 ACCIONES**

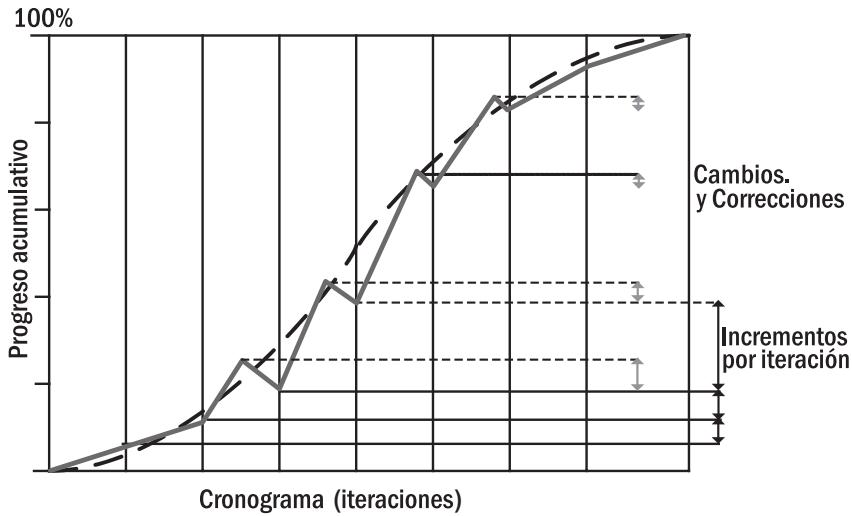
Solo deseo enfatizar que en todo momento y frente a cualquier evento y cualquiera sean las acciones que se deban tomar es esencial fundamentarlas. La mayor cantidad de conflictos y desacuerdos que he presenciado se debieron a acciones no entendidas por algunos de los involucrados. Para que cada decisión que tomemos sea entendida es importante contar con el canal de comunicación apropiado para publicarla.

**Conclusión:** no tome decisiones o implemente acciones sin antes comunicarlas y fundamentarlas debidamente.

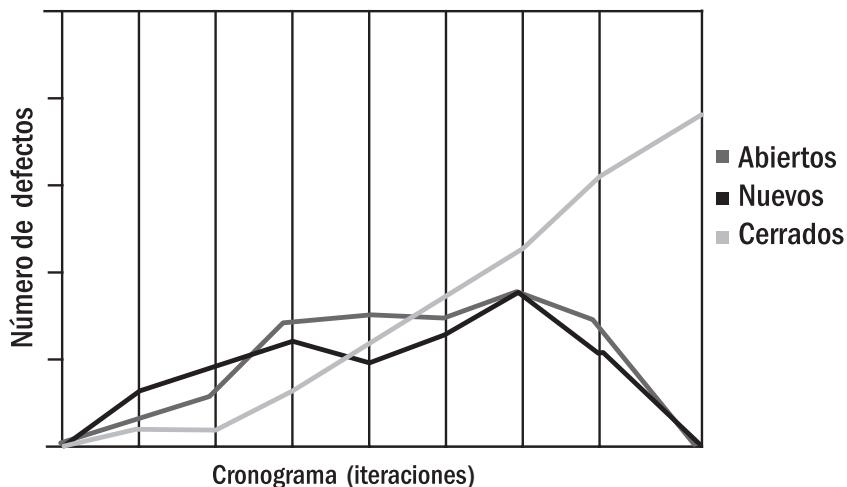
#### **5.3.7 MÉTRICAS**

Por alguna razón, que no me queda del todo clara, cada vez que tratamos el tema “métricas” las personas proponen medir cantidad de errores (bugs) como si ese fuera el único indicador válido que nos muestra información de utilidad. Podríamos medir una gran cantidad de parámetros asociados a diferentes aspectos de un proyecto. Sin embargo hay dos grupos de métricas que deseo analizar.

Unas nos permitirán contar con un indicador de la estabilidad del proyecto cuando trabajamos en forma iterativa. Estas medidas ya las presentamos en la sección Medidas de estabilidad y sus gráficas se reproducen en las figuras 5.11 y 5.12 para un mejor análisis. Su importancia radica en que al variar las correcciones y los requerimientos implementados de la forma que se muestra, creciendo y luego decreciendo, es un indicador de la evolución estable del proyecto. Por otro lado la relación entre errores detectados y corregidos, así como su cantidad en los diferentes momentos del proyecto, indica la misma estabilidad mencionada.



**Fig. 5.11** - Métricas que muestran la evolución de los incrementos del sistema en desarrollo y los errores y cambios pedidos a lo largo de las iteraciones.

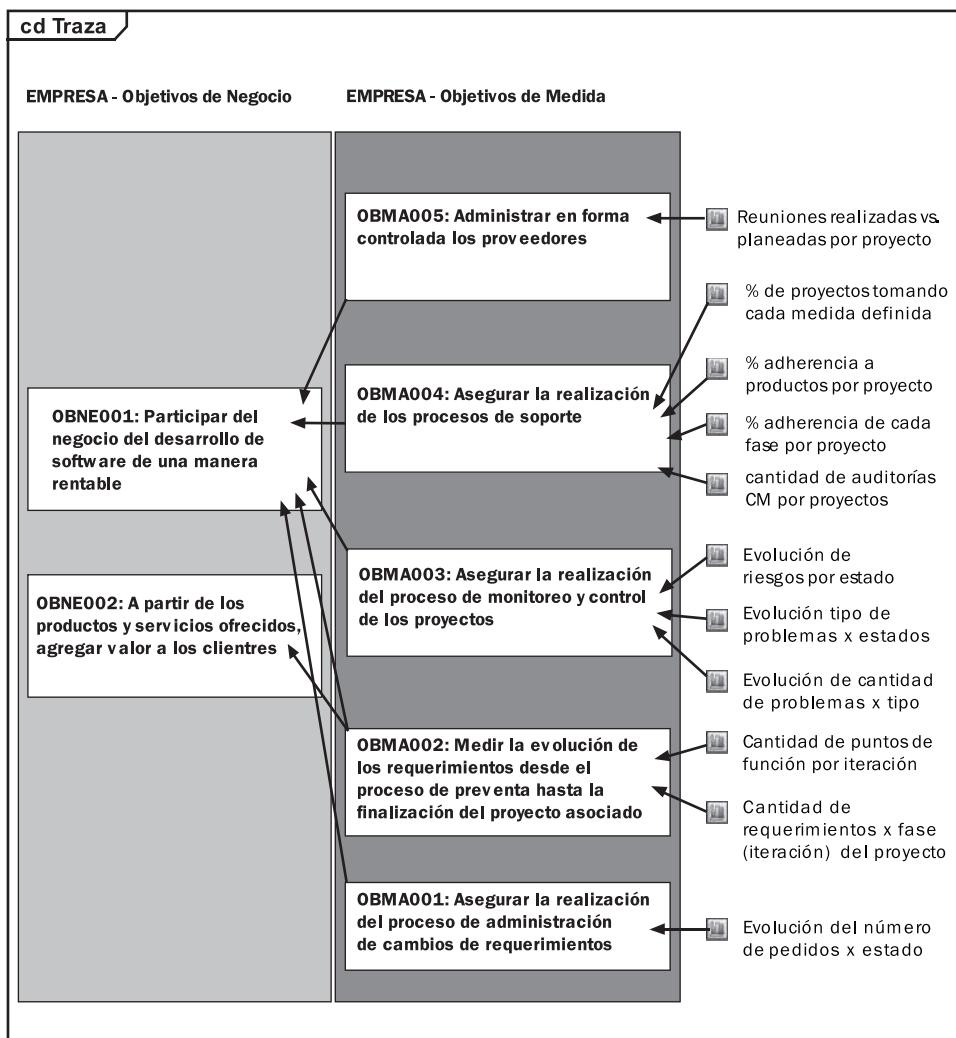


**Fig. 5.12** - Evolución de los defectos de un proyecto de desarrollo estable

Un segundo grupo de métricas funda su estrategia en su alineación con el negocio. Esta es conocida como Goal-Question-Metric (GQM) y en GQM<sup>45</sup> se la presenta muy bien. Se basa en que las preguntas realizadas al negocio nos deben conducir a los objetivos de medida desde los objetivos de negocio y así a los indicadores a construir.

45 | Basili, Victor. Gianluigi Caldiera. H. Dieter Rombach. *The Goal Question Metric Approach*. 1994.

En función de estos indicadores seleccionaremos las métricas a tomar. En la figura 5.13 se muestra un ejemplo de estas relaciones derivadas de la manera descripta.



**Fig. 5.13** - Objetivos de negocio, de medidas e indicadores relacionados

Además de basar los objetivos de medida en el negocio, las métricas se elaboran desde la perspectiva de una mejora de procesos. Por esta razón estas irán evolucionando y cambiando con el tiempo según se progrese con las mejoras mencionadas. Se buscará mensurar cada aspecto que mencionaremos a continuación a medida que avance el proyecto.

1. **Adherencia:** inicialmente se organizan las medidas con el objetivo de medir la incorporación y el cumplimiento de los procesos y procedimientos del proyecto de mejora de procesos.
2. **Estabilidad:** una vez establecidos los procesos que se están definiendo se buscará medir la estabilidad de los mismos a efectos de tomar acciones para que se comporten de forma estable.
3. **Capacidad:** cuando estos procesos se comporten en forma estable se buscará evaluar el cubrimiento de los objetivos planteados por los mismos. Para esto se obtendrán las medidas que se definan a tal fin.
4. **Comportamiento:** se tomarán medidas orientadas a evaluar la eficiencia y eficacia de los procesos anteriores.
5. **Mejora:** llegado a este estadio, se continuará con el proceso de medidas y análisis a efectos de lograr una mejora continua de los procesos definidos para la organización.

## 5.4 | CONCLUSIÓN

---

A efectos de lograr una buena gestión de los proyectos de desarrollo de software de forma que generen productos de calidad se debe:

- Elaborar una estrategia en base a los objetivos, prioridades y riesgos
- Volcar en una planificación los esfuerzos estimados según la estrategia
- Realizar el seguimiento resolviendo los problemas que se generen y escalando los que excedan nuestras responsabilidades
- Detenerse ante los problemas y no continuar hasta resolverlos
- Explicar cada decisión y comunicarla a todos los involucrados
- Tomar métricas que nos provean indicadores de la evolución de los proyectos de alineados a los objetivos del negocio y a los procesos definidos para la organización

---

<b>Capítulo 6: Trabajo con la Implementación - Diseño Codificación y Pruebas</b>	
<b>6.1   Diseño, codificación y pruebas</b>	139
6.1.1 Problemas .....	140
6.1.1.2 Proceso de diseño .....	140
6.1.1.3 Coordinación de la construcción .....	140
6.1.1.4 Pruebas .....	140
<b>6.2   Pruebas de software</b>	141
6.2.1 Trabajo con el repositorio .....	145
6.2.2 Test sistemáticos y automáticos.....	146
6.2.3 Cómo adoptar la nueva forma de trabajo .....	146
6.2.3.1 Obstáculos para automatizar las pruebas .....	147
6.2.3.2 Qué debería automatizarse .....	148
6.2.3.3 Qué no debería automatizarse.....	148
6.2.3.4 Estrategia para comenzar la automatización .....	148
<b>6.3   Integración continua</b>	149
6.3.1 Forma de trabajo .....	149
6.3.1.1 Pasos .....	149
6.3.1.2 Principios.....	150
6.3.2 Infraestructura .....	151
6.3.3 Resultados .....	153
<b>6.4   Revisiones de diseño y código.</b>	154
6.4.1 Revisiones .....	154
6.4.1.1 Objetivos .....	155
6.4.1.2 Beneficios .....	155
6.4.1.3 Métricas guía .....	156
6.4.1.4 Indicadores .....	157
6.4.1.5 Polimétrica de complejidad .....	158
<b>6.5   Conclusiones</b>	158
<b>6.6   Herramientas</b>	158

# 6

## TRABAJO CON LA IMPLEMENTACIÓN - DISEÑO CODIFICACIÓN Y PRUEBAS

### 6.1 | DISEÑO, CODIFICACIÓN Y PRUEBAS

---

Durante muchos años uno de los problemas más difíciles de resolver en el desarrollo de software fue el de la logística de un proyecto. Entendemos por logística “entregar en el tiempo acordado el sistema funcionando en el lugar preestablecido”. En las iteraciones en las que se escribe código se pone de manifiesto las bondades y falencias de la metodología de desarrollo, las virtudes y defectos de los miembros del grupo de desarrollo y las capacidades de los líderes para gestionar el proyecto tomando acciones correctivas y acordando con los clientes. Durante estas iteraciones se acentúa el caos y, a pesar de contar con mayor información que en las etapas tempranas del proyecto, se generan momentos de incertidumbre. En estas iteraciones los desarrolladores elaboran el diseño detallado vinculado a cada caso de uso luego de refinar su especificación y análisis para luego escribir el código y realizar las pruebas. Son momentos de intensa actividad en el seno del proyecto. Es importante desarrollar criterios acerca de cómo resolver los problemas que se plantean. Estos están asociados a la respuesta a la siguiente pregunta que nos podemos hacer en cualquier momento de estas iteraciones: ¿Qué hemos desarrollado y qué falta desarrollar?

Para darnos una respuesta certera no son válidas alternativas tales como:

“Pablo está terminando el caso de uso que le asignaron, o sea que cuando termine, Juana podrá integrarlo a su desarrollo”. “María ya desarrolló alrededor del 50% de sus casos de uso y cuando termine podremos hacer la prueba del módulo en construcción”.

Estas son especulaciones, necesitamos una respuesta.

En las secciones que siguen trataremos de dar una respuesta certera a la pregunta anterior y por lo tanto un criterio de trabajo y algunas buenas prácticas que nos ayuden en

esta fase del ciclo de vida de nuestros proyectos a mejorar la calidad de nuestro trabajo y por consiguiente de los productos que generemos.

### 6.1.1| PROBLEMAS

#### 6.1.1.2 Proceso de diseño

Asumimos que la arquitectura del sistema en desarrollo fue revisada y probada y que estamos frente al diseño detallado de cada caso de uso. Además de capacitar a nuestros desarrolladores en temas de diseño tales como falencias comunes en el diseño, criterios de buen diseño y patrones de diseño, cuáles son otras acciones que podríamos tomar para que nuestros diseños sean de mejor calidad. Una buena práctica que ha mostrado dar muy buenos resultados es la revisión de los diseños en sesiones con la participación de pares y otros miembros de la organización con probada experiencia y conocimientos. Yo he participado de estas sesiones y a decir verdad en muchas de ellas el producto a revisar era tan importante como muchos otros que no se revisaron. Entonces, qué revisamos, cómo seleccionamos los productos a revisar. Necesitamos un criterio de selección que nos permita, por ejemplo, revisar las partes críticas, pero también las partes con potenciales falencias. Pero ¿cómo saber cuáles son?

#### 6.1.1.3 Coordinación de la construcción

Desde hace tiempo el software de los sistemas está compuesto de un gran volumen de código. Todos los miembros de los grupos de desarrollo aportan a este código. Todos los días en el ciclo de vida del proyecto cada desarrollador usa y modifica al código escrito por otro desarrollador. Esta es la fuente de muchos errores de difícil solución. Tradicionalmente una vez desarrollada una porción de código los desarrolladores la prueban con test unitarios y cuando el código está integrado se prueba esta integración para luego comprobar su funcionalidad y entonces relevar los errores y corregirlos. Este miniciclo dentro de cada iteración implica el desarrollo de una ceremonia muy cara en recursos. Por esta razón es necesaria una forma de trabajo que facilite y acelere este proceso. Además si consideramos importante la realización de revisiones de código, se nos plantea un inconveniente parecido al que describimos con el diseño. Necesitamos una guía para seleccionar los productos a revisar, de lo contrario revisaremos código no importante o quizás sin falencias y dejaremos sin revisar otros con potenciales problemas.

#### 6.1.1.4 Pruebas

Existe un error conceptual en el ambiente de desarrollo de software acerca de cómo se puede garantizar la calidad de los productos que las empresas desarrolladoras generan. Muchos proyectos son sometidos a pruebas por parte de empresas especialistas en testing y como consecuencia de esto, los responsables de estos proyectos piensan que de esta forma están cubiertas las actividades orientadas a la mejora de la calidad. Sin embargo lo que sucede en la mayoría de estos casos es que les cobran por decirles cuántos errores su software todavía tiene. Realizar este tipo de pruebas no contribuye a la mejora de la calidad sino a la detección de errores y en el mejor de los casos direcciona posibles soluciones.

Para que un software mejore su calidad además de someterlo a una acción destructiva con sadismo, como dice Myers<sup>46</sup>, hace falta una acción constructiva durante su desarrollo a efectos de evitar algunos de los errores a partir de un mejor diseño y codificación. En la sección Pruebas de Software describiremos una forma de trabajo con estos objetivos.

## 6.2 | PRUEBAS DE SOFTWARE

En la sección anterior nos referimos a la forma de trabajo tradicional con referencia a las pruebas que presentamos en la figura 6.1.

### Trabajo con tests manuales

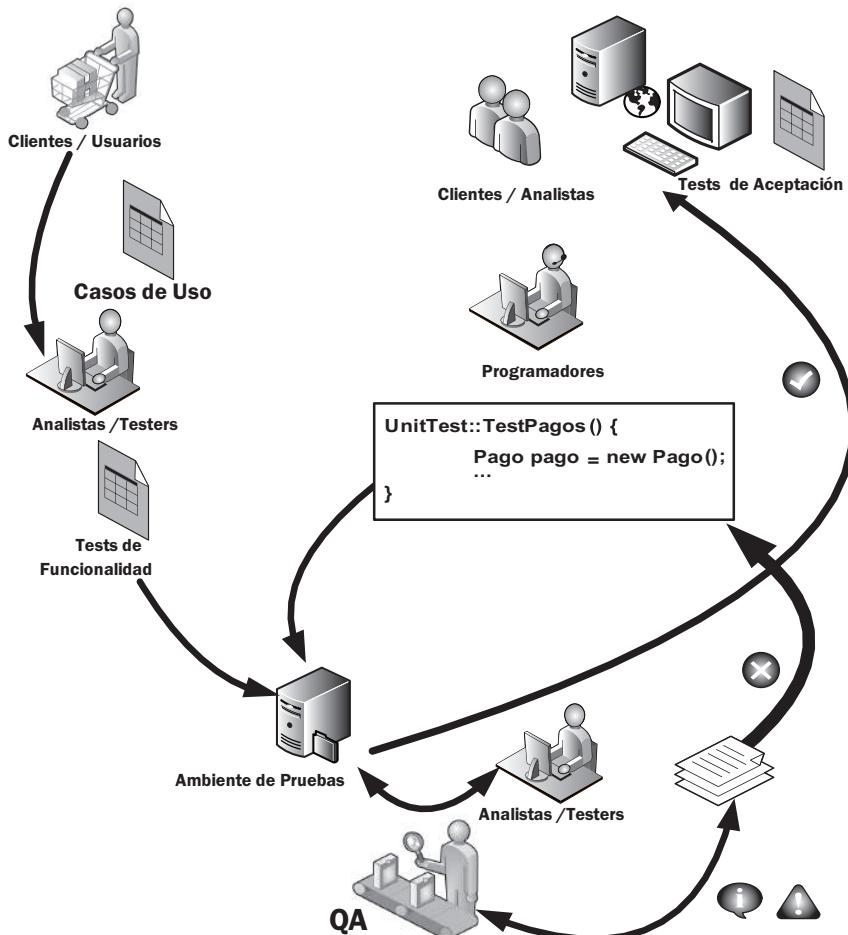


Fig. 6.1 - Forma de trabajo tradicional en relación con las pruebas

Es conocido y hasta casi trillado mencionar la aseveración de que cuanto más temprano se introducen errores en el ciclo de vida del desarrollo de un software, más caros serán, debido al esfuerzo necesario para reparar un sistema mal concebido o mal diseñado cuando ya está construido. Por esta razón es deseable detectarlos cuanto antes para poder corregirlos<sup>47 48</sup>. Pero, además, sería sumamente valioso elaborar diseños y escribir código de manera de asegurarnos que las causas de los errores anteriores fueron eliminadas, por lo cual los errores nunca estuvieron presentes. Para lograr esto es necesario incorporar las pruebas en etapas anteriores, introducirlas mientras el sistema se construye, influir desde la perspectiva de las pruebas en el diseño. Lo que proponemos y promovemos es el cambio, sufrido hace ya varios años, de la forma de trabajo tradicional a la forma de trabajo sistematizada y automatizada, cuya evolución se muestra en el esquema siguiente.

En el esquema de la figura 6.2 se muestra la forma de trabajo tradicional y en el de la figura 6.3 que en la forma automatizada algunas actividades fueron adelantadas ya que se incorporaron pruebas durante el desarrollo. Estas actividades se muestran sin color cuando son llevadas adelante en la forma tradicional.

Por lo general en la forma tradicional de trabajo, más allá de la ayuda de alguna herramienta puntual, las distintas actividades se desarrollan manualmente. Esta forma de trabajo determina que las pruebas muchas veces no se repitan por el esfuerzo que ello implica. Además, durante la ejecución manual siempre existe la posibilidad de cometer errores.

Como conclusión podemos mencionar entonces que las razones para cambiar la forma de trabajo con las pruebas son:

- El ciclo de prueba manual es muy largo
- El proceso de prueba manual es propenso a errores
- Liberar a la gente para realizar tareas creativas
- Generar un ambiente de confianza soportado por los test
- Obtener realimentación de forma temprana y con alta frecuencia
- Generar conocimiento del sistema en desarrollo a partir de los test
- Generar documentación consistente del código
- Generar una mejor utilización de los recursos a partir de menores costos
- Desarrollar un criterio de diseño basado en las pruebas

47 | Rakitin, Steven R. Software Verification and Validation for Practitioners and Managers. Second Edition, Artech House. 2001.

48 | McConnell, Steve. *Code Complete*. Redmond, Wa. Microsoft Press. 2.<sup>nd</sup> Edition. 2004.

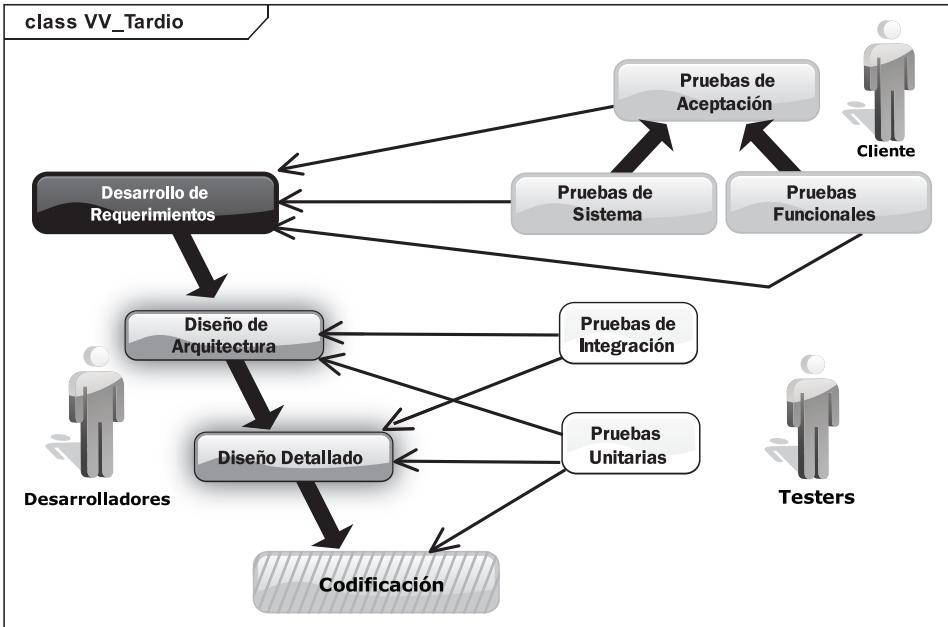


Fig. 6.2 - Forma de trabajo tradicional

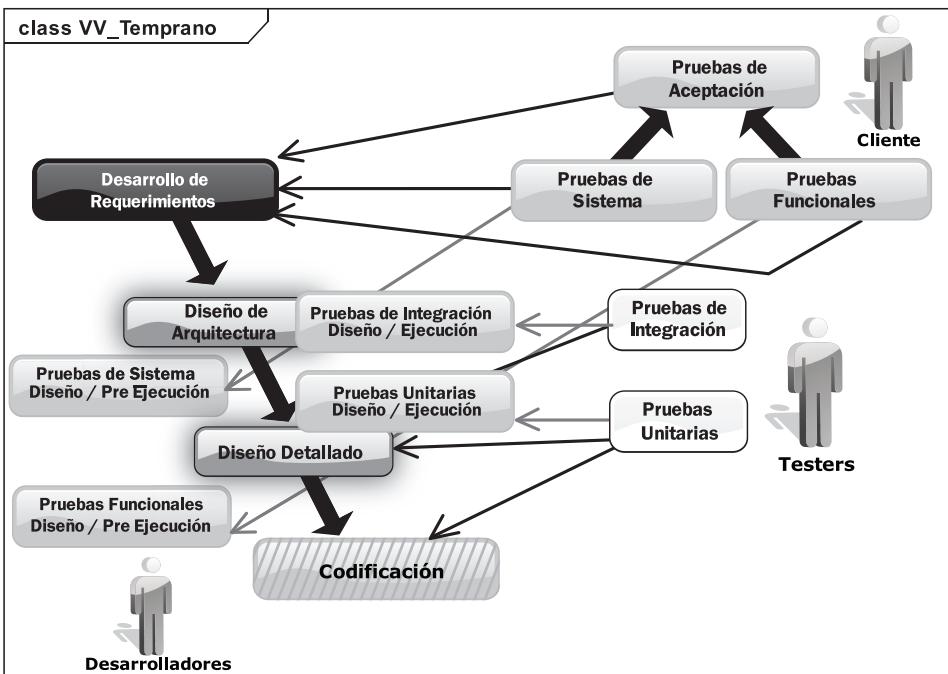


Fig. 6.3 - Forma de trabajo propuesta con las pruebas automatizadas y formando parte del criterio de diseño

Como consecuencia del análisis anterior extraeremos un par de objetivos que determinarán las actividades que presentaremos en las próximas secciones.

*Por todo lo dicho necesitamos:*

- Automatizar y sistematizar las pruebas
- Desarrollar un criterio de diseño basado en las pruebas

Si logramos cumplir con estos objetivos habremos eliminado los problemas descriptos y seguramente habremos contribuido a la mejora de la calidad de los productos de software de nuestros proyectos de desarrollo.

En la figura 6.4 se muestra la nueva forma de trabajo en la que se puede observar la influencia de las pruebas durante el desarrollo (relación más gruesa en el dibujo entre analistas, testers y programadores) y además se ha incorporado un nuevo componente en este escenario. A efectos de sistematizar y automatizar las pruebas para contar con una realimentación temprana y con alta frecuencia aparece un servidor de integración continua que describiremos en la sección Integración Continua.

### Trabajo con tests automatizados

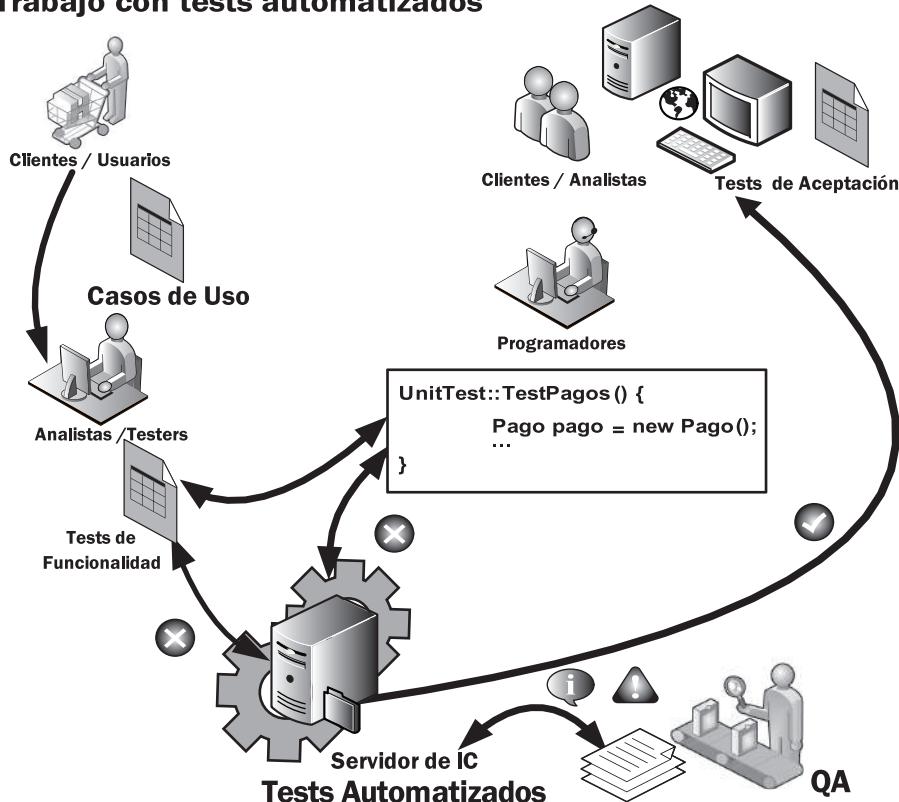


Fig. 6.4 - Ambiente de desarrollo con pruebas automatizadas

### 6.2.1| TRABAJO CON EL REPOSITORIO

Uno de los aspectos claves a la hora de organizar el trabajo de un grupo de desarrollo de software de hoy día es el correcto uso del repositorio de código fuente. Digo hoy día ya que este concepto a pesar de haber sido introducido hace muchos años<sup>49</sup>, con los sistemas de tipo “enterprise”, recientemente tomó mayor relevancia, la que está dada por el volumen del código de estos sistemas y el trabajo con código compartido.

Para resumir mencionaremos un listado de buenas prácticas en el trabajo de los programadores con repositorios de código en las actividades diarias. En el apéndice A pueden encontrar un procedimiento escrito como guía a este trabajo. En el libro de Berczuk<sup>50</sup> podrán leer el tema presentado en formato de patrones.

- Haga commit con frecuencia
- Construya binarios con cada cambio
- No tome del repositorio (check out) a su estación de trabajo código que no funciona
- Escriba test unitarios sistemáticos para el código que desarrolla
- Antes de subir al repositorio (check in) el código desarrollado y todos los tests (suyos y los de los otros desarrolladores) deben ser ejecutados con éxito
- Corrija los errores (tests sin funcionar) que su código generó en forma inmediata

Si se siguen estas buenas prácticas, se contará con un repositorio estable, consistente y que se constituirá en el primer componente de la forma de trabajo que estamos buscando. Otras buenas prácticas se pueden ver en el libro mencionado de Berczuk.

En el uso de esta manera de un repositorio hay implícita una dinámica que constituye la esencia de esa forma de trabajo. Por un lado el repositorio es el único punto de acceso al código compartido que evita inconsistencias varias. Por otro, la regla que establece que no puede ser subido a él código que previamente no fue probado trata de mantener en el repositorio código que funciona. Además, por el hecho de realizar el trabajo con el código compartido desde un inicio y probado día a día, distribuye y facilita la integración de las diferentes partes. Martin Fowler llamó a esto “Integración Continua”<sup>51</sup>. Evita que el grupo desarrolle quince días y esté otros quince integrando su trabajo como sucedía hace veinte años.

49| A pesar de los años todavía encuentro en el mercado organizaciones cuyos grupos de desarrollo no utilizan un repositorio. Nota del autor.

50| Berczuk (Author), et al. *Software Configuration Management Patterns: Effective Teamwork. Practical Integration* (Software Patterns Series). 2008.

51| Fowler, Martin, Continuous Integration, <http://martinfowler.com/articles/continuousIntegration.html>

## 6.2.2 | TEST SISTEMÁTICOS Y AUTOMÁTICOS

Cuando alrededor del año 2000 Kent Beck y Erich Gamma concibieron JUnit, un framework para realizar pruebas sistemáticas en Java, seguramente no solo pretendían construir una herramienta sino una forma de trabajo. El criterio de diseño de este software se convirtió en el seguido por una gran familia de herramientas conocidas como xUnit que hoy día sirven para hacer pruebas sistemáticas en diferentes lenguajes de programación<sup>52</sup>. Entre las pautas de ese diseño estaban:

- Mejorar la calidad del software
- Entender el SUT (System Under Test)
- Reducir los riesgos
- Tests fáciles de ejecutar
- Tests fáciles de escribir y mantener, si es posible en el mismo lenguaje en que los desarrolladores programan el sistema
- Tests con mínimo mantenimiento ante evolución del sistema

Como podemos ver estas pautas están alineadas con la forma de trabajo que buscamos, en especial las que mencionan la mejora de la calidad y la facilidad de ejecución.

Analicemos estas dos pautas:

**La facilidad de ejecución está directamente ligada** a la constante ejecución de los test. Esto es importante por los reiterados cambios a que es sometido el software mientras se desarrolla. Test que no se ejecuta sistemáticamente y con gran frecuencia no sirve. La experiencia indica que si son fáciles de ejecutar se ejecutan, sino no. La facilidad está también directamente vinculada con la automatización, la cual determina un mínimo esfuerzo en el ciclo de edición, compilación, lindeo y ejecución. Por otro lado se vincula a la verificación autónoma (si fallan el framework registra el fallo), la repetibilidad sin intervención manual y a la posibilidad de independencia a partir de una ejecución aislada que facilita el mantenimiento.

**La mejora de la calidad se produce por** la incorporación en etapas tempranas de los programadores al proceso de pruebas, lo que determina la posibilidad de conocer de antemano cómo será utilizado el software y como consecuencia de ello cómo debe ser diseñado para facilitar las pruebas a partir de la forma de uso. Esto contribuye al desarrollo de un criterio de diseño basado en las pruebas. Se han realizado grandes progresos desde el año 2000 y hoy contamos con la posibilidad de implementar pruebas a todas las capas de un sistema con herramientas de tipo xUnit. Para un sistema desarrollado en Java, por ejemplo, se muestran en la tabla 6.1 un conjunto de herramientas de esta familia.

Un comentario adicional es necesario para la automatización de las pruebas de la capa de presentación y para las pruebas relacionadas con las reglas de negocio. Con relación a las primeras la utilización de frameworks, como los tests de Watir, es una

52 | Meszaros, Gerard. *xUnit Test Patterns: Refactoring Test Code*. Addison-Wesley, 2007.

solución interesante, en la cual se continua trabajando para mejorarlas. En relación con las segundas, el framework desarrollado por Rick Mugridge y Ward Cunningham “Fitnesse” (FitFitnesse, Pruebas de funcionalidad y aceptación. Basado en una Wiki para Java, <http://fitnesse.org/>) propone<sup>53</sup> la automatización con la participación de analistas y testers que trabajan en estrecha relación con los programadores, ya que si bien es factible diseñar los casos de prueba sobre planillas de cálculo se necesita escribir código para vincularlas con el SUT. De esta forma se promueve la automatización de las pruebas a partir de una forma de trabajo integrada de los distintos roles.

<b>Herramientas xUnit por capa</b>	
Capa	Herramienta
Presentación	Tests de Watir, <a href="http://wiki.openqa.org/display/WTR/Tutorial">http://wiki.openqa.org/display/WTR/Tutorial</a> HttpUnit, <a href="http://httpunit.sourceforge.net/">http://httpunit.sourceforge.net/</a>
Acceso al Negocio	ServletUnit, <a href="http://httpunit.sourceforge.net/doc/servletunit-intro.html">http://httpunit.sourceforge.net/doc/servletunit-intro.html</a> JUnit, <a href="http://www.junit.org/">http://www.junit.org/</a>
Negocio	FIT Fitnesse, <a href="http://fitnesse.org/">http://fitnesse.org/</a> JUnit, <a href="http://www.junit.org/">http://www.junit.org/</a>
Persistencia	DBUnit, <a href="http://www.dbunit.org/">http://www.dbunit.org/</a> JUnit, <a href="http://www.junit.org/">http://www.junit.org/</a>

**Tabla 6.1** - Herramientas xUnit por capa de una aplicación enterprise

### 6.2.3 | CÓMO ADOPTAR LA NUEVA FORMA DE TRABAJO

A continuación presentaremos una breve guía de las cuestiones a tener en cuenta al momento de decidir adoptar la forma de trabajo que implica pruebas sistemáticas y automáticas.

#### 6.2.3.1 Obstáculos para automatizar las pruebas

- Actitud de los programadores: es necesario inducir la cultura de la prueba y el trabajo con los analistas y testers, es decir, una forma de trabajo donde los roles se comuniquen abiertamente.
- Inversión inicial: es necesario capacitar a los programadores en el diseño de pruebas y uso de herramientas xUnit.
- Código que siempre cambia: es el escenario más propicio para aplicar esta forma de trabajo.

53 | FitFitnesse, Pruebas de funcionalidad y aceptación. Basado en una Wiki para Java, <http://fitnesse.org/>

- Sistemas legacy: deben aprenderse técnicas de pruebas para testear estos sistemas.
- Temor: siempre viene de la mano del cambio, hay que administrarlo.
- Viejos hábitos: lentamente hay que dejarlos de lado

#### 6.2.3.2 Qué debería automatizarse

- Pruebas unitarias y de componentes
- Pruebas de funcionalidad sin interfaces de usuario
- Pruebas de sistema con interfaces de usuario

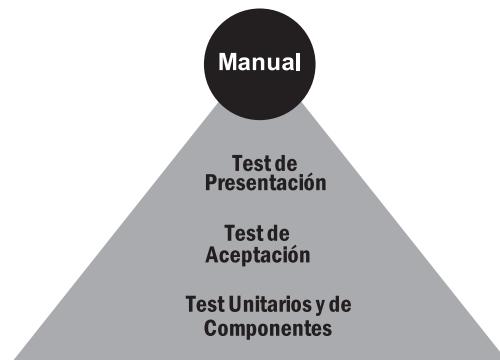


Fig. 6.5 - Pirámide de pruebas que indica la automatización de los diferentes test

#### 6.2.3.3 Qué no debería automatizarse

- Pruebas de usabilidad
- Pruebas exploratorias
- Pruebas que no fallarán
- Tareas únicas de fácil ejecución manual y de difícil automatización<sup>54</sup>

#### 6.2.3.4 Estrategia para comenzar la automatización

- Capacitación a analistas, testers y programadores
- Seleccionar una forma de trabajo
- Seleccionar herramientas
- Desarrollar proyectos pilotos
- Institucionalizar

54 | Mugridge, Rick. Ward Cunningham. *Fit for Developing Software: Framework for Integrated Tests*. Prentice Hall PTR. 2005.

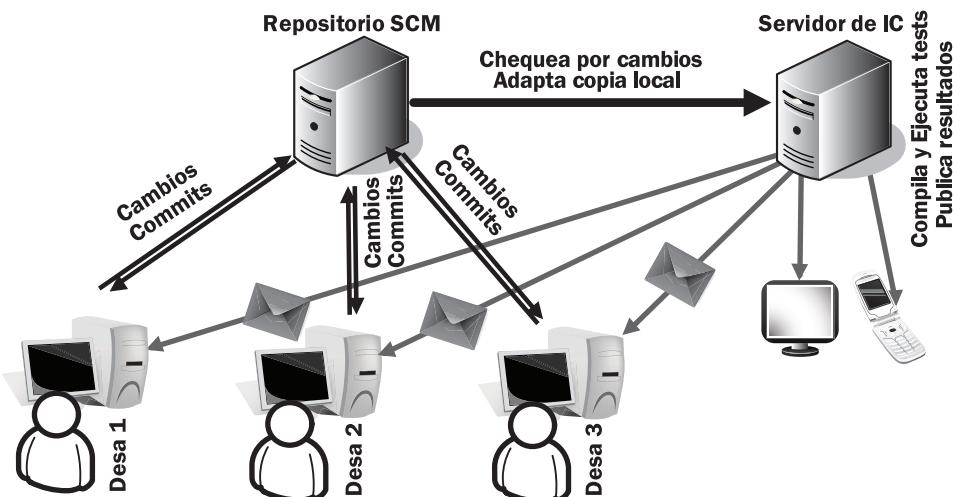
## 6.3 | INTEGRACIÓN CONTINUA

### 6.3.1 | FORMA DE TRABAJO

Si bien hemos resuelto algunos de los problemas planteados en el inicio del capítulo, aún restan otros para los que ahora propondremos una solución. Por cierto propusimos la mejora de la calidad de nuestros diseños incorporando a las pruebas como criterio y sistematizamos dichas pruebas con la utilización de las herramientas xUnit. Veamos entonces una forma de automatizarlas y de extraer indicadores que nos guien en la selección de productos de software (diseño y código) a revisar. En la figura 6.6 se muestra un esquema de un ambiente de trabajo de “integración continua”. La dinámica de la forma de trabajo se indica en los pasos que se listan a continuación.

#### 6.3.1.1 Pasos

1. Un desarrollador realiza un commit (cambios) sobre el repositorio (SCM) mientras el administrador de integración continua (IC) lo consulta por cambios con una frecuencia determinada.
2. Después del commit el administrador de IC detecta el cambio, toma del repositorio las últimas versiones y ejecuta los scripts que integran todo el software (compila y linea), ejecuta los tests, toma métricas y elabora informes.
3. El administrador de IC informa por mail a los miembros del grupo de desarrollo de los resultados del build.
4. El administrador de IC continua consultando al repositorio (SCM) con la frecuencia determinada.



**Fig. 6.6** - Ambiente de trabajo con integración continua

### 6.3.1.2 Principios

Los criterios sobre los que se fundamenta esta arquitectura se resumen en los principios mostrados en la tabla 6.2, que fueron adaptados de Paul M. Duvall with Andrew Glover and Steve Matyas<sup>55</sup>.

Principios de integración continua	
Repositorio	Construya binarios con cada cambio Haga commit con frecuencia Corrija errores que generan fallas de los test en forma inmediata No haga commit de código no probado Nombre cada entregable e indique con qué componentes fue construido generando capacidad para volver atrás
Construcción de binarios	Automatizar Separar del IDE (integrated development environment) Ejecute comandos simples Crear estructura de directorios consistente Automatice la integración de la base de datos Utilice una copia privada de la base de datos
Pruebas y revisiones	Automatice y categorice los tests Ejecute los tests en momentos diferentes Escriba tests para fijar errores Haga a los tests repetibles Controle el cubrimiento de los tests Defina estándares de la organización para los diseños y el código mantenidos por QA Promueva el uso de estándares ejecutando revisiones guiadas por métricas Tome métricas sobre el diseño y el código
Integración continua	Utilice un servidor independiente para la integración continua Provea información para la publicación de resultados a los involucrados Publique resultados de la integración en forma continua Genere informes con cada ejecución y guárdelos
Informes	Construya informes y publíquelos a los desarrolladores y a QA Promueva la lectura de los informes

**Tabla 6.2** - Listado de principios de integración continua

55 | Paul M. Duvall with Andrew Glover & Steve Matyas. *Continuous Integration: Improving Software Quality and Reducing Risk*. 2008.

### 6.3.2 | INFRAESTRUCTURA

A efectos de ejemplificar un posible ambiente de integración continua se muestran en las figuras que siguen las herramientas integradas y los nodos sobre los cuales estas fueron instaladas. La arquitectura del ambiente mostrado en el esquema se presenta en las dos figuras. En la figura 6.7 se muestran las herramientas y sus roles, y en la 6.8, los nodos en los cuales están instaladas.

Se indican, con diferentes colores, los roles de las herramientas en relación con lo expuesto en la tabla 6.3. Pueden verse una gran cantidad de módulos orientados a tomar métricas y armar informes relacionados. La presentada es solo una posible arquitectura usada por nosotros en algunos proyectos en los que se utilizó tecnología Java. Una referencia donde pueden encontrarse vínculos a otras herramientas es el libro de Duvall (op. cit.).

Descripción de herramientas por rol		
Rol	Herramienta	Propósito
Repositorio	Subversion 1.4.2	Administración del repositorio
Construcción de binarios	Maven 2.0.9	Organización del proyecto Definición de ciclo estándar de generación Generación de binarios Definición de directorios Administración de dependencias Administración de repositorios de binarios Generación automática de entregables Integración de plug ins de informes
	Ant 1.7.1	Ejecución de scripts de construcción
Pruebas y métricas para revisiones	JUnit 3.8.1 JDepend ChecStyle PMD/CPD Clover JavaDoc Doxygen JIRA 3.13 Enterprise Architecture	Pruebas automáticas y sistemáticas Analizar la relación entre paquetes Chequear los estilos de programación Detectar problemas de programación Controlar el cubrimiento de los tests Documentar código del sistema y tests Documentar diseño desde el código Seguimiento de problemas y errores Documentación del proyecto
Integración continua	Cruise Control 2.7.3	Detección de cambios en el repositorio Actualización de su ambiente de trabajo Ejecución de las fases de construcción de Maven Informar a involucrados de los resultados Almacenar historia de ejecuciones
Informes	Tomcat 5.5	Publicar informes con resultados

Tabla 6.3 - Herramientas según su rol

Al final del capítulo se presentan los vínculos a los sitios web de las herramientas mencionadas.

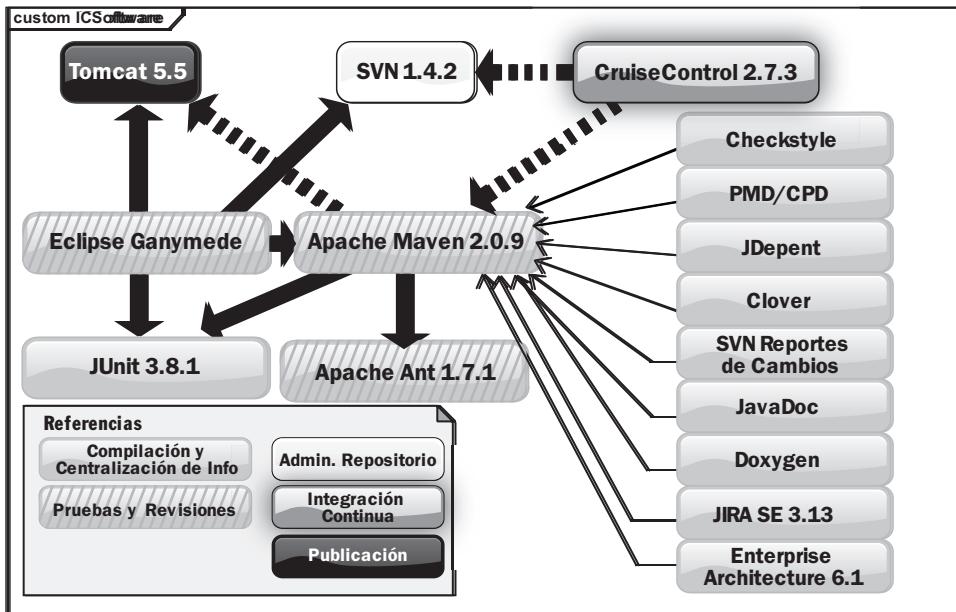


Fig. 6.7 - Herramientas y sus roles utilizadas en un ambiente de integración continua

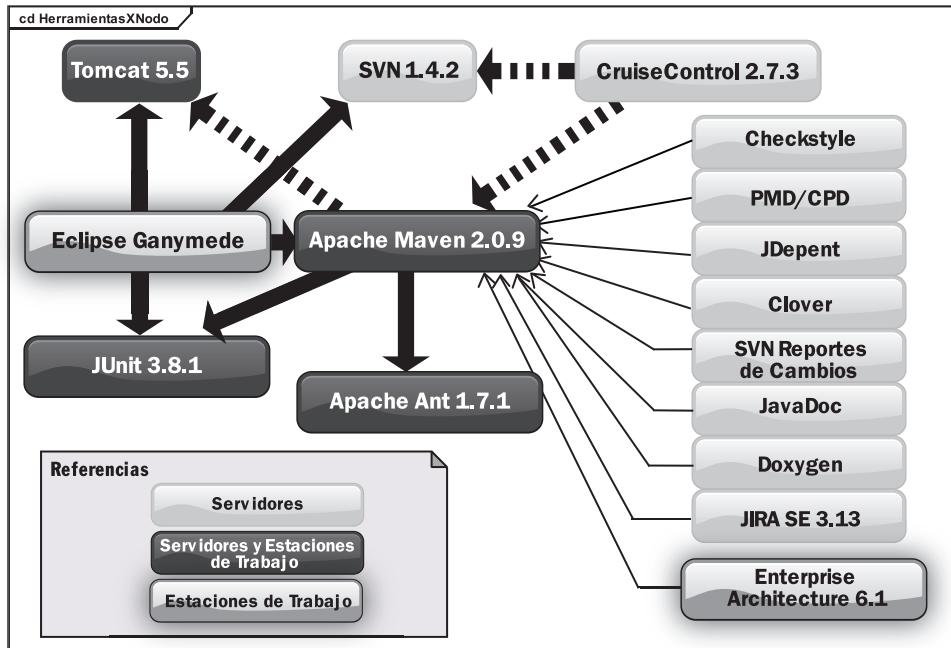


Fig. 6.8 - Herramientas utilizadas en un ambiente de integración continua y el nodo asociado donde fueron instaladas

### 6.3.3 | RESULTADOS

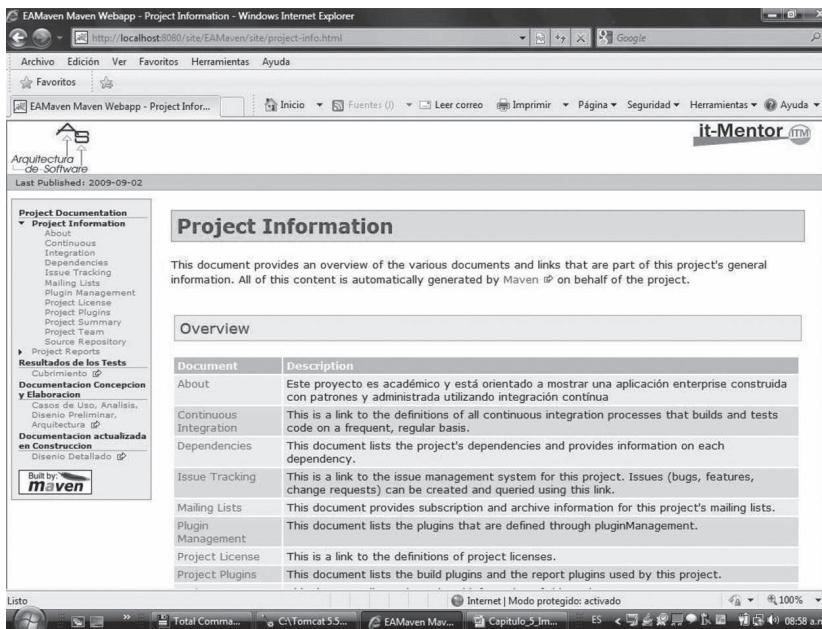
Un aspecto de gran importancia que deseamos volver a mencionar es la posibilidad que ofrecen Maven y Cruise Control de generar un sitio para el proyecto donde se publican los informes con resultados y métricas.

Este sitio es el punto de referencia de los involucrados en el proyecto así como para los miembros del área de calidad de la organización, ya que ellos encontrarán información de gran utilizada para realizar su trabajo con relación al proyecto.

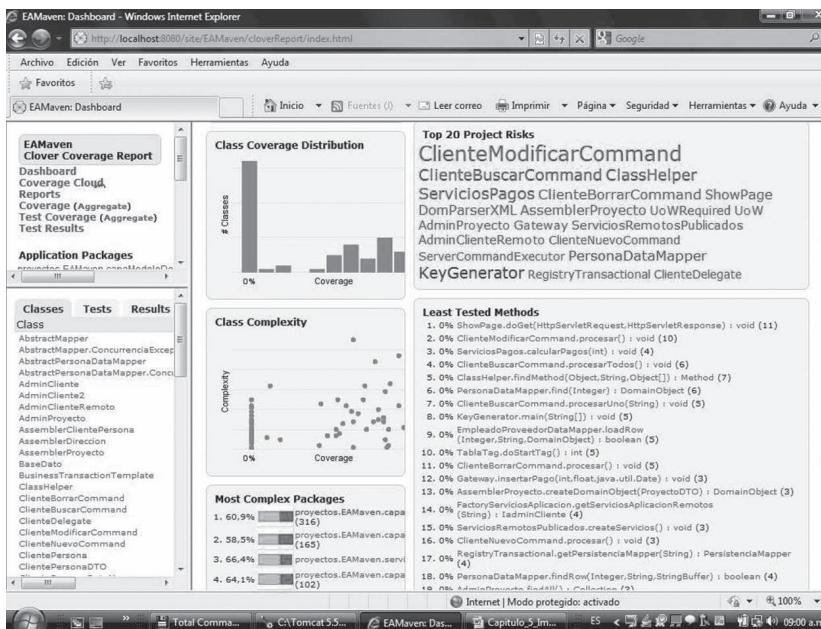
En la figura 6.9 se presenta una imagen de la portada de uno de estos sitios.

Es importante tener en cuenta que la información allí publicada es actualizada con cada construcción de binarios, por lo que constituye la fuente de información más actualizada y precisa del proyecto.

De esta forma hemos logrado los objetivos iniciales planteados en el comienzo del capítulo en relación con la visibilidad y la actualidad de la información deseable a efectos de respondernos la pregunta ¿qué hemos desarrollado y qué falta desarrollar?



**Fig. 6.9** - Sitio de un proyecto Maven2



**Fig. 6.10** - Vista del sitio construido con Maven2, donde se muestra la cobertura de los tests, elaborada con Clover

En la figura 6.10 se muestra una vista del mismo sitio correspondiente al informe de cubrimiento de los tests construida con Clover.

## 6.4 | REVISIONES DE DISEÑO Y CÓDIGO

### 6.4.1 REVISIONES

Las revisiones de productos por parte de pares y/o especialistas externos ha probado ser un método efectivo para la detección de errores. Cuando se alcanza madurez y habilidad para su desarrollo, los números indican que el 60% de los errores de un proyecto son detectados por este medio contra el 40% restante detectados por los tests.

En la figura 6.11 se esquematiza los diferentes roles que participan de una revisión.

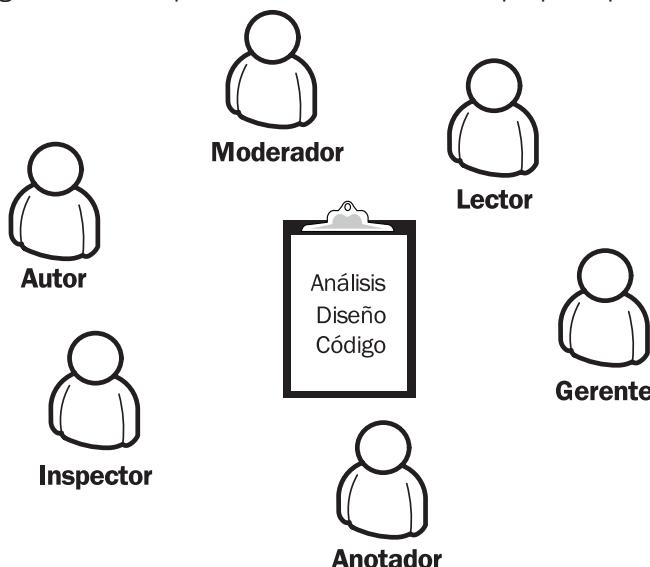


Fig. 6.11 - Ambiente de revisión de artefactos de software

Las revisiones deben ser rigurosas y en profundidad; su objetivo debe ser detectar errores en los productos revisados.

#### 6.4.1.1 Objetivos

1. Detectar problemas de análisis, diseño y código en forma temprana
2. Definir y acordar criterios de retrabajo para su resolución
3. Verificar que se resolvió de acuerdo al criterio acordado

#### 6.4.1.2 Beneficios

1. Genera datos acerca del producto y del proceso de desarrollo
2. Genera datos acerca del producto y del proceso de desarrollo
3. Genera conocimiento entre miembros del grupo de desarrollo
4. Aumenta la efectividad de la validación y verificación
5. Contribuye a la instalación del concepto de calidad

Estas revisiones pueden ser más o menos formales. En la siguiente tabla se muestran las diferencias entre ambas.

**Formales:** Con roles y responsabilidades, y un procedimiento definido.

**Informales:** Con roles desdibujados y sin procedimiento.

Revisores		
Característica	Formal	Informal
Objetivos	Detectar errores	Detectar errores
	Verificar retrabajo	Discutir alternativas de solución
	Focalizada sobre si sí o no los productos cubren los requerimientos	Focalizada en demostrar cómo los productos cubren los requerimientos
Decisiones	Decisiones concensuadas	Decisiones del autor
Responsable	Moderador entrenado	Autor
Asistentes	Pares con asistencia registrada	Pares y responsables técnicos, sin registrar
Material	Presentador por el lector	Presentado por el autor
Métricas	Requeridas	Opcionales
Procedimiento	Formalmente registrado	Informal
Entrenamiento	Requerido para todos los roles	Requerido para todos los roles

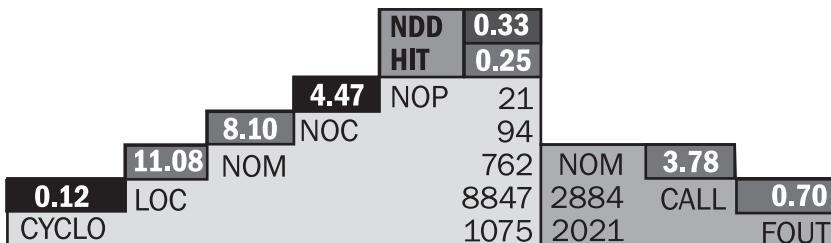
**Tabla 6.4** - Características de la revisiones formales e informales

#### 6.4.1.3 Métricas guía

Como mencionamos en la sección Problemas-Proceso de Diseño es de gran importancia la selección de los productos a revisar. Algunos indicadores obtenidos como resultado de la integración continua pueden ser utilizados como tal. Otros indicadores sobre los que se está trabajando y prometen ser un aporte sumamente valioso son las métricas de código asociadas a las desarmonías de diseño<sup>56</sup>. En las figuras 6.12 y 6.13 se muestra una vista de una de estas métricas presentadas en forma gráfica. Ambas fueron obtenidas con el framework inCode para el mismo sistema. La primera muestra índices de volumen, uso de mecanismo de herencia y acoplamiento. La segunda, una polimétrica de la complejidad del sistema. Estos indicadores son un excelente punto de partida en la revisión de los productos de un proyecto.

56| Michele Lanza, Radu Marinescu. *Object-Oriented Metrics in Practice - Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems*. Springer. 2006.

#### 6.4.1.4 Indicadores



#### Interpretation of the Overview Pyramid for module EAMaven-TRUNK

**Class Hierarchies** tend to be **tall** and of average width (i.e. inheritance trees tend to have many depth-levels and base-classes with several directly derived sub-classes)

#### Classes tend to:

- be rather **large** (i.e. they define many methods);
- be organized in rather **fine-grained packages** (i.e. few classes per package);

#### Methods tend to:

- be rather **long** yet having a rather **simple logic** (i.e. few conditional branches);
- call **many methods** (high coupling intensity) from **many other classes** (high coupling dispersion);

More details on the Overview Pyramid can be found in the *Object-Oriented Metrics in Practice* book or by contacting us at [incode@cs.upt.ro](mailto:incode@cs.upt.ro) For any questions or suggestions, please visit our web site and forum.

**Fig. 6.12** - Métricas de tamaño, uso de herencia y acoplamiento inCode

#### 6.4.1.5 Polimétrica de complejidad

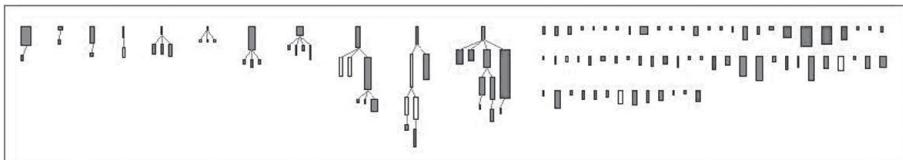


Fig. 6.13 – Métrica de complejidad inCode

## 6.5 | CONCLUSIONES

### 6.6 | Herramientas

La preocupación que expresamos en el comienzo del capítulo fue conocer, en cada momento de las iteraciones de la construcción de un sistema en desarrollo, el estado actualizado de los productos generados por el proyecto.

Cuestionamos las especulaciones que generalmente se utilizan para dar una respuesta certera a esta pregunta y para ello promovimos la utilización de un repositorio de código fuente con ciertas reglas de uso, la sistematización y automatización de las pruebas, la integración continua por parte de un servidor destinado a tal fin y la revisiones de productos conducidas por métricas obtenidas de los informes de resultados.

En definitiva, *buenas prácticas orientadas a garantizar la calidad de los productos de los proyectos organizando y controlando la logística*.

## 6.6 | HERRAMIENTAS

1. **Subversion:** <http://subversion.tigris.org/>
2. **Ant:** <http://ant.apache.org/>
3. **Maven:** <http://maven.apache.org/>
4. **Eclipse:** <http://www.eclipse.org/>
5. **Junit:** <http://junit.sourceforge.net/>
6. **Tomcat:** <http://tomcat.apache.org/>
7. **CruiseControl:** <http://cruisecontrol.sourceforge.net/>
8. **CheckStyle:** <http://checkstyle.sourceforge.net/>
9. **JavaDoc:** <http://java.sun.com/j2se/javadoc/>
10. **Doxxygen:** <http://www.stack.nl/~dimitri/doxygen/>
11. **PMD/CPD:** <http://pmd.sourceforge.net/cpd.html>
12. **JDepend:** <http://clarkware.com/software/JDepend.html>

13. **Enterprise Architecture:** <http://www.sparxsystems.com.ar/>
14. **Clover:** <http://www.atlassian.com/software/clover/>
15. **JIRA:** <http://www.atlassian.com/software/jira/>
16. **FitFitness:** <http://fitnesse.org/>
17. **JMeter:** <http://jakarta.apache.org/jmeter/>

---

## **Capítulo 7: Trabajo con Modelos de Desarrollo - CMMI**

---

<b>7.1   Modelos de referencia.....</b>	160
7.1.1 CMMI .....	162
7.1.2 Relación entre áreas de proceso .....	167
7.1.3 Desmistificando el modelo .....	169
7.1.3.1 <i>Por qué CMMI</i> .....	169
7.1.3.2 <i>Con quién trabajar</i> .....	169
7.1.3.3 <i>Cómo es el proceso de mejoras con CMMI</i> .....	171
7.1.3.4 <i>Qué recursos se necesitan</i> .....	175
7.1.3.5 <i>Cómo es la evaluación con CMMI (SCAMPI)</i> .....	177
<b>7.2  Mejora de procesos utilizando el modelo CMMI .....</b>	181
7.2.1 Estrategia general .....	182
7.2.1.1 <i>Políticas y Procesos</i> .....	183
7.2.1.2 <i>Interpretación y mapeo de objetivos y tareas</i> .....	184
7.2.2 Institucionalización .....	187
7.2.2.1 <i>Relación entre áreas de proceso y objetivos genéricos</i> .....	187
<b>7.3  Modelos y metodologías.....</b>	189
7.3.1 Metodologías y modelos .....	189
7.3.2 CMMI y metodologías .....	190
<b>7.4  Madurez .....</b>	191
<b>7.5  Conclusiones .....</b>	192

# 7

## TRABAJO CON MODELOS DE DESARROLLO - CMMi

### 7.1 | MODELOS DE REFERENCIA

---

¿Cómo procedemos cuando tenemos que planificar una mejora de procesos?, ¿qué referencia utilizamos?, ¿cómo evaluamos el desempeño del proyecto asociado? Podríamos usar nuestra experiencia e implementar todas las buenas prácticas que conocemos. Sin embargo, la comunidad informática optó por elegir el modelo de referencia CMM (Capability Maturity Model)<sup>57</sup> que de facto se convirtió en el estándar del mercado de desarrollo de software de los años 90 y su versión perfeccionada y aumentada CMMi (Capability Maturity Model Integration)<sup>58</sup> en el año 2000.

En adelante trabajaremos con el modelo CMMi, al cual llamaremos “el modelo”. Este estándar posee dos propiedades que deseamos puntualizar.

Una es que fue concebido en términos generales y con la intención de abarcar todo tipo de organizaciones. Sin embargo, debido al espónsor del trabajo (Department of Defense of United States of America - DoD-EEUU), al origen de sus autores (grandes corporaciones) y a los objetivos que persiguían (ordenar y clasificar a los proveedores del DoD) hicieron que el modelo se corresponda con grandes organizaciones. Esto se convirtió en todo un desafío a la hora de implementar el modelo en pequeñas y medianas empresas (pymes).

La otra es su ambigüedad en la definición de las prácticas para lograr sus objetivos, lo cual deja a criterio de los implementadores cómo hacerlo. Esto último ha sido una de las

---

57 | M. C. Paulk, B. Curtis, (et al.). *Capability Maturity Model for Software*. Software Engineering Institute, CMU/SEI-91-TR-24, ADA240603. 1991.

58 | Capability Maturity Model ® Integration (CMMi SM), Version 1.1, CMMi SM for Systems Engineering and Software Engineering (CMMISE/SW, V1.1), Continuous Representation, CMU/SEI-2002-TR-001, ESC-TR-2002-001.

---

causas de fracaso en la implementación del modelo ya que es utilizado por profesionales que no saben cómo adaptarlo a la organización en la que trabajan y además creen que el modelo por sí mismo expresa una implementación. Por esta razón, en este libro lo llamamos “modelo de referencia”, evidenciando que el contenido del modelo, es decir el cómo, deberemos aportarlo nosotros.

En este capítulo nos referiremos al modelo en general, realizaremos un mapeo con un Proceso de Desarrollo y analizaremos los errores comunes que las empresas cometan en su implementación.

### 7.1.1 CMMi

CMMi es un modelo de capacidad y madurez. Posee dos vistas que permiten un enfoque diferente según las necesidades de quien vaya a implementarlo. Una estructurada en cinco niveles de madurez y otra en seis de capacidad. En la figura 7.1 se muestra un esquema del modelo. Las áreas de procesos están agrupadas en Soporte, Ingeniería, Administración de Proyectos y Administración de Procesos.

Los niveles de capacidad indican qué tan bien se desempeña la organización en un área de proceso individual. El nivel de capacidad de cero (CL 0) indica que los procesos no se realizan, lo que significa que uno o más de los objetivos específicos del área de proceso no se cumplen. Los niveles de capacidad aumentan hasta el nivel de capacidad cinco (CL 5), donde el proceso se realiza bien y se está mejorando continuamente.

Los niveles de madurez indican cómo se desempeña una organización en base a la capacidad y madurez en un conjunto de áreas de proceso. El nivel de madurez uno (ML 1) indica que los procesos no se realizan, lo que significa que los proyectos se desarrollan de manera impredecible y descontrolada. Los niveles de madurez aumentan hasta el nivel de madurez cinco (ML 5), en el que los proyectos se desarrollan de una forma cuantitativamente definida para la organización y mejorando continuamente.

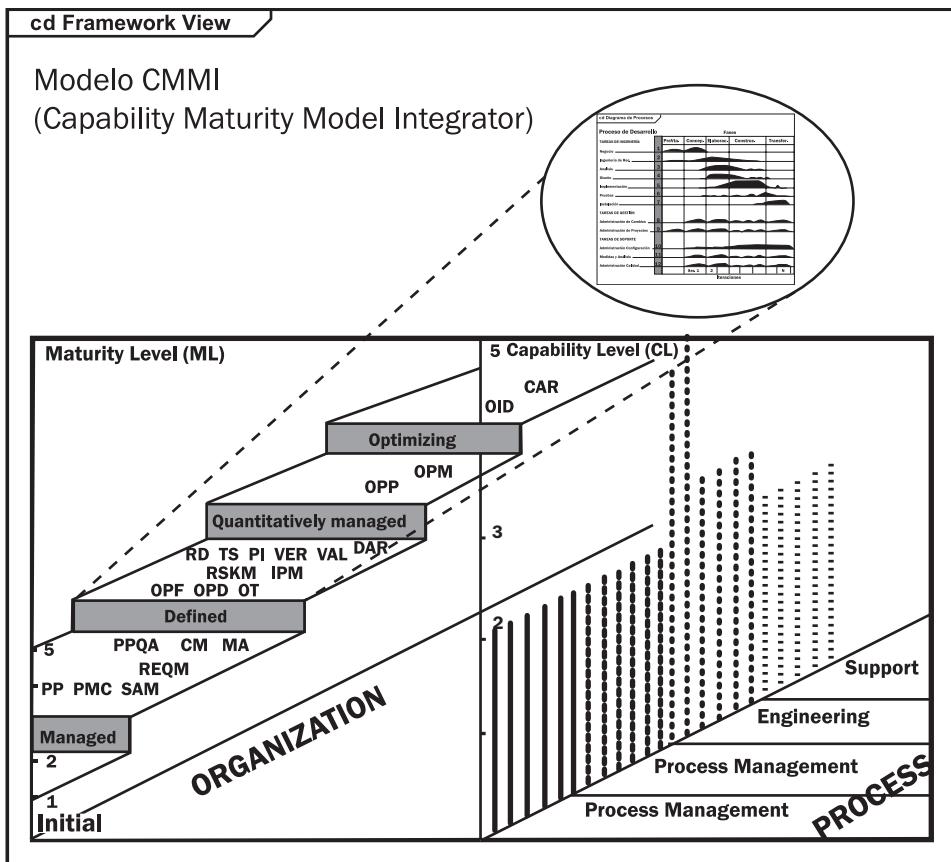
El modelo está compuesto por los siguientes componentes: **objetivos, prácticas y subprácticas**.

Los **objetivos son componentes requeridos**, es decir que al momento de evaluar a una organización deben estar satisfechos.

Estos objetivos son de dos tipos: específicos, por área de proceso, y genéricos, relacionados a la institucionalización.

Las **prácticas son esperadas**, es decir que si se alcanzaron los objetivos se espera contar con las prácticas que lo hicieron posible. Estas prácticas son específicas, asociadas a los objetivos específicos, y genéricas, asociadas a los objetivos genéricos.

Las **subprácticas son componentes informativos** que ayudan a la interpretación e implementación de las prácticas. Estas están constituidas por los activos de trabajo y las distintas disciplinas.



**Fig.7.1** - Esquema del modelo CMMI<sup>59</sup>

En la tabla 7.1 se presenta el modelo con sus niveles de madurez, áreas de proceso y características esenciales. Los nombres de los niveles y áreas de proceso se mantienen en idioma inglés.

59 | Modelo elaborado en it-Mentor. Nota del autor.

Nivel de Madurez	Características de Procesos	Áreas de Proceso
ML5 Optimizing	Foco en la mejora continua en base a indicadores.	OID - Organizational Innovation and Deployment CAR - Causal Analysis and Resolution
ML4	Procesos gestionados en base a indicadores.	OPP - Organizational Process Performance
Quantitatively managed		QPM - Quantitative Project Management
ML3 Defined	Procesos definidos para la organización, más aspectos tecnológicos.	RD - Requirements Development TS - Technical Solution PI - Product Integration VER - Verification VAL - Validation OPF - Organizational Process Focus OPD - Organizational Process Definition OT - Organizational Training IPM - Integrated Project Management RSKM - Risk Management DAR - Decision Analysis and Resolution
ML2 Managed	Procesos focalizados en lograr acuerdos en los proyectos.	REQM- Requirements Management PP - Project Planning PMC - Project Monitoring and Control SAM - Supplier Agreement Management MA - Measurement and Analysis PPQA - Process and Product Quality Assurance CM - Configuration Management
ML1 Initial	Procesos impredecibles, descontrolados y reactivos	

**Tabla 7.1** - Áreas de proceso por niveles de madurez

Es interesante hacer notar que su nivel más bajo (2), como se observa en la figura 7.1, es consistente con nuestras observaciones extraídas del informe del Standish Group, ya que busca que los proyectos se desarrollen dentro de una planificación y un presupuesto establecidos. Por esta razón las áreas de proceso incluidas abarcan la administración de cambios a los requerimientos (Requirements Management), la planificación de los proyectos (Project Planning), su seguimiento (Project Monitoring and Control) y la administración de los proveedores (Supplier Agreement Management). Estas áreas tienen como objetivo central establecer acuerdos. El resto de las áreas de este nivel son de soporte y buscan crear el ambiente de desarrollo apropiado para que estos acuerdos sean factibles y además recogen métricas con vistas a avanzar en una organización más sistemática de los proyectos. No son las cuestiones técnicas las

incluidas en este primer nivel. Estas son dejadas para mejorar en niveles superiores (3). Esto es también consistente con las conclusiones que analizamos a partir de los Informes de Caos mencionados, en los que estas cuestiones aparecían como causas de fracaso de los proyectos detrás de las relacionadas a la planificación, los compromisos, el involucramiento y los acuerdos entre las partes de un proyecto.

En la tabla 7.2 se presentan los objetivos genéricos en relación con los niveles de capacidad y las prácticas genéricas asociadas.

Nivel de Capacidad	Objetivos Genéricos	Prácticas
CLO Not Performed	No goal.	
CL1 Performed solo en la representación continua	GG 1: The process supports and enables achievement of the specific goals of the process area by transforming identifiable input work products to produce identifiable output work products.	
CL2 Managed	GG 2: The process is institutionalized as a managed process.	GP 2.1 Establish an Organizational Policy GP 2.2 Plan the Process GP 2.3 Provide Resources GP 2.4 Assign Responsibility GP 2.5 Train People GP 2.6 Manage Configurations GP 2.7 Identify and Involve Relevant Stakeholders GP 2.8 Monitor and Control the Process GP 2.9 Objectively Evaluate Adherence GP 2.10 Review Status with Higher Level Management
CL3 Defined	GG 3: The process is institutionalized as a defined process.	GP3.1 Establish a defined process GP3.2 Collect improvement information
CL4 Quantitatively Managed (solo en la representación continua)	GG 4: The process is institutionalized as a quantitatively managed process.	GP4.1 Establish Quantitative Objectives for the Process GP4.2 Stabilize Sub process Performance
CL5 Optimizing (solo en la representación continua)	GG 5: The process is institutionalized as an optimizing process.	GP5.1 Ensure Continuous Process Improvement GP5.2 Correct Root Causes of Problems

**Tabla 7.2** - Objetivos genéricos por niveles de capacidad

En la tabla 7.3 se indica la equivalencia entre ambas vistas del modelo. Se puede ver que hasta los niveles 3 se da una equivalencia, pero para niveles superiores esto no sucede, debido a que los niveles 4 y 5 no agregan nuevas áreas de proceso, sino que las áreas correspondientes operan sobre las de los niveles inferiores. Por esta razón, desde la perspectiva continua del modelo, una organización podría administrar algunas áreas en forma cuantitativa u optimizada (niveles 4 y 5) y otras solo como un proceso definido (nivel 3) o como un proceso administrado (nivel 2).

Niveles de Capacidad			Nivel de Madurez	Áreas de Proceso		
NA	NA	Equiv. 5	5	OID - Organizational Innovation and Deployment		
			5	CAR - Causal Analysis and Resolution		
		Equiv. 4	4	OPP - Organizational Process Performance		
			4	QPM - Quantitative Project Management		
		Equiv. 3		RD - Requirements Development TS - Technical Solution PI - Product Integration VER - Verification VAL - Validation OPF - Organizational Process Focus OPD - Organizational Process Definition OT - Organizational Training IPM - Integrated Project Management RSKM - Risk Management DAR - Decision Analysis and Resolution		
		Equiv. 2		REQM- Requirements Management PP - Project Planning PMC - Project Monitoring and Control SAM - Supplier Agreement Management MA - Measurement and Analysis PPQA - Process and Product Quality Assurance CM - Configuration Management		
CL5	CL4	CL3	CL2	CL1	ML	Areas de proceso

**Tabla 7.3** - Equivalencia entre las vistas por niveles de madurez y de capacidad del modelo CMMI

### 7.1.2 RELACIÓN ENTRE ÁREAS DE PROCESO

A efectos de vincular conceptos y contar así con una visión más clara del modelo presentaremos a continuación la relación entre las áreas de proceso de nivel 3. En las figuras que siguen se muestran estas relaciones, las que fueron adaptadas de CMMI SM (op. cit.). En ellas puede verse cómo las áreas de niveles inferiores (2) sirven de Soporte al resto, figura 7.2, y las áreas vinculadas a la Gestión aparecen vinculadas a los Proyectos, figura 7.3. Las áreas de Ingeniería, figura 7.4, aparecen intensamente relacionadas entre sí y ligadas al proceso de administración de cambios en los requerimientos, mientras las áreas asociadas a la administración de Procesos, figura 7.5, aparecen definiendo el marco para todas las anteriores.

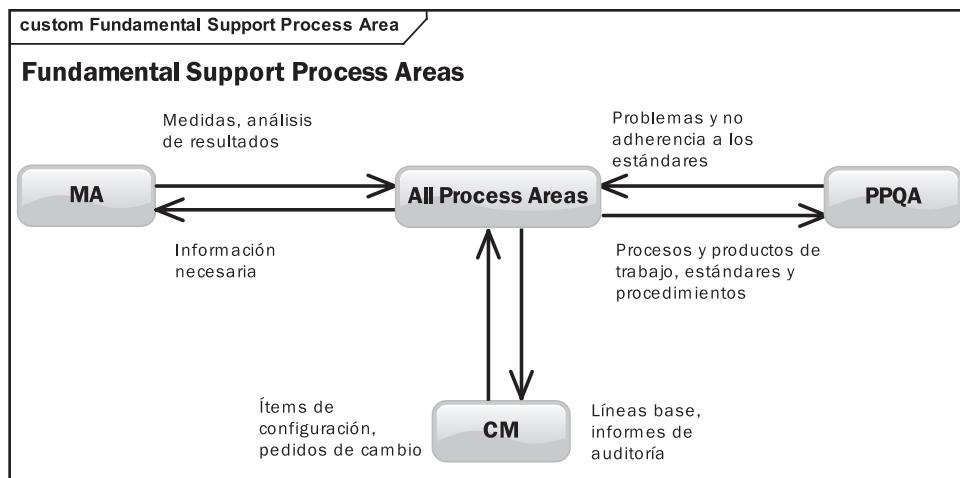


Fig. 7.2 - Áreas de proceso de soporte y su relación con el resto de las áreas

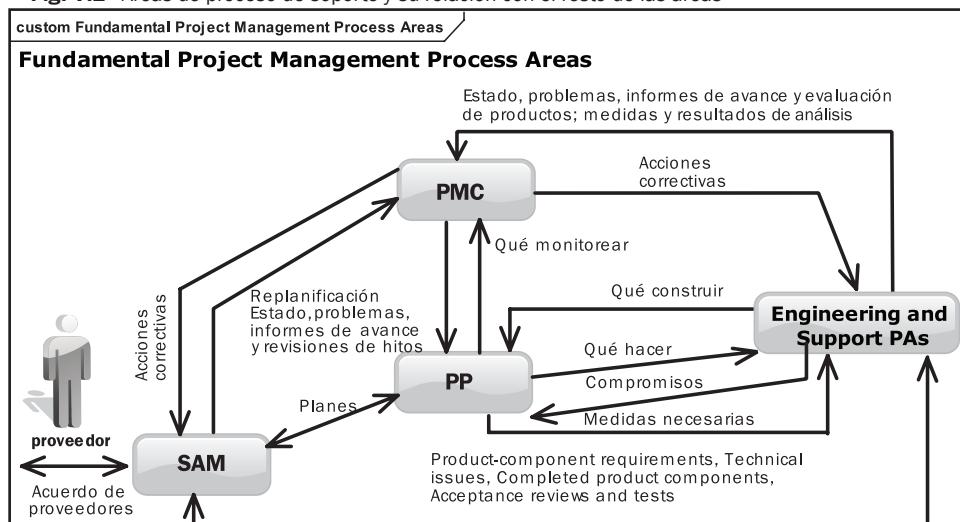


Fig. 7.3 - Áreas de proceso de administración de proyectos y su relación con el resto de las áreas

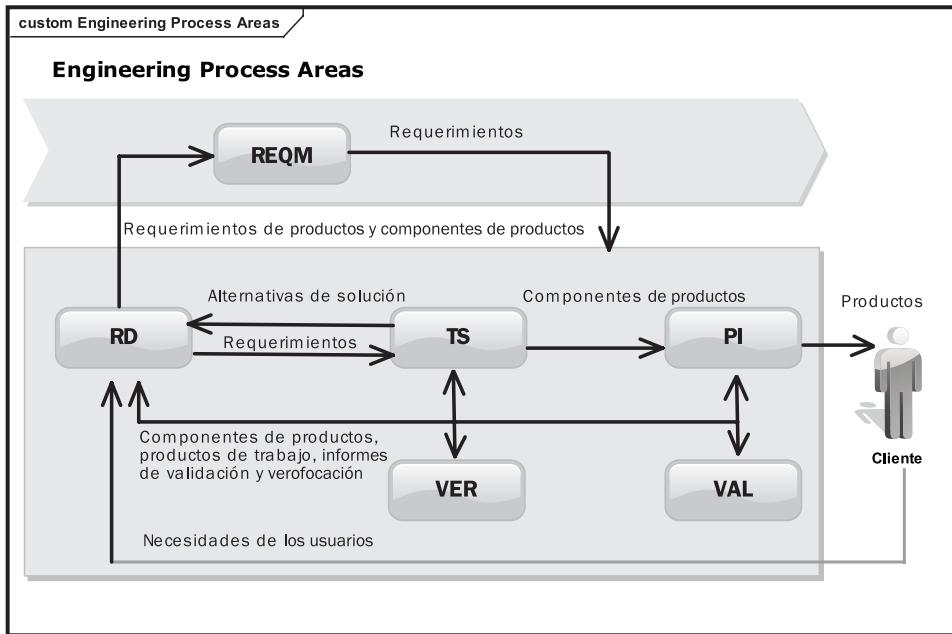


Fig. 7.4 - Áreas de proceso de ingeniería

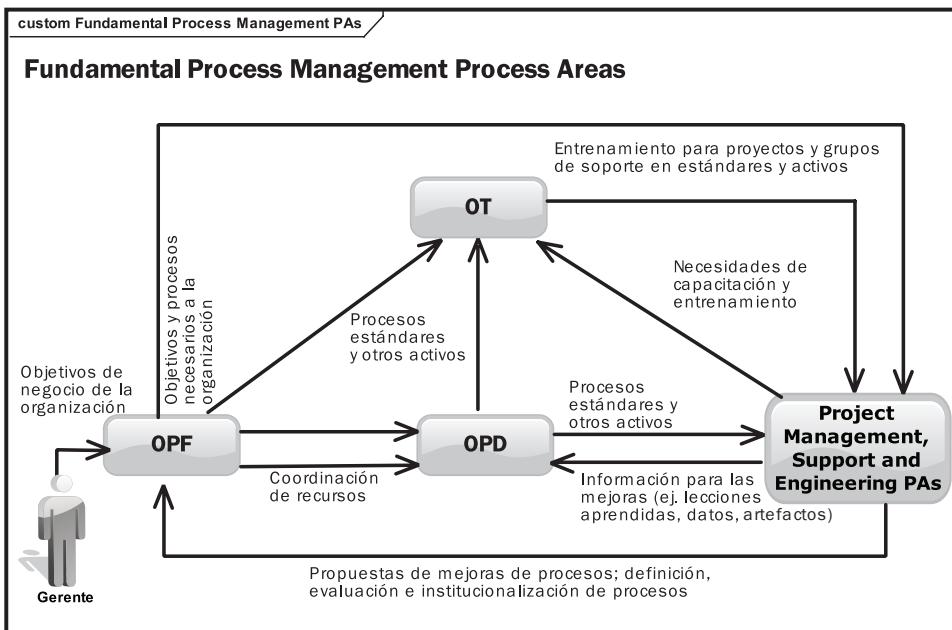


Fig. 7.5 - Áreas de administración de procesos y su relación con el resto de las áreas

### 7.1.3 | DESMISTIFICANDO EL MODELO

En esta sección nos referiremos a las preguntas que se hacen los responsables de las empresas al momento de tomar la decisión de comenzar con un proyecto asociado a una mejora de procesos orientado a ser evaluadas según un modelo de referencia.

#### 7.1.3.1 Por qué CMMi

Hay una gran cantidad de modelos y de normas pensados para servir de referencia en la organización del área de desarrollo de software de una empresa.

La creciente popularidad del modelo CMM y su extensión CMMi, en gran parte de la mano del crecimiento de las software factory de la India, significó que ese país cuente con el mayor número de empresas evaluadas en este modelo.

Hacia fines de los 90, en Europa hubo una tendencia a trabajar con empresas consultoras y software factory de esa región que se revirtió en los últimos años a raíz de la gran cantidad de proyectos fallidos. Entre las razones de estos fracasos se cuentan las diferencias culturales, horarias, la distancia y la dificultad de construir una forma de trabajo común. En mi país, al momento de escribir este libro, existe un impaz en el uso del modelo y los procesos de mejoras asociados. La ola de evaluaciones CMMi llegó al mercado de desarrollo de software argentino de la mano de la llamada “Ley del software”, la cual otorga excenciones impositivas a las empresas que cumplen con una serie de requisitos. Entre estos, se exige una certificación u evaluación en algún modelo o norma tales como CMMi o ISO. Después de seis años de trabajo la cantidad de organizaciones o áreas de desarrollo evaluadas exitosamente ha crecido sustancialmente. Sin embargo las experiencias son bien distintas y más allá de la evaluación positiva los resultados a mediano plazo han sido, en algunos casos, muy malos. Algunas empresas han desaparecido, otras se han reducido sustancialmente y en algunas no existe rastro alguno de que hace apenas unos años se trabajó en la organización utilizando el modelo. Si bien la causa de estos sucesos no fue únicamente el haberse involucrado en un proceso para ellos desconocido y además mal guiados por mentores sin criterio para adaptar el modelo a las pymes, los recursos gastados en los proyectos vinculados a estas mejoras contribuyeron de una manera importante al deterioro mencionado.

Los resultados no fueron tan frustrantes ni tuvieron un impacto tan importante para aquellas organizaciones que eligieron ISO en lugar de CMMi. Sin embargo debemos hacer notar que la norma y el modelo resultan incomparables, ya que fueron pensados para objetivos diferentes.

En definitiva, CMMi es un modelo bien pensado por gente que conoce acerca del desarrollo de software, pero muy mal utilizado por una gran cantidad de personas que desconocen acerca de este tema.

#### 7.1.3.2 Con quién trabajar

¿Por qué deberíamos trabajar con el apoyo de un mentor externo y no con recursos propios? Nos interesa contestar esta pregunta a responsables de pymes porque son

estas empresas las que deben acudir a la ayuda externa. Una empresa de estas características no posee los recursos necesarios para contratar en forma permanente personas con conocimientos expertos en procesos de mejoras, por lo tanto la opción de contratar temporalmente a un mentor es la alternativa. Es difícil hacer una valoración de un grupo mentor que ayuda a las empresas en el camino a una evaluación exitosa y con resultados perdurables en el tiempo. Sin embargo, a la hora de contestar esta pregunta deberíamos por lo menos tratar de evitar lo siguiente:

### **Evite trabajar con mentores que...**

*Hayan nacido al desarrollo de software con el modelo* porque en general evidencia características mercantilistas además de la falta de experiencia. La tentación de obtener buenos honorarios hizo que nacieran consultoras especializadas en el modelo sin historia ni tradición en el negocio del desarrollo de software. Con miembros que hablan de procesos asociados al desarrollo sin haber escrito en su vida una sola línea de código.

*Toda su experiencia en el desarrollo de software se reduzca al modelo* porque el trabajo con este tipo de mentores genera resultados insuficientes. No tienen la capacidad para vincular las definiciones de procesos y procedimientos con las prácticas diarias de los desarrolladores. Se construye un proceso vacío de contenidos que la mayor parte de las veces no es asumido ni reconocido como propio por los miembros de su empresa ya que es percibido como excepcionalmente abstracto y de poca utilidad.

*No manifiesten un compromiso creíble con el proceso de mejoras* porque cuando llegue el momento de aconsejar y tomar decisiones que generen algún conflicto no lo harán. Porque no compartirán los objetivos del proceso de mejoras y en los momentos críticos adoptarán la postura de consultores externos.

*Enarbolen numerosos antecedentes de procesos exitosos de evaluación de empresas que luego terminaron como los mencionados*, porque evidencia una forma de trabajo que no genera resultados perdurables en el tiempo.

*No propongan un trabajo diario con los miembros de su organización* porque no entienden o no saben implementarlo. El acompañamiento diario es la única forma de acortar la distancia entre las definiciones y las prácticas diarias.

*No acrediten conocimientos actualizados de buenas prácticas de desarrollo de software* porque seguramente no puedan compartir el trabajo diario con los miembros de su organización.

*No manifisen y planifiquen una estrategia de administración de los cambios generados por la mejora de procesos* porque si descuidan estos aspectos fundamentales se retrasará el avance del proyecto y hasta quizás se cancele.

*Expresen de antemano una fecha certera de terminación del proyecto sin antes conocer a su empresa* porque tienen la visión de que todos los procesos son iguales, cuando siempre cada empresa es única y presenta sus propias fortalezas y debilidades. Porque seguramente no escucharán sino que solo asistirán a hablar del modelo y sus características sin importarles cómo este se debe adaptar a su empresa.

**Trabaje con mentores que...**

Acrediten experiencia en el desarrollo de software más allá del modelo porque aportarán contenido al proceso a definir.

Se muestren comprometidos con el proceso de mejoras porque seguramente llevarán el proyecto a buen término en forma compartida.

Enarbolen antecedentes de procesos realistas, algunos exitosos, otros con problemas, porque evidencia que ha habido un proceso de aprendizaje de los errores cometidos.

Propongan un trabajo diario con los miembros de su organización porque evidencia capacidad para acompañar a sus miembros a definir el proceso de desarrollo de su organización y darle contenido a partir del trabajo compartido.

Acrediten conocimientos actualizados de buenas prácticas de desarrollo de software porque se comunicarán de una mejor manera con los desarrolladores de su empresa y se facilitará la construcción de los vínculos entre el proceso y sus prácticas diarias.

Manifiesten la importancia de administrar los cambios generados por las mejoras y propongan una estrategia para ello porque evidencia conocimiento de los fenómenos no técnicos que deben ser tenidos en cuenta según tratamos en el capítulo dedicado a la mejora de procesos.

Expresen más que una fecha, un horizonte de tiempo que dependerá del estado de su organización y del trabajo de sus miembros porque evidencia la postura de adaptar el modelo a su empresa a partir de escuchar a sus miembros y conservar todas las buenas prácticas ya instaladas sin perjuicio de trabajar para incorporarlas al proceso.

### 7.1.3.3 Cómo es el proceso de mejoras con CMMI

**Estrategia**

- La estrategia para llevar adelante estos proyectos consiste en realizar un relevamiento de la empresa, desde sus procesos de preventa, generación de políticas de gestión, gestión, liderazgo, conformación de grupos de desarrollo y prácticas de desarrollo. Este relevamiento presta especial atención a las interacciones entre las dimensiones tecnológicas, organizacionales y humanas de la empresa. De esta manera, buscará explicitar los mecanismos cotidianos de la organización que se han hecho transparentes para sus miembros y permitirá prepararlos para los cambios que la siguiente etapa requiera.
- Elaborar un documento del estado de la empresa que contendrá los resultados del trabajo de relevamiento y que será utilizado como línea base para el proyecto.
- Exponer a todos los miembros de la organización los resultados, manifestando fortalezas y debilidades. Esto está basado en una cuestión clave: el proyecto debe ser compartido por todos los miembros de la organización. No es un proyecto impuesto por sus dirigentes, sino que es un proyecto al cual se invita a participar a todos. Todo aquel que crea poder aportar algo podrá

expresarlo y contribuir a este proyecto, y los mentores son los responsables de encausar esta dinámica.

- En base a los resultados anteriores se elaborará un plan de mejora que incluirá, entre otras cosas, un cronograma preciso con una distribución en el tiempo del trabajo a realizar, los objetivos parciales a alcanzar y la asignación de recursos humanos y materiales. Este plan será desarrollado por los mentores con la participación de miembros de la empresa.
- La empresa deberá asignar una persona (el impulsor) responsable del proyecto. Su dedicación variará con el tiempo, yendo de menor a mayor, cuando el proyecto avance. El perfil de esta persona debe ser de un *ex programador con experiencia, madurez y con acceso a todos los miembros de la organización*. Esta persona será la que lleve adelante los procesos una vez terminado el proyecto.
- Los mentores realizarán reuniones de exposición de los resultados del trabajo que serán anunciadas y programadas con el tiempo adecuado, y llevarán adelante el proyecto con una *forma de trabajo continua*, es decir con una dedicación parcial diaria de sus miembros.
- Cuando la etapa de definiciones haya sido superada y los miembros hayan sido entrenados en las prácticas derivadas de estas, se seleccionará un proyecto nuevo o en curso de la empresa y se aplicarán (Proyecto Piloto).
- La empresa formará un grupo (Engineering Process Group EPG) que será responsable de las definiciones de los procesos de la organización. Los mentores aconsejarán en la selección de los miembros de este grupo. Por el tamaño de la empresa (pyme), este grupo estará formado por el impulsor en forma permanente y por referentes de los distintos temas en forma temporal.
- En una última etapa se pondrán en práctica, en todos los proyectos de la empresa, el producto generado por el trabajo de los distintos grupos. Desde el primer día de trabajo los mentores generarán un ambiente de participación orientado a concientizar a todos los miembros de la empresa de que el proyecto es conducido por ella pero que es un proyecto en el que todos sus miembros tendrán una participación plena y activa. *Es decir que los miembros de la empresa deben adueñarse del proceso definido y probado.*

### **Metodología de trabajo**

Una metodología posible a seguir es la descripta en el capítulo “Mejora de Procesos” y está basada en el modelo IDEAL (Initiating, Diagnosing, Establishing, Acting and Learning). Las fases del proceso de mejora según este estándar son:

- **Inicio:** esta fase comienza con la aplicación de un SCAMPI (Standard CMMI Appraisal Method for Process Improvement) informal especialmente adaptado a la naturaleza y contexto de la empresa, y con el establecimiento de la línea base del proyecto de mejora en base a los resultados del mismo.
- **Diseño:** se escriben las políticas, los procesos, procedimientos y estándares.

- **Piloto:** se entrena a los participantes y se prueban los procedimientos en algunas áreas de proceso. Los procedimientos son adaptados, cuando sea necesario, basados en los resultados de la prueba piloto.
- **Implementación:** se extiende la implementación de los procedimientos a todos los proyectos de la organización y se mide su efectividad.

A continuación se detallan las actividades de cada una de las fases:

**Inicio:** en esta fase se relevan los procesos, tareas, actividades y activos con que cuenta la empresa, así como las políticas generadas por la conducción de la organización. Esto es llevado a cabo en las áreas de preventa, desarrollo y gestión. El resultado de este relevamiento se compara con los definidos por el modelo CMMI para los objetivos fijados por la empresa. En base a la *diferencia resultante* se determina el alcance del proyecto.

El método que CMMI propone para la realización de este relevamiento es el SCAMPI, el cual consiste en un conjunto estructurado de actividades tales como entrevistas, revisión de documentos, presentaciones y análisis de respuestas a cuestionarios. iT-Mentor ha desarrollado adaptaciones especiales de las recomendaciones de SCAMPI para las características y el contexto de empresas argentinas.

**Diseño:** basados en las debilidades, fortalezas y oportunidades de mejora encontradas en el SCAMPI se elabora el *Plan de Implementación de Mejoras* (PI) y los *Planes de Acción* (PAs). En el PI se define la estructura del Engineering Process Group (EPG) responsable de la definición de los procesos de la organización; de los Process Action Team (PAT) los cuales llevan a cabo los PAs; se define y documenta la forma de trabajo de los mismos; se planea el entrenamiento necesario para todos los involucrados en las actividades mencionadas en los planes; se determina el alcance de la capacitación en aspectos generales del CMMI; se planifica la realización de los proyectos pilotos y la estrategia a seguir para la implementación final. En los PAs se asignan objetivos, cronogramas, personas y actividades a desarrollar.

En esta fase se *trabaja a nivel de procesos* realizándose sus *definiciones* en base a los pasos que los componen, los procedimientos que definen estos pasos, los estándares y otros activos a generar. Se definen las métricas a utilizar para las distintas áreas de proceso. Se implementa la capacitación. Se construye un repositorio donde se instalan todos los productos generados en el proyecto. Se trabaja con los líderes de proyecto, analistas y desarrolladores seniors. Se comunican los resultados a todos los miembros.

**Piloto:** de acuerdo a los objetivos planteados en cada PATs y al producto resultante de su trabajo (proceso, tarea, actividad, estándares), se capacita a los miembros del grupo del proyecto piloto y se prueban las prácticas correspondientes. Se da soporte a todos los miembros del grupo de desarrollo para la implementación del proyecto en el marco definido. En base al análisis de los resultados obtenidos en esta ejecución, se realizan las modificaciones correspondientes y se *institucionaliza* el producto resultante.

**Implementación:** en esta fase, la empresa institucionalizará el producto resultante del trabajo realizado con los proyectos pilotos en el resto de sus proyectos.

Si bien los mentores pueden acompañar a la empresa en esta tarea, la experiencia indica que una vez desarrollado los pilotos es una buena estrategia que las empresas realicen esta tarea por sí mismas para terminar de “adueñarse” de los nuevos procesos sin depender de apoyo externo.

Un proceso de mejoras de software que finalice con una evaluación exitosa CMMI nivel 2 de una organización depende de su tamaño, de los recursos disponibles, la capacidad y madurez de sus miembros y de la organización, y de las oportunidades de mejora presentes en la organización. La experiencia indica que su duración puede variar entre diez y catorce meses. La misma empresa deberá invertir alrededor de doce meses adicionales para ser evaluada al nivel 3 de CMMI.

En la figura 7.6 se muestra un esquema del organigrama típico de una empresa o área de desarrollo y en la figura 7.7 el mismo organigrama complementado con el correspondiente al proyecto de mejora de procesos.

## Organigrama de la Organización

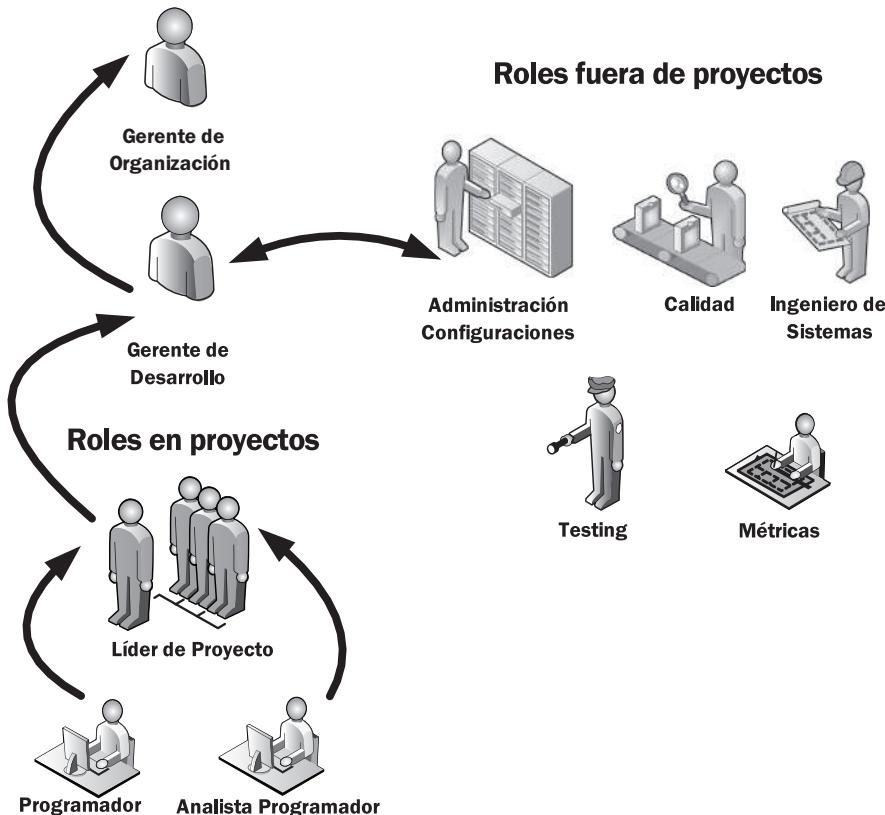


Fig. 7.6 - Organigrama de una organización típica de desarrollo de software

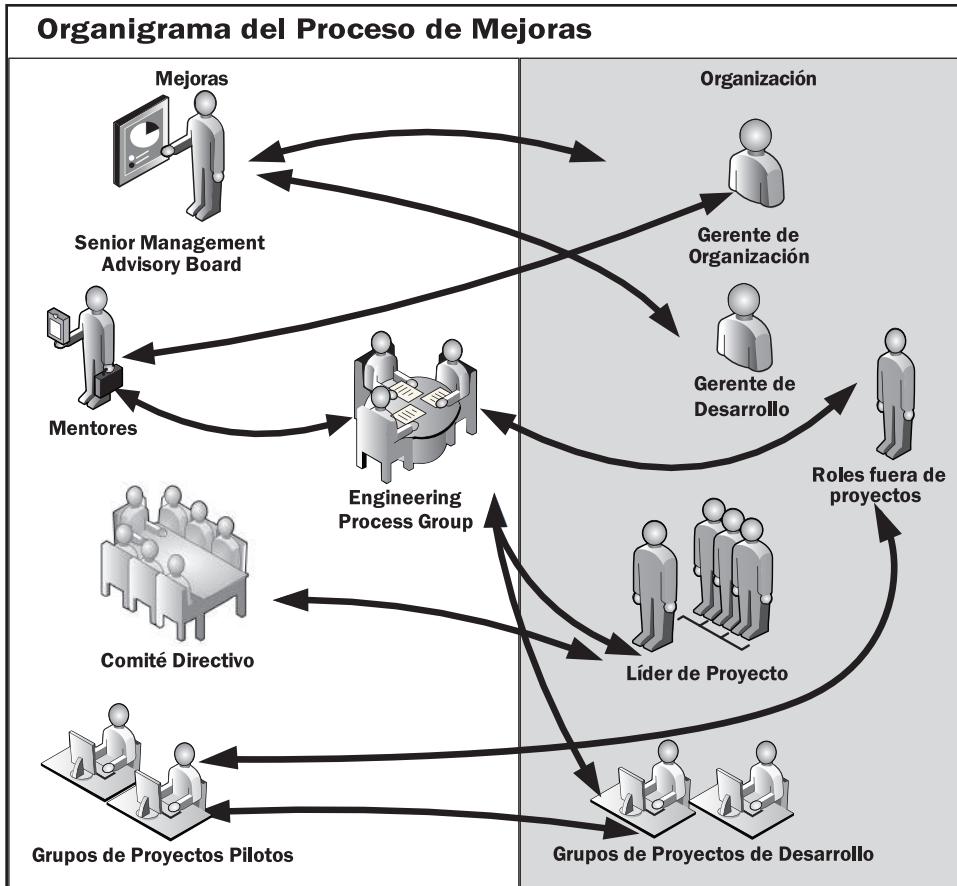


Fig. 7.7 - Organigrama del proyecto de mejora de procesos

#### 7.1.3.4 Qué recursos se necesitan

Es importante responder esta pregunta ya que muchos responsables de empresas deben elaborar un presupuesto para el proyecto de mejora de procesos y deben incluir tanto honorarios de profesionales como costo de herramientas, capacitaciones e infraestructura.

Mayoritariamente los recursos a invertir corresponden a honorarios profesionales de miembros propios y externos. Eso, como propondremos, si las herramientas a utilizar son open source y free, las cuales hay de excelente calidad.

Respecto de los recursos humanos es fundamental la participación con dedicación exclusiva de un miembro de la empresa que llamaremos el *impulsor* y que será el responsable del proyecto. Esta persona debe contar con experiencia en desarrollo de software y además poseer atributos de liderazgo. Si la empresa no cuenta con esta persona deberá contratarla; y debido a que será la que al término

del proyecto habrá participado en todas las definiciones del proceso de desarrollo, es una práctica común que permanezca como miembro de la organización, tal vez como responsable del SPEG (Software Process Engineering Group). Este grupo, en una pyme, puede estar formado por esta persona en forma permanente y por miembros temporales referentes de distintos aspectos que hacen a proyectos de desarrollo como requerimientos, planificación, configuración, calidad, diseño y codificación. Este organismo, que se irá formando a medida que el proyecto de mejoras avance, es el que promueve los cambios, los analiza y elabora propuestas. La idea es que participen los mejores recursos de cada especialidad.

Otro rol que muchas veces las empresas no tienen cubierto y con el que deberán contar a partir de la mejora de procesos con CMMi es el *responsable de calidad*. Es importante que la persona que ocupe dicho rol participe de la definición de los procesos y procedimientos, razón por la cual deberá ser asignada al comienzo del proyecto. Este es otro recurso a tener en cuenta a la hora de elaborar el presupuesto. A veces el mismo impulsor ocupa este rol al término del proyecto, una vez obtenida la evaluación. El modelo es muy demandante de documentación en lo referente a activos (estándares, guías de uso, etc.), definiciones de políticas, procesos y procedimientos. Para este trabajo es de gran importancia contar con una persona que posea *muy buenos conocimientos y experiencia en herramientas de tipo office, elaboración de páginas web, etc.* Este recurso es de participación intensiva en todo el proyecto, pero su dedicación es menor una vez alcanzada la evaluación.

Además de lo mencionado, *todos los miembros de la organización tendrán una participación con distinta dedicación* en el proyecto que debe tenerse en cuenta en el presupuesto, ya que el tiempo asignado a las mejoras es tiempo que dejarán de producir en un proyecto de desarrollo de la empresa.

Otro recurso a tener en cuenta es la participación de los mentores externos la cual dependerá de la forma de trabajo definida.

En la tabla 7.4 se muestra una asignación típica de recursos humanos a un proyecto de mejoras que puede servir de guía para la planificación. Un tratamiento claro de los roles y las responsabilidades asociadas lo podemos encontrar en *Real Process Improvement Using the CMMI*<sup>60</sup>.

Recursos asignados	Fases del proceso de mejora		
	Inicio	Diseño	Pilotos
Senior Mentor	1 ½ [P]	1 ½ [P]	1 ½ [P]
Gerente Empresa	1[M]	1[M]	1[M]
Senior Empresa (Impulsor)	1[S]	1[T]	1[T]
Desarrolladores miembros de cada PAT de Empresa	--	1[P]	1[P]
Desarrolladores de Empresa	Todos [M]	Todos [M]	Todos [M]
Documentadores de Empresa	--	1[P]	1[S]

**Tabla 7.4** - Asignación de recursos para el proyecto de mejora de procesos

60 | Michael West. Auerbach Publications. *Real Process Improvement Using the CMMI*. 2004.

**Referencia de la dedicación:**

- [T] total, dedicación exclusiva al proyecto (8 horas diarias)
- [P] parcial, dedicación semiexclusiva al proyecto (4 horas diarias)
- [S] simple, dedicación colaborativa al proyecto (8 horas semanales)
- [M] mínima, dedicación informativa al proyecto (1/4 hora diaria en promedio)

**7.1.3.5 Cómo es la evaluación con CMMI (SCAMPI)**

*El proceso de evaluación del modelo se denomina SCAMPI (Standard CMMI Appraisal Method for Process Improvement). El mismo está compuesto por tres fases que en general se extienden por unos quince días. En el siguiente listado se presenta una partición de las tareas que se llevan adelante en cada fase.*

- **Fase de planear y prepararse para el Appraisal**
  - Analizar los requerimientos
  - Desarrollar un Plan
  - Seleccionar y preparar al Team
  - Obtener y analizar evidencia objetiva inicial
  - Prepararse para recolectar evidencia objetiva
- **Fase de conducir el Appraisal**
  - Examinar evidencia objetiva
  - Verificar y validar evidencia objetiva caracterizando las prácticas
  - Documentar evidencia objetiva
  - Generar resultados calificando a la Unidad Organizacional (UO)
- **Fase de informar resultados del Appraisal**
  - Publicar resultados del Appraisal
  - Empaquetar y archivar activos del Appraisal

Los aspectos salientes son:

- Un Lead Appraisal (Conductor de la Evaluación) conduce el SCAMPI con la ayuda de un team compuesto de cinco a seis personas dependiendo del tamaño de la organización a evaluar.
- Se seleccionan entre tres y cinco proyectos a evaluar del grupo de proyectos de la organización.
- Para estos se buscará evidencia objetiva directa (activos), indirecta (mails, minutas) y afirmaciones (entrevistas con los miembros) en relación con las prácticas de los diferentes objetivos de las áreas de proceso en evaluación.
- Se caracterizan las diferentes prácticas para los proyectos y luego se consolidan para la Unidad Organizacional (UO) (empresa de desarrollo o área de desarrollo de una empresa).

- Se califica la evaluación de acuerdo a la implementación de las prácticas en la UO.
- Se publican los resultados del SCAMPI haciendo recomendaciones de mejoras basadas sobre las fortalezas y debilidades encontradas.

En las figuras 7.8, 7.9, 7.10 y 7.11 se muestra conceptualmente la realización de estas tareas<sup>61</sup>.

En la primera de ellas, figura 7.8, se presenta la revisión de la evidencia objetiva en todas las formas ya mencionadas para cada uno de los proyectos en relación con los objetivos y prácticas del modelo.

De acuerdo al grado de cumplimiento de las prácticas de cada objetivo de cada una de las áreas de proceso, a estas se las caracteriza como Fully Implemented (FI), Largely Implemented (LI), Partially Implemented (PI), Not Implemented (NI) o Not Yet (NY). Este proceso de caracterización de las prácticas se muestra en la figura 7.9, en la que pueden verse también aspectos de los criterios de caracterización.

La información así elaborada se consolida para todos los proyectos y se obtiene la caracterización de las prácticas para la UO. Este proceso está expuesto en la figura 7.10.

En la figura 7.11 se señala el proceso de calificación de la UO en función de la caracterización de las prácticas resultante de los procesos descriptos más arriba.

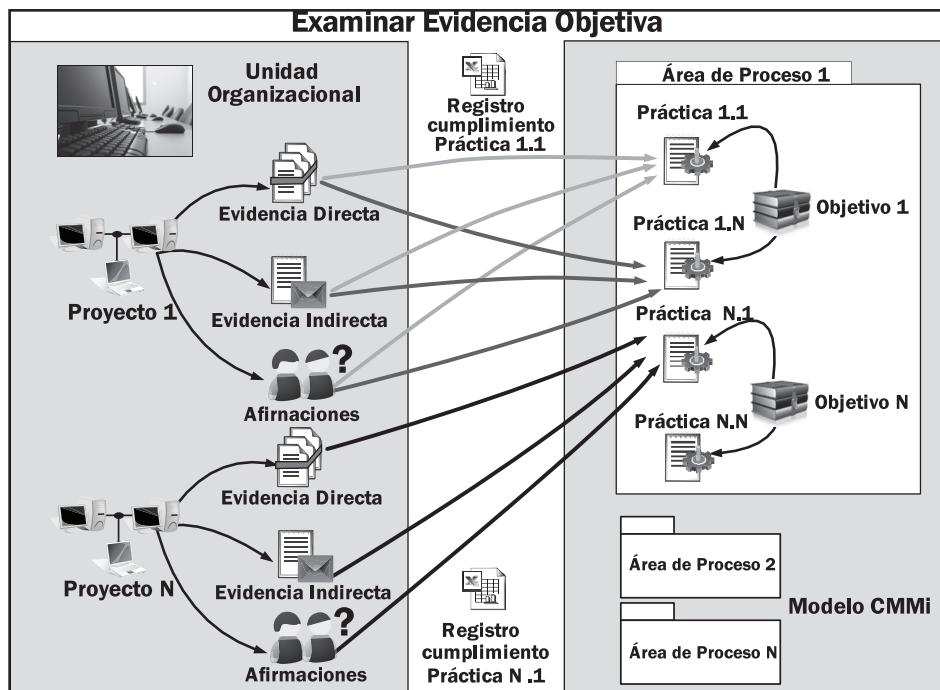
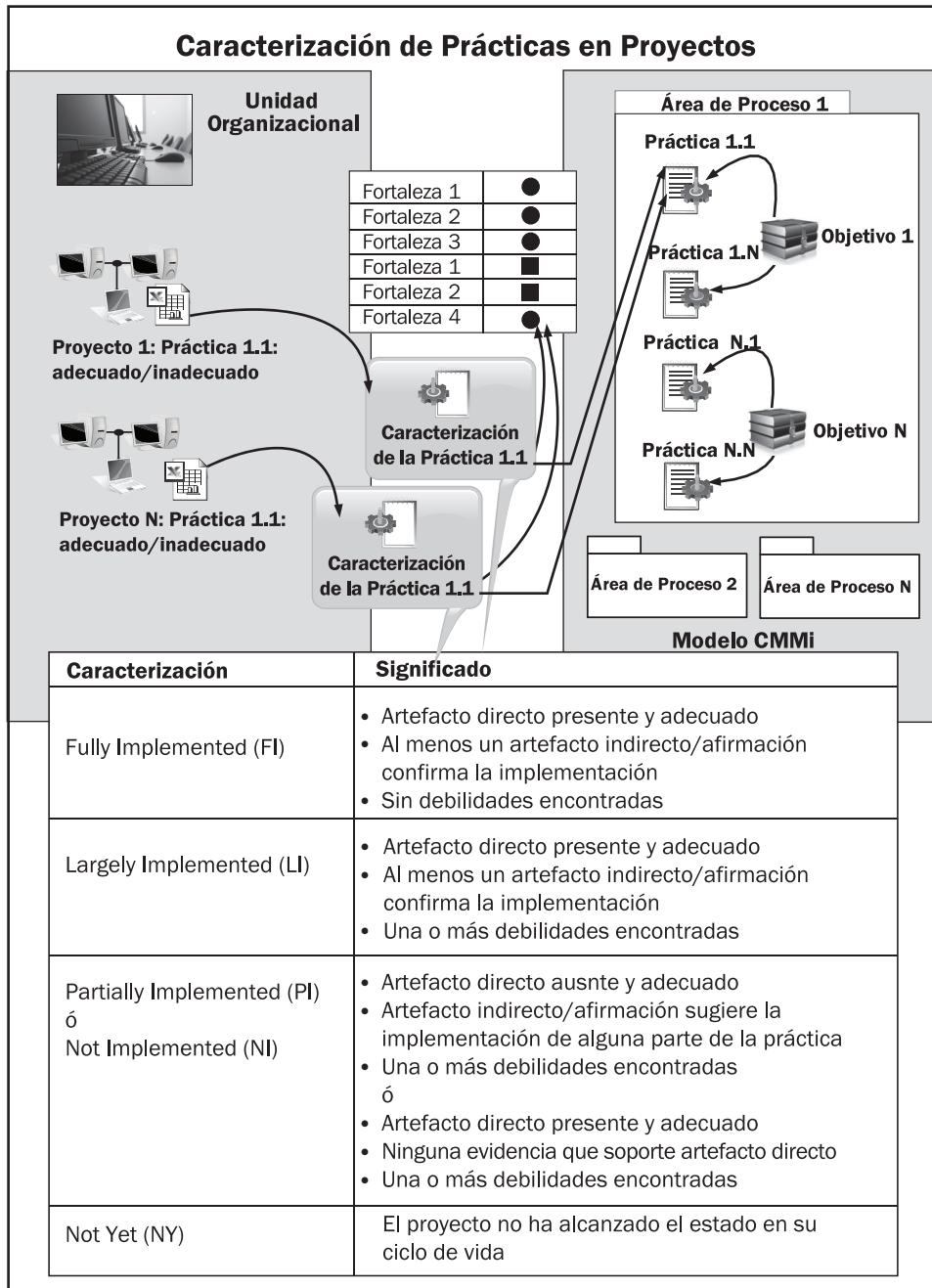
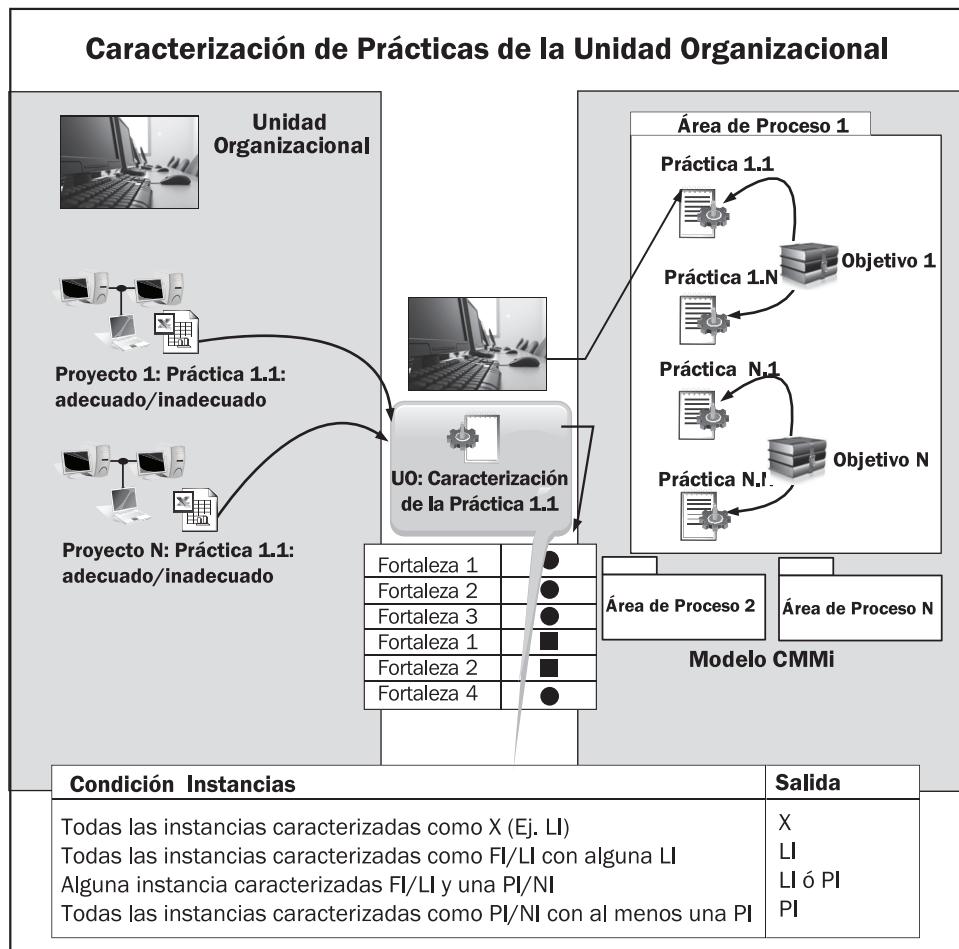


Fig. 7.8 - Examinar evidencia objetiva para cada uno de los proyectos seleccionados

61 | Modelo elaborado en it-Mentor. Nota del autor.



**Fig. 7.9** - Caracterizar prácticas de los proyectos según la evidencia encontrada



**Fig. 7.10** - Caracterizar prácticas de la Unidad Organizacional según la evidencia de los proyectos

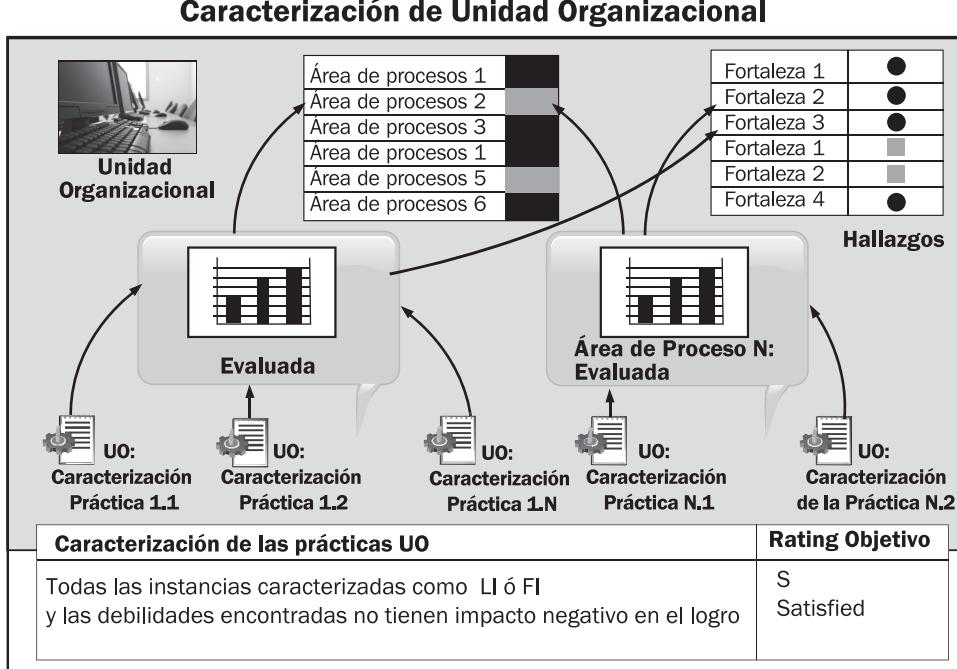


Fig. 7.11 - Calificar a la Unidad Organizacional

En Standard CMMI Appraisal Method for Process Improvement (SCAMPI)<sup>62</sup>se puede consultar información detallada del proceso descripto. Con los cambios introducidos al modelo en su versión 1.2 se estableció que las evaluaciones deben ser periódicas, cada dos años.

## 7.2 | MEJORA DE PROCESOS UTILIZANDO EL MODELO CMMI

En esta sección se exponen las bases del criterio que usamos al mapear un proceso de desarrollo con el modelo CMMI. Se presenta la estrategia general y algunas tácticas orientadas a prácticas puntuales que pueden ser extrapoladas a otras no expuestas en este texto. El contexto en el que se desarrollan las siguientes secciones es el de una empresa que trabaja con un proceso de desarrollo basado en una metodología conducida por planes, con una forma de trabajo iterativa, implementada en forma ágil y tomando algunas buenas prácticas de metodologías ágiles y de XP (Extreme Programming) y su mapeo con el nivel 2 de madurez del modelo CMMI. En la figura 7.12 se muestra conceptualmente la relación que estableceremos<sup>63</sup>.

62 | Standard CMMI Appraisal Method for Process Improvement (SCAMPI), Version 1.1: Method Definition Document, Members of the Assessment Method Integrated Team, HANDBOOK CMU/SEI-2001-HB-001, 2001.

63 | Modelo elaborado en it-Mentor. Nota del autor.

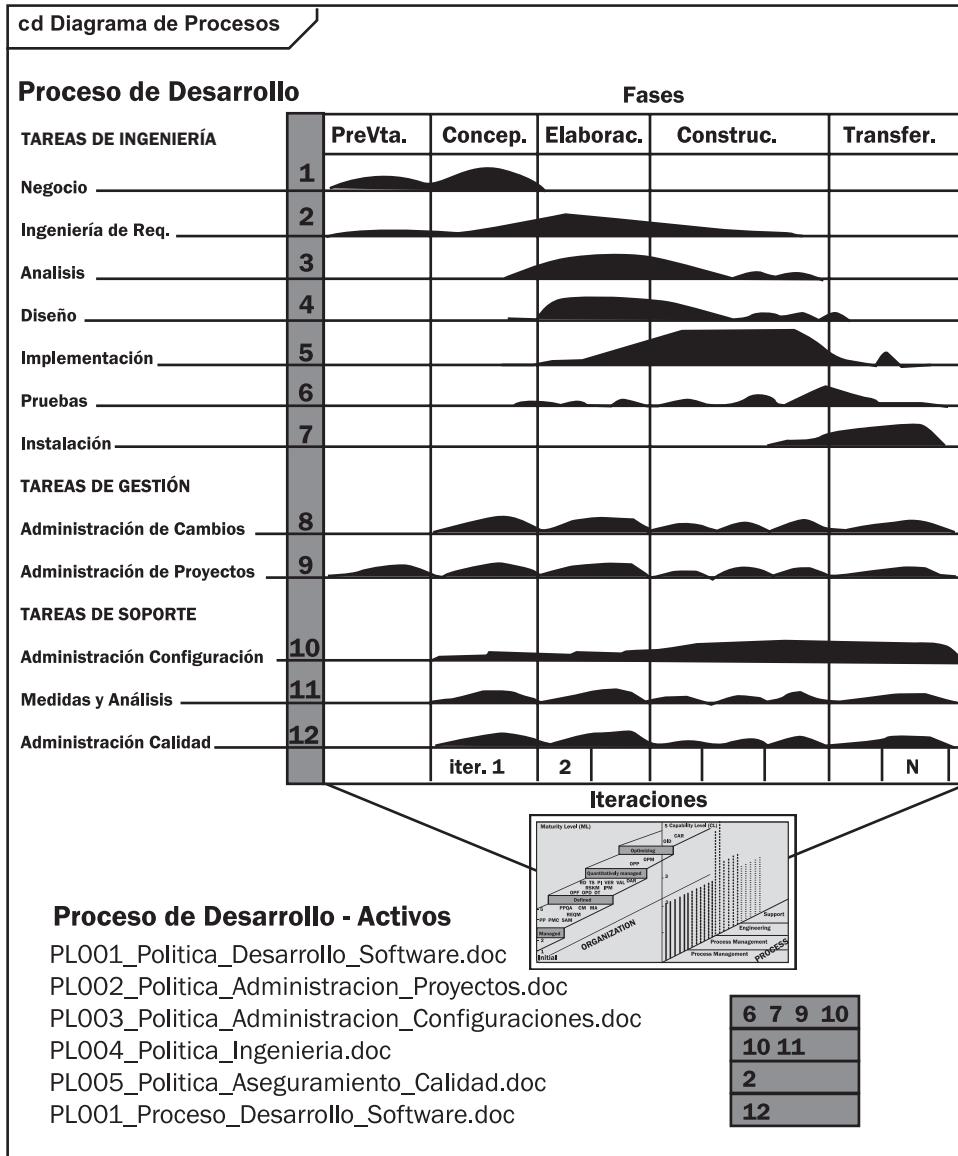


Fig. 7.12 - Relación entre el proceso de desarrollo y el modelo CMMI

### 7.2.1 | ESTRATEGIA GENERAL

Pueden leerse algunas guías acerca de la implementación de mejoras utilizando el modelo en: CMMI, Guidelines for Process Integration and Product Improvement (2003), de M. B. Chrissis, M. Konrad y S. Shrum. También CMMI ® Distilled: A Practical

Introduction to Integrated Process Improvement, Second Edition<sup>64</sup>; Interpreting the CMMI: A Process Improvement Approach<sup>65</sup>; y Practical Insight into CMMI<sup>66</sup>.

### 7.2.1.1 Políticas y Procesos

A efectos de definir las políticas demandadas por el modelo en una empresa dedicada al desarrollo de software es necesario explicitar claramente cuál será el proceso de desarrollo seleccionado y cuáles serán las definiciones asociadas a los aspectos generales en relación con este negocio.

En la figura anterior se observa que se ha decidido contar con las siguientes políticas:

1. Desarrollo de Software
2. Administración de Proyectos
3. Administración de Configuraciones
4. Ingeniería
5. Aseguramiento de la Calidad

De esta manera se ha definido el proceso de desarrollo que la empresa seguirá en el desarrollo de sus proyectos.

Estas definiciones apuntan a cumplir con el objetivo genérico (GG 2: The process is institutionalized as a managed process) y su práctica genérica (GP 2.1 Establish an Organizational Policy) cuyas relaciones se muestran en la figura 7.11.

Cada una de estas políticas define su propósito, *alcance*, *responsables* y la forma de verificación de cumplimiento.

Así, la primera de ellas establece la decisión de trabajar con el proceso de desarrollo definido y en el marco del modelo CMMI nivel 2 (*propósito*). Su *alcance* incluye los tipos de proyectos y el área de desarrollo en la cual la política es válida. Se establecen al Gerente de Desarrollo, a los Líderes de Proyectos y al EPG como los *responsables* de hacer que los proyectos de desarrollo se lleven adelante dentro de lo establecido por estas políticas y al área de Aseguramiento de la Calidad la *verificación del cumplimiento* de políticas y procesos.

Cada una de las restantes políticas incluye definiciones puntuales referidas a aspectos vinculados al desarrollo tales como la determinación de administrar los proyectos de una forma definida, administrar configuraciones, desarrollar la ingeniería de los proyectos de una forma definida y establecer un área de aseguramiento de la calidad con funciones definidas.

64| Dennis M. Ahern. Aaron Clouse. Richard Turner. *CMMI, Guidelines for Process Integration and Product Improvement* (2003), de M. B. Chrissis, M. Konrad y S. Shrum. También *CMMI ® Distilled: A Practical Introduction to Integrated Process Improvement*. Addison Wesley. 2<sup>nd</sup> Edition. 2003.

65| Margaret K. Kulpa & Kent A. Johnson. *Interpreting the CMMI: A Process Improvement Approach*. Auerbach Publications. 2003.

66| Tim Kasse. *Practical Insight into CMMI*. Artech House. 2004.

El proceso de desarrollo establece las fases de su ciclo de vida, su forma iterativa de trabajo, la definición de sus roles y responsabilidades, la dinámica de sus iteraciones, la forma de acordar alcance y prioridades, cómo se utilizarán los riesgos y estimaciones de esfuerzo en la construcción de una planificación. Cuáles serán su entradas y salidas; cuáles serán las condiciones que determinarán su comienzo y finalización y cómo se verificará y validará su desarrollo. En el apéndice A se presenta un template orientado a la construcción de procesos y procedimientos.

### 7.2.1.2 Interpretación y mapeo de objetivos y tareas

En relación a las áreas de proceso y a sus objetivos y prácticas específicas presentaremos dos ejemplos. Uno orientado a mapear un conjunto de tareas bien conocidas y presentadas en el capítulo “Trabajo con Requerimientos” y otro a dar forma a un área nueva, “Aseguramiento de la Calidad”, como seguramente lo es en una empresa que no trabaja en el marco del modelo CMMi.

#### Trabajo con requerimientos como un conjunto de tareas a mapear

Nos guiarímos por la matriz inspirada en la visión extendida de Zachman que utilizamos en el capítulo cuatro de Trabajo con Requerimientos. Las tareas que se llevan adelante en los distintos niveles para desarrollar los productos según los diferentes enfoques y los objetivos que se buscan lograr son:

- **Nivel Intención:** Analizar el problema para achicar el riesgo del no conocimiento del negocio y entender las necesidades de los usuarios.
- **Nivel Alcance:** Administrar el alcance del sistema a desarrollar para establecer las prioridades del negocio y acordar el alcance del proyecto y la planificación.
- **Nivel Alto Nivel:** Definir el Sistema para establecer relaciones y prioridades del proyecto que ayuden a la elaboración de la planificación.
- **Nivel Detalle:** Refinar el Sistema para realizar validaciones.

Una tarea que no está expresada en la tabla 7.5 es la *administración de cambios a los requerimientos* que fue tratada como esencial. Por eso la agregamos a la tabla que sigue en la que se muestra el mapeo entre las tareas reconocidas como necesarias para trabajar con los requerimientos y las prácticas expresadas por el modelo.

En la tabla 7.5 el mapeo se indica con flechas, desde las buenas prácticas hacia la prácticas del modelo. Se observa que la tarea de Refinar el Sistema, marcada con un adorno, la cual involucra la especificación de casos de uso, la asignación de reglas del negocio a clases del modelo de dominio y las pruebas de validación asociadas, no mapea a ninguna práctica. Estas tareas corresponden a ingeniería de requerimientos y serán mapeadas cuando se trabaje en el nivel 3 del modelo.

También hay una tarea marcada con un adorno, Planear el Trabajo con requerimientos que no mapea a una práctica específica, sino que está implementada a efectos de satisfacer el objetivo genérico GG1 y mapea a sus prácticas genéricas GP 2.2 Plan the Process y GP 2.3 Provide Resources.

Modelo CMMI - Requirements Management		Proceso de Desarrollo	
Objetivos Específico	Prácticas Específicas	Tareas del Trabajo con Requerimientos	Objetivos
SG1 Managed Requirements	SP1.1 Obtain an understanding of requirements <b>a b</b>	<b>a</b> Analizar el problema	Achicar el riesgo del no conocimiento del negocio y validar los requerimientos
	SP1.2 Obtain commitment to requirements <b>d</b>	<b>b</b> Entender las necesidades de los usuarios	
	SP1.3 Manage requirements changes <b>f</b>	<b>c</b> Definir el problema	Establecer relaciones y prioridades del proyecto que ayuden a la elaboración de la planificación
	SP1.4 Maintain bidirectional traceability of requirements. <b>c</b>	<b>d</b> Administrar el alcance del sistema	Acordar el alcance del sistema en desarrollo y establecer prioridades
	SP1.5 Identify inconsistencies between project work and requirements <b>d</b>	<b>e</b> Refinar el sistema ●	Realizar validaciones
		<b>f</b> Administra cambios a los requerimientos	Evitar inestabilidad en el proyecto
		<b>g</b> Planear el trabajo ● con requerimientos	Proveer y administrar recursos

**Tabla 7.5** - Mapeo de las tareas del trabajo con requerimientos del proceso de desarrollo y las prácticas del área de proceso de CMMI nivel 2 Requirements Management

Para esta área de procesos definiremos los siguientes activos o productos de trabajo:

- **ERS** (Especificación de Requerimientos de Software)
- **Procedimiento de especificación de requerimientos** (orientado a la construcción de la ERS)

La ERS fue descripta en el capítulo de Trabajo con Requerimientos y el procedimiento se presenta en los apéndices.

#### **Aseguramiento de la Calidad como un conjunto de nuevas tareas a llevar adelante**

En la siguiente tabla se muestran las tareas implementadas como parte del proceso de desarrollo y cómo estas mapean a las prácticas específicas del área de procesos Process and Product Quality Assurance. El fin de las tareas de las celdas marcadas con un adorno y que no mapean a ninguna práctica específica del modelo es satisfacer el objetivo genérico GG1 y mapean a sus prácticas genéricas GP 2.2 Plan the Process y GP 2.3 Provide Resources. En la sección Institucionalización, de este capítulo, se trata el tema de los objetivos y prácticas genéricas.

Modelo CMMI - Process and Product Quality Assurance		Proceso de Desarrollo	
Objetivos Específico	Prácticas Específicas	Tareas de aseguramiento de la calidad	Objetivos
SG1 Objetively evaluate processes and products	SP1.1 Objetively evaluate processes <span style="background-color: #cccccc; padding: 2px;">d</span>	<span style="background-color: #cccccc; padding: 2px;">a</span> Planear tareas de aseguramiento de la calidad	Definir criterios de calidad para el proyecto de desarrollo y complementar la planificación del proyecto insertando las revisiones y auditorías en su cronograma
	SP1.2 Objetively evaluate work products and services <span style="background-color: #cccccc; padding: 2px;">d</span>		
SG2 Provide objective insight	SP2.1 Establish records <span style="background-color: #cccccc; padding: 2px;">c</span>	<span style="background-color: #cccccc; padding: 2px;">b</span> Crear ambiente de aseguramiento de la calidad	Proveer la infraestructura y otros recursos
	SP2.2 Communicate and ensure resolution of noncompliance issues <span style="background-color: #cccccc; padding: 2px;">d</span>	<span style="background-color: #cccccc; padding: 2px;">c</span> Ejecutar tareas de aseguramiento de la calidad	Realizar revisiones, auditorías y ejecutar checklist registrando resultados
		<span style="background-color: #cccccc; padding: 2px;">d</span> Informar y colaborar en el análisis de los resultados	Garantizar el análisis y corrección de no conformidades

**Tabla 7.6** - Mapeo de las tareas del trabajo con la calidad del proceso de desarrollo y las prácticas del área de proceso de CMMI nivel 2 Product and Process Quality Assurance

Para esta área de procesos definiremos los siguientes activos o productos de trabajo:

### Procedimientos

- Planear aseguramiento de la calidad
- Crear ambiente de aseguramiento de la calidad
- Ejecutar tareas de aseguramiento de la calidad

### Activos

- Plan de aseguramiento de la calidad.
- Estándares de programación.
- Matriz de seguimiento de no conformidades.

### Checklists de procesos

- Fase Preventa
- Fase Concepción
- Fase Elaboración

- Fase Construcción
- Fase Transferencia
- Proceso de aseguramiento de la calidad (PPQA)
- Administración de la configuración
- Implementación de medidas y análisis
- Administración de proveedores
- Iteración

### **Checklists de productos**

- ERS
- Plan de Proyecto
- Plan de QA (Aseguramiento de la calidad)
- Plan de CM (Administración de la configuración)
- Plan de MA (Medidas y análisis)
- Cronograma del proyecto
- Plan de Iteración

#### **7.2.1 | INSTITUCIONALIZACIÓN**

Se entiende por institucionalización “la forma establecida que una organización sigue de forma rutinaria para hacer negocios y que es parte de su cultura corporativa”. En las evaluaciones CMMI, la institucionalización es juzgada por el logro de los objetivos genéricos en el nivel de capacidad o nivel de madurez adecuado. Los procesos definidos deben institucionalizarse con diferente grado según el nivel del modelo que se trate.

En particular, en la vista continua del modelo, para un área de proceso determinada, el nivel de capacidad logrado depende del grado de institucionalización, como se ve en la tabla 7.2.

##### **7.2.1.1 Relación entre áreas de proceso y objetivos genéricos**

Cada una de las áreas de proceso debe contribuir a la institucionalización del trabajo realizado en ellas como lo especifican las prácticas genéricas. En la tabla 7.7 se muestra cómo esto es implementado para el trabajo con los requerimientos en el área de proceso Requirements Management.

Modelo CMMI nivel 2		Proceso de Desarrollo	
Objetivos Genéricos	Prácticas Genéricas	Tareas de Trabajo con Requerimientos	Objetivos
<b>GG 2: The process is institutionalized as a managed process</b>	GP 2.1 Establish an Organizational Policy	Incluir en la política de desarrollo de software las actividades del trabajo con requerimientos	Dar al trabajo con requerimientos el espacio y soporte necesarios
	GP 2.2 Plan the Process	Para cada proyecto se planificará e implementará el proceso de trabajo con requerimientos (capítulo 4) adaptándolo a las características del proyecto	Como parte de las tareas del proyecto de desarrollo se coordinará y gestionará
	GP 2.3 Provide Resources	Para cada proyecto se gestionarán y administrarán los recursos necesarios	Contar con personas, activos e infraestructura para trabajar con los requerimientos
	GP 2.4 Assign Responsibility	Uno de los miembros del grupo de desarrollo desarrollará el rol de analista	Contar con la doble visión definida en el capítulo 4 acerca de la interfaz de negocio y del desarrollo de este rol
	GP 2.5 Train People	Los miembros del grupo de desarrollo recibirán capacitaciones actualizadas acerca de técnicas y métodos de relevamiento y análisis	Contar con analistas que faciliten el trabajo con requerimientos a partir de habilidades y conocimientos
	GP 2.6 Manage Configurations	Todos los activos y documentos generados por el trabajo con requerimientos serán identificados como ítems de configuración y versionados en el repositorio del proyecto	Administrar líneas base de requerimientos y versionado de todos los activos del proyecto

	GP 2.7 Identify and Involve Relevant Stakeholders	<p>Para un mejor conocimiento del negocio se mantendrá un vínculo estrecho entre los requerimientos y sus solicitantes, usuarios, conocedores y aprobadores, como parte del trabajo</p>	<p>Contar con información de primera mano y responsables de decisiones</p>
	GP 2.8 Monitor and Control the Process	<p>Como se definió en el capítulo 5, se realizará un seguimiento, una evaluación y corrección del trabajo como parte de la gestión del proyecto utilizando las métricas definidas</p>	<p>Generar visibilidad, promover acuerdos y gestionar problemas</p>
	GP 2.9 Objectively Evaluate Adherence	<p>Para la forma de trabajo definida se evaluará consistencia entre lo realizado y lo definido, como parte del aseguramiento de la calidad de procesos y productos utilizando métricas sobre los proyectos</p>	<p>Generar patrones de trabajo y comportamiento</p>
	GP 2.10 Review Status with Higher Level Management	<p>Analizar medidas e informes. Implementar acciones correctivas</p>	<p>Informar a las gerencias y tomar acciones con autoridad</p>

**Tabla 7.7** - Intitucionalización de las actividades del trabajo con los requerimientos

## 7.3 | MODELOS Y METODOLOGÍAS

### 7.3.1 | METODOLOGÍAS Y MODELOS

La confusión entre metodologías y modelos es una de las causas frecuentes de fracaso de proyectos de mejora de procesos en organizaciones de desarrollo de software.

Una metodología es una forma de trabajo organizada para un grupo de desarrollo en base a la dinámica de las actividades de los diferentes roles. Establece qué, cuándo y cómo realizan sus tareas y cómo utilizan los demás roles los productos generados por ellos.

Un modelo de referencia como CMMI establece áreas de proceso con objetivos y prácticas a efectos de calificar y comparar grupos de desarrollo. Ya hemos mencionado

la ambigüedad con que estos modelos definen subprácticas y activos de trabajo. Dicen qué, pero no les interesa cómo.

Esta falta de precisión acerca de la implementación del modelo es lo que genera que sea necesaria una experiencia importante en los diferentes aspectos que hacen al desarrollo para interpretar e implementar adecuadamente dicho modelo. Las organizaciones deben seleccionar con cuidado a las personas que llevarán adelante estos proyectos, de lo contrario corren el riesgo de terminar con la implementación de procesos vacíos de contenido.

Parte de la confusión que estamos analizando se extiende a los roles de las organizaciones y afecta no solo a la forma de trabajo sino a las responsabilidades y actividades de sus miembros. Por alguna razón es una práctica común en el mercado argentino de desarrollo de software confundir calidad con testing. Esto hace que los roles asociados sean ocupados por personas que no cuentan con experiencia de desarrollo, sino de cómo realizar pruebas. Es decir, los roles centrales en la mejora de procesos son ocupados por profesionales que se han dedicado buena parte de su vida profesional a contar errores en lugar de trabajar para evitarlos. Garantizar la calidad de los productos de software consiste en generar acciones orientadas a evitar la presencia de errores a partir de la promoción, capacitación y soporte de buenas prácticas durante el proceso de desarrollo. Por esta razón es fundamental una buena selección de los miembros de la organización para los roles del proyecto de mejora de procesos, los cuales deben contar con experiencia en desarrollo y ser conocedores de distintas formas de trabajo.

### 7.3.2 | CMMI Y METODOLOGÍAS

Por las razones que ya analizamos, la implementación del modelo CMMI requiere una definición precisa de la forma de trabajo plasmada en alguna forma de documentación y la recolección y administración de evidencia que muestre la consistencia entre las definiciones previas y lo realizado en los proyectos.

Por esta razón en una primera aproximación parecen ser más apropiadas las metodologías conducidas por los planes para la implementación del modelo más que otras centradas en las personas, es decir las conocidas como “ágiles”. En el apéndice A me refiero a la comparación entre estas formas de trabajo. En mi opinión algunas de las metodologías ágiles, en esencia, se plantean objetivos e implementan prácticas que garantizan la calidad de los productos generados y por lo tanto dan contenido a la implementación del modelo. Sin embargo no está en su espíritu la predefinición documentada de su forma de trabajo ni la recolección de evidencia del modo en que el modelo lo requiere. Por esta razón los grupos que dicen practicar alguna de estas metodologías y con el fin de implementar el modelo adaptan su forma de trabajo, dejan de ser ágiles. Lo importante no es aferrarse a una metodología sino desarrollar el criterio para su adaptación, aunque sigo pensando que es importante ser conscientes, para aquellos que eligieron ser ágiles, que al momento de implementar el modelo dejarán de serlo.

Personalmente me ha tocado implementar una metodología conducida por los planes con la inclusión de varias prácticas de algunas metodologías ágiles con

muy buenos resultados. Recomiendo considerar la forma iterativa de trabajo, la programación de a pares, la integración continua y la asignación de roles de acuerdo al momento del ciclo de vida del proyecto en el grupo de desarrollo como excelentes prácticas para cualquier forma de trabajo seleccionada.

## 7.4 | MADUREZ

Un aspecto importante de la mejora de procesos que utiliza el modelo es la madurez alcanzada en los procesos definidos por una organización. La madurez significa, según el diccionario de la lengua castellana, “plenitud, esplendor, nivel de desarrollo” y se dice de alguien maduro que es “desarrollado, completo, formado”. El modelo promueve el alcance de este estado de madurez y es una condición esencial para lograr los objetivos. Sin embargo la perfección no existe, tampoco en este lado del mundo, ya que los casos que personalmente conozco y en los que me ha tocado participar son mayoritariamente de empresas que lograron ser evaluadas con resultado positivo pero que estaban lejos de haber alcanzado un estado de madurez en sus procesos. ¿Cuáles son las razones para que esto suceda? y ¿cuáles son algunas de las consecuencias de que esto suceda?

Entre las primeras se encuentran los tiempos cortísimos asociados a objetivos no realistas y la presión por la falta de recursos para que los proyectos puedan extenderse por más tiempo. Esta causa es común en el mercado y es responsabilidad de los conductores de las empresas y de los consultores que colaboran en la elaboración de este tipo de planificaciones. Sin embargo resta aún conocer cuál es la razón por la cual los lead appraisal aprueban las evaluaciones en estas condiciones de falta absoluta de madurez. No haré especulaciones ni opinaré acerca de este punto.

Las consecuencias de estos procesos implementados bajo presión por falta de recursos, con miembros de la organización que hacen trabajos extra a destiempo, tratando de utilizar procesos y procedimientos en forma forzada ya que aún no han sido adquiridos como hábito por falta de tiempo, es decir madurez, son catastróficas. Conozco una organización con quince o veinte años de vida productiva que después de lograr ser una empresa CMMI nivel 2, sufrió el alejamiento de todas las personas que llevaron adelante dicho proyecto, se redujo alrededor de un treinta por ciento en recursos y hoy día tiene serios problemas para subsistir.

Conozco dos empresas, vinculadas a organizaciones multinacionales, que después de haber obtenido un nivel de CMMI hace algunos años, hoy día trabajan de la misma forma que lo hacían antes de comenzar con la mejora de procesos que los llevó a un nivel 2 ó 3. En ambas organizaciones el número de miembros que participaron en dichos procesos de mejoras y que aún pertenecen a estas se reduce a un par y no atraviesan problemas financieros por su vínculo global.

Conozco una software factory que fue una organización de avanzada en el mercado por haber instalado el concepto de calidad en la industria; después de obtener su nivel 3 del modelo, se redujo hasta hoy ser solo un sello. Conozco una pyme, líder en el mercado de desarrollo, que después de obtener su nivel 2 del modelo se redujo substancialmente y sufrió el alejamiento de todos los miembros que participaron de la mejora de procesos y hoy día trabajan como lo hacían antes.

Todos estos casos son ejemplos de una mala interpretación del modelo y una mala adaptación e implementación con objetivos no realistas. En esencia a todos ellos les faltó madurez para potenciar las mejoras de una forma sostenible en el tiempo.

También conozco organizaciones que implementaron el modelo con tiempo, recursos y buen criterio y hoy día son empresas CMMI de niveles 4 y 5. Sin embargo menciono los casos anteriores porque generalmente se hace una lectura positivista de los informes que muestran a Argentina como un país de avanzada en relación con el número de empresas evaluadas positivamente por el modelo. Sin embargo no he tenido oportunidad de ver lecturas realistas con un análisis de casos y una evaluación de resultados a mediano plazo.

## 7.5 | CONCLUSIONES

---

- Entender las necesidades de la organización y realizar un análisis de factibilidad del proyecto asociado a las mejoras utilizando CMMI.
- Confirmar el compromiso de los dirigentes de la organización para con el proyecto en forma periódica.
- Adaptar el modelo a la realidad de la organización.
- Administrar los cambios como se trató en el capítulo tres.
- Hacer planificaciones realistas basadas en el gap análisis.
- Rescatar todo lo bueno instalado en la organización previamente al inicio de la mejora de procesos.
- Seleccionar cuidadosamente a los mentores.
- Disponer y asignar los recursos materiales y humanos necesarios.
- Dar contenido a las subprácticas ya que estas representan las actividades diarias de los miembros de la organización para inducir a la adherencia al modelo.
- Trabajar hasta alcanzar un grado de madurez que facilite la perdurabilidad en el tiempo de las mejoras logradas.



---

## **Apéndice: Temas Varios**

---

<b>Apéndice - Ejemplos de Activos .....</b>	195
Especificación de requerimientos de software (ers) .....	195
Modelo de especificación de casos de uso.....	197
Ejemplo de priorización de requerimientos .....	199
Modelo de informe de avance .....	202
Modelo de procedimiento .....	203
Procedimiento de trabajo con código compartido (cc) en ambiente de ic .....	204
Descripción .....	204
Forma de trabajo .....	204
Condiciones de entrada .....	205
Entradas .....	205
Roles .....	205
Activos .....	205
Pasos de la actividad .....	206
Salida .....	206
Condiciones de salida .....	206
Métricas .....	207
Verificación y validación .....	207

# APÉNDICE: TEMAS VARIOS

## APÉNDICE - EJEMPLOS DE ACTIVOS

---

En este apéndice se presentan modelos de algunos de los activos mencionados en los diferentes capítulos. Para cada uno de ellos se enfatizan los ítems que consideramos de importancia en relación con los objetivos perseguidos en los diferentes momentos del ciclo de vida del proyecto en los que estos activos son utilizados.

### ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE (ERS)

En la ERS es fundamental la inclusión de un listado de requerimientos que determinarán el alcance del proyecto. El acuerdo entre las partes se basará en este listado. Nuestra experiencia nos ha mostrado que además es de gran utilidad listar los problemas que la organización cliente tiene en la actualidad. Esta enumeración sirve para analizar los requerimientos en relación con la solución a estos problemas. Otro ítem que resulta útil es la mención explícita del alcance de las funcionalidades no incluidas. Esto ayuda a desambiguar diferentes interpretaciones y a determinar el alcance con mayor precisión. Otra sección importante es la relación entre casos de uso y paquetes ya que esta define el primer esbozo de la arquitectura y la relación entre requerimientos y casos de uso. La relación entre estos y el diseño será la herramienta que se utilizará en el análisis de impacto cuando en el futuro aparezcan los cambios.

## DESCRIPCIÓN GENERAL

### Objetivos

- De esta etapa del proyecto (concepción)
- Del proyecto

### Documentos Relacionados

### Destinatarios

### Participantes

### Contexto

- Objetivos y estrategias del negocio
- Sistema actual
- Problemas actuales

### Descripción

- Descripción general del sistema a desarrollar
- Beneficios esperados

### Alcance

- Funciones incluidas
- Funciones excluidas
- Interfaz con otros sistemas y/o dispositivos de hardware

## REQUERIMIENTOS

- Requerimientos funcionales
- Paquete i

### Requerimientos no funcionales

### Acuerdo

## ACTORES

### Modelo de Actores

### Descripción de Actores

## PAQUETES LÓGICOS

### Modelo de Paquetes Lógicos

### Casos de uso preliminares de paquetes lógicos

### Lista de casos de uso preliminares

- Paquete i

## MODELO DE CASOS DE USO

Paquete i

- Diagrama de casos de uso
- Especificación de casos de uso
- Interfaz
- Modelo de dominio

## RELACIONES

Matriz de realización de requerimientos - Casos de Uso

## ALTERNATIVAS DE SOLUCIÓN

*Alternativa 1*

*Alternativa 2*

*Alternativa 3*

## APÉNDICE

*Estimación de tamaño de software*

*Análisis de riesgos*

*Glosario*

*Control de Cambios*

## MODELO DE ESPECIFICACIÓN DE CASOS DE USO

Hemos mencionado en este libro la importancia de la especificación textual de los casos de uso. A continuación presentamos un modelo en el que se muestra esta especificación. Hacemos notar para este activo la importancia en la tabla de diseño conceptual de interfaces (prototipo) de la relación con las clases del modelo de dominio. Esta relación es fundamental ya que es el vínculo entre la aplicación (funcionalidad) y el negocio.

## ESPECIFICACIÓN DE CASOS DE USO DEL SISTEMA

*Diagrama de casos de uso*

*Especificación de casos de uso*

### **ID\_Caso: Nombre\_Caso**

Objetivo

Prototipo de interfaz

#### **Especificación del caso de uso**

- Actores:
- Precondiciones:
- Prioridad:
- **Flujo básico:**

Nombre del flujo básico

1. Primer paso del flujo básico

2. Segundo paso del flujo básico

3. La información mostrada en la interfaz es:

DatoMostrado-1

DatosMostrado-2

DatoMostrado-3

4. Tercer paso del flujo básico

5. El sistema responde mostrando una interfaz de entrada de los siguientes datos:

DatoAIngresar-1

DatosAIngresar-2

DatoAIngresar-3

6. Cuarto paso del flujo básico

7. Quinto paso del flujo básico

8. El caso termina

**Flujos alternativos:**

- 6.<sup>a</sup> Nombre del flujo alternativo
  9. Primer paso del flujo básico
  10. Segundo paso del flujo básico
  11. Tercer paso del flujo básico
  12. El caso vuelve al paso 6

**Postcondición:****Requerimientos no funcionales:****DISEÑO CONCEPTUAL DE INTERFACES**

Interfaz i						
Campo	Tipo	Obligatorio	Validación	Valor Default	Descripción	Modelo de dominio

**CONTROL DE CAMBIOS****EJEMPLO DE PRIORIZACIÓN DE REQUERIMIENTOS**

Un aspecto importante al momento de realizar la planificación de los proyectos es el establecimiento de las prioridades de los requerimientos, las cuales permitirán ordenar el desarrollo del sistema. Estas prioridades deben ser analizadas por parte de la organización cliente en el proyecto. Esto es así debido a que las necesidades del negocio son las que fijan dichas prioridades, las que deben ser acordadas con los responsables del proyecto por parte de la empresa desarrolladora que elabora la planificación. Para ayudarse en esta tarea existen pautas derivadas desde el valor agregado y evolución financiera del proyecto, deseo de los usuarios y riesgos de implementación (ver Agile Estimating and Planning, Mike Cohn, Prentice hall, 2005).

Para exemplificar estas tareas describiremos el análisis realizado en el siguiente proyecto:

La empresa le compró minutos telefónicos a una proveedora de telecomunicaciones (TELCO) y luego los revende. Los productos ofrecidos son servicios de telefonía fija de alcance local, nacional e internacional y voz sobre IP (Internet Protocol).

La organización posee un switch telefónico a través del cual interactúa con la TELCO, rutea y establece las llamadas de sus clientes.

Hace unos años que opera en este mercado y actualmente tiene ocho mil clientes. No posee un sistema para llevar adelante el negocio, lo hace con pequeñas aplicaciones y otros activos, como planillas de cálculo. Presta servicios de Help Desk a sus clientes.

En su momento, la empresa encargó el desarrollo de un sistema con el objetivo de vender a través de la web (actualmente no lo hacen) y ordenar los procesos de negocio cuyas falencias se evidencian en la mala facturación y en el consiguiente caos del Help Desk que se ve desbordado cada vez que los clientes reciben sus facturas.

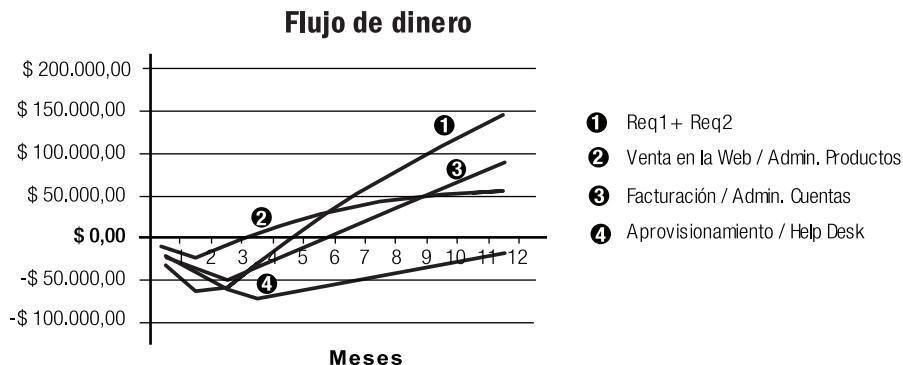
Realizamos un análisis de flujo financiero del proyecto para cada módulo según la tabla A.1. Este análisis fue extrapolado a doce meses, tiempo estimado para el proyecto.

Módulo de ventas en la web / administración de productos: para este módulo se requerían dos meses de desarrollo con el equipo de proyecto por lo que por ese lapso de tiempo no ingresarían sino egresarían recursos. Se especuló que se captaría cincuenta nuevos clientes con un ingreso promedio como ingresos nuevos, que el 10% de los actuales, por el solo hecho de encontrar a su empresa en la web, comprarían nuevos productos por un valor promedio lo cual representaba ingresos incrementados. No se identificaron ingresos retenidos y no se detectó mejoras en la eficiencia operacional. El flujo financiero del proyecto para este módulo se muestra en la figura A.1 en color azul. Puede verse que a partir del cuarto mes, de ser posible instalar el sistema por partes, el desarrollo de este módulo generaría ingresos.

Un análisis similar se realizó para los otros módulos con los resultados mostrados en la figura A.1 con diferentes colores.

Módulo						
Meses	Ingresos nuevos	Ingresos incrementados	Ingresos retenidos	Eficiencia operacional	Costo desarrollo	Flujo dinero

**Tabla A.1** - Análisis del flujo financiero del proyecto



**Fig. A.1** - Flujo financiero del proyecto para los diferentes módulos del sistema

El desarrollo prioritario del módulo azul fue una gran tentación para los gerentes y propietarios de la empresa ya que significaba ingresos genuinos a partir del cuarto mes. Sin embargo esta decisión contradecía la visión del proyecto que establecía un ordenamiento de los procesos de negocio. El desarrollo del Help Desk por sí solo no generaría ingresos y no resolvería la causa de desborde de esta área, la cual se producía por problemas en la facturación. Por esta razón y en base a estos indicadores resolvimos desarrollar como primera prioridad dos módulos compuestos: la venta por Internet / administración de productos; y el de facturación / administración de cuentas, cuyo indicador compuesto se muestra con color turquesa en la figura A.1. Luego de terminado estos se desarrollaría el resto.

Además del análisis de flujo financiero se realizó un análisis del deseo de los involucrados por parte de la empresa, el cual se utilizó para calcular prioridades. Este se muestra en las dos tablas que siguen. La primera, tabla A.2, corresponde a las respuestas de los responsables de la empresa (gerentes, responsables de área, dueños). La segunda, tabla A.3, corresponde a los usuarios no jerárquicos, usuarios del sistema.

Requerimiento	Beneficio relativo	Penalidad relativa	Valor total	% Valor	Estimación de% esfuerzo	% Costo	Prioridad
Ventas en la web / Administración de productos	8	6	14	41,18	536	12,57	3,28
Facturación / Administración de cuentas	6	3	9	26,47	1448	33,96	0,78
Aprovisionamiento / Help Desk	6	5	11	32,35	2280	53,47	0,61
<b>Total</b>	<b>20</b>	<b>14</b>	<b>34</b>	<b>100,00</b>	<b>4264</b>	<b>100</b>	

**Tabla A.2** - Prioridades de los miembros jerárquicos de la empresa

Requerimiento	Beneficio relativo	Penalidad relativa	Valor total	% Valor	Estimación de% esfuerzo	% Costo	Prioridad
Ventas en la web / Administración de productos	4	2	6	15,79	536	12,57	1,26
Facturación / Administración de cuentas	8	6	14	36,84	1448	33,96	1,08
Aprovisionamiento / Help Desk	10	8	18	47,37	2280	53,47	0,89
<b>Total</b>	<b>22</b>	<b>16</b>	<b>38</b>	<b>100,00</b>	<b>4264</b>	<b>100</b>	

**Tabla A.3** - prioridades de los miembros no jerárquicos de la empresa

Puede verse que a diferencia del grupo dirigente de la empresa, los usuarios, que son los que sufrían el caos generado por las falencias de la actual gestión, priorizan de manera distinta la necesidad de los módulos.

Por último se realizó un análisis de prioridades de acuerdo al riesgo de implementación y al valor agregado al negocio. Los resultados se muestran en la tabla A.4.

Requerimiento	Conocimiento negocio	del Valor agregado al negocio	Riesgos	Prioridad
Ventas en la web / Administración de productos	Alto	Medio	Bajo	Baja
Facturación / Administración de cuentas	Alto	Alto	Alto	Alta
Aprovisionamiento / Help Desk	Alto	Bajo	Medio	Media

**Tabla A.4** - Prioridades basadas en el valor agregado y los riesgos de implementación

La estrategia que se deriva de este análisis es: primero implementar (alta prioridad) aquellos módulos de alto valor agregado y alto riesgo. Segundo seguir con los de alto valor agregado y menor riesgo. Para la priorización de los casos de uso de cada módulo se realiza un análisis similar.

La combinación de todos estos factores se utiliza en el establecimiento de prioridades de los requerimientos de los proyectos.

## MODELO DE INFORME DE AVANCE

Como se analizó en el capítulo Trabajo con la gestión de proyectos, el buen seguimiento de la evolución es lo que permite visibilidad y un accionar a tiempo sobre los eventos que afectan al proyecto. Además de la información vinculada a los números actualizados del proyecto es muy importante que estos informes estén relacionados, es decir, constituyan una secuencia a través de la cual se pueda reconstruir la evolución. Para esto se incluyen los problemas anteriores y luego los nuevos, para realizar la gestión de la resolución de unos y planificar las acciones sobre los otros.

*Datos del reporte*

*Grado de avance del proyecto*

- Observaciones

*Entregables*

*Facturación*

*Problemas surgidos*

- Problemas anteriores
- Problemas nuevos

*Seguimiento de riesgos*

- Análisis de la evolución
- Acciones

*Cambios de requerimientos acordados*

*Monitoreo de acuerdos*

*Monitoreo de compromisos*

*Monitoreo de involucramiento*

*Monitoreo de hitos*

*Monitoreo de administración de datos*

*Acuerdo de avance*

## **MODELO DE PROCEDIMIENTO**

El modelo de procedimiento que se presenta a continuación fue tomado de las recomendaciones del SEI (Software Engineering Institute) y adaptado a través del uso que nosotros le dimos en múltiples proyectos. El objetivo de este modelo es comunicar. Los miembros de la organización que lo lean deben encontrar en él la información clara y precisa para llevar adelante la tarea que el procedimiento documenta. Consideramos fundamental la sección Condiciones de entrada y salida, ya que son estas las que identifican cuándo la tarea puede comenzar y cuándo está terminada. Las métricas están orientadas a relevar información para futuras mejoras en la ejecución de la tarea. Mientras que la verificación y validación, a poder hacer un control de calidad de esta.

*Descripción*  
*Condiciones de entrada*  
*Entradas*  
*Roles*  
*Activos*  
*Pasos de la actividad*  
*Resumen de Tareas*  
*Pasos del procedimiento*

- Tarea 1
- Tarea 2

*Salida*  
*Condiciones de salida*  
*Métricas*  
*Verificación y validación*

## **PROCEDIMIENTO DE TRABAJO CON CÓDIGO COMPARTIDO (CC) EN AMBIENTE DE IC**

### **Descripción**

Este proceso está orientado a generar en cada momento del proyecto los activos que muestran el trabajo realizado. La estrategia que se sigue es la de generar un nuevo build con cada cambio detectado. Opera en forma automática y, además de los productos del proyecto, genera reportes de las pruebas y provee información de análisis de la calidad de dichos productos.

### **Forma de trabajo**

1. Un desarrollador realiza un commit (cambios) sobre el repositorio de software del proyecto mientras el administrador de IC lo consulta por cambios con una frecuencia determinada.
2. Despues del commit el administrador de IC detecta el cambio, toma del repositorio las últimas versiones y ejecuta los scripts que integran todo el software y ejecuta los test.
3. El administrador de IC informa por mail acerca de los resultados a los miembros del grupo de desarrollo.

4. El administrador continúa consultando al repositorio con la frecuencia determinada.

### Condiciones de entrada

Las condiciones que deben cumplirse antes de comenzar con las tareas incluidas en este proceso son:

- Casos de uso de la próxima iteración especificados
- Arquitectura definida
- Ambiente de desarrollo definido
- Primera iteración de construcción por comenzar

### Entradas

Los artefactos y la información necesaria para llevar adelante la actividad son:

- Plan de proyecto: listado de los miembros del grupo de trabajo con la dirección de email correspondiente.
- Plan de CM (Configuration Management): estructura del repositorio de software.
- EDA (especificación de la arquitectura, incluyendo la definición del ambiente de desarrollo).

### Roles

Los roles responsables de llevar adelante la actividad son:

- Desarrolladores
- Miembros de QA
- Líder de proyecto
- Administrador de infraestructura

### Activos

Los activos a utilizar en el desarrollo de este proceso son:

- Software de la plataforma de desarrollo
- Software de administración del repositorio
- Software de administración del ambiente de desarrollo
- Software de compilación y generación de binarios
- Software de administración de pruebas
- Software de control y automatización

- Software de seguimiento de eventos.
- Software de reportes de actividades y atributos de calidad.
- Software de métricas de código

### Pasos de la actividad

Tarea	Roles	Entrada	Salida
Planificación	<ul style="list-style-type: none"> <li>• Líder de proyecto</li> <li>• Líder técnico</li> <li>• Admin. de infraestructura</li> <li>• Responsable de QA</li> </ul>	Plan proyecto	Plan IC
Instalación del ambiente	<ul style="list-style-type: none"> <li>• Líder técnico</li> <li>• Admin. de infraestructura</li> </ul>	Plan IC	Ambiente instalado y probado
Ejecución	<ul style="list-style-type: none"> <li>• Desarrolladores</li> <li>• Miembros QA</li> <li>• Líder de proyecto</li> <li>• Admin. de infraestructura</li> </ul>	Repositorio del proyecto	<ul style="list-style-type: none"> <li>• Reportes de fallas</li> <li>• Reportes de seguimiento</li> <li>• Reporte de actividades</li> <li>• Reporte de calidad</li> <li>• Binarios instalados</li> </ul>

### Salida

Las salidas generadas por este proceso son:

- Binarios instalados en el ambiente definido
- Reportes de actividades del proyecto
- Reportes de calidad de diseño y código del proyecto
- Reportes de fallas y los avisos correspondientes a los desarrolladores.
- Reportes de resultados de pruebas

### Condiciones de salida

Las condiciones que deben alcanzarse antes de que la actividad pueda considerarse completada son:

- Última iteración de la planificación terminada
- Pruebas de aceptación superadas
- Acuerdo de cierre del proyecto

## Métricas

Las medidas útiles que soportan la realización de la actividad o futuras realizaciones son:

- Cantidad de fallas reportadas por el sistema automático
- Números de los indicadores de las métricas de código
- Reportes de la actividad de cambios en el repositorio
- Números del sistema de seguimiento que muestran la dinámica de la cantidad de fallas nuevas y cantidad de resueltas

## Verificación y validación

Las técnicas para verificar y validar la realización del proceso son:

- Análisis de los reportes del sistema automático
- Análisis de los reportes de seguimiento

