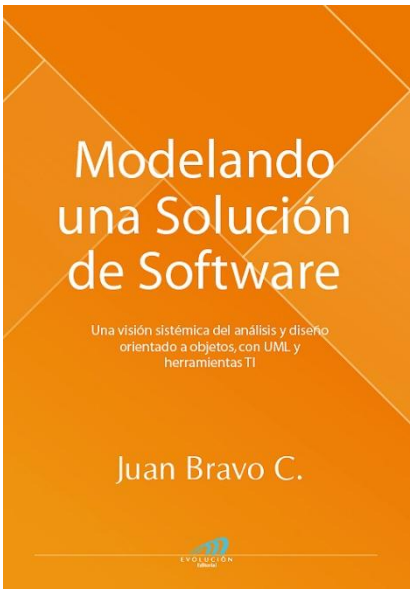
Modelando una Solución de Software

(Alineados con la estrategia)

Versión resumida

(Extractos de la versión original)



Juan Bravo Carrasco

www.EVOLUCIÓN.cl

Inscripción Nº 169.936 del 24 de marzo de 2008
I.S.B.N.: 978-956-7604-15-9 del 24 de marzo de 2008

(392 páginas., 24,5 x 17 cm.)

Puede adquirir la versión completa en formato papel o digital desde la página www.evolucion.cl.

EDITORIAL EVOLUCIÓN S.A.
www.evolucion.cl, info@evolucion.cl
Santiago de Chile

Contenido

Introducción

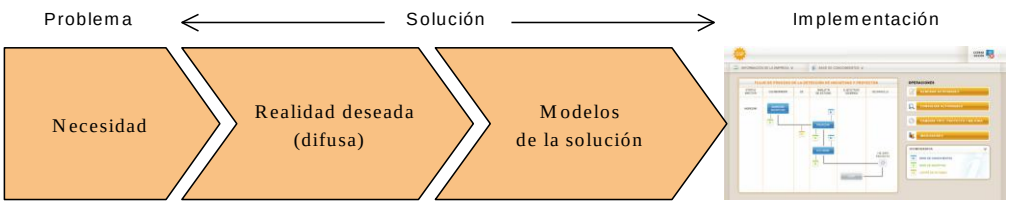
Modelar una solución de software es una labor bella y creativa. Por esta razón, frecuentemente se obtienen muy buenos productos que son verdaderas "obras de arte", tal como si a un artista le encargaran una obra (requerimientos) y él, utilizando sus propios métodos y herramientas de trabajo, diera vida a una creación única e irrepetible.

¿Será posible profesionalizar conservando la creatividad? ¡Sí! y de esta forma los métodos y herramientas van a recibir el aporte de muchas personas. Modelar soluciones de software puede ser arte y tecnología a la vez.

Este es el desafío, modelar soluciones de software con técnicas normalizadas, buscando simplicidad, eficiencia y adaptabilidad al cambio, en el contexto de un proceso general de desarrollo que permita trazabilidad y productos repetibles.

¿Por qué modelar? Porque es necesario representar formalmente una realidad deseada, que de otra forma resultaría muy difícil de transmitir, en este caso una solución de software. Lo más probable es que la realidad deseada se encuentre vagamente escrita y que principalmente esté en la mente de las personas, más como un deseo difuso que como un requerimiento. La función del modelamiento es hacer tangible y aclarar esa realidad para que luego se pueda implementar.

Si esa realidad deseada da respuesta a una necesidad por una parte y por otra los modelos de esa realidad son factibles de implementar, entonces la probabilidad de éxito es alta. Eso es lo que quiere mostrar la siguiente figura.



Shari Pfleeger explica en su libro *Ingeniería de software* (2002, p. 226): "Se utiliza la especificación de requerimientos para definir el problema. Entonces, podemos declarar que algo es una solución a un problema si satisface todos los requerimientos planteados en la especificación. En muchos casos el número de soluciones es prácticamente ilimitado".

¿Siempre es necesario modelar la solución antes de implementar?

Depende, si usted sólo quiere extender una pared interior de 2 metros, es posible que no requiera los modelos, puede contratar un experto y sólo darle instrucciones verbales, tal vez le resulte y ahorre tiempo y dinero. Sin embargo, si quiere construir su casa necesitará de maquetas y muchos planos (obra gruesa, cañerías de agua, cables de electricidad y todo lo demás).

En el software es igual, toda aplicación desde un costo de algunos miles de dólares ya hace necesario un modelar formalmente. La idea es superar la construcción artesanal de software (por ejemplo, construir sin planos) y aplicar los nuevos conocimientos sobre teoría de modelos, normalización (tal como orientación a objetos y UML) y construcción con herramientas CASE.

Contenido del libro

Lo primero sucede en esta misma introducción, una selección de los modelos más relevantes de una solución de software, donde sólo se muestra un boceto de ellos para lograr una visión de conjunto (el detalle de cada uno está en el resto del libro).

Esta presentación será nuestra guía y a partir de esta experiencia extraeremos una conclusión vital: *las competencias que debería tener un profesional* de la informática. Las competencias son conocimientos, habilidades y actitudes necesarias para modelar aplicaciones computacionales (o soluciones de software), en este texto se presentan en la forma de capítulos:

1. Un método completo para la gestión de proyectos y así situar el modelamiento de soluciones de software en su contexto.
2. La profesionalización de conocimientos que aporta la ingeniería de software para comprender todos los aspectos técnicos de los modelos, las normas y estándares que existen.
3. Los múltiples aportes de la nueva teoría de modelos, en particular el modelamiento de funciones y el criterio curso normal de los eventos.
4. El indispensable modelamiento de datos.
5. Los necesarios conocimientos de la orientación a objetos.
6. El estándar UML.
7. Las herramientas de la tecnología de información.

Cada capítulo, en el mismo orden, profundiza en la competencia correspondiente, produciéndose un avance que parece una pirámide, tal como se aprecia en la siguiente figura.

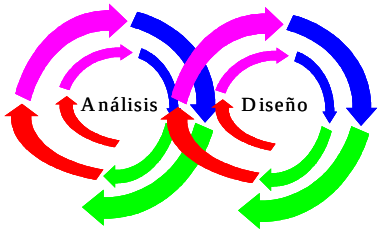
Competencias necesarias para modelar aplicaciones computacionales

Síntesis de modelos de una solución de software

Los modelos se crean en las etapas de *análisis* y *diseño* de la Gestión Sistémica de Proyectos (GSP)[1]. El camino para dibujarlos es iterativo, es decir, se van perfeccionando mediante borradores sucesivos.

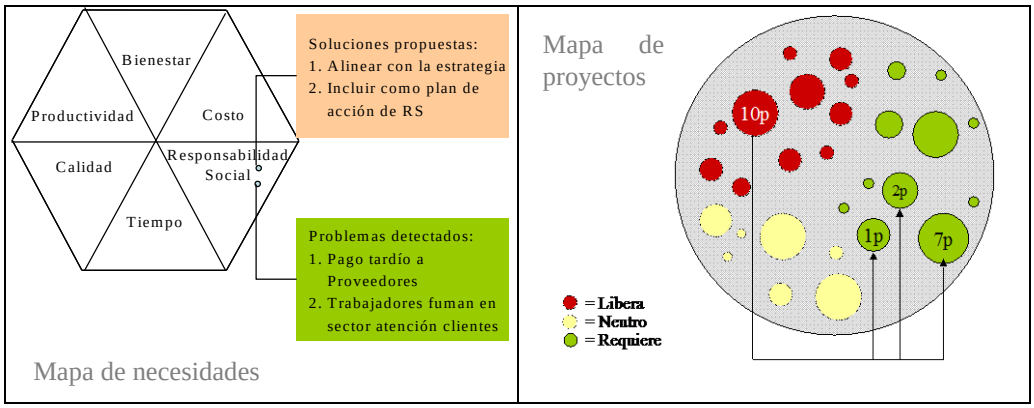
Es recomendable contar cuanto antes con un boceto de todos los modelos que se considerarán para cada etapa de la aplicación particular, una primera versión destinada a lograr una visión de conjunto. Ese es el sentido de esta introducción.

Seguimos la idea de una doble espiral que se traslapan parcialmente: la primera del análisis y la segunda del diseño, tal como vemos en la figura.

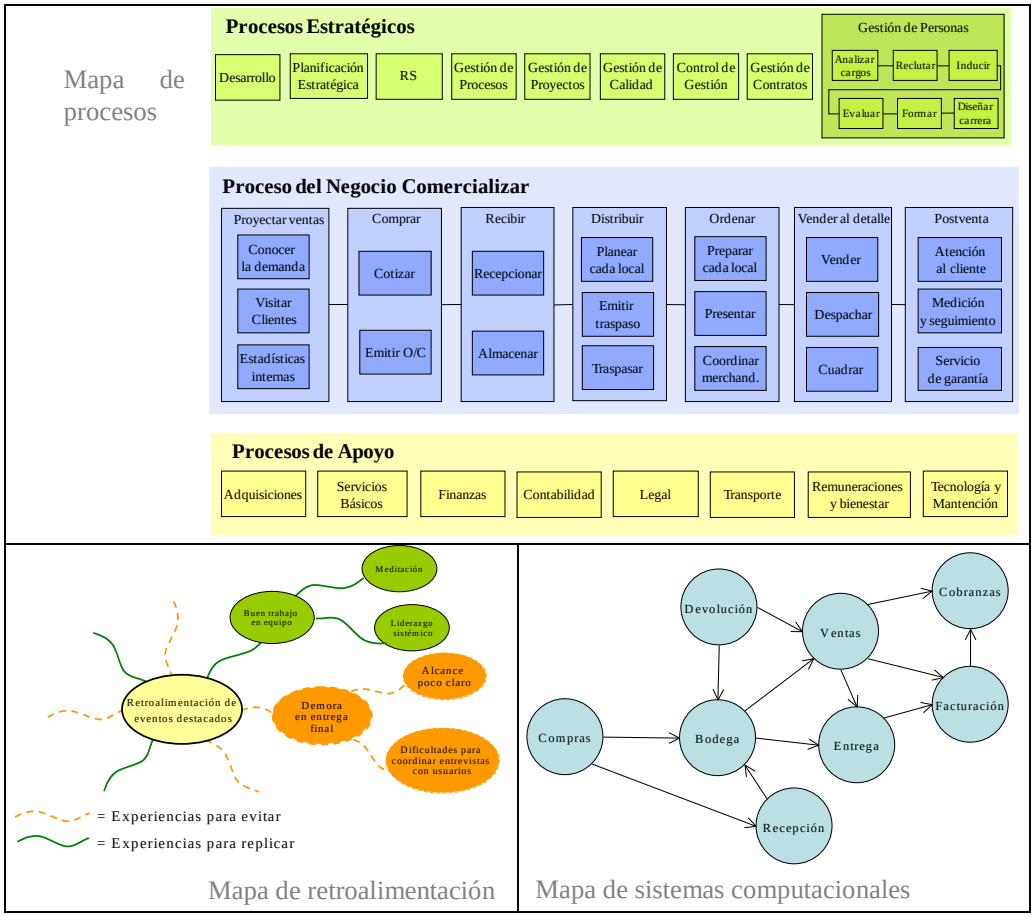


Mapas para la visión de conjunto

Conviene observar estos cinco mapas que deberían existir en la organización previo al desarrollo de una aplicación específica. Son los mapas de necesidades, proyectos, procesos, retroalimentación y sistemas.



[1] En el capítulo 1 veremos el método completo.

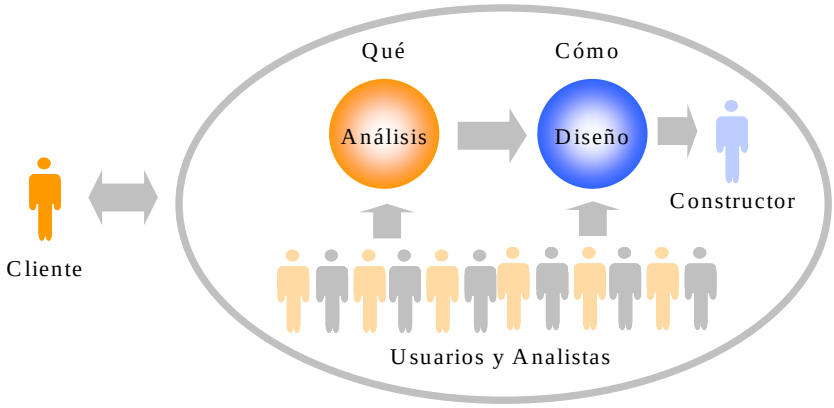


Estos mapas son modelos que ayudan a lograr la visión de conjunto para luego formalizar en el análisis y diseño la solución de software. El detalle de cada uno se puede apreciar en el capítulo 1.

La visión de conjunto es vital en la *visión sistémica*, cosmovisión que guía todo el trabajo de este libro, tanto en los mapas como en los siguientes modelos, los cuales tienen la características de contextualidad, es decir, dependen del método y nivel de madurez de cada organización.

Los modelos que se presentan a continuación para análisis y diseño son sólo una muestra de las posibilidades del modelamiento. Cada empresa debe tener su propia ruta metodológica e incluso adaptada según tipos de proyectos.

Se presentan los modelos ordenados según las etapas de análisis y diseño. En la siguiente figura se aprecia el objetivo y actores de cada una. Todo el modelamiento está orientado al cliente (externo, quien paga), la etapa de análisis se orienta a definir el qué y la de diseño el cómo, en ambas participan analistas y usuarios. Una vez que el diseño está completo, se envía al constructor (aunque sea el mismo analista en otro rol).

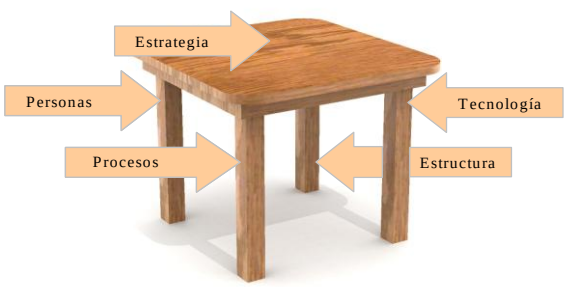
- 

Modelos de la etapa de análisis

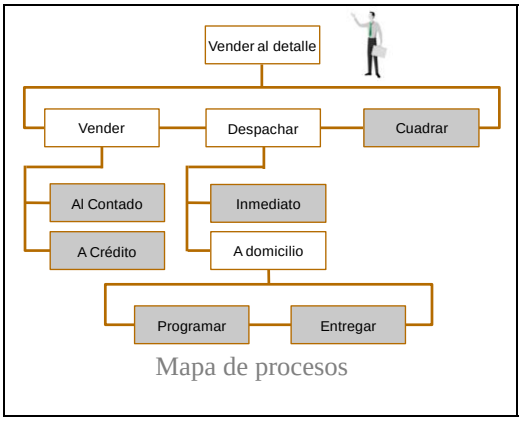
La siguiente selección de modelos no pretende ser exhaustiva, se incluye con el único objetivo de lograr una visión global, no se explican aquí porque el detalle de cada uno está en los capítulos del libro.

En este caso, los modelos siguen la lógica del método GSP, en la empresa corresponden a los contenidos del proceso de desarrollo de software que ésta se haya dado.

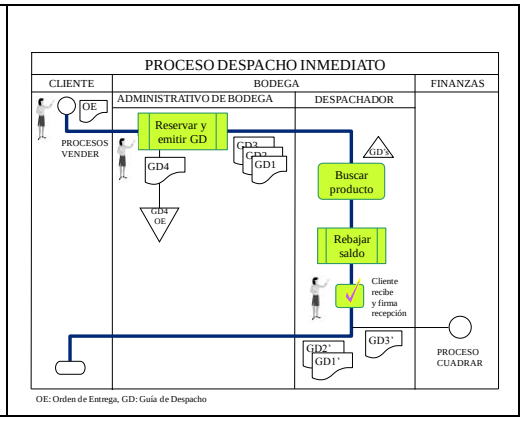
- En esta etapa todo el trabajo se centra en el modelo de negocios de la solución, con cinco elementos que se representan con la metáfora de una mesa, la cubierta es la estrategia (alineando la de la empresa y la de del proyecto, incluye responsabilidad social) y las 4 patas son: personas (incluyendo ambiente), procesos, estructura (organizacional y física) y tecnología (de todo tipo). En esencia: corresponde decidir *Qué hacer*, comenzando por la cubierta de la mesa (detalle de la figura en el capítulo 1).



Luego se comienza a trabajar en la pata de los procesos: levantamiento detallado y propuestas de cambio. Se emplean principalmente los modelos mapa de procesos y flujograma de información, los cuales se observan en la siguiente figura (detalle en los capítulos 1 y 3 respectivamente).



Mapa de procesos



Desde aquí surgirán definiciones para las otras patas de la mesa: personas, estructura y tecnología. Lo cual está descrito en el capítulo 1.

Para definir el alcance de la solución de software en la etapa de análisis, se puede emplear esta serie de modelos (una buena técnica es por borradores sucesivos, comenzando por cualquiera de ellos). El objetivo es decidir qué incluye el modelo de negocios (detalle en los capítulos 1, 2 y 3).

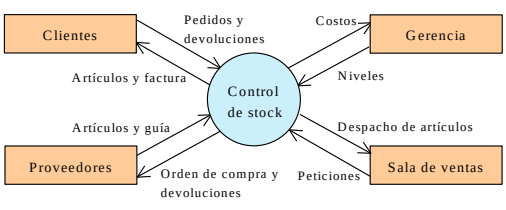
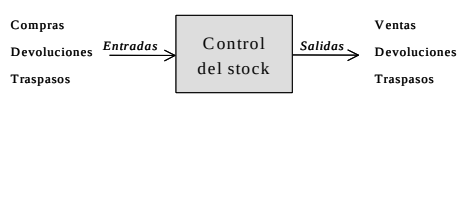
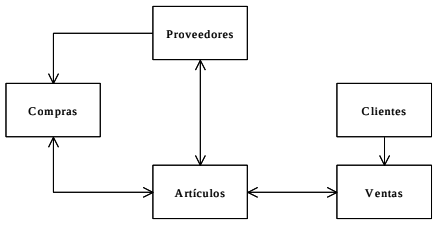


Diagrama de contexto



Modelo orientado al objetivo principal del sistema



Modelo orientado a los datos

Maestros / Transacciones	Clientes	Artículos	Proveedores	Cuentas Contables	Historial Ventas	Historial Compras
Ventas	X	X		X	X	
Compras		X	X	X		X
Devolución ventas	X	X		X	X	

Modelo orientado al flujo de transacciones

Diseño general de la interfaz

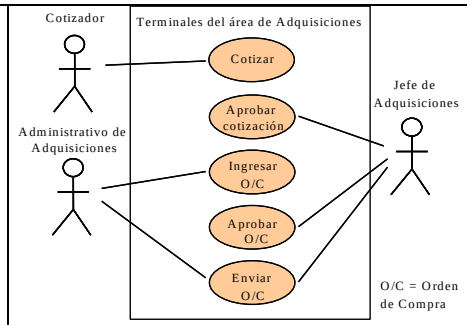
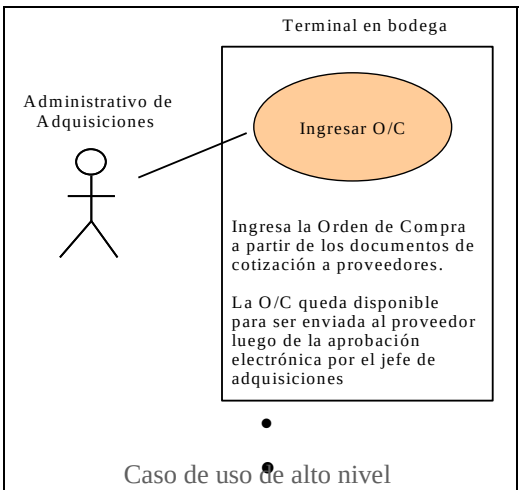
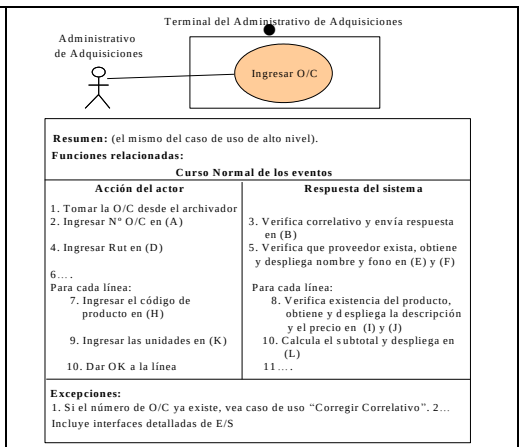


Diagrama de casos de uso

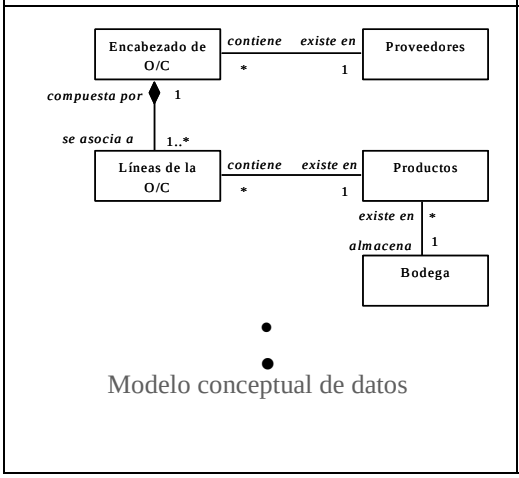
Una vez lograda la decisión respecto a los *qué*, es necesario profundizar en los requerimientos principales de la solución de software, en tal caso, la recomendación es trabajar con estos nuevos modelos (detalle en los capítulos 5 y 6).



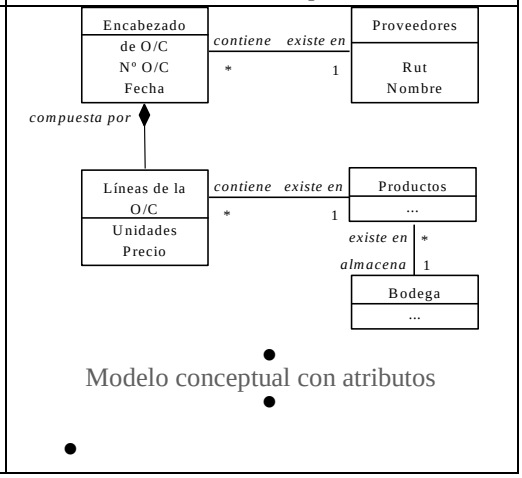
Caso de uso de alto nivel



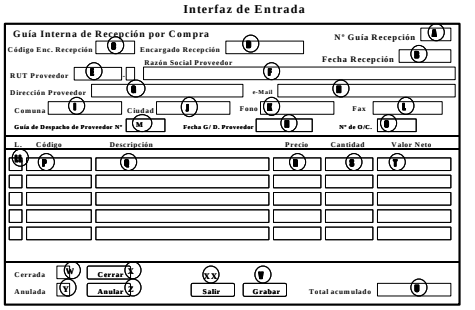
Caso de uso de expandido



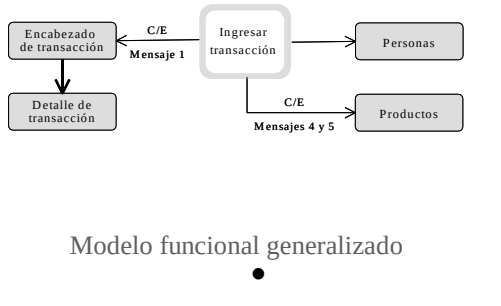
Modelo conceptual de datos



Modelo conceptual con atributos



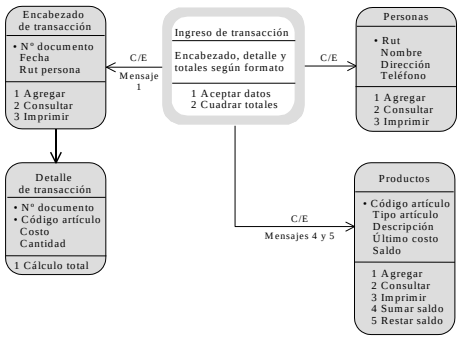
Interfaz visual detallada



Modelo funcional generalizado

Modelos de la etapa de diseño

De la misma forma que la etapa de análisis, es decir, mediante borradores sucesivos y la técnica de desarrollo en espiral (ver anexo 3) se avanza en el diseño, sin mayores problemas de volver a veces a la etapa de análisis, porque ambas forman una totalidad que llamamos *modelar una solución de software* (el detalle de estos modelos está en los capítulos 5 y 6).



Modelo funcional generalizado y detallado

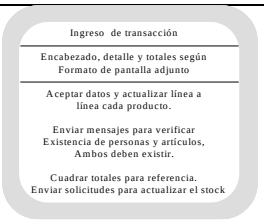


Tabla de objetos, clase Ingreso de transacción

Objeto	Atributos	Funciones
Ingreso de ventas	Indicar stock del producto	Deben cuadrar totales, stock mayor a unidades por vender. Mensaje 5
Ingreso de compras		Crear proveedor y artículo si no existen. Mensaje 4

Visión interna de la clase ingreso de transacción con la tabla de objetos

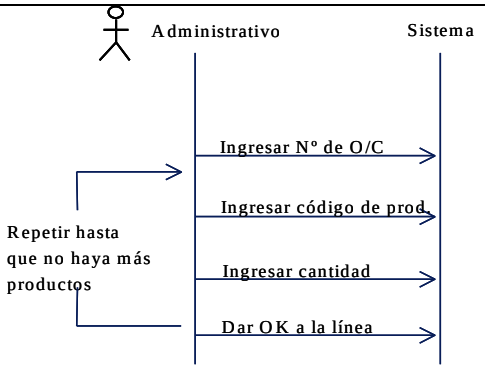


Diagrama de secuencia

Contrato

Identificación: Dar OK al ingreso de la línea
Responsabilidades: con cada ingreso de línea los conceptos deben ser consistentes.
Tipos de datos: afecta a los conceptos Encabezado de O/C y Detalle de O/C.
Referencias cruzadas: no hay
Notas: nada especial
Excepciones: la no existencia de la línea en el sistema ya fue validada con el ingreso de O/C.
Salida: no hay
Precondiciones: no existe la línea.
Poscondiciones:

- Se creó una línea en el concepto detalle.
- Se actualizó el contador de líneas en el encabezado.
- Se actualizó la asociación entre encabezado y detalle de O/C.

Contrato

Los dos modelos más característicos del diseño desde el punto de vista de UML son el de diseño de clases y colaboración (detalle en el capítulo 6).

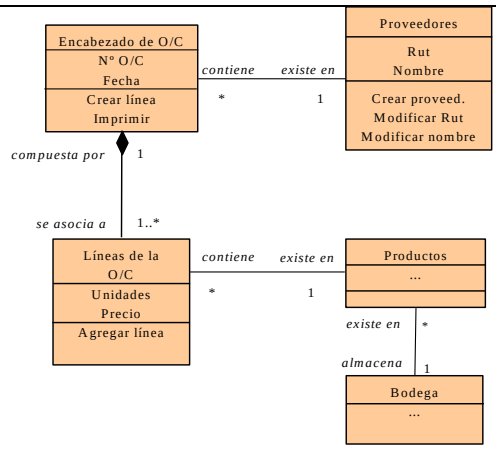


Diagrama de diseño de clases

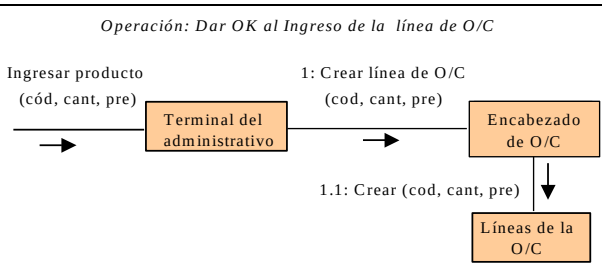


Diagrama de colaboración

Capítulo 1.
Método para la gestión de proyectos

Este capítulo introduce en los conceptos y la necesidad de contar con un método completo para la gestión de proyectos en la organización, no sólo en el ámbito tecnológico.

Esta es la primera competencia considerada para apoyar la elaboración de modelos de una solución de software, tal como se aprecia en la siguiente figura (en la introducción se incluyó la visión global de las competencias). Es necesario que el analista conozca la totalidad de pasos de un proyecto para insertar su aporte. Podríamos decir que es un conocimiento de tipo horizontal, con "visión de procesos", porque se refiere a entender la totalidad de la gestión de proyectos, independiente de que su foco estará en las etapas de análisis y diseño.

La visión global, sistémica, que ofrece un método es indispensable para entender la totalidad que surge de necesidades concretas en la empresa que los proyectos ayudarán a resolver creando una habilidad organizacional.

Al método que presentamos en estas páginas le hemos llamado GSP (Gestión Sistémica de Proyectos), es resultado de extensas investigaciones acerca de las mejores prácticas de proyectos y es un extracto del libro *Gestión de proyectos de procesos y tecnología*, señalado en el prólogo.

Trabajar metodológicamente es una competencia indispensable para todo profesional del área de proyectos y para todo tipo de proyectos, ya sea que estén orientados a procesos del negocio, de apoyo o estratégicos. Por otra parte, toda organización debe contar con un método para la gestión de sus proyectos.

Las *etapas* son los grandes bloques que aporta el método GSP: concepción, factibilidad, análisis, diseño, implementación, despliegue y operación.

Las etapas están agrupadas en tres *fases*: estudio, desarrollo y mejora continua. Tanto las etapas como las fases se pueden traslapar en los límites.

También existen prácticas transversales a las etapas, es decir, aplican en algunas o en todas las etapas del método. Son 28: dirección del proyecto, plan de la etapa, exposición de los planes, retroalimentación, equipo de trabajo, entrevistas, comunicación, informes, técnicas, herramientas de apoyo, trazabilidad, *Quick wins*, costos y duración, gestión de riesgos, gestión de la calidad, responsabilidad social, inserción, orientación al cliente, sensibilización, capacitación, seguimiento, cuidar la solución anterior, continuidad operacional, plan de recursos físicos del proyecto, *kill time*, control de cambios, gestión del cambio y gestión de proveedores.

Veremos:

- Trabajar metodológicamente

- Claves de la implantación de método
- Adaptación y profesionalismo
- Etapas genéricas
- Prácticas transversales

• • •

Capítulo 2.
Ingeniería de Software

El objetivo más importante de la ingeniería de software es lograr la producción profesional de software, donde se aumente la calidad y la productividad y se disminuyan los riesgos del proyecto mediante una excelente planificación y modelamiento. No se refiere a la producción en serie, sino a la obtención de un producto creativo y personalizado, desarrollado con método, técnicas y herramientas conocidas.

Esta es la segunda competencia considerada para apoyar la elaboración de modelos de una solución de software, tal como se aprecia en la siguiente figura (en la introducción se incluyó la visión global de las competencias). Es necesario que el analista conozca acerca de la ingeniería de software para que inserte su aporte en el contexto de esta disciplina. Es una competencia de tipo vertical, porque se profundiza en una especialización, el desarrollo de software.

Se trata de avanzar desde aplicaciones computacionales que son "obras de arte únicas", hacia productos de programación normalizados, con documentación automatizada, fáciles de construir y de mantener, para lograr aumentos de productividad de los desarrolladores de software.

No hay una contradicción entre obtener productos creativos trabajando metodológicamente.

Hay que desmitificar la producción de software, transformándola en una actividad mucho más amistosa, a través de la aplicación de técnicas simples y herramientas poderosas, con una finalidad revolucionaria: *permitir que los usuarios calificados puedan participar activamente en todo el ciclo de desarrollo.* Usuarios calificados son profesionales y ejecutivos que poseen entrenamiento formal en tecnología de la información, quienes conocen su problema y saben como estructurarlo.

La orientación del capítulo y de todo el libro, es hacia la producción de software que apoye directamente los procesos de la organización teniendo siempre al cliente como norte (al cliente final, el que paga).

La ingeniería de software incluye la producción de otros productos de software, como sistemas operativos, herramientas de productividad o de automatización de oficinas. Éstos siguen patrones parecidos a la producción de software administrativo e incluyen algunos aspectos más especializados, como el énfasis en la programación orientada al objeto o la utilización de lenguajes que provean máxima eficiencia.

Todo lo que se refiere al apoyo automatizado para el desarrollo de software (Herramientas CASE) se incluyó en el capítulo 7, sobre herramientas de la tecnología de información.

Veremos:

- Alcance de la ingeniería de software
- Planificación en informática
- Sistema de productividad en el desarrollo de software
- Criterios de desarrollo de software
- Métodos para la producción de software
- Apoyo del diseño en la explotación del sistema
- Diseño de interfaces
- Normas y estándares aplicados a los modelos TI

• • •

Capítulo 3.
Teoría de Modelos Aplicada

La nueva teoría de modelos aporta avances vitales que deben ser conocidos para enriquecer la creación de modelos de una solución de software.

Esta es la tercera competencia considerada para apoyar la elaboración de modelos de una solución de software, tal como se aprecia en la siguiente figura (en la introducción se incluyó la visión global de las competencias). En este caso, el analista debe conocer acerca de la teoría de modelos como una simple responsabilidad profesional que deriva de su misión de modelador de una realidad deseada.

Los aportes de visión sistémica y de la gestión de procesos, en particular el criterio del curso normal de los eventos, son claves en esta misión.

Veremos:

- Marco teórico de los modelos
- Modos de procesamiento
- Once claves de los modelos computacionales
- Modelamiento de funciones
- Fundamentos del modelamiento de funciones
- Criterio curso normal de los eventos

•••

Capítulo 4.
Modelamiento de Datos

El modelamiento de datos logra una visión de conjunto de los datos de la aplicación y de su contexto. En todo caso, se requiere un gran modelo con los conjuntos de datos de toda la organización para que las diferentes aplicaciones "vean" y trabajen con la porción que les corresponde.

Esta es la cuarta competencia considerada para apoyar la elaboración de modelos de una solución de software, tal como se aprecia en la siguiente figura (en la introducción se incluyó la visión global). Es indispensable que el analista conozca acerca del modelamiento de datos como simple responsabilidad profesional porque es una habilidad básica de su labor.

Es vital la visión de conjunto que provee el modelo conceptual de datos de toda la organización, permite comprender, ubicarse y evitar inconsistencias como la de crear dos veces la misma tabla. De esta forma, el aporte de la solución de software será aportar nuevas tablas o modificar las existentes.

Veremos:

- Definiciones sobre el modelo de datos
- Criterios básicos de normalización de datos
- Enfoque de bases de datos

•••

Capítulo 5.
Orientación a Objetos

La orientación a objetos (OO) provee una forma simple y natural para crear los modelos de la solución de software. Los objetivos que se pretenden lograr son ambiciosos: aumentar la productividad, mejorar la calidad, facilitar la mantención, incorporar al usuario, reducir los riesgos y reutilizar el trabajo previo, entre otros. Cabe destacar dos características: la mayor naturalidad y el aporte a los contenidos a través de una biblioteca de clases que se va perfeccionando en el tiempo.

Esta es la quinta competencia considerada para apoyar la elaboración de modelos de una solución de software, tal como se aprecia en la siguiente figura (en la introducción se incluyó la visión global de las competencias). Es necesario que el analista sea muy hábil en la orientación a objetos como parte de su responsabilidad profesional, porque es una competencia central de su labor que tiene un impacto mucho más allá de las etapas de análisis y diseño, tiene que ver con la visión de trabajar con integración y componentes.

Con la orientación a objetos es posible que la solución de un desarrollador sea comprendida más fácilmente por otros, obteniéndose mayor independencia del modelo respecto a su creador; así, la empresa usuaria se beneficia doblemente, porque no se repiten soluciones a los mismos problemas y porque hay una inversión en inteligencia al ser posible que nuevos especialistas aprovechen todo o parte del avance de sus predecesores, más aun cuando el diseño queda registrado en alguna herramienta de apoyo para esta etapa.

En este libro se aborda la orientación a objetos desde el punto de vista del desarrollo de las aplicaciones computacionales que apoyarán el negocio de la organización.

Veremos:

- Fundamentos de la orientación a objetos
- Definiciones sobre orientación a objetos
- Conceptos de la orientación a objetos
- Proceso de generalización
- Fases de la orientación a objetos
- Incorporación de la tecnología de objetos

• • •

Capítulo 6.
Unified Modeling Language (UML)

UML significa literalmente *Lenguaje Unificado de Modelamiento*, aunque la idea queda mejor expresada con: *Modelamiento Visual del Software*, expresión que se está utilizando cada vez más en español. UML está orientado a la especificación, visualización y documentación de los productos de software. Se le considera parte del desarrollo tecnológico de un modelo de negocios, enfocado en la definición de los requerimientos de la aplicación.

Esta es la sexta competencia considerada para apoyar la elaboración de modelos de una solución de software, tal como se aprecia en la siguiente figura (en la introducción se incluyó la visión global de las competencias). Es indispensable que el analista maneje UML porque es el único estándar en esta materia, por lo tanto, también se trata de una responsabilidad profesional, porque hoy es considerado como parte de la calidad estar integrado al mundo (y en este caso es literal porque UML es el lenguaje utilizado para solicitar servicios de desarrollo al otro lado del planeta).

UML surgió de los aportes combinados de tres pioneros en el campo del modelamiento orientado a objetos, los doctores Grady Booch, Jim Rumbaugh e Ivar Jacobson, a petición de la OMG (Object Management Group), organización creada por las empresas líderes mundiales de la industria del software (entre las cuales se encuentran IBM, Unisys, Alcatel, Toshiba y Microsoft) destinada a fijar estándares en la industria con la tecnología de objetos.

La primera versión de UML estuvo disponible en 1997. Ha sido perfeccionado en el tiempo y la versión actual es la 2.0.

Es mucho lo que se puede decir de UML, en este capítulo veremos los modelos y en el anexo 7 podrá ver un caso completo, el cual puede bajar desde la página www.evolucion.cl.

Veremos:

- Modelos de UML
- Aplicación de los modelos UML en la etapa de análisis
- Aplicación de los modelos UML en la etapa de diseño

•••

CAPÍTULO 7.
Herramientas de la Tecnología de Información

Las *Herramientas de la Tecnología de Información* son todos los productos de software que permiten aumentar la productividad de especialistas y usuarios. Éstas incluyen desde poderosos procesadores de texto hasta sofisticados sistemas administradores de bases de datos, pasando por múltiples productos de ayuda directa a usuarios y especialistas en desarrollo de aplicaciones.

Esta es la séptima competencia considerada para apoyar la elaboración de modelos de una solución de software, tal como se aprecia en la siguiente figura (en la introducción se incluyó la visión global de las competencias). El analista debe tener una visión global de las herramientas de la tecnología de información y debe conocer en detalle las que apoyan directamente su labor. Es una responsabilidad profesional para aumentar la productividad del modelamiento y para compartir en equipos de trabajo.

En el caso de las herramientas de ayuda en la producción de software conviene actuar con prudencia, por ahí aparecen mensajes del siguiente tipo: es posible que las aplicaciones se construyan casi solas o que el mismo usuario pueda construir su propio software sin depender del programador. Esos y otros mensajes se demuestran con aplicaciones muy pequeñas y estructuradas, extrapolándose las conclusiones al resto del software. Sin embargo, siendo válido, esa es una parte pequeña de la realidad, lo habitual es que el especialista en informática tenga la responsabilidad de construir aplicaciones de mayor nivel de complejidad, en las cuales la herramienta será sólo un apoyo y siempre que se pueda integrar al esquema de desarrollo del departamento de sistemas en particular.

Para tranquilidad de muchos programadores, hasta hoy no se ha visto el reemplazo de un especialista por una herramienta de productividad. Sí ocurre que la incorporación de la nueva herramienta acarrea a veces nuevas contrataciones, debido a la complejidad de su manejo y al aprovechamiento de las nuevas oportunidades que genera su mayor potencial.

Abordaremos el tema con una introducción general a los lenguajes de cuarta generación, para apreciar la evolución que derivó en las herramientas de la tecnología de información. Luego estudiaremos las herramientas de uso específico, es decir, aquéllas orientadas a temas precisos, como procesadores de texto, planillas electrónicas o consultas de bases de datos; algunos productos los pueden usar directamente los usuarios finales. Después, revisaremos las soluciones de software generalizadas, para concluir con las herramientas de apoyo para la producción de software, más conocidas como herramientas CASE.

Veremos:

- Evolución de los lenguajes de computador
- Herramientas de uso específico
- Una pirámide de soluciones
- Herramientas de apoyo para la producción de software

•••

Fin resumen

Puede adquirir la versión completa en formato papel o digital desde la página www.evolucion.cl o escribir a silviabravo@evolucion.cl. Cel. 9-2252004.

Si desea estudiar estos temas con mayor profundidad, en nuestra página www.evolucion.cl puede apreciar nuestros programas de cursos, diplomado y máster.