# COMPONENTES DE LÓGICA DEL NEGOCIO DESARROLLADOS A PARTIR DE PATRONES DE PROCESOS

**Oscar Barros V.**
Centro Gestión (CEGES)
Departamento de Ingeniería Industrial
Facultad de Ciencias Físicas y Matemáticas
Universidad de Chile

## Resumen

*Se presenta un enfoque que permite definir componentes genéricos de software, materializados por medio de* frameworks*, a partir de patrones de procesos de negocios. Tales componentes enfatizan la lógica del negocio que puede apoyar las diferentes actividades de un proceso. Un* framework *preconstruido permite desarrollar soluciones computacionales de apoyo a un proceso de negocio de manera mucho más rápida que partiendo desde cero y adaptadas o especializadas a las particularidades de un proceso específico.*

## 1. La Necesidad

Para satisfacer las necesidades de apoyo con Tecnologías de la Información a los diversos procesos de negocios que se ejecutan en las empresas, éstas han recurrido, básicamente, a dos enfoques. El primero es el desarrollo de soluciones a la medida, el cual implica, si es hecho de la manera correcta, rediseñar los procesos de negocios y, a partir de éstos, diseñar y construir los apoyos computacionales que permitan ejecutar tales procesos mejorados [3]. El segundo consiste en usar software empaquetado del tipo ERP, CRM o paquetes más primarios, los cuales proveen algunas funcionalidades que contienen implícitamente ciertas prácticas de negocios. Estas deberían dar lugar a un cambio en los procesos de negocios, pero habitualmente sólo se implementa el software. Los paquetes de mayor nivel,

que son también los más caros, facilitan la adopción o adaptación de ciertas prácticas negocios, por medio de las llamadas parametrizaciones, las cuales permiten un cierto grado de flexibilidad [3]. Con esto es posible rediseñar los procesos en alguna medida, lo cual se hace en escasas oportunidades, dado lo complejo y caro de las parametrizaciones y el riesgo de tener que rehacer el trabajo cuando hay cambio de versión del software.

En resumen, la tendencia actual, al usar software empaquetado, es hacer adaptaciones mínimas del mismo a los procesos de negocios de la empresa y ojalá usarlo tal como es, para acelerar y disminuir el costo de las implementaciones. Esto es evidentemente verdadero, con mayor razón, cuando se utilizan paquetes económicos –que son esencialmente rígidos- en las PYMES.

Se presenta aquí un enfoque original que trata de incorporar las mejores características de los enfoques anteriormente bosquejados: la flexibilidad del desarrollo a la medida –que permite mejorar los procesos de negocios por medio de un rediseño y tener software perfectamente adaptado al mismo- y la rapidez y potencial más bajo costo de una solución de software preconstruida, lista para usar, que provee el segundo enfoque.

Nuestra propuesta se basa en dos ideas fundamentales: el rediseño de los procesos de negocios a base de patrones preestablecidos [3] y el uso de componentes de software preconstruidos, los cuales están desarrollados a partir de los patrones.

La estrategia que proponemos consiste, entonces, en comenzar con un rediseño de un proceso de negocio de una empresa, a partir de un patrón que establece cómo debería funcionar un proceso en un cierto dominio, de acuerdo a las mejores prácticas conocidas [3]. Al hacer esto, se toman en cuenta las condiciones existentes en la empresa donde existe el proceso, para factibilizar las mejoras. Una vez establecido un rediseño, se recurre a los componentes de software, previamente desarrollados –a partir de los mismos patrones que apoyan el rediseño-, los cuales se especializan, usando tecnología de orientación a objetos, al caso particular, para conformar una solución de software específica para el proceso en cuestión.

En este documento mostramos cómo, utilizando una metodología original y herramientas de punta para diseño e implementación de software orientado a objetos, es posible generar componentes genéricos de software, a partir de patrones de procesos que incluyen mejores prácticas. El centro de atención está en procesos que tengan una alta componente de lógica del negocio y que involucren decisiones importantes en relación a cómo satisfacer los requerimientos de los clientes y tender a un uso óptimo de los recursos de la empresa. Además, mostramos cómo los componentes de software genéricos pueden internalizar la lógica del negocio de una manera flexible, la cual permite a una empresa optar por diferentes posibilidades de manejo de las decisiones del proceso y complementarla si fuese necesario. Para lograr esto desarrollamos el concepto de lógica de negocio incremental.

Por último, estableceremos un procedimiento de especialización de los componentes a un rediseño particular para la conformación de una solución de software al proceso en cuestión.

Nuestro enfoque se diferencia de otros trabajos, que han propuesto ideas para desarrollar componentes de software que implementen verdaderos objetos del negocio [5, 7, 9, 10] en que logra una integración explícita con el diseño del negocio inédita hasta ahora.

Por ser las PYMES las posibles mayores beneficiadas con un enfoque como el propuesto –dada la imposibilidad que tienen de contratar asesoría cara para mejorar sus procesos y desarrollar software a la medida, o comprar software empaquetado de clase mundial-, restringimos nuestro planteamiento a lógica del negocio simple, pero rigurosa, adecuada para este tipo de empresas. Esto no implica que el enfoque no sea generalizable a soluciones con mayor grado de sofisticación para grandes empresas.

Creemos que el enfoque propuesto define una nueva manera de hacer software de apoyo a procesos, que puede originar la creación de una industria de software proveedora de componentes orientados al negocio, lo cual permitiría soluciones más flexibles y económicas, particularmente relevantes para PYMES, como ya se indicó.

## 2. Patrones de Procesos de Negocios

### 2.1. Planteamiento general

La idea de patrones de procesos se encuentra detallada en un libro [3], otras publicaciones [2] y el sitio web del autor [12]. Por lo tanto, damos aquí un muy breve resumen de sus ideas fundamentales.

Los procesos existen en las empresas, pero su funcionamiento ha sido el fruto de la historia y la experiencia. Dada la naturaleza funcional de las organizaciones, los cambios y mejoras han sido puntuales, en las actividades bajo una gerencia de área, pero rara vez sistémicos y orientados al funcionamiento y al cumplimiento de los objetivos de una empresa en su conjunto, lo cual hace que, en general, los procesos de negocios sean extremadamente ineficientes. De aquí se originó la idea de rediseño de procesos, que consiste en tomar las actividades de un proceso en su totalidad y someterlas a un cambio fundamental –el cual habitualmente implica un uso intensivo de Tecnologías de la Información– que garantice un desempeño claramente mejorado del mismo. Por ejemplo, en el caso de crédito hipotecario, a base de un flujo electrónico (*workflow*) de los documentos, eliminando pasos y autorizaciones innecesarias, conseguir una reducción significativa del tiempo de curse de una operación.

La experiencia muestra que el rediseño de procesos lleva a soluciones similares en procesos del mismo tipo. Por esto, no hay razón alguna para pensar que un rediseño optimizado del proceso de crédito hipotecario debiera ser muy diferente de un banco a otro. Asimismo, el proceso de satisfacción de pedidos rediseñado en una empresa de distribución no tiene que tener diferencias fundamentales de otra del mismo rubro y el proceso de atención de urgencias rediseñado en un hospital público no debería diferir del de otro.

La idea de procesos comunes a muchas empresas puede desarrollarse observando que, en cualquier organización, hay un número pequeño de procesos –entre 7 y 15– y que cada uno de ellos, además de tener una arquitectura o estructura común que comparte con los otros, es muy parecido en su esencia en diferentes contextos [1]. Así, este autor ha demostrado que los procesos de crédito hipotecario en un banco, la satisfacción de pedidos en una empresa manufacturera, la atención de urgencia en un hospital y muchos otros corresponden a instancias de una estructura común –con actividades y relaciones del mismo tipo– de un proceso que ocurre en todas las organizaciones y que genera el producto o servicio que los clientes externos demandan. A esta estructura común la llamamos patrón de proceso [3].

La consecuencia de definir patrones de procesos en detalle –que toman la forma de modelos gráficos de fácil comprensión– reside en que en ellos se pueden internalizar las mejores prácticas desarrolladas en muy diferentes contextos, conformando una acumulación de conocimiento normativo respecto a cómo debe ejecutarse un determinado tipo de proceso y, en consecuencia, de cómo debe realizarse la gestión. Ahora, si estos patrones son públicos y muchas empresas pueden cooperar en su desarrollo y usufructuar de ellos, se tiene un instrumento muy poderoso y estructurado para compartir conocimiento de gestión. Esto permitiría que numerosas organizaciones medianas y pequeñas pudieran mejorar sus procesos, sin tener que empezar desde cero, lo cual tiene obvias implicancias en cuanto a aumento de productividad. Esta idea la está implementando el autor por medio de publicar los patrones que ha desarrollado en su sitio web [12].

## 2.2. Un patrón de proceso particular

Para ilustrar en más detalle un proceso específico y, en particular, cómo se especifica la lógica del negocio en forma genérica, presentamos algunas partes seleccionadas del patrón Macro1 [3]. Este patrón cubre el dominio de aplicación de las empresas que venden y satisfacen requerimientos de productos y/o servicios de cualquier tipo. Para poder ser más precisos -dada la generalidad de este patrón- especializaremos Macro1 al dominio de empresas PYMES que venden productos o servicios fabricados o generados en sus propias instalaciones; por ejemplo, hemos tomado como casos específicos, para definir esta especialización, una elaboradora de quesos, una fábrica textil, una imprenta, una empresa de capacitación que ofrece cursos y los ejecuta en sus instalaciones, un hospital de una asociación de seguridad que ofrece servicios de tratamiento de lesiones causadas por accidentes y varios otros con características similares.

El primer nivel de detalle de Macro1 se muestra en la Figura 1, donde se modela el proceso usando la metodología IDEFO. De este modelo, detallamos la actividad "Administración relación con el cliente" en la Figura 2. De aquí tomamos "Decidir satisfacción requerimientos" para especificarla más aún. Esta incluye el conjunto de actividades que permiten decidir, en este caso, si satisfacer o no un pedido por productos o servicios realizado por un cliente particular. Aquí nos centramos en la especificación de la lógica de negocio que debe ejecutarse para tomar esta decisión. De acuerdo al objetivo de los patrones y de este trabajo, intentamos sintetizar una lógica genérica aplicable a muchas empresas dentro del dominio, a partir de las mejores prácticas conocidas y la experiencia de varias empresas. Por tratarse de empresas PYME, tratamos de que la lógica sea lo más simple posible, pero suficiente para una buena decisión. Además, hacemos una serie de simplificaciones adicionales, en la lógica misma, para facilitar su entendimiento. Sin embargo, ella puede complementarse para incluir todos los aspectos no considerados.

De las mejores prácticas y la experiencia con este tipo de empresas, existen dos aspectos de la lógica que son válidos para todas las empresas de este dominio y que son separables. El primero es la verificación de la factibilidad de proveer el producto o servicio y el segundo, de la confiabilidad del cliente. La primera lógica es muy simple e incluye: verificación de si existe disponibilidad o capacidad para proveer el producto o servicio; reserva de disponibilidad o capacidad en caso positivo; y generación de alguna solución o alternativa en caso negativo.

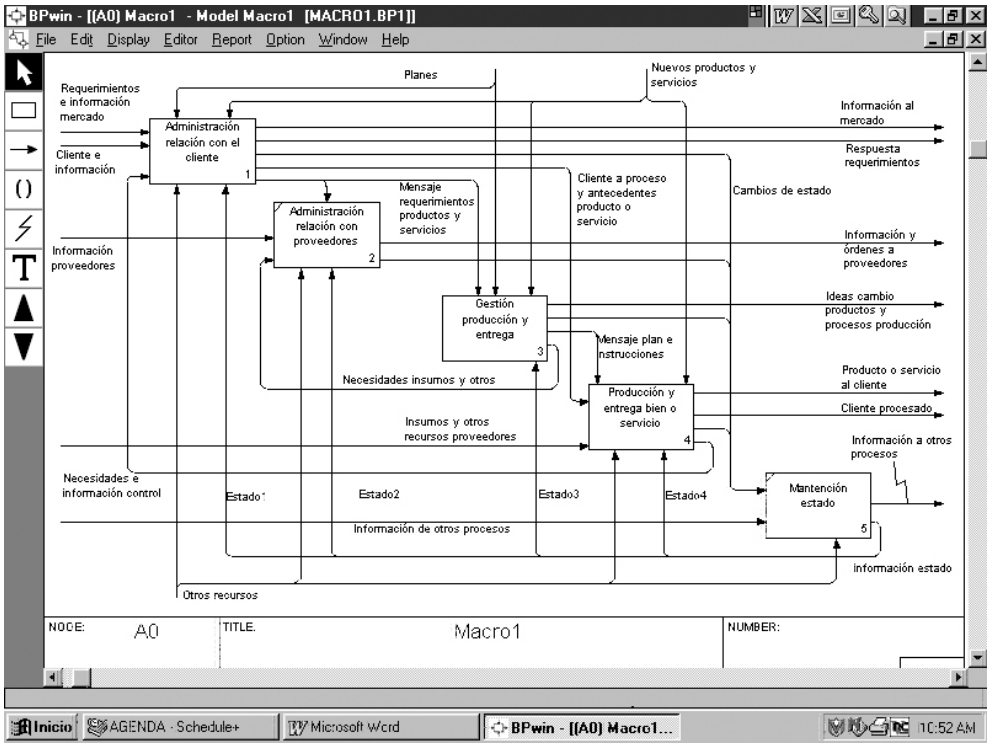



Figura 1. Macro1

La lógica de evaluación del cliente para darle el crédito normal de pago diferido de facturas (30 días o más) o de otro tipo, es más elaborada y ofrece múltiples posibilidades, las cuales entregamos, a continuación, en orden de complejidad:

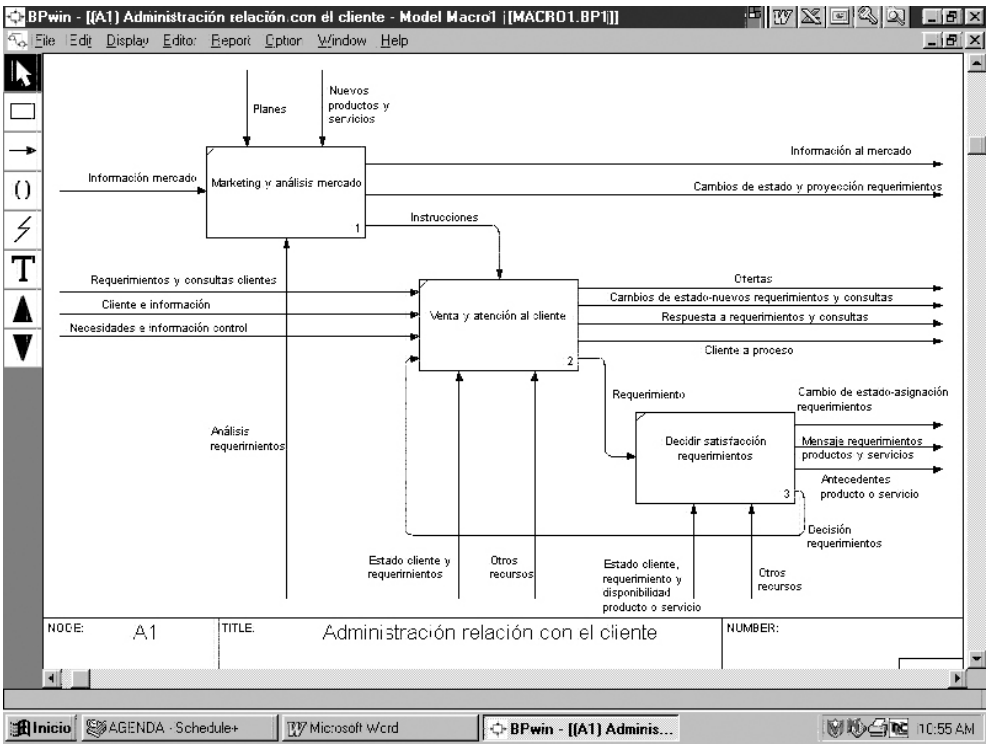



Figura 2. Descomposición de "Administración relación con el cliente"

a) Lógica primaria de clasificación, la cual se basa en la idea de clasificar los clientes de acuerdo al comportamiento histórico de ventas o actividades realizadas para ellos. Suponiendo que los clientes son empresas, las variables de evaluación son: *Ventas anuales* efectuadas a una empresa; *Impagos,* que son las facturas a la empresa declaradas incobrables; y *Atrasos,* el cual es un contador que establece cuántas veces en la historia de la relación con la empresa, ésta se ha atrasado en el pago de las facturas. Con estas variables se puede formalizar la lógica de la Figura 3 que asigna una empresa a una categoría entre cinco. Obviamente, la lógica está simplificada: se podrían considerar más combinaciones de variables e introducir variables adicionales como DICOM.

```
//Calcular clasificación primaria
    Si (Ventas anuales ≥ A e Impagos = 0)
        Clasificación primaria  = 5
    Sino
        Si (Ventas anuales ≥ B e Impagos = 0 y Atrasos = 0)
        Clasificación primaria  = 4
        Sino
            Si (ventas anuales ≥ C e Impagos = 0 y Atrasos = 0)
                Clasificación primaria = 3
               Sino
                Si (Impagos = 0 y Atrasos > 0)
                    Clasificación primaria = 2
                Sino
Clasificación primaria = 1
```

Figura 3. Lógica de Clasificación Primaria

b) Lógica secundaria de clasificación, que tiene como fundamento evaluar la solvencia financiera del cliente a partir de su balance. Esto puede hacerse con índices estándares derivados del mismo, como el test ácido de liquidez. Entonces, la lógica de negocio consiste en asignar los clientes a varias categorías de acuerdo a que los diferentes índices o combinaciones de ellos cumplan con ciertas condiciones, la cual no detallaremos aquí.

c) La lógica primaria de aprobación de crédito por un producto o servicio se basa en la clasificación primaria y se bosqueja en la Figura 4.

d) La lógica secundaria de aprobación sería similar a (c), pero basada en la clasificación secundaria.

Evidentemente, se podrían definir lógicas incrementales ternarias, cuaternarias, etc., agregando nuevos niveles de complejidad a los previamente establecidos. Estas lógicas incrementales contienen opciones que un diseñador de procesos podría utilizar en un caso particular; por ejemplo, podría elegir sólo la lógica primaria o la primaria y secundaria, etc., para, a partir de ellas, elaborar un rediseño específico.

```
//Lógica primaria de aprobación crédito
        Si Clasificación primaria = 5
            Aprobación crédito = sí
        Sino
            Si Clasificación primaria = 4 ó 3
                Si Saldo cuenta corriente + Monto pedido ≤ Límite crédito
                    Aprobación crédito = sí
                    Sino
                    Aprobación crédito = no
                Sino
                  Aprobación crédito = no
```

Figura 4. Lógica primaria de aprobación crédito

Consideremos, ahora, la actividad del proceso Macro1 de "Gestión producción y entrega", cuyo detalle se encuentra en la Figura 5. Concentrémonos en la actividad "Planificación y control producción" y veamos la lógica de negocio de ella. Para precisar, sigamos en el mismo dominio de PYMES que tienen actividades de fabricación de productos o generación de servicios que se satisfacen por medio de sus instalaciones. Supongamos que hay una cierta rutina –que no especificaremos- que genera requerimientos por productos o servicios, que llamaremos, en general, Tareas, las cuales deben ejecutarse de acuerdo a una cierta prioridad. Por ejemplo, pueden ser Ordenes de Trabajo (O/T), órdenes de tratamiento de pacientes, órdenes de ejecución de cursos de capacitación, etc.

Las Tareas pueden ejecutarse en varios recursos opcionales, pero utilizan sólo uno de ellos; por ejemplo, líneas de envasado de bebidas, yoghurt, queso y otros productos alimenticios, máquinas impresoras de distinta capacidad, equipos de rayos X, pabellones de operación, salas de clases, etc.

La lógica primaria es, en este caso, simple y consiste en la programación de las Tareas para satisfacer las siguientes condiciones:

i) Un Recurso programado con una Tarea debe tener la funcionalidad requerida por ésta.

ii) Un módulo horario disponible de un Recurso debe asignarse a una sola Tarea.

iii) Las Tareas deben programarse en orden de prioridad; por defecto es FIFO.

iv) Tratar de maximizar el uso de Recursos en el tiempo.

v) Los Recursos deben usarse en forma equilibrada: todos deben tener una carga similar.

Por supuesto, hay simplificaciones implícitas, como que el orden de las Tareas no importa, lo cual sí es relevante en casos en que los tiempos de preparación entre Tareas sean significativos y dependan del orden en que se ejecutan las mismas; por ejemplo, líneas de producción que hay limpiar después de fabricar un producto, donde el orden –de menos contaminante a más contaminante- puede influir en los tiempos de lavado.

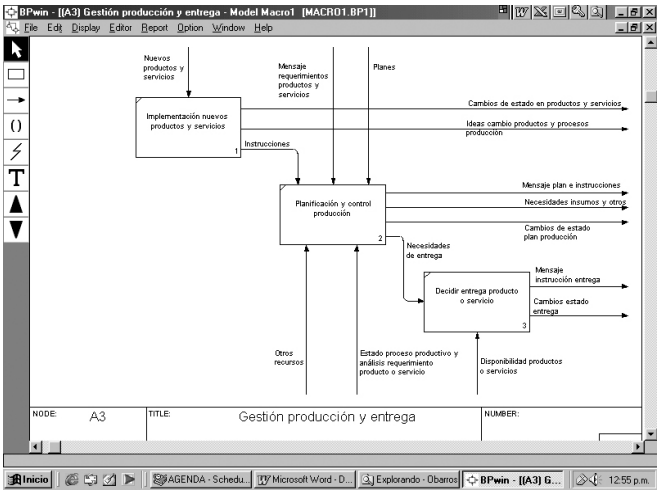



Figura 5. Detalle de "Gestión producción y entrega"

Un segundo nivel incremental de complejidad para la lógica, en este caso lógica secundaria, obvía el problema anterior. Esto se puede hacer definiendo grupos de Tareas que deberían, dentro de lo posible, programarse en forma secuencial. Adicionalmente, se define un factor de calidad de un Recurso para ejecutar una determinada Tarea; o sea, algunos Recursos son más apropiados que otros. Por lo tanto, la lógica secundaria reordenaría las asignaciones de Tareas a Recursos, respetando los siguientes criterios, adicionales a los de la lógica primaria:

i) Las Tareas correspondientes a un grupo deben –dentro de lo posible- asignarse secuencialmente en las líneas de tiempo de un Recurso.

ii) Los grupos deben asignarse al mejor Recurso disponible.

iii) Hay que respetar, dentro de cada grupo y a nivel global, las prioridades, particularmente si éstas están relacionadas con fechas requeridas de término de la Tarea.

Se puede definir un tercer nivel de complejidad para situaciones en las cuales participa más de una línea o máquina en el procesamiento de una Tarea, de manera similar a lo ya explicado, lo cual omitimos en este documento.

La idea de lógica incremental que hemos propuesto puede funcionar bien en muchos casos. De hecho existen precedentes de buenas heurísticas que utilizan esta estrategia; por ejemplo, en ruteo de vehículos para distribuir pedidos a múltiples ubicaciones, se ha propuesto asignar primero los pedidos a los vehículos y después determinar las rutas de éstos (6). Sin embargo, en algunos casos, puede haber demasiada interrelación entre una lógica primaria y otra secundaria, lo cual hace necesario refundirlas en una sola y proveerla como otra lógica alternativa. En tal caso, un diseñador tendría como opciones una lógica primaria, una primaria seguida de una secundaria o una alternativa que refunde a las anteriores. La racionalidad que está detrás de estas opciones es que un diseñador pueda elegir el nivel estrictamente necesario de complejidad de lógica para su caso particular. El uso de esta idea se facilita con la tecnología de orientación a objetos, ya que, como lo veremos más adelante, estas opciones se pueden empaquetar en componentes de software de entre los cuales se puede seleccionar en forma eficiente.

Las lógicas presentadas para las dos actividades de Macro1 son, evidentemente, parciales, y sirven sólo para ilustrar el enfoque propuesto. En forma similar se podrían completar tales lógicas y desarrollar lógica adicional para otras actividades de Macro1, como gestión de stock bajo "Administración relación con proveedores" y gestión logística bajo "Decidir entrega producto o servicio".

## 2.3. Aplicación a un caso particular

Ilustramos cómo se aplica un patrón general como el del Punto 2.2, y su correspondiente lógica genérica, a un caso particular, considerando el caso de una fábrica que confecciona productos para stock y vende sólo a empresas.

A nivel de representación del proceso no hay cambio alguno; lo que se especializa es la lógica de negocio, de la siguiente manera:

a) La verificación de la factibilidad de proveer los productos solicitados consiste en determinar su disponibilidad en stock; reserva de stock en caso positivo y oferta de entrega parcial en caso parcialmente positivo; y verificación de productos en fabricación para ofrecer entrega futura.

b) La lógica primaria de clasificación aplica sin cambio alguno.

c) La lógica secundaria de clasificación podría obviarse en este caso, dado que las empresas manufactureras PYME tienen pocas posibilidades de obtener los balances de sus clientes.

d) La lógica de evaluación de pedido aplica también sin cambio, restringiéndose al uso de la lógica primaria.

e) La lógica primaria de programación de las Tareas se convierte en la asignación de Ordenes de Trabajo (O/T) –que se generan para reposición de stock- a líneas de producción o máquinas, siendo los criterios de la misma directamente aplicables en este caso.

f) La lógica secundaria de programación de Tareas no es aplicable, ya que en la fabricación considerada no existen tiempos de preparación significativos.

g) Una posible lógica de programación terciaria sería relevante si las O/T's requirieran un procesamiento en más de una máquina. Aquí la opción simple es determinar el cuello de botella de la fabricación –si es que existe- y aplicar (e) y (f) al mismo.

# 3. Componentes genéricos para implementar lógica del negocio

## 3.1. Derivación de un *framework*

A partir de las lógicas de negocios asociadas a los patrones –desarrolladas en el Punto 2.2-, es posible definir en forma directa clases de objetos genéricas o componentes que conforman un *framework,* el cual puede ser aplicado, como punto de partida, para desarrollar una solución de software para apoyar un proceso particular. Los *framework* que presentaremos se distinguen de los habitualmente descritos en la literatura [11] en que contienen lógica del negocio, en vez de lógica computacional –conformando verdaderos **objetos del negocio** (*business objects*)–; son modelados utilizando UML especializado a aplicaciones que serán implementadas en una plataforma Internet [8], en la cual existirán, a lo menos, tres capas: un cliente con browser, un servidor de aplicaciones y un servidor de datos [4]; y que permiten un uso opcional e incremental, idea no desarrollada hasta ahora en la literatura. Asimismo, aclaramos que el énfasis de los *framework* estará en la lógica, y sólo se incluirán los datos estrictamente necesarios. Una metodología de cómo generar modelos genéricos más orientados a los datos se da en [3].

El primer *framework* que desarrollamos es el correspondiente al apoyo a "Decidir satisfacción requerimientos". A partir de la lógica definida en el Punto 2.2, se deduce el *framework* que se presenta en la Figura 6. En éste hay clases

tradicionales que mantienen y manipulan datos y clases que manejan sólo lógica. La idea detrás de esta separación -habitual en aplicaciones en Internet [8]– es que los datos van a residir en servidores de datos y la lógica, en servidores de aplicaciones; además tener la lógica aislada posibilita la especialización del *framework* a casos particulares, agregando lógica especial de éstos. Ello es facilitado por la división de la lógica en niveles de complejidad incremental –primario, secundario, etc.–, que pueden ser utilizados parcial o totalmente, de la manera que explicaremos más adelante.

Las lógicas (métodos) que contienen las clases respectivas son, evidentemente, las ya definidas en el Punto 2.2.

La manera en que estas clases colaboran para procesar pedidos en línea de los clientes se muestra, por medio de un diagrama de secuencia UML, en la Figura 7, a un nivel conceptual. Este requiere, obviamente, de una serie de detalles adicionales –los que se proveerían por especialización- para ser realizable y ejecutable en un caso particular.

De la misma manera, se puede desarrollar un *framework* para apoyo a la actividad "Planificación y control de producción" de Macro1. De la lógica incremental presentada en el Punto 3.2, se deduce el *framework* de la Figura 8. La idea que implementa este *framework* es que una empresa podría elegir utilizar sólo la parte central del mismo, usando sólo la lógica primaria y otros complementos particulares del caso, por especialización. Esto llevaría a una programación de las Tareas en los Recursos.

Complementariamente o incrementalmente, otra empresa podría decidir usar el *framework* completo con la lógica primaria y secundaria. En tal caso, sobre la programación de la primera se ejecuta la agrupación que produce la segunda, reprogramando el comienzo de cada Tarea y cambiando el estado de cada Recurso.

Asimismo, alternativamente, se podría tener un componente adicional que refundiera la lógica primaria y secundaria, para casos en que la separación no produce buenos resultados.

Presentamos, a continuación, una aplicación diferente de las dadas anteriormente, para mostrar el uso del *framework* anterior y la versatilidad del enfoque.

Consideremos una empresa de capacitación cuyas Tareas son Cursos que deben ser programadas en los Recursos, que son Salas. La especialización del *framework* de la Figura 8 a este caso se muestra en la Figura 9, considerando que es relevante sólo el nivel primario. Si hubiera que asignar, además, profesores, se podría aplicar separadamente el mismo *framework* para ello, con los cambios obvios. Pero, para la asignación de profesores, podría haber la necesidad del uso de lógica secundaria de agrupación, ya que los cursos asignados a un profesor deberían ser, deseablemente, contiguos para -al ser contratados para hacer clases por horas- hacer más eficiente el uso de su tiempo.

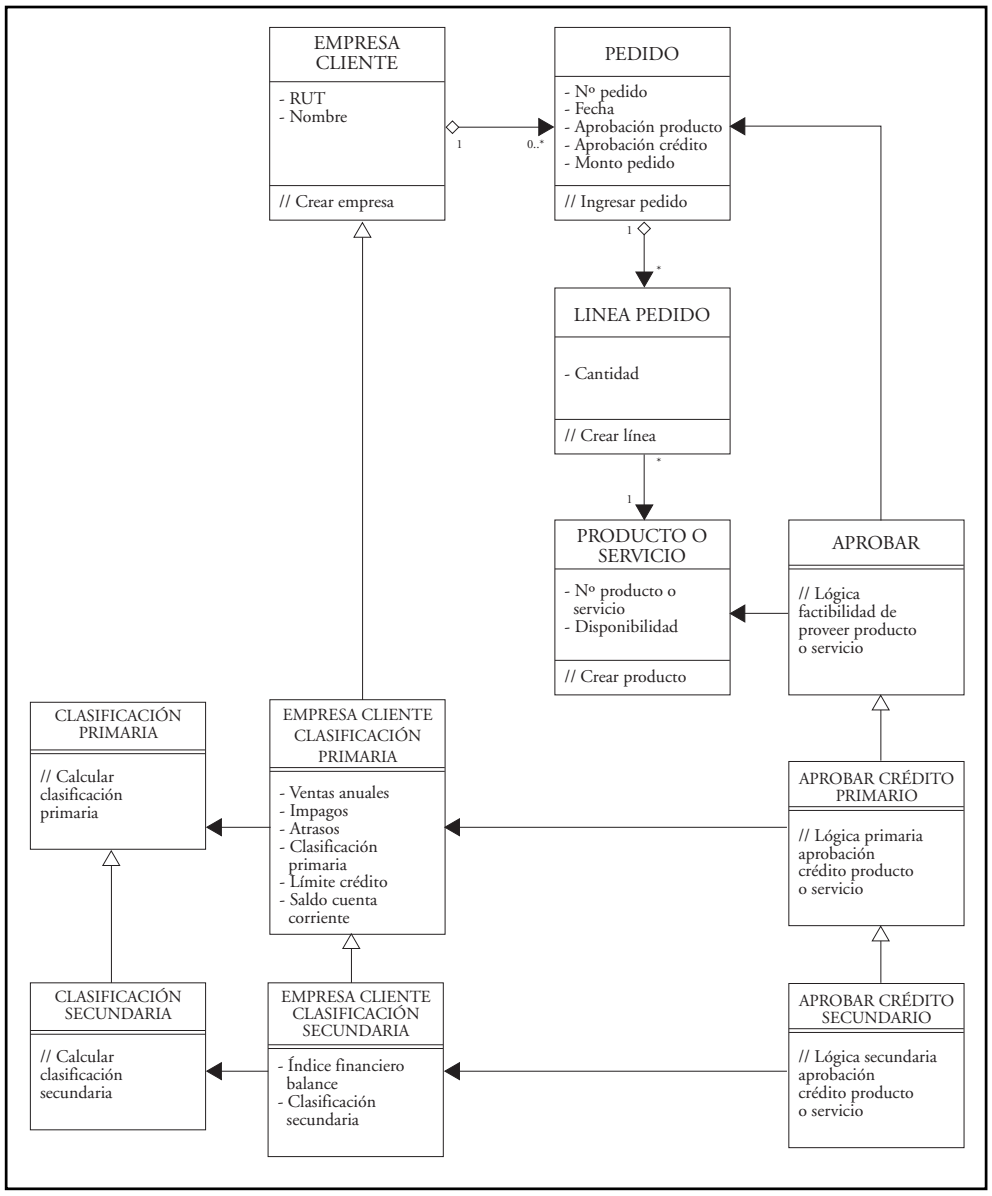



Figura 6. *Framework* de lógica satisfacción requerimientos

Es evidente que las lógicas incrementales que hemos propuesto, particularmente en el caso de la asignación de Tareas, son una muy gruesa aproximación a lo que sería una solución rigurosa del problema de optimización subyacente. Sin embargo, la idea fuerza es que en las PYMES se requieren, en muchos casos, soluciones factibles y que garanticen condiciones mínimas, como orden en el uso de recursos y cumplimiento de plazos, lo cual -comparado con la ausencia absoluta de mecanismos formales de evaluación de clientes y asignación de recursos que actualmente se da- puede significar beneficios considerables.

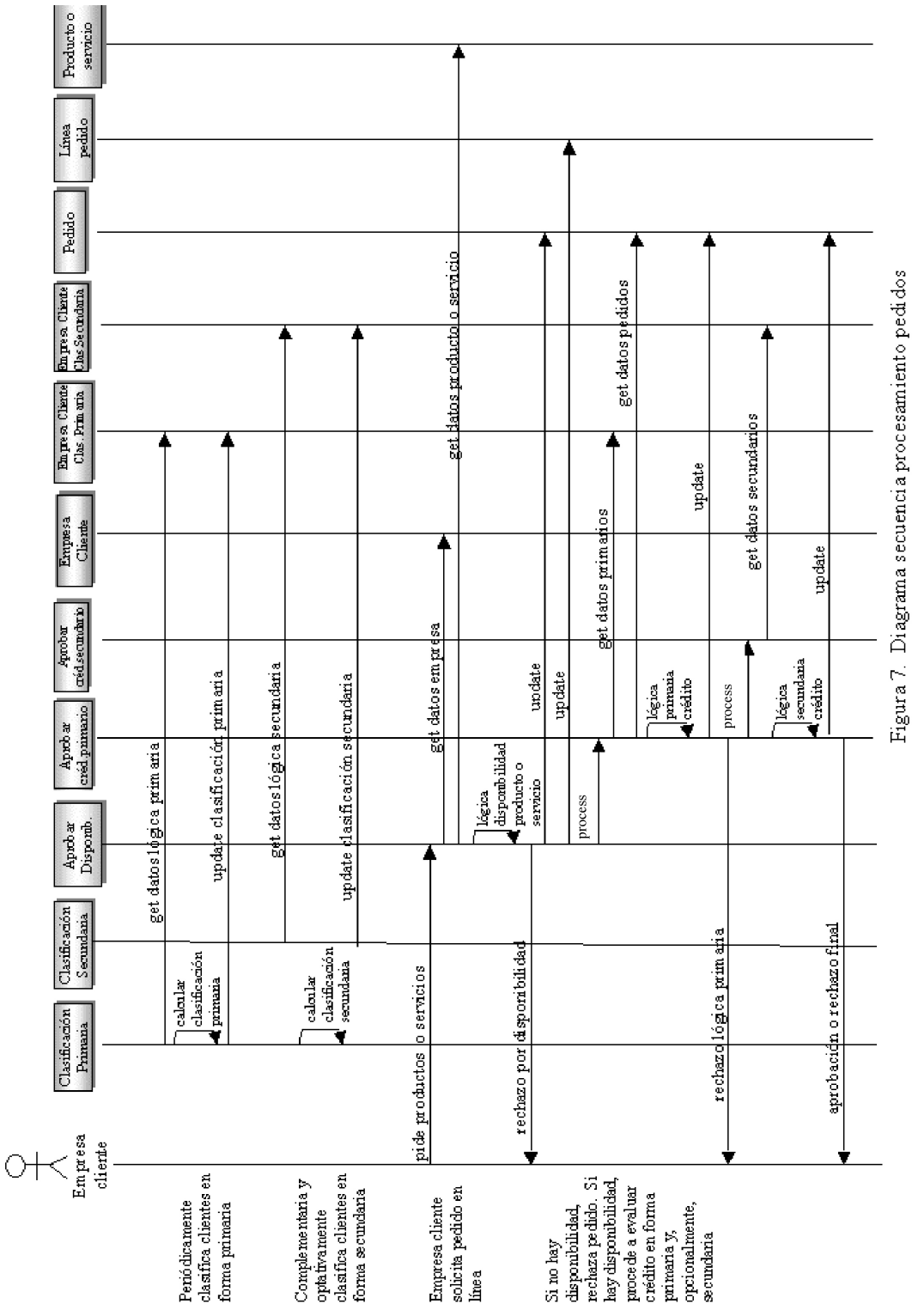
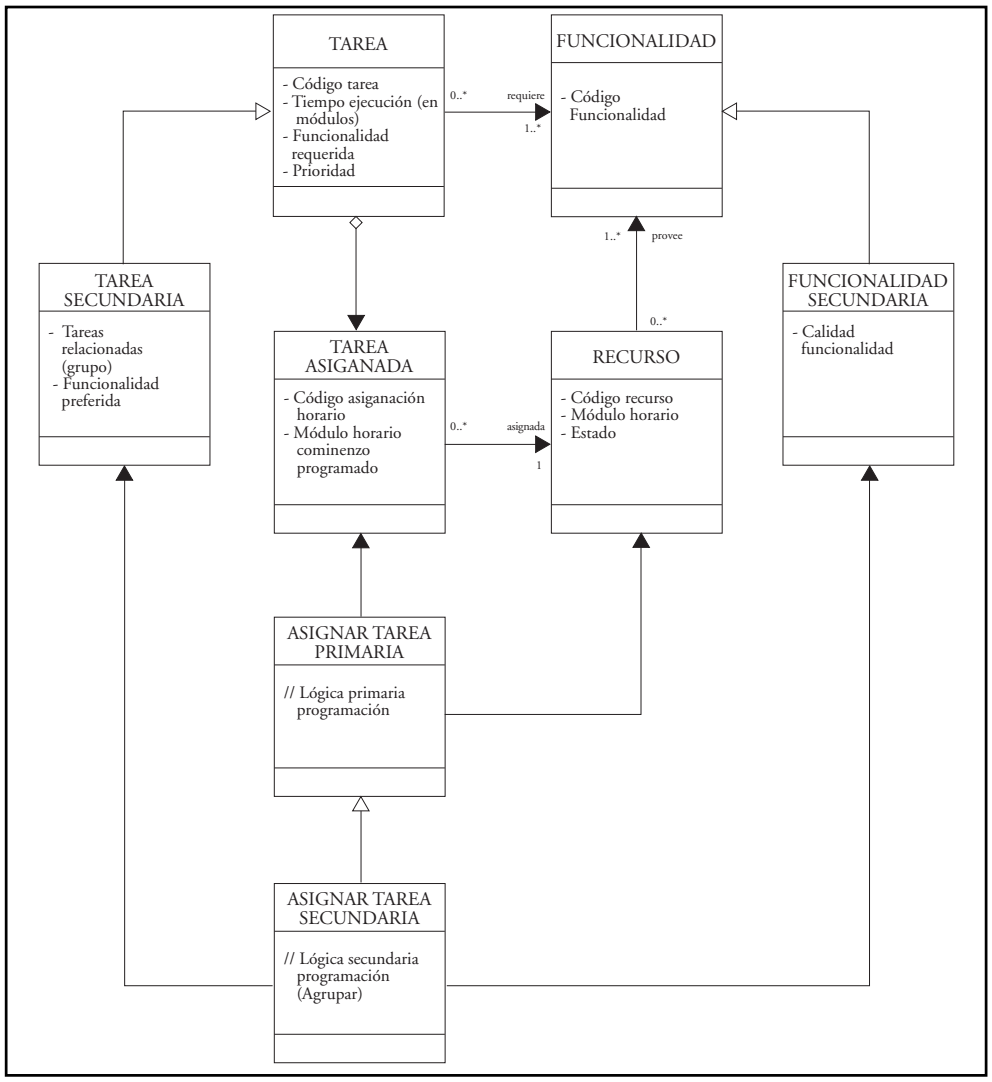



Figura 7. Diagrama secuencia procesamiento pedidos




* Pueden ser varias; relación recursiva

Figura 8. *Framework* para programación de Tareas

Por lo tanto, la ausencia de optimización –que sería bastante más cara de alcanzar- puede que no signifique una pérdida demasiado significativa de beneficios netos.

De manera parecida a lo explicado, se podrían desarrollar lógicas del negocio y *framework* para otras actividades del patrón Macro1, como "Administración relación con el proveedor", en particular gestión de stock de productos terminados y de materias primas; y "Decidir entrega producto o servicio", por ejemplo, para el armado de cargas para vehículos y ruteo de la distribución.

## 3.2. Uso de un *framework*

Como ya se ha bosquejado, un *framework,* como los descritos en el punto anterior, es el punto de partida para desarrollar una solución de software de apoyo a un

proceso. Para que esto ocurra, el proceso tiene que haber sido rediseñado a partir del patrón de proceso que originó el *framework,* como se muestra en la Figura 10. Entonces, a partir del rediseño y el *framework* se genera una especialización de este último, la cual incluye la selección del nivel de lógica incremental con el cual se trabajará y la complementación del mismo con lógica adicional y datos necesarios. A partir de este *framework* especializado y el rediseño del proceso se puede, entonces, construir el software definitivo de apoyo a éste, desarrollando una serie de elementos complementarios, como interfaces con los usuarios, integración de datos utilizados en diferentes *framework,* programas para incluir lógica y datos adicionales en el *framework*, programas que implementen la colaboración entre clases, de acuerdo a una especificación como la presentada en la Figura 8, y otras tareas similares.

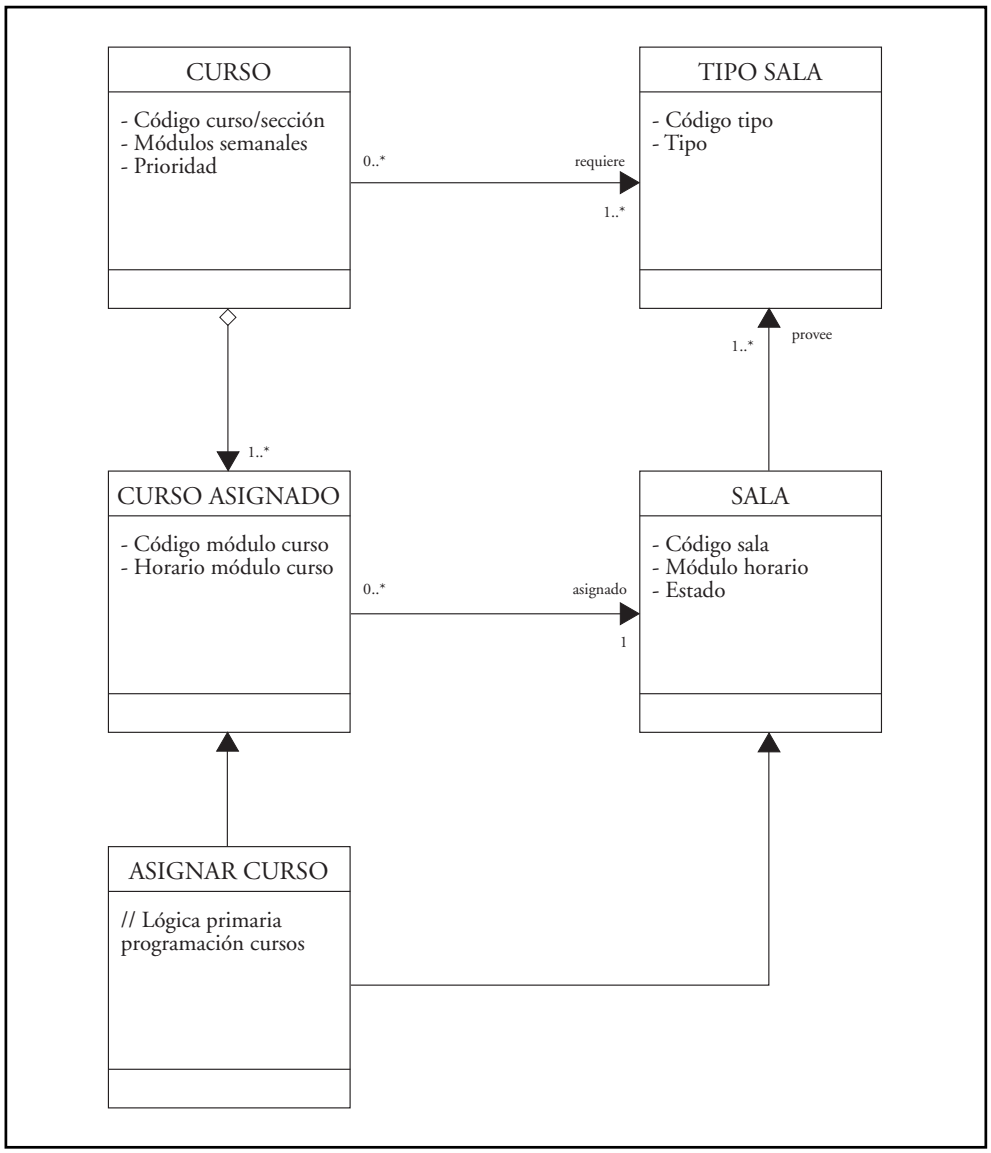



Figura 9. Aplicación *framework* programación Tareas a asignación de cursos

El esquema presentado de generación y utilización de un *framework* ha sido validado a base de programar las lógicas de satisfacción de requerimientos de la Figura 6 y de asignación de tareas de la Figura 8, para utilizarlas en la generación de software para casos particulares(*). Para ello se usó una base de datos relacional para implementar las clases que manejan datos, Java para programar las clases de lógica y JDBC para acceder a datos desde la lógica.

La idea de uso incremental de la lógica de los *framework* se probó factible en la práctica, al construir aplicaciones específicas tomando parte de las clases de un *framework* y programando especializaciones y complementos a ellas para construir una solución particular.

## 4. Consecuencias y extensiones

Este trabajo ha mostrado la factibilidad preliminar de un enfoque que permite desarrollar apoyos de software a un proceso rediseñado, a partir de componentes preconstruidos. Estos están centrados en la lógica del negocio y permiten su uso parcial y flexible, a partir de las ideas de lógica incremental y lógica alternativa. Si bien otros autores han propuesto ideas en la misma dirección de este documento para crear verdaderos objetos del negocio [5, 7, 9 y 10], las publicaciones y oferta comercial existente –de acuerdo al conocimiento del autor- no han logrado el grado de integración entre el diseño del negocio y los componentes genéricos de software de apoyo aquí desarrollado, o se han quedado en planteamientos generales que

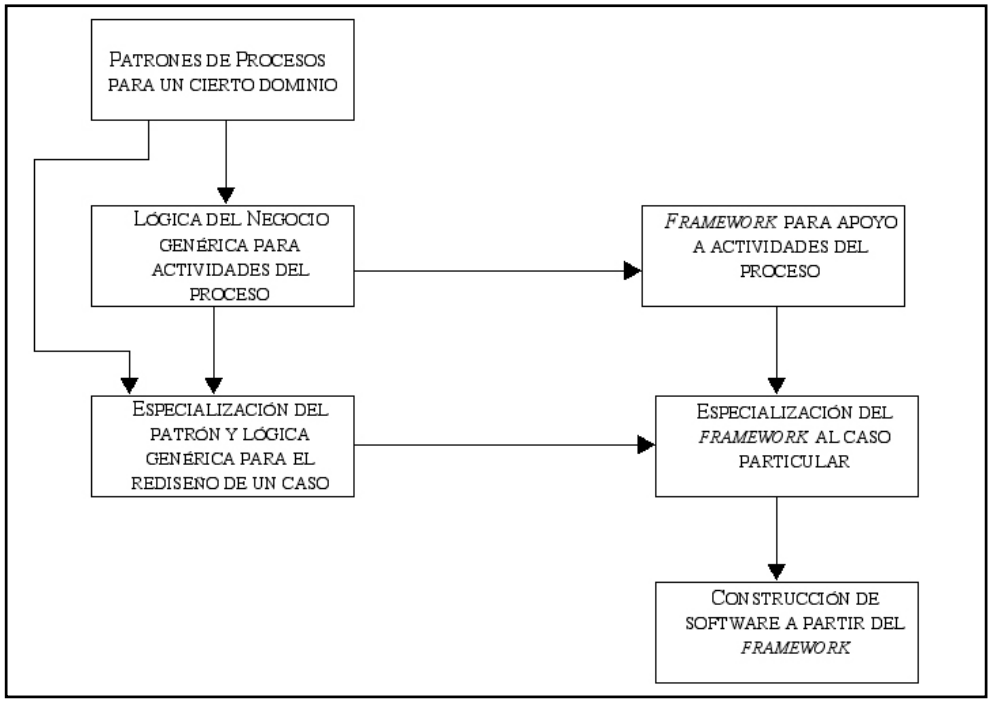



Figura 10. Diagrama de flujo para utilización de patrones y *framework*

no son operacionalizables. Lo que distingue nuestro enfoque y lo hace factible es su sólido fundamento en patrones prediseñados de procesos de negocios. Esta idea es única y no ha sido propuesta en ninguno de los otros trabajos conocidos.

Creemos que el enfoque tiene un buen potencial para realizar rediseño de procesos y construir soluciones adaptadas de apoyo a los mismos en las empresas. Esto debido a que la existencia de patrones de procesos de negocios prediseñados, adaptados a un cierto dominio y lógica de negocio con diversas opciones, hace factible que comunidades de empresas usuarias y desarrolladores de software se coordinen en la construcción de componentes genéricos –a partir de los *framework*- adaptables a situaciones particulares. Evidentemente, esto requiere un acuerdo dentro de un grupo de empresas, lo cual se puede conseguir de varias maneras.

En el caso de una organización que incluye muchas instancias del mismo tipo de negocio –estructuradas por regiones, por ejemplo-, como el caso de cadenas comerciales, holding de empresas productivas, sistemas hospitalarios públicos y privados, servicios públicos del estado y empresas financieras, la decisión de usar un enfoque como el propuesto es una política corporativa. De hecho, el autor ha estado trabajando con un grupo del sector telecomunicaciones y otro de empresas industriales en la implementación del enfoque y está elaborando un proyecto para servicios públicos.

Pero el autor ve mayor potencialidad y necesidad de un enfoque con el propuesto en las empresas PYMES, que, como ya se ha dicho, no tienen otra alternativa de calidad aceptable. Para éstas el problema de acuerdo y coordinación es más difícil, pero no insuperable, ya que hay presiones competitivas que hacen necesaria una mayor tecnologización y eficiencia de tales empresas. Además, el Gobierno ha reconocido el problema y apoya varias iniciativas al respecto. Por lo tanto, aparece como factible unir empresas PYMES en sectores particulares - exportadores, por ejemplo- para aplicar un enfoque como el propuesto.

Alternativamente a lo anterior, el autor está desarrollando una oferta de patrones para empresas PYMES –que estarán públicamente disponibles a través de un sitio web- a partir de la experiencia de muchas empresas de este tipo con las cuales se han realizado proyectos de rediseño de procesos. De aquí se derivará una primera generación de *framework* y componentes preconstruidos que se ofrecerá a tales empresas. Con éstos se harán aplicaciones en empresas específicas para mostrar la viabilidad del enfoque.

Es claro que lo que existe como posibilidad –de resultar el enfoque en la práctica para empresas PYMES- es la creación de una industria original de software para tales empresas, con buenas posibilidades de exportación, ya que no hay una buena solución disponible a nivel internacional para ellas, particularmente si se consideran los aspectos de mejora de procesos del negocio. Al respecto, la experiencia con el software chileno de una generación anterior para empresas PYMES es alentadora, ya que con una solución absolutamente tradicional, los desarrolladores nacionales lograron un buen posicionamiento a nivel latinoamericano.

## Referencias


1. Barros, O. *Reingeniería de Procesos de Negocios: Un Enfoque Metodológico*. Dolmen, 2ª edición, 1995.
2. Barros, O. Patrones de Procesos de Gestión: Compartiendo Conocimiento para Aumentar la Productividad. Documento de Trabajo Nº 10, CEGES, Depto. de Ingeniería Industrial, U. de Chile, 1999.
3. Barros, O. *Rediseño de Procesos de Negocios Mediante el Uso de Patrones*, Dolmen, 2000.
4. Barros, O. Arquitectura de Aplicaciones en e-Business. Documento de Trabajo Nº 26, CEGES, Depto. de Ingeniería Industrial, U. de Chile, 2001.
5. Bohrer, K., V. Johnson, A. Nilsson y B. Rubin. Business Process Components for Distributed Object Applications. *Communications of the ACM*, Vol 41, Nº6, pp. 43-49, 1998.
6. Caldentey, E. DSLOG, Software de ruteo de camiones, Versión Beta 1.0, 2001.
7. Cline, M. y M. Girou. Enduring Business Themes. *Communications of the ACM*, Vol 43. Nº 5, pp. 101-106, 2000.
8. Conallen, J. Modeling Web Application Architectures with UML. *Communications of the ACM*, Vol. 42, Nº 10, pp. 63-70, 1999.
9. Fan, M., J. Stallaert y A.B. Whinston. The Adoption and Design Methodologies of Component-Based Enterprise Systems. *European Journal of Information Systems*, Nº9, pp.25-35, 2000.
10. Fowler, M. *Analysis Patterns: Reusable Object Models*. Addison-Wesley, 1996.
11. Pree, W. *Design Patterns for Object-Oriented Software Development*. ACM Press Books/ Addison-Wesley, 1994.
12. http://obarros.cl