Business and Information Systems Design Based on Process Patterns and Frameworks

Oscar H. Barros
Departament of Industrial Engineering
University of Chile
República 701, Santiago, Chile
obarros@dii.uchile.cl – www.obarros.cl

Business and Information Systems Design Based on Process Patterns and Frameworks

Abstract

We present a novel approach for encapsulating high level business knowledge and logic of a given application domain in patterns and frameworks. These constructs can be applied to improve a process for a given business in the domain and to develop an Information System to support such process. The Business Process Patterns we have developed synthesize best business practices found in hundreds of real projects of process redesign. From these patterns, software Business Objects Frameworks, which encapsulate business logic to support their processes, have been derived. This approach provides a very flexible way, based on reusable components, to develop solutions and software for complex business decisions, which is a complement to packaged products. The approach is exemplified by using a specific application domain and applied to a real case in the domain. Our main contribution is that a good characterization of the processes and associated decisions, by means of Business Process Patterns, allows including complex generalized business decisions logic in Business Objects Frameworks. Such logic is built upon the best current knowledge of analytical methods, such as business intelligence, optimization and heuristics.

Key words: Business Processes, Business Patterns, Business Objects, Frameworks, Information System Development

Business and Information Systems Design Based on Process Patterns and Frameworks

1. Introduction

A common theme in the more recent IT and Business Process (BP) literature is the search for approaches that allow formalizing domain knowledge into structures, patterns or frameworks, which can be reused to facilitate process redesign and support systems development [2, 3, 7, 9, 15, 17, 27, 28, 29, 33]. The objective of these structures is to simplify and accelerate process innovation and system design. The approaches differ in scope and business knowledge content, so, in order to properly put our proposal into a context, we first attempt to develop a classification scheme for them. The classification variables we chose are business scope, which can vary from approaches that consider a single business activity to others that cover a whole business, and degree of business knowledge content, which can go from little to detailed content. In Table 1 we present our classification scheme and show the more representative approaches within it, where we have also located our proposals. In briefly reviewing such approaches, we group them into two classes: the ones in the two top rows of Table 1, which concentrate on business and process structure and the ones in the last row, which emphasize the definition of IT objects and their relationships.

For the business and process structure, the better known proposals for different domains are: SCOR, which provides reference models for supply chain management processes that are structured as a tree of typical sub-processes and component activities, together with business practices, performance attributes and metrics for them [27]; eTOM, a BP architecture for telecommunication companies, which is similar to SCOR, but oriented to the telecommunication processes domain [29]; FEA, the Federal Enterprise Architecture, which is a collection of interrelated reference models for the whole business of the US Government [33]; MIT's Process Handbook project, a collection of business practices that are structured along BP of different domains [22, 23]; and Model Driven Architecture (MDA), which is an approach, not a structure, to automatically built software by a series of transformations, starting with a business model (Computation Independent Model: CIM) for which an OMG Task Force is developing a BP metamodel [24].

Business scope	Little business knowledge content	Some business knowledge content	Detailed business knowledge content
Whole business			eTOM [29] SCOR [27] FEA [33]
Business processes (network of activities)		MDA [24]	Business Process Patterns [5] MIT's Process Handbook [23,24] Business Object Frameworks [6] Fowler's patterns [17] Archetype patterns [3]
Single activity	Problem frames [21]	Domain theory [28] e-Business patterns [2]	

Business knowledge content

Table1. Classification scheme for patterns and frameworks

For the IT structure, more relevant proposals are: Fowler's patterns [17], which are published objects structures in domains such as accounting, billing and payroll; archetype patterns, which are configurable UML models for a given business context, such as CRM and inventory [3]; Domain Theory, which basically proposes reusable class models to automate typical business activities, such as inventory, accounting and hiring [28]; e-Business patterns, which are mostly hardware and software reusable structures for typical business in Internet environments, such as User to Business (e.g. B2C), B2B and e-Marketplaces [20]; and Problem Frames, which are abstract descriptions of recognizable classes of problems, such as commanded behavior, information display and transformation [21]. As it is clear from Table 1, these IT structures, despite their technical orientation, consider some aspects of business and process structure, since they are related to a particular BP or business context and contain at least some business logic. For example, in the archetype pattern *Inventory* proposed by Arlow and Neustadt [3], besides the common UML entity classes such as *product type, inventory entry* and the like, there is a *restock policy* class that allows for some business logic for item reordering.

In what follows we summarize our proposal for the two levels of structure defined above, which has been developed independently of the approaches reviewed. It is based on the following research methodology.

Starting with the knowledge derived in hundred of real life cases, where the redesign of many different types of processes has been performed, we have developed Business Process Patterns (BPP) that are valid for given domains. These BPP are generalized process models, including activities, flows connecting them and business logic, of how a business in a given domain should be run, according to best practices known [5]. The word pattern is used here for structures that are very different than software patterns [3, 17, 20]. As we will explain in more detail below, BPP can be built for very general domains, such as the value chain of any business. These general BPP can then be specialized for specific domains; e.g., the process pattern associated to the value chain in a hospital. This generates a tree of BPP from more general at the top to more specific at the bottom. These patterns have been validated by showing their applicability to many new cases, where redesigns have been performed based on the structure and practices recommended by such patterns.

Given the BPP above, we have developed Business Object Frameworks (BOF) that are sets of related generalized classes that define the IT support that can be given to the process in a pattern. Here we adopt the definition of a framework used in software development as "a body of code that implements the aspects that are common across an entire domain, and exposes as extension points those elements of the domain that vary between one application and another" [11]. In developing a framework of this type, associated to a BPP, a UML model that defines the Business Objects (BO) involved is first derived. As show in Table 1, BOF can be developed for a single activity or for a business process.

Our research diverges and advances with respect to the BP structures reviewed above in the following aspects. The BPP we propose are better defined and detailed than SCOR, eTOM and others reviewed, since they explicitly consider not only the generalized activities of a BP but their interrelationships by means of well-defined information and physical flows; they also include explicit business logic for all the activities. Of course, this logic will be more detailed when the domain is smaller. For example, a business structure such as SCOR models the supply chain with five basic processes: *plan, source, make, deliver, return*. These are then decomposed into sub-processes and activities. As we will detail below, our approach also allows to model the supply chain by detailing its processes and their component activities. But, besides this, it adds explicit relationships that should exist among activities, at all levels of detail, in order to have the required coordination, and also specifies the best practices, in the form of business logic, that each of them should carry out in order to assure a given performance. The BPP we propose have been developed during a period of more than ten years, based on several hundreds of real cases of BP redesign, and tested with over one hundred of new cases where the patterns have been used to generate new BP designs in very different domains [5, 13, 18, 31].

Our frameworks differ from traditional ones in that they are mostly oriented to code business logic, in particular complex business decision logic using analytics, according to current best practices taken from OR/MS, Business Intelligence and Statistics. This makes them very different from the IT structures above that only include simple business information processing type of logic that, in many cases, duplicates logic included in packages such as ERP. For example, as compared with the archetype pattern *Inventory* of Arlow and Neustadt [3], mentioned above, our BOF include not only simple rules for inventory restock, but classes that perform explicit logic on how to predict inventory demand – based on time series or causal models – and to optimize inventory management with mathematical models. This orientation has allowed that, in practical applications of BOF, their advanced flexible logic has complemented or been more effective than the one contained in competitive ERP and other types of packages.

We remark that our BOF are fully integrated with the BPP, while the other proposals for the two-level structure above have been developed independently. To remedy this lack of integration, such proposals have extended IT systems structure to include business aspects, but this only allows considering a few aspects of the more important issues of process relationships. We will illustrate this point when we present our BOF.

In the following sections we present the results of our research and its practical application. Due to the space available we can only give the flavour of how our BPP and BOF have been developed and we emphasize the way in which they can be specialized to specific cases to perform redesign and develop Information Systems.

We present the BPP and report on their use in Section 2, and show how to specify the business logic for such patterns in Section 3.

We will show how to derive BOF from BPP and their use in Section 4.

Finally, in Section 5, we present a real application of our BPP and BOF and, in Section 6, conclusions.

2. Business Process Patterns

Business Process Patterns are models of how a business in a given domain should be run, according to the best practices known [5]. Hence they are based on empirical knowledge of how activities of a process in the best companies of a given domain are performed. Such knowledge can be obtained from the best practices literature, reported experiences and direct observation of firms [14, 16, 19, 26, 30]. Our patterns have benefited from the knowledge derived by hundred of cases in which processes of many different companies have been modeled, analyzed and redesigned and previous experience with the modeling of Information Systems [4].

We have found that beyond the best practices for a given domain, usually expressed in the form of specific business logic, BPP share a common structure of activities and flows. Thus, products or services provision processes – such as manufactured goods, health, justice and financial services, etc – share such a common structure. A first level of detail of such a process structure or BPP is shown in Figure 1.

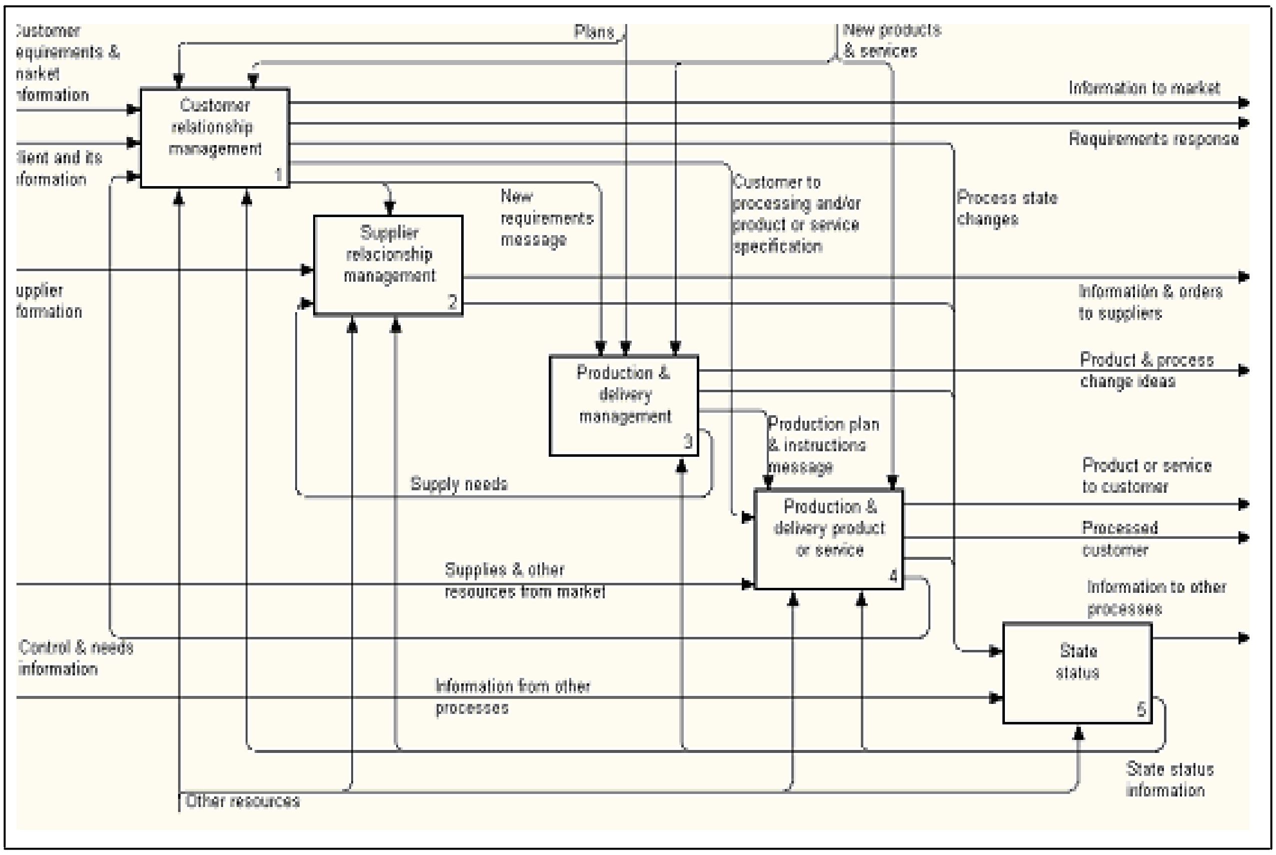
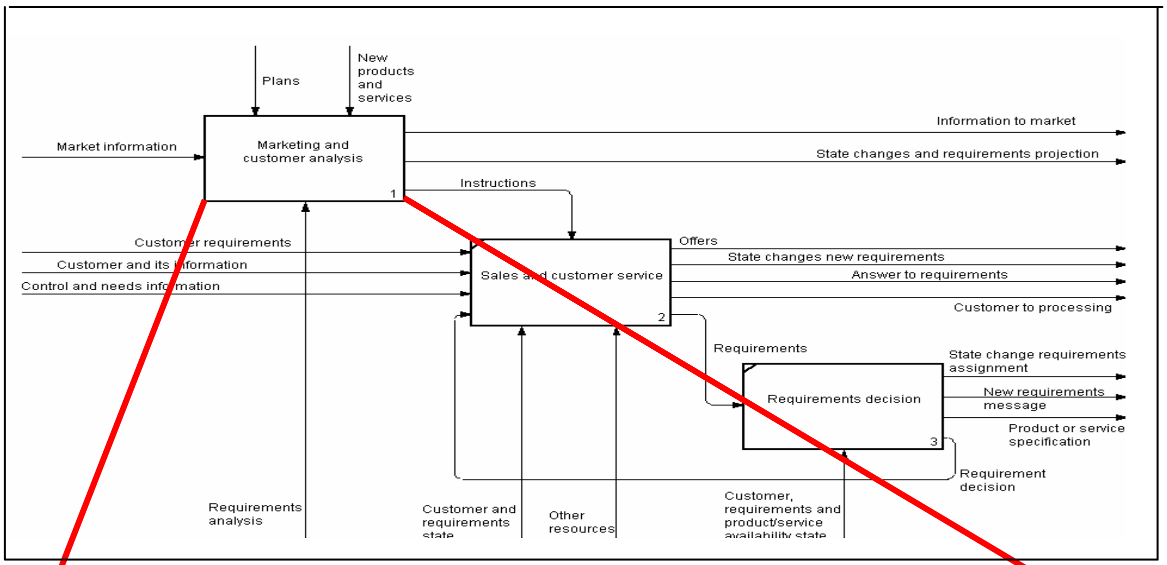

ustomer equirements & narket nformation
lient and its nformation
upplier formation
Control & needs information
Plans
New products & services
Customer relationship management 1
Supplier relacionship management 2
Production & delivery management 3
Production & delivery product or service 4
State status 5
New requirements message
Customer to processing and/or product or service specification
Process state changes
Information to market
Requirements response
Informatión & orders to suppliers
Product & process change ideas
Production plan & instructions message
Product or service to customer
Processed customer
Information to other processes
Supply needs
Supplies & other resources from market
Information from other processes
State status information
Other resources

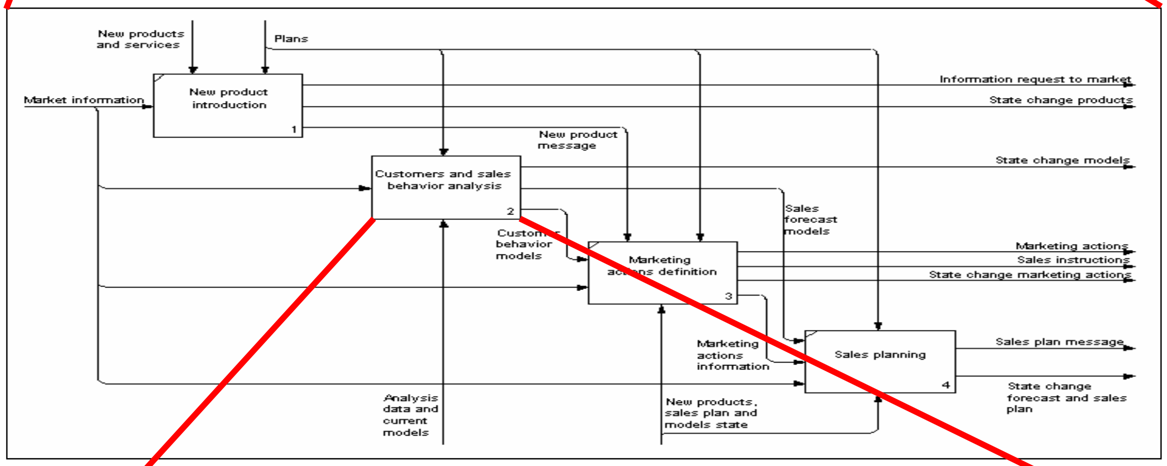
Figure 1. Business Process Pattern for products or services provision (Macro1)

This structure is a process model of what is commonly known as the value chain, which is a BPP we call Macro1. It includes the sub-processes of *Customer relationship management, Supplier relationship management, Production and delivery management, Production and delivery of products or services* and *State status.* It also shows the necessary relationships among such sub-processes by means of information and physical flows, such as supplies and products [5]. Now, following the IDEF0 modeling scheme, we can detail such a pattern by partitioning each of its sub-processes, as it is shown in Figure 2(a) for *Customer relationship management.* Our BPP do not depend on IDEF0, so other modeling approaches can be used, such as the one proposed in Dalai *et al* [12] or any implementation of the OMG business process modeling language [8].

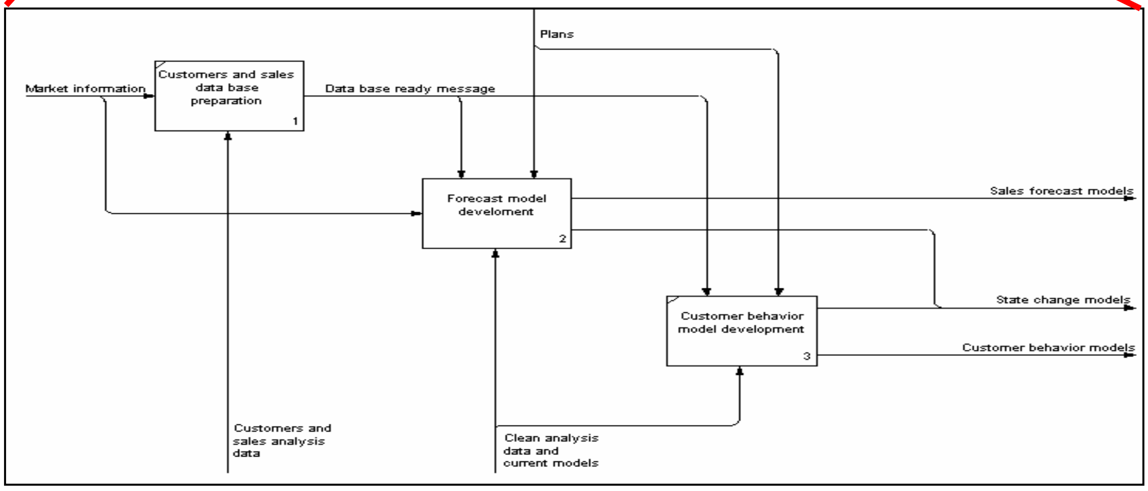
If we want to give further detail of a BPP, we have to be more specific about the domain, so that we can define decompositions of sub-processes into generic activities, their business logic and the flows that connect them with precision. In order to show how to do this, we synthesize our experience of many real cases in the following domain definition for the generic activity *Marketing and customer analysis* of Figure 2(a). We assume situations where businesses sell physical products or services to


Plans
New products and services
Market information
Marketing and customer analysis
1
Information to market
State changes and requirements projection
Instructions
Customer requirements
Customer and its information
Control and needs information
Sales and customer service
2
Offers
State changes new requirements
Answer to requirements
Customer to processing
Requirements
State change requirements assignment
New requirements message
Requirements decision
3
Product or service specification
Requirement decision
Requirements analysis
Customer and requirements state
Other resources
Customer, requirements and product/service availability state

a) Detail of *Customer relationship management*


New products and services
Plans
Market information
New product introduction
1
Information request to market
State change products
New product message
Customers and sales behavior analysis
2
State change models
Sales forecast models
Customer behavior models
Marketing actions definition
3
Marketing actions
Sales instructions
State change marketing actions
Marketing actions information
Sales planning
4
Sales plan message
State change forecast and sales plan
Analysis data and current models
New products, sales plan and models state

b) Detail of *Marketing and customer analysis*


Plans
Market information
Customers and sales data base preparation
1
Data base ready message
Forecast model develoment
2
Sales forecast models
State change models
Customer behavior model development
3
Customer behavior models
Customers and sales analysis data
Clean analysis data and current models

c) Detail of *Customer and sales behavior analysis*

Figura 2. Decomposition of *Customer relationship management*

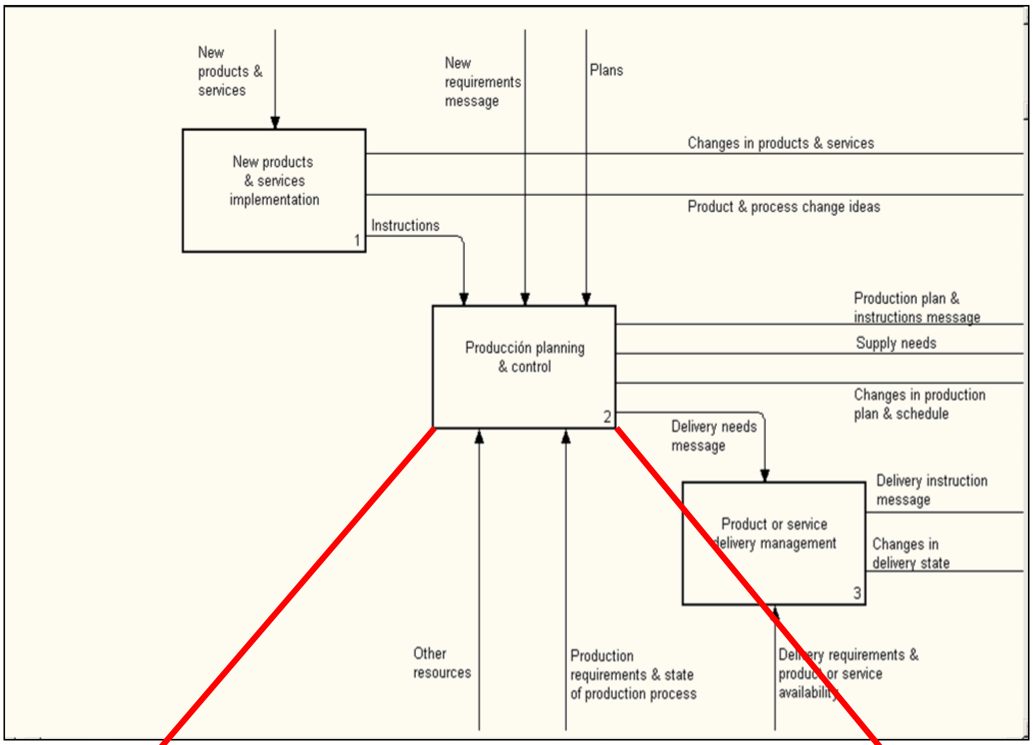
a large number of customers; specifically we include cases such as retail using any channel and direct sales by manufacturing or service firms.

Under previous assumptions, we decompose the said activity in Figure 2(b), where we will concentrate on *Customer and sales behavior analysis.* For such an activity in this specific domain, which is further decomposed in Figure 2(c), we can be very precise about a best practice business logic. Business logic, which guides the action in an activity, determines the exact information flows that are required and that are produced, including the updating of the process changes in *State Status* of Figure 1 and the state information this generates, shown below each activity, which is necessary for its performance. We will show, in the next section, how to specify such logic.

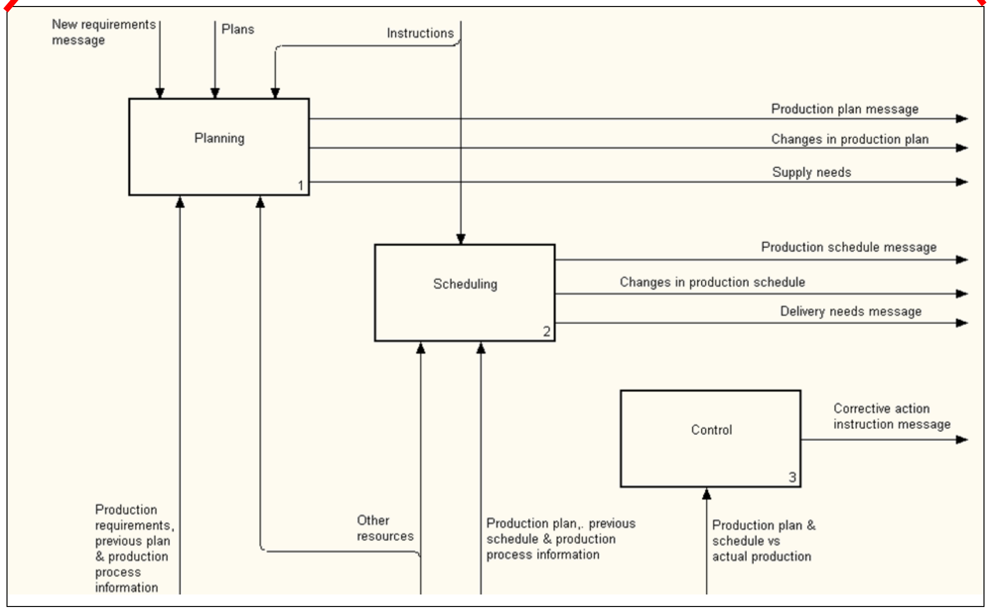
The pattern includes decomposition for all the activities in Figure 1. To further illustrate such task, we decompose *Production and delivery management,* considering the same domain previously defined. This is shown in Figure 3, where we have concentrated on the activity of *Scheduling,* for which we will provide generalized business logic in the next section. Details for the other activities of this BPP are given in Barros [5].

Now a key and unique feature of our approach is to make explicit how the different activities of a pattern interact to produce coordinated actions. This is done by means of the information flows that activities interchange and the business logic that each of them executes to process such flows. For example, in the case of applying the previous pattern to production to stock, such type of relationship is the one between *Forecast model development* of Figure 2c, *Sales Planning* of Figure 2b, *Scheduling* of Figure 3b and *Supplier relationship management* of Figure 1. Clearly, forecast models produced by the first activity will be used to generate a sales forecast and sales plan, in the second, which will then trigger production lots that will be sequenced by *Scheduling.* The production schedule will, in turn, be used by the last activity to generate materials orders to suppliers. We present this coordination chain as a sub pattern in Figure 4, where we show a "horizontal" process view mixing activities from several "vertical" hierarchical decomposition views, including the explicit role of *State status* in terms of updating and information provided[*]. In such chain each activity executes a business logic that

[*] This view is given to facilitate the understanding of the coordination issues; experienced modelers can gather the same information from the "vertical" views in Figures 1, 2 and 3. Of course, there are many other sub patterns that can be derived from such views.


New products & services
New requirements message
Plans
New products & services implementation
1
Changes in products & services
Product & process change ideas
Instructions
Producción planning & control
2
Production plan & instructions message
Supply needs
Changes in production plan & schedule
Delivery needs message
Product or service delivery management
3
Delivery instruction message
Changes in delivery state
Other resources
Production requirements & state of production process
Delivery requirements & product or service availability

a) Detail of *Production & delivery management*


New requirements message
Plans
Instructions
Planning
1
Production plan message
Changes in production plan
Supply needs
Scheduling
2
Production schedule message
Changes in production schedule
Delivery needs message
Control
3
Corrective action instruction message
Production requirements, previous plan & production process information
Other resources
Production plan,. previous schedule & production process information
Production plan & schedule vs actual production

b) Detail of *Production planning a control*

Figure 3. Decomposition of *Production and delivery management*

is designed to generate the required performance: to produce what is needed according to the forecast, assuring the required materials and minimizing production costs. Part of such logic can be specialized from the one we will give in the next section.

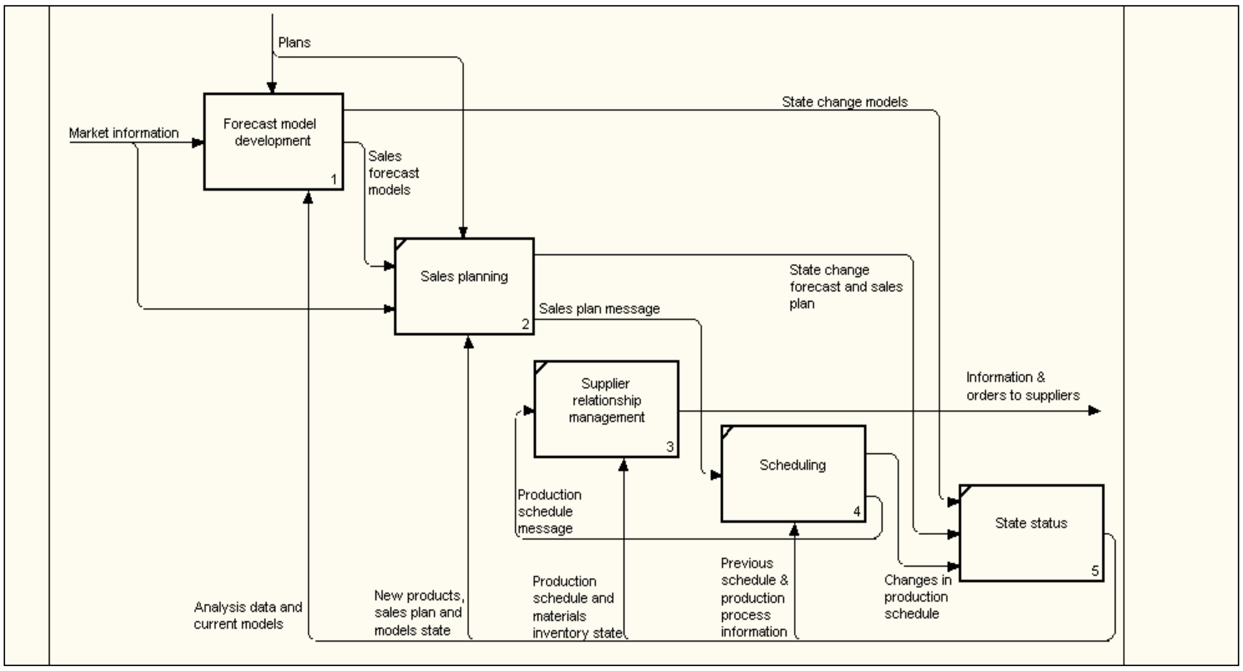

Plans
Market information
Forecast model development
1
State change models
Sales forecast models
Sales planning
2
Sales plan message
State change forecast and sales plan
Supplier relationship management
3
Information & orders to suppliers
Scheduling
4
State status
5
Production schedule message
Analysis data and current models
New products, sales plan and models state
Production schedule and materials inventory state
Previous schedule & production process information
Changes in production schedule

Figure 4. Coordination sub pattern

We have illustrated the concept of BPP with just one process pattern for a very general domain. Other BPP have been developed for particular cases or specializations of the general domain, such as situations in which there is not production and sale of products and services is made from stock bought from others, e.g. distribution companies or telephone companies, manufacturers that sells their own products, and hospital or financial services [5, 32]. Also, BPP have been developed for other types of processes, complementary to the value chain: Management of resources –human, financial and fixed assets- (Macro4), Development of new products (Macro3) and Business planning (Macro2) [5]. These types of BPP and their specializations conform, as advanced above, a tree of BPP from which we can select the closest to a situation one wants to redesign. A partial version of such tree is given in Figure 5, where the root *General Architecture* refers to the fact that all the BPP are built starting with a generic structure, well founded from a theoretical and empirical point of view, which we do not detail here [5]. We have only given a very small sample of the BPP we have developed and of the details that each of them includes, because of space limitations.

3. Business logic specification

Our aim is to give generalized business logic for the activities of a BPP in a specific domain. Consider the activity of *Forecast model development* of Figure 1(c). Under previous domain assumptions, a forecast based on sales history is possible. We also assume that, previously, a datamart with a relevant and clean history has been set up in the activity *Customer and sales data base preparation*. Then, an analyst that performs former activity will have a system support with a business logic that allows him to do the following:

i. For all current forecast models for sales items, made available through *Clean analysis data and current models,* to calculate forecast error by comparing a selected history of forecasts and actual sales.
ii. For selected sales items and forecast models -e.g. exponential smoothing, Box-Jenkins, neural networks and supper vector machines [1, 2, 5], to fit historical sales data to models, proposing parameters and providing estimated forecast error.
iii. To update models selected by analyst for routine use in forecasting in *Sales planning* of Figure 1(b).

This a simplification of real cases we have performed where the logic can be more complex, involving model identification and training with more analyst responsibility than the one outlined.

In order to give a flavour of the business logic included in the system support above, we outline a portion of the logic corresponding to the fitting and estimation of a selected model to historical data. This logic, which is shown in Table 2, corresponds to a simplified version of the identification and estimation of a Box-Jenkins model. The logic is mostly composed of statistical calculations, with some analyst intervention for the more qualitative aspects. Hence it leaves little room for introducing causal business factors and decisions, such as economic environment, promotions and pricing policies. However, other methods, such as neural networks and support vector machines [25], allow to consider such factors, in which case

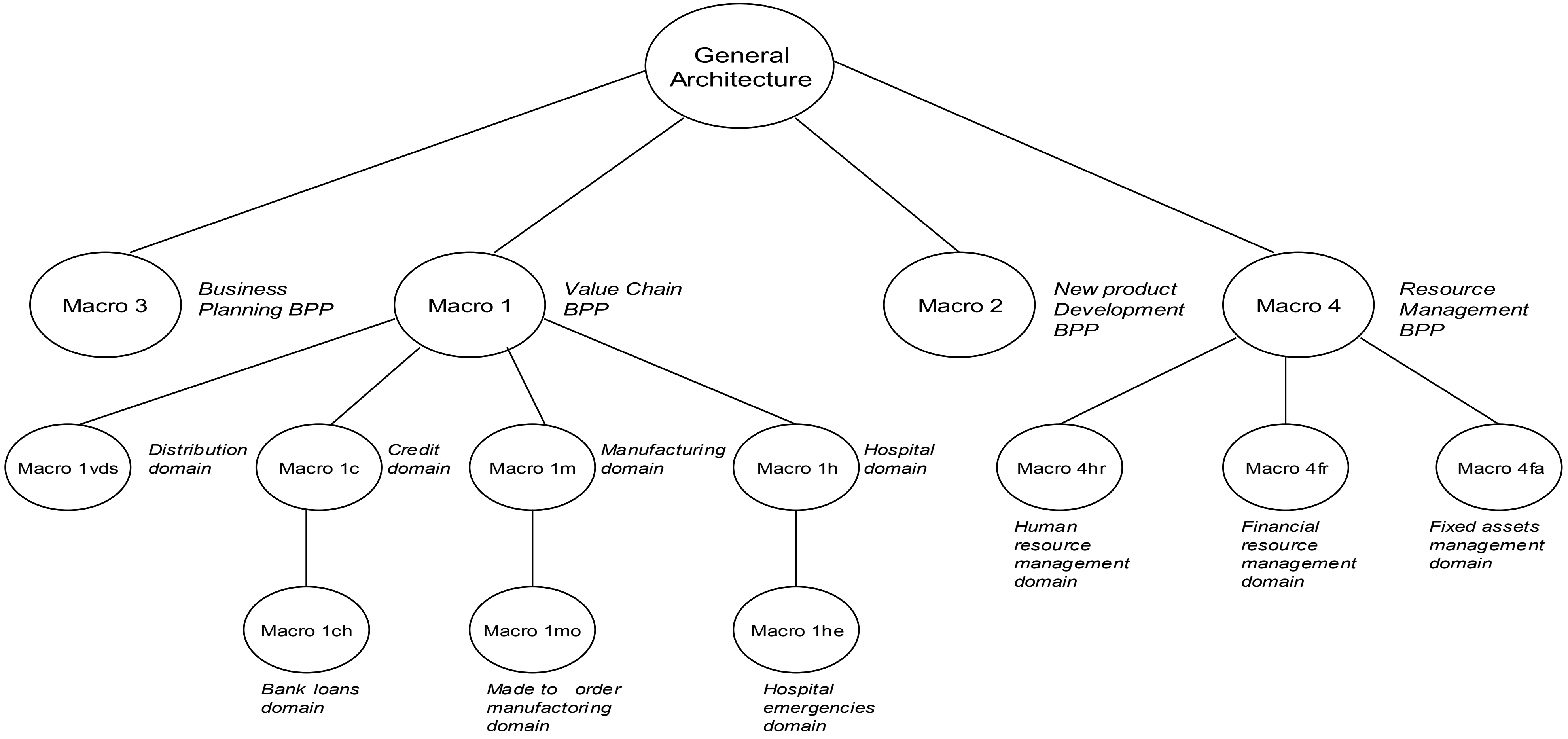


Figure 5. Tree of Business Process Patterns

```
//Business logic outline for Box-Jenkins model identification and estimation

//Previously, several tests for determining the suitability of Box- Jenkins against other
methods have been performed; e.g. presence of tendency, seasonality and white noise
and consideration of the number of observations

//Model identification

Calculate autocorrelations and partial autocorrelation functions
Test for determining if series is stationary
//Autocorrelation decay is evaluated
If series is not stationary
    Do the series difference until it is stationary
    //Times series is differenced corresponds to parameter d
Endif

//Series is stationary
Establish behavior of autocorrelation and partial autocorrelation functions: decay,
oscillation, truncation, large particular values, etc.
Identify type of model: MA(q), AR(p), ARMA(p,q) or ARIMA(p,q,d)
//This is based on functions behavior
Show the analyst autocorrelation and partial autocorrelations graphics and the
proposed model
Accept the analyst approval of proposed model or own values for parameters p and q

//Models estimation and testing

Estimate constants for model
Perform model goodness test (Box-Pierce)
Test model by using new historical data to forecast and calculate forecast error
Show analyst estimated model and tests
If the analyst accepts the model or decides that no model can be fitted
    Update the model
Else analyst establishes new analyses to be made
//This may mean going back to the model identification or selecting a model different
from Box-Jenkins.
```

Table 2. Business logic for Box-Jenkins identification and estimation

business logic would explicitly consider the interaction between commercial decisions and forecast [1]. These possibilities are considered in the full version of our BPP.

The key for developing generalized business logic, valid for all particular cases in a domain, is to structure the decisions in the domain in cases, according to certain variables or parameters that determine the relevant algorithm or heuristic for each case. If we apply this idea to the forecasting activity considering, in a simplified way, the variables of sales stability and marketing style as determining the right type of forecasting model, we show in Table 3 the models that apply for combinations of such variables (cases). This determines the logic to forecast sales in each case. Then a logic as in Table 2 can be specified for each case. The same idea as above can be applied to the *Scheduling* activity. We consider the general situation where a list of waiting tasks, which can be products to be manufactured or services to be performed, is to be assigned to servers – machines, people, trucks, etc – for their completion. In this case structure depends on the variables in Table 4, where the relevant heuristics are also shown.

		DEMAND	
		STABLE	NOT STABLE
MARKETING	*PASSIVE*	Exponential smoothing	Box- Jenkins
	ACTIVE	Exponential smoothing and Box-Jenkins	Box-Jenkins and neural networks

Table 3. Appropriate forecasting models for various cases

		NUMBER OF SERVERS		
		1	2	>2
NO SET UP TIME BETWEEN TASKS	NO COMPLETION DATES REQUIRED	Greedy heuristic (shortest task first)	(Series servers) Greedy heuristic	(Series servers) Grouping of servers plus greedy heuristic
	COMPLETION .DATES REQUIRED	Greedy heuristic with time windows	(Series servers) Grouping of servers plus greedy heuristic with time windows	
SIGNIFICATIVE SET UP TIME	NO COMPLETION DATES REQUIRED	Greedy heuristic with set up times (smallest set up plus processing time task first)	(Parallel servers) Assignment of tasks to servers plus greedy heuristic with setup time for each server	
	COMPLETION DATES REQUIRED	Greedy heuristics with set up times and time windows	(Parallel servers) Assignment of tasks to servers plus greedy heuristic with setup times and time windows for each server	

Set up Time: Time needed between execution of any two tasks; e.g. change of tooling in manufacturing, travel time between locations in telephone repair and cleaning of surgical facilities between medical operations.

Server: Facilities that execute tasks; e.g. machines, repairmen and surgical facilities

Table 4. Appropriate scheduling heuristics for various cases

In order to show the type of logic for *Scheduling*, we give, in Table 5, the general logic for all the cases of Table 4 and one example of specific logic for the case with set up times, no completion or lead time and one server. Though they may look mathematically complex, they are a formalization of simple decision heuristics: for the general case, to first assign tasks to servers using an appropriate criteria and then sequence them in each of the servers in the best possible order; and for the particular case, choose a sequence by selecting always the task with the smallest remaining set up plus processing time (greedy heuristic). The algorithms for the rest of the cases and their justification are given in Barros and Varas [6].

4. From Business Process Patterns to Frameworks

From the BPP support and business logic of the previous sections, we can map BOF with BO that incorporate the knowledge about the solution of relevant problems in the given domain. These BOF have as a purpose to provide generalized solutions to the problems, which can be used to develop an object-based software application for any particular real-life problem in the domain.

The mapping from BPP and business logic to a BOF is as follows:

i) The structure of the BPP system support and the business logic of the domain give a first cut definition of the BO classes that encapsulate the algorithms or heuristics that solve the problem for different cases in the domain.
ii) The structure of the BO can then be modeled using UML class diagrams, and operations or methods for classes defined according to business logic.
iii) The data needed to execute operations can then be derived from the data included in the business logic.
iv) The data can be structured into data classes that interact with BO in (ii). A complete class diagram with BO and databases can then be modeled using UML.

We follow the steps above for the activity *Forecast model development* in Figure 2c.

```
Heuristic Schedule ($n°, m$)
        $\pi_j = 0, j = 1,...,m$
        $n = n°$
        SelectSet ($m, n, \Phi$)
        GetTask ($\Phi, \pi$)
        If $(m > 1)$ then
                ImproveSchedule ($\pi$)

GetTask2 $(\Phi, \pi)$
        $q = \arg\min_i \{\min_j \{S_{ijk} + P_{jk} \mid i, j \in \Phi\}\}$
        $\pi = insert(q, \varnothing, 0)$
        $\Phi = \Phi \setminus \{q\}$
        $while(\Phi \neq \varnothing)$
        {
        $q = \arg\min_j \{S_{qjk} + P_{jk} \mid j \in \Phi\}$
        $\pi = insert\ (q, \pi, |\pi|)$
        $\Phi = \Phi \setminus \{q\}$
        }
```

where

- Routine *SelectSet()* selects a subset of tasks for every facility, where the result is given in the *m*tuple $(\Phi_1, \Phi_2, ..., \Phi_m)$ of subsets of Φ .
- Routine *GetTask*() selects a sequence π_j for the set of tasks Φ_j of each facility $j\ 1 \le j \le m$; its implementation will depend on the specific case.
- Routine *ImproveSchedule*() improves the current schedule π for all facilities.
- $insert(j, \pi, i)$ returs a new schedule with element j inserted in schedule π just after place i
- m is the number of different types of facilities or machines.
- n is the number of distinct tasks types.
- P_{ik} is the processing time of task i at facility k
- S_{ijk} is the set up time if task type i is processed immediately before task type j at facility k .

Table 5. Example of business logic for *Scheduling*

.

The structure of the system support in 3i,ii,iii and the business logic in Tables 3 and 4 lead us directly in to the BO structure of Figure 6, where we also show the data classes and the operations for each class. We use common OO conventions for frameworks and adopt some of the ideas in Conallen for web applications [10] to organize classes. The structure is not complete, since it should be integrated with all the components that support the activities of *Customer and Sales database preparation* and *Sales planning* of Figure 2, where the datamart is created, in the first, and forecast models are actually run, in the second, to produce forecasts that are needed for generating sales plans. We have avoided such integration to simplify presentation.

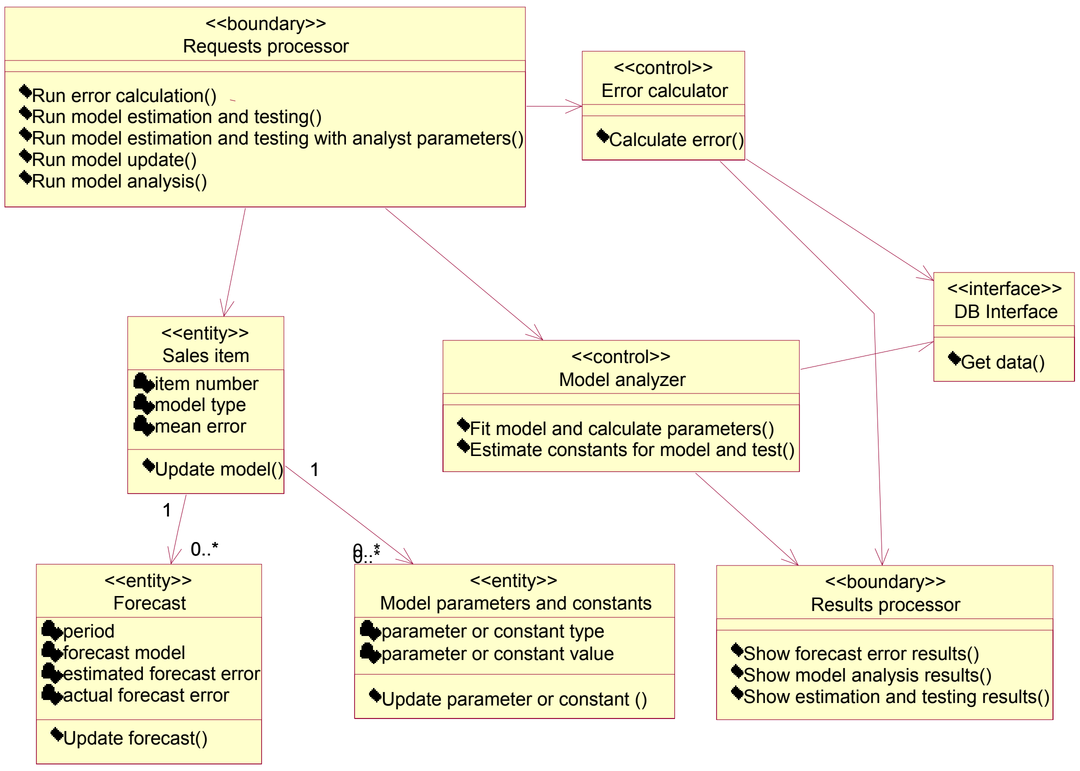


Figure 6 Framework for forecasting model development

The BO of the framework is structured according to the type of cases in the domain, defined in Table 3. Then the analysis can be tailored and made more specific for each particular case. In Figure 7 we show, in a simplified way, how this is done in our forecasting framework. The key is to structure the *Model analyzer* BO in component cases, which provide different solutions according to the characteristics of the business problem that the framework intends to solve. In Figure 7 such structure is organized according to the variables of stability and marketing type previously mentioned. Then, for each case in the structure, an appropriate logic is provided, as in Table 2, based on the experience obtained from the results generated with the use of the most important analysis methods [1, 25]. Hence, the main feature of the framework is to offer solutions for several different cases that are part of the domain, so that an analyst can choose the one that fits his particular case.

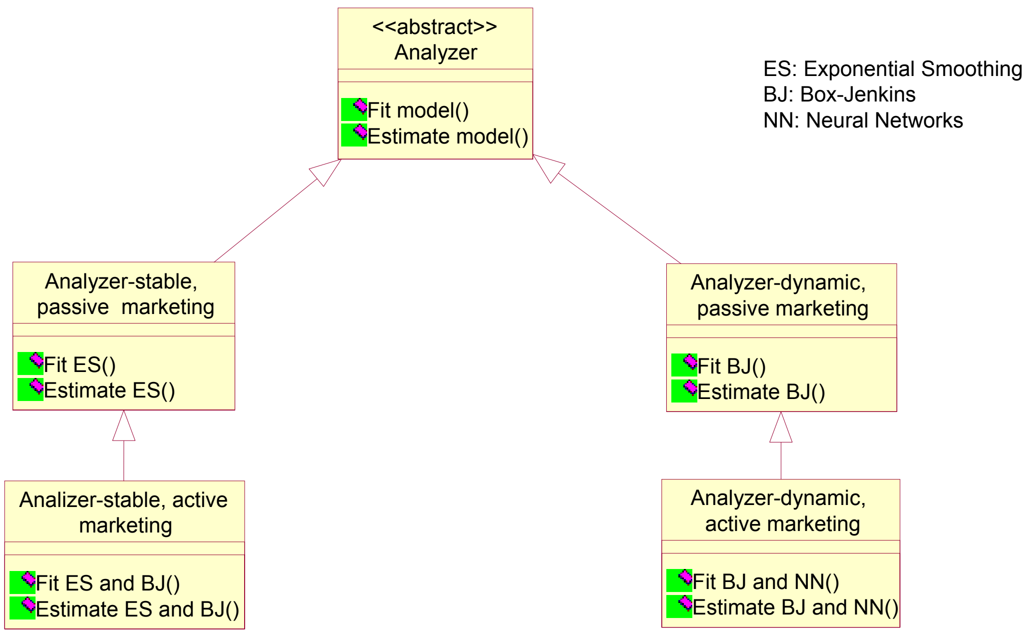


Figure 7. Structure of *Model Analyzer*

In the same way, a framework for *Scheduling* can be developed, which is shown in Figure 8. This framework is based on the idea that all the problems in the domain share a common structure (BOF) with several different cases, defined in terms of number of processors and configurations (series, parallel and network), as in Table 4, which can be selectively used according to the characteristics of the problem al hand.

An important characteristic of these frameworks is to offer the possibility of having incrementally more complex logic for the cases in a problem, defined as outlined above. Thus, for example, for the *Scheduling* activity, the structure of Figure 8, which is a specialization one, shows that there are methods (*HeuristicSchedule, SelectSet, GetTask*, and *ImproveSchedule*), which are used by all specialization classes. Then the three branches starting at the class *Scheduler*, define three alternative cases: one with lead time, another with set up time, and one with both. Each branch has a method that is a specialization of *GetTask* (*GetTask1*, *GetTask2* and *GetTask3),* which is inherited by cases immediately below in such branch. The same is true for other branches below this one. All these methods are specified by means of a business logic of the type given in Table 5. Now each branch has classes increasingly more complex; e.g., the one at the left of Figure 8, corresponding to the case with led time but no set up time, has *Scheduler1* corresponding to the case with one facility, *Scheduler11,* with two facilities and *Scheduler111,* with three facilities in series. Each class inherits methods from classes above in the branch and uses them in their own methods. This means that more complex cases are built based on methods of simpler ones. Thus when using the framework to develop an application for a particular case, just one branch and some of the classes of the branch will be selected: just the ones appropriate for the particular case at hand. The framework can be extended by adding more specialization classes at the last level selected of the branch. We will illustrate this idea with a real case in the next section.

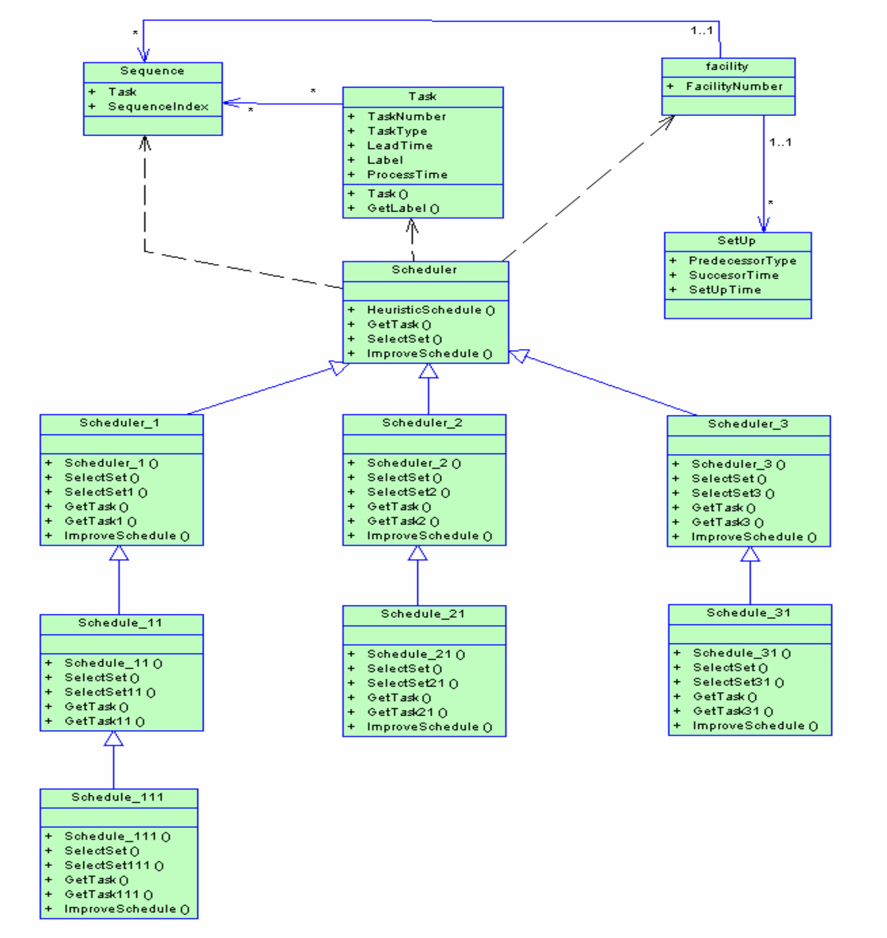


Figure 8. Framework for *Scheduling*

Hence, in the application of a BOF to a particular situation, the user of the framework will select the minimum level of complexity that solves his problem Thus, for example, a developer will select, in Figure 8, just the class *Scheduler 1,* because he has a case with lead time, no set up time and just one facility.

The implementation of this feature, which allows for the selection and use of incrementally more complex solutions for a problem, is based on OO inheritance. We have coded our framework based on this feature and determined that is very simple to select and combine the options that they offer and to specialize them to particular applications. This idea of selection from a given framework is related to the notion of configuration of Arlow and Neustadt [3], where formal models have been developed to define such general structures and automatically generate valid particular cases. Our approach depends more on analyst specialization.

The frameworks we have used as examples have been presented as stand alone, which is not realistic. In some cases they would be integrated with frameworks for others activities in a process, in order to develop an Information Systems that support a coordination chain for such activities, as outlined in Section 2; in others cases, they can be used without integration, but they should be, at least, be connected with other applications though business DB. For example, the forecasting framework supports *Forecast model development* of Figure 4 and the scheduling one, the activity *Scheduling.* Clearly an integrated Information System support for the whole coordination sub pattern of such figure should include these two pattern plus the ones to support *Sales Planning* and *Supplier Relationship Management,* all of them connected by means of classes in these patterns; such as the ones necessary to support *Sales Planning* to generate, for a given forecast stored in the framework of Figure 6, a sales plan and then convert it into production requirements (lots) to be stored in *Task* of Figure 8.

We have developed working frameworks for several activities of the process in Figure 1, which contain the best practices that can be automated in applications to support such activities. In particular we have frameworks for customer evaluation and order processing which includes automated customer classification based on history and balance sheet information; the frameworks we have presented in a very simplified way in this paper; inventory management that includes JIT and Reorder Point cases, with probability considerations for demand and lead times; and production planning with mathematical models.

5. Using Frameworks for Information System Development

In using a framework of the type derived in the previous section for developing an Information System to support a process in a real-life case, within the frameworks domain, the following procedure is used [6]:

i) Select the relevant substructure of the framework applicable to the case.
ii) Specialize the substructure to the characteristics of the case, adding data and logic as needed.
iii) Design in detail for an available or selected technology and code.

We illustrate this procedure with the *Scheduling* framework. For this we use a real-life case, which we have actually solved. It deals with the process of attending telephone repair calls and the day to day scheduling of such calls (tasks) for the largest telephone company in Santiago, Chile. In attending calls, hundreds of repairmen (facilities) are available. Then the problem is to assign calls to each repairman and give him a route to attend the calls. The objective is to minimize the sum of all repairmen travel and repair time, equivalent to set up time between tasks plus processing time in the general formulation, subject to the maximum work load that can be assigned to each of them. Additionally, we would like that each repairmen has an assigned zone, where he will get to be known by customers and develop a good relationship with them; then each repairmen should hopefully be assigned calls in such zone, but trying to keep the work load balanced among repairmen by eventually assigning him, if he has time available, calls from other zones where the repairman is overloaded.

This case corresponds to *Schedule21* (with set up time but no lead time) in Figure 8. So the relevant classes for such node in the structure are selected, which are shown in Figure 9.

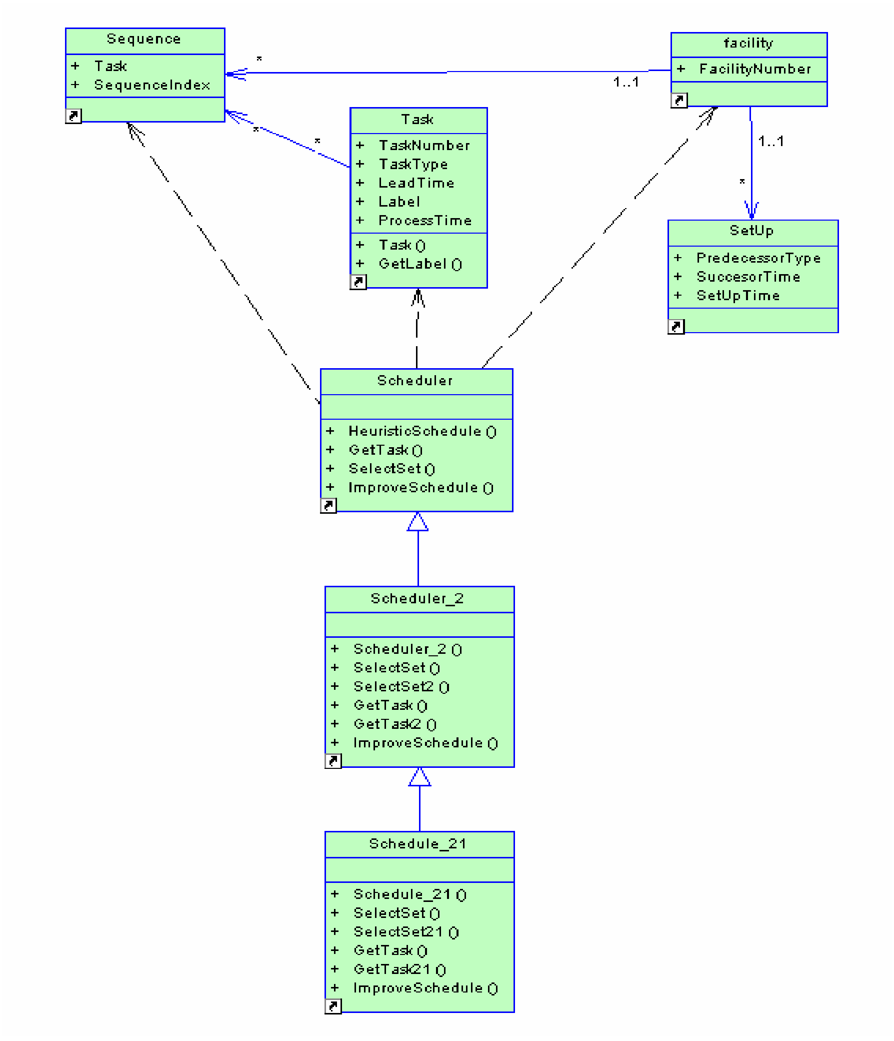


Figure 9. Relevant classes for repairmen scheduling

The specialization of Figure 10, according to previous case description, proceeds with adding a class *ScheduleCalls* with methods *SelectSet211*, *GetTask211* and *ImproveSchedule211*; a *Repairmen* data class with attribute *zone*; a *RepairCall* data class with attribute *location*; and a class *SetupGenerator*, which calculates distances for all pair of locations of standing calls and creates the data of the *Setup* class. In calculating such distances, a dummy task (node) that represents the location from which all repairmen start, is created. All these classes inherit from the selected structure as shown in Figure 10.

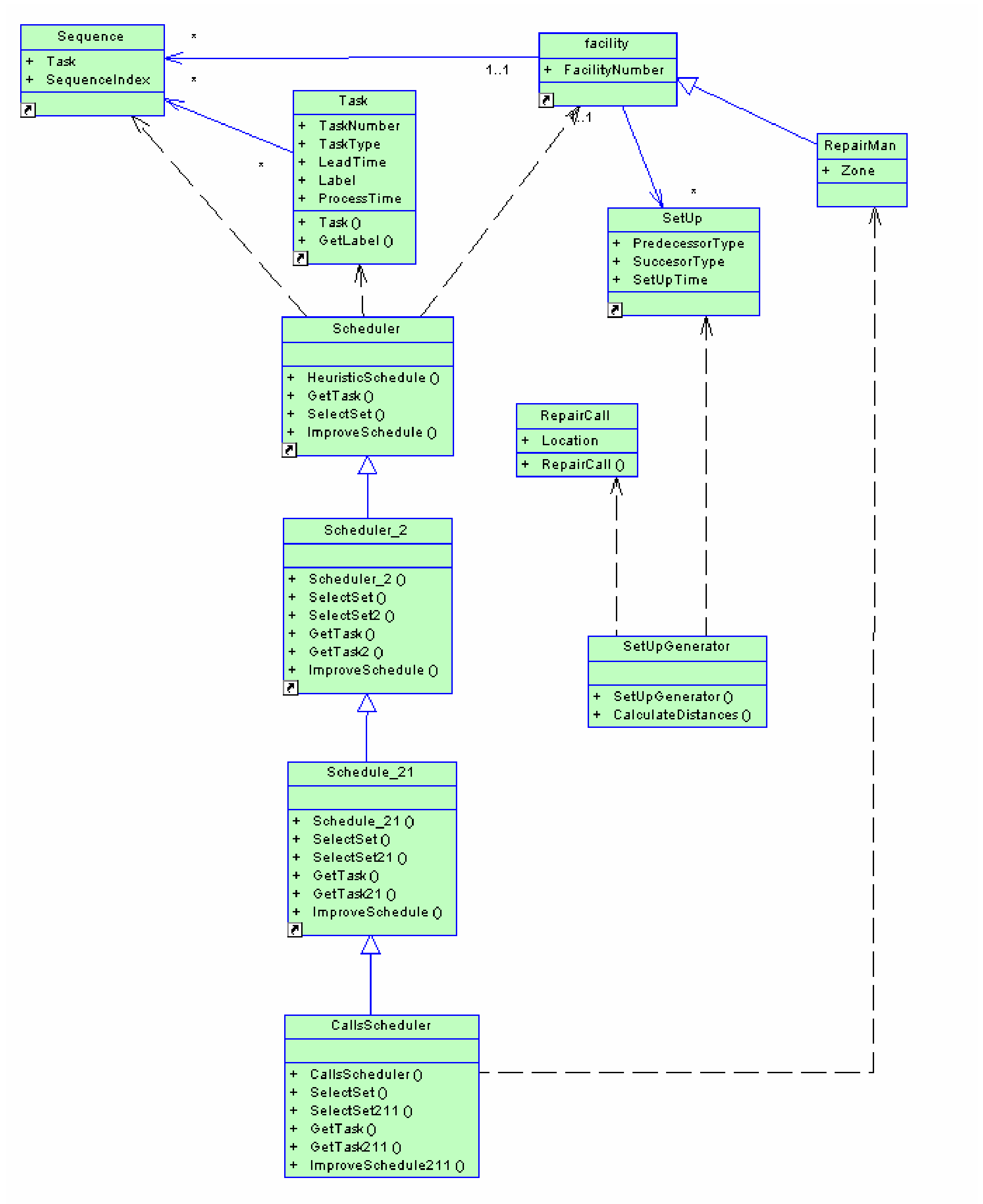


Figure 10. Specialized framework for repairmen scheduling

Then the business logic for *SelectSet211* is obtained from class *SelectSet21*, and the sets are sequenced with *GetTask21(Φ,π)*, where all sequences start at the dummy node defined above.

We have added a method *ImproveSchedule211* to balance the work load among repairmen by reassigning calls from overload zones to the nearest one with available time.

The coding and actual use of the framework in solving the repairmen scheduling problem has confirmed the benefits we expected in its practical application. The coding effort of the framework was about 2000 lines of Java code. Now in terms of its use, it is clear that the framework offers a pre built solution that allows the developer to select, among several possibilities, the functionality that solves his problem and specialize it, if it does not have a complete fit with it. So it requires much less work than an ad hoc solution and does not require the ability to develop complex algorithms. On the other hand, a packaged solution has the same advantages of the framework, but it is not customizable by mean of specialization; so if it does not fit the problem, it use is not feasible. In our case we did not find a package that was able to balance load among zones. For specializing the code of the framework in Figure 9, we wrote about 500 lines of code in Java.

6. Conclusions and Future Work.

We have shown in detail the workings of our approach for developing BOF based on BPP that contain more business logic than other known patterns and frameworks. This included the presentation of realistic example frameworks. In particular we have presented a working procedure that can incorporate domain knowledge in providing generalized solutions that are able to be reused and specialized for integrated business process and Information Systems design in a given application domain. This also provides an approach to solve in a generalized and rigorous, business-design-based way, the requirements problem in system development for situations where complex business logic is involved.

So it is apparently feasible to have the best of two worlds in the support of complex business decisions: the advantages of pre-built software based on frameworks, with savings in developing costs, and also the option to easily customize a solution according to the specific characteristics of a given case.

Our research is continuing in several directions. Firstly, we have successfully applied the *Scheduling* framework of this paper to the actual assignment and routing of installation requests in a cable TV/Internet provider. Secondly, the forecasting framework is being extended to include cases not included in it, in particular for situations with complex variables such as fashion, using techniques such as neural networks and vector support machines [25]. Thirdly, frameworks for other activities in the value chain defined in this paper are being developed, such as supply chain management, production and operations planning, and logistics. Also we are working on the integration of these frameworks. In particular we are developing an integrated BPP and framework, that covers the whole value chain, with practices adapted to small and medium sized companies, and another one specialized to firms that develop engineering, construction, software and other types of projects. Finally, we are improving the way to package these frameworks to facilitate their practical use. In particular we are evaluating technologies such as EJB and web services as tools for this purpose.

REFERENCES

1. L. Aburto, R. Weber, Demand forecast in a supermarket using a hybrid intelligent system, in: A. Abraham et al. (Eds.), Design and application of hybrid intelligent systems, IOS Press, Amsterdam, Berlin, 2003, pp. 1076-1083.

2. J. Adams, S. Koushik, G. Vasudeva, G. Galambos, Patterns for e-business: A Strategy for Reuse, IBM Press, 2001.

3. J. Arlow, I. Neustadt, Enterprise Patterns and MDA, Addison Wesley, 2003.

4. O. Barros, Modeling and evaluation of alternatives in Information Systems, Information Systems 16 (5) (1991) 537-558.

5. O. Barros, Rediseño de Procesos de Negocios mediante el Uso de Patrones, Dolmen, 2000.

6. O. Barros, S. Varas, Frameworks derived from business process patterns. Technical Report 56, 2004, Industrial Engineering Department, University of Chile . Available at www.obarros.cl.

7. K. Bohrer, V. Johnson, A. Nilsson, R. Rubin, Business process components for distributed object applications. Communications of the ACM 41 (6)(1998) 43-49.

8. Business Process Management Initiative, Business Process Modeling Language, http://www.bpmi.org/bpml-spec.htm

9. M. Cline, M. Girou, Enduring business themes. Communications of the ACM 43 (5)(2000) 101-106.

10. J. Conallen, Modeling web application architectures with UML. Communications of the ACM 42 (10)(1999) 63-77.

11. S. Cook, Domain–specific modeling and Model Driven Architecture, MDA Journal, BPTrends, Janary (2004). Available www.bptrends.com.

12. N.P.Dalai, M. Kamath, W.J. Kolarik, S. Sivaraman, Toward an Integrated Framework for Modeling Enterprise Processes, Communications of the ACM, Vol. 47 Nº 3 , (2004) pp 83-87.

13. A. Díaz, Introducción de tecnología de inteligencia de negocios al proceso de ventas del área residencial de Telefónica CTC Chile, Professional Thesis, Industrial Engineering Department, University of Chile. 2002. Available at www.obarros.cl.

14. DMOS, Benchmarking and Best Practices, http://dmoz.org/Business/Management/Benchmarking_and_Best_Practices/

15. D. F. D´Sousa, A. C. Wills, Objects Components and Frameworks with UML. Addison-Wesley, 1999.

16. DTI, Best Practice, http://www.dti.gov.uk/bestpractice/

17. M. Fowler, Analysis Patterns: Reusable Objects Models. Addison-Wesley, 1996.

18. A. González, Diseño del procedimiento de administración de causas para el Ministerio Público, Professional Thesis, Industrial Engineering Department, University of Chile. 2001. Available at www.obarros.cl.

19. R. Hieleber, T. B. Kelly, Ch. Ketterman, Best Practices, Simon & Schuster, 1998.

20. IBM, Patterns for e-business, http://www-128.ibm.com/developerworks/patterns/

21. M. Jackson, Problem Frames, ACM Press Books, Addison-Wesley, 2001

22. T. W. Malone, K. Crowston, G. A. Herman, Organizing Business Knowledge: The MIT Process Handbook, MIT Press, 2003.

23. MIT, The MIT Process Handbook Project, http://ccs.mit.edu/ph/

24. OMG, OMG Model Driven Architecture, www.omg.org/mda

25. D. Pyle, Business Modeling and Date Mining, Morgan Kaufmann Publishers, 2003.

26. Siebel, Best practices, www.siebel.com/bestpractices

27. Supply Chain Council, Supply-Chain Operations Reference Model, http://www.supply-chain.org/page.ww?name=Home&section=root

28. A. Sutcliffe, The Domain Theory: Patterns for Knowledge and Software Reuse, Ealrbaum Assoc., 2002

29. Telemanagement Forum, Enhanced Telecommunication Map (eTOM), http://www.telemanagementforum.com/

30. The Economist, Economist Intelligence Unit, www.ebusinessforum.com

31. R. Vallette, Apoyo tecnológico a la administración de solicitudes de pabellón a partir de la arquitectura de un e-business, Professional Thesis, Industrial Engineering Department, University of Chile, 2002. Available at www.obarros.cl.

32. P. Vitharama, H. Zadei, Jain, Design, retrieval , and assembly in component-based software development, Communications of the ACM 42(11)(2003) 97-102.

33. White House E-Gov, Federal Enterprise Architecture, http://www.whitehouse.gov/omb/egov/a-1-fea.html