Business Information System Design Based on Process Patterns and Frameworks

Oscar H. Barros
Universidad de Chile, Departamento de Ingeniería Industrial, República 701, Santiago, Chile
obarros@dii.uchile.cl

Abstract

In this paper, I will present a novel approach for encapsulating high level knowledge and business logic in the design of Business Objects Frameworks. These frameworks are derived from formal and explicit Business Process Patterns, which are generalized designs that include best practices for businesses in a given application domain. A pattern and a framework derived from it can be applied to the design of a process for a given business in the domain and to develop an Information System to support such process. This provides a very flexible way, based on reusable components, to develop solutions and software for complex business decisions, and is an alternative to packaged products. The approach is exemplified by using a specific application domain.

1. Introduction

Competitive pressure, globalization, and the wide availability of Internet have made necessary the formal design of businesses. While in the past, business practices--rules, routines, procedures, and processes--could evolve in a piecemeal, isolated, and historical way, the likes of Amazon, Dell and FedEx, today, need a rigorous and systemic design of such practices, to insure that customers’ requests for products and services are processed at the speed of the Internet. In turn, this requires that the value chain process--from receipt of orders to the delivery of the product or service--be formally designed in an integrated way to assure a smooth flow of orders, in a mostly automated way. In order for automated practices to perform well, their design--expressed as business logic--should insure optimized management of key variables. For example, sales management should be based on sound analytical techniques--e.g., time series analysis and data mining--to predict customer behavior, and act proactively in connection with it. Credit management should also use predictive analysis--e.g., Neural Networks--to evaluate risk; supply management should apply mathematical models to optimize stock; and operations management should plan production or services execution to assure orders satisfaction and optimize the use of resources. Furthermore, the relationships among these decisions should be taken into account; thus, supply should be based on sales plans that consider customers’ behavior and/or production/service plans, which should also use sales plans.

The natural way to perform the design outlined above is to use a business process based approach, where all the variables of sales, supply, and production are managed in an integrated way, considering their interactions. This is the key proposal of this paper, but with an original addition: to base the design of a process in real cases on Business Process Patterns (BPP) that incorporate best practices that guide such design. These patterns--which have been tested and used in hundred of real situations--are presented in Section 2 of this paper.

The formalization of business design by means of BPP—which incorporate optimized business logic based on sound models and analytics--makes it also possible to incorporate automated system support as part of such design. This means that the usual Information System
requirements are derived from the business design in a formal way, which can then be translated to common software models, e.g., UML. We show how this is done in Section 3. Finally, generalized, reusable BPP and associated system requirements allow us to derive reusable software frameworks that can be used in tandem with the patterns to give the basis for joint design of the business processes and support systems in practical situations. Framework derivation from BPP is presented in Section 4.

There have been several attempts to provide approaches similar to parts of what we propose. In particular, in the idea of frameworks, several authors [7, 8, and 14] have established the need for Business Objects (BO) that represent things and behavior in a business domain and provide a solution to generalized, recurring problems in it. Such Business Objects (BO) would be organized [8, 14] in a framework that is not necessarily executable, but can be adapted and specialized to solve particular business problems. The value of a Business Objects Framework (BOF) depends on the relevance—in terms of impact on business results—of the business situation it represents, the quality of the support it gives to such situation, and the effort needed to make it work. Examples of specific well known attempts to implement these ideas are as follows:

i) The San Francisco Project [7] that, based on requirements derived for a vertical domain defined by several IBM’s business partners, developed an extendable component-based development platform. This included basic business logic for common business functions—e.g., financial management, order management, and the like—to be enhanced and extended by developers; Common Business Objects (CBO) that perform processing functions used in many applications domains; and a Foundation, which provides an infrastructure that is used to build the business logic and the CBO. These components were commercially available for a few years and are no longer marketed by IBM.

ii) Fowler’s patterns [14], that are published frameworks in domains such as accounting, billing, and payroll. They identify object structures and associated logic that synthesize generalized solutions in such domains. The logic considered is mostly processing logic and not true decision oriented business logic.

iii) The Catalysis approach [13], which proposes frameworks similar to Fowler’s, but for a wider range of domains. It attempts to cover some business decision logic, but at a basic, naive level.

All the above approaches share a common weakness, which is that they do not start with an explicit business process domain model that defines with precision the high level decision logic needed to run a business according to best practices.

In terms of a business design, recent proposals are as follows: One of the approaches proposes to specify business logic as a formal set of rules, before doing system or application design [21]. Another one, very popular at the time of this writing, is a process-based approach that is founded on a formal language (Business Process Modeling Language: BPML) that allows us to model processes and would eventually have execution facilities to run such models [22, 23]. These models do not use any domain specific semantics; they allow us to write business logic, but do not have any predefined one. It is clear that none of these approaches starts with a formal normative business model from which process and system design are derived.

An older proposal in the line of BPP is the MIT Process Handbook project which is a sort of knowledge base of business practices structured along business processes of different domains [19]. However, publicly available business practices in the website [17] of the project are very general and qualitative and do not consider formal relationships among them. Furthermore, it is not clear how to do process design using the handbook, and no connection with practices system support is considered.
This paper is also related to a recent paper in the CACM [25] that proposes a four-step development life cycle for component-based software. The authors of that paper aim to provide an approach driven by the developer business strategy. They do this by considering techno-economics managerial goals--cost effectiveness, ease of assembly, customization, reusability, and maintainability--in software development, goals which are adequate, but do not consider domain knowledge in component design. Our approach complements the above ideas by using domain knowledge in formalizing domain analysis--business design in our terminology--and component design, which are the first two steps in their proposal. Thus, our approach tries to assure that these components have a content that is right from the point of view of domain requirements, and can then be optimized in terms of structure design, according to said techno-economics managerial goals.

In summary, as compared to the approaches above, the most distinctive characteristic of our proposal is that it is closer to the most important decisions of a business than any previously proposed one and provides a very flexible, reusable component-based approach for supporting such decisions. It has been widely tested in real-life situations in Chile.

2. Business Process Patterns

Business Process Patterns (BPP) are models of how a business in a given domain should be run according to the best practices known [4]. 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 books [16], web sites [26, 27, 29], and direct observation of firms. Our patterns have benefited from the knowledge derived from hundred of cases in which processes of many different companies have been modeled, analyzed, and redesigned∗, and from previous experience with the formal modeling of Information Systems [3].

We have found that beyond best practices for a given domain--usually expressed in the form of a specific business logic, BPP share a common structure of activities and flows. Thus, products or services provision processes--such as manufactured goods, health services, justice services, financial services, etc.--share a common structure. A first level of detail of such a process structure for a very large domain is shown in Figure 1, where an activity-based modeling scheme that uses IDEF0 is shown. This pattern is a more precise version of the value chain of a firm [20]. Such BPP establishes which activities and relationships, by means of information and physical flows, should exist in practice in order that the type of business it realizes is well run. One activity in the model, called State status, is of particular interest, since it represents the centralized IT-based storage of data needed to support the process. Thus the BPP assumes that every transaction that occurs in the activities -- other that State status -- is informed to this store, and state of relevant entities is updated and fed back to former activities, by means of State status information, so that they can act upon the knowledge received.

Detail of flows--by means of attributes definition—and actions of activities, described by business logic, are given in the BPP dictionary, which is supported by software that runs IDEF0∗.

Further detail of any activity can be given by decomposition of it, following the IDEF0 scheme. For example, Figure 2 shows the detail of activity 1 in Figure 1. At this level of detail, our domain is still as general as the first one.

∗ Representative cases are published in the web site www.obarros.cl (in Spanish) and also examples of dictionary use.
If we want to give further detail, we have to be more specific about the domain, so that we can define the business logic and flows with precision. In order to show how to do this and use the same example for the rest of the document, we synthesize our experience of many real cases in the following domain definition for the activity **Marketing and customer analysis** of Figure 2.
Figure 2. Detail of *Customer relationship management*

We assume a domain where private firms sell products in a competitive market. Under this assumption, we decompose *Marketing and customer analysis* as shown in Figure 3.
Finally, to give more details of activity *Customer and sales behavior analysis* of Figure 3, we reduce the domain to situations where businesses sell physical products to a large number of customers; specifically, we have in mind cases such as retail using any channel--face to face, telephone, Internet, etc., and wholesale distribution and direct sales by manufacturing or service firms--e.g., telecommunications. Under this assumption, we decompose said activity in Figure 4, where we will concentrate on *Forecast model development*. For such activity in this specific domain, we can be very precise about the business logic that produces an optimal or near optimal solution that represents a best practice. Business logic, which guides the action in an activity, determines the exact information flows that are required and that are produced. We will show in the next section how such logic is specified.
We have shown details for a fourth level of decomposition of just one activity of a given domain. In a real-life situation, where a BPP is to be used to redesign a whole process, all the lowest level activities of it should be detailed, which, of course, we do not do here, because we are just presenting the way our approach works. Also, all the logic for the different activities should be consistent, since they generate the flows that allow the interaction among themselves, as shown in Figures 1, 2, and 3. Thus, for example, the logic for producing Sales plan message using Sales forecast models in Figure 3 should the right one in terms of the generation of information needed by Production & delivery management and Supplier relationship management in Figure 1.

Of course, BPP can be developed for any business domain of interest, which, besides the cases already presented, may include new product development, business planning, human resource management, financial resource management, etc. We have developed many of these BPP, which have been applied to business and process design in cases such as telecommunications customer analysis and service [12], surgical facilities management [24], and justice administration [15].

3. Business Logic Specification

Our aim is to give generalized business logic for a specific domain. In the case we are presenting, we have defined our domain, as outlined in previous section, as a situation--representative of many real life experiences--that can be formalized as follows.
Consider the activity of Forecast model development of Figure 4. We assume we have a situation where, due to the sales to a large number of final customers in a competitive market, a forecast based on sales history is possible. We also assume that, previously, a datamart with relevant and clean history has been set up in activity Customer and sales data base preparation of Figure 4. Then we can model the situation as in Figure 5, where an analyst in Forecast model evaluation will have System support for evaluation 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, calculate forecast error—e.g., mean absolute percentage error—by comparing selected history of forecast and actual sales.

ii) For selected sales items and forecast methods—e.g., Exponential Smoothing, Box-Jenkins, Neural Networks, fit data to model using historical sales data, proposing adequate model parameters, and providing estimated forecast error to analyst.

iii) Update models selected by analyst in State status for routine use in forecasting in Sales planning of Figure 3.

This is a simplification of real cases we have performed [1, 2, 11, 18] where the logic can be more complex, involving model identification and training with more analyst responsibility than the one outlined. Such additions and detail logic are included in a working framework that is currently being applied to sales forecasting in a supermarket chain.

The support modeled in Figure 5 is the typical requirement specified by a User Case of UML. Since our representation is more precise and consistent with the process design, we will use it to directly model system support in more detail by means of a Sequence Diagram. This is shown in Figure 6.

Figure 5. System support for model development
In order to give a flavor of the detailed business logic included in the system support shown in Figures 5 and 6, we outline a portion of the logic corresponding to the fitting and estimation of a selected model to historical data of the later figure. This logic, which is shown in Figure 7, corresponds to a highly simplified version of the identification and estimation of a Box-Jenkins model, once it has been determined that it is the most suitable for the series at hand. The logic is mostly 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, pricing policies, and the like. However, other methods, such as Neural Networks—stand alone or combined with Box-Jenkins—allow us to consider such factors, in which case business logic would explicitly consider the interaction between commercial decisions and forecast. These possibilities are considered in the full version of our framework.

4. From Business Process Patterns to Frameworks

From the BPP system support and business logic of the previous sections, we can derive BOF with BO that incorporate the knowledge about the solution of a relevant problem in the given

* An actual application of this idea was performed in the case reported in [1].
The purpose of this BOF is to provide a generalized solution to the problem that 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 [5]:

i) The structure of the BPP system support and the business logic of the domain gives a first cut definition of the BO classes that encapsulates 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.

//Business logic outline for Box-Jenkins model identification and estimation
//Previously, several tests for determining the suitability of Box-Jenkins v/s other methods have been performed; e.g., presence of tendency, seasonality and white noise, and consideration of 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 series difference until 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 analyst autocorrelation and partial autocorrelations graphics and proposed model.
Accept 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 analyst accepts model or decides that no model can be fitted
    Update model.
Else analyst establishes new analyses to be made.
    //This may mean going back to model identification or selecting a model different from Box-Jenkins.

Figure 7. Business logic for forecasting model development support

iii) Data needed to execute operations can then be derived from the data included in the business logic.

iv) 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 and collaboration among classes specified with a Sequence Diagram.

We follow steps above for the activity Forecast model development of Figure 5.

The structure of the system support and business logic in Figures 6 and 7 leads us directly into the BO structure of Figure 8, where we also show the data classes and the operations for each class.
We use common OO conventions for patterns and adopt some of the ideas in [9] to organize classes. The structure is not complete, since it should be integrated with all the components that support Sales planning of Figure 3, where forecast models are actually run to produce forecasts that are needed for generating sales plans, which we have avoided to simplify presentation.

The BO structure or framework of Figure 8 allows us to detail the way classes collaborate to support the development of forecast models, which is shown in Figure 10.

The BOF of the framework can be organized according to type of cases in the domain. For example, in forecasting model development, a typology can be defined according to the characteristics of sales data and commercial policy: cases with active marketing--such as promotions, opportunistic pricing and the like--and cases more passive; cases with stable sales behavior in terms of tendency and seasonality, and cases with no stability. It is obvious that analysis can be tailored and made more specific for each particular case. In Figure 10 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 problem. In a way, this is a structure of the application domain. In Figure 10 such structure is organized according to the variables of stability and type of marketing previously mentioned. Then, for each case in the structure, an appropriate analysis is provided, based on experience obtained from the results generated with the use of the most important analysis methods [1, 2, 11, 18]. Hence, when using the framework, only its relevant parts can be selected.

![Figure 8. Framework for forecasting model development](image-url)
Figure 9. Class collaboration for forecasting model development.

Another framework in which we use this idea of structure of cases is presented in [6]. This is a framework for scheduling—an activity that appears when decomposing Production and delivery management of Figure 1—in a domain that includes machine, on-site customer service (e.g., telephone repairs) and hospital surgical facilities scheduling. This framework shows 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—which can be selectively used according to the characteristics of the problem at hand.
Another characteristic of these frameworks is the possibility of having incrementally more complex logic for each of the cases in a framework, defined as outlined above. Thus, for example, for the forecasting case defined as stable sales behavior with tendency and seasonality, with little market intervention (stable, passive marketing of Figure 10), a first level of complexity offered by the framework could be simple Exponential Smoothing with tendency and seasonality, which does not need any statistical knowledge for its use and can be adequate for small and medium sized firms. A second level of complexity could be the Box-Jenkins method presented above, which requires statistical skills to be able to exploit its possibilities. This can be appropriate for larger businesses where a more active marketing may need a greater precision and longer range forecasting capabilities can be of value, which will justify providing the necessary skills for its use. Hence, in the application of a BOF to a particular situation, the user of the framework may select the minimum level of complexity that solves his problem. Thus, for example, some developers will select in Figure 10 just the class Analyzer-stable, passive marketing, according to the recommendations above. Others will select both the former and Analyzer-stable, active marketing, in which case some sales items could be forecasted by using simple Exponential Smoothing, while others, with more complex behavior, could be forecasted by using Box-Jenkins. The framework advises, in this case, which method is the most appropriate.

The implementation of this feature, which allows the selection and use of incrementally more complex solutions for a case, is based on OO inheritance. We have coded our framework based on this feature and determined that it is very simple to select and combine the options that they offer and specialize them to particular applications. For example, the second selection of the previous paragraph will mean that the method Estimate ES and BJ will inherit Estimate ES, and this method itself can be inherited by a further specialization to include analyses tailored to a particular application.

The framework we have used as an example has been presented as stand alone, which is not realistic. In some cases this would be integrated with other frameworks for other activities in a process, as outlined in Section 2; in others, it can be used without integration, but it should be at
least connected to the business data bases, which contain data needed by the framework, instead of duplicating it.

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

These frameworks are now being routinely used to develop specific applications for real-life situations in their domains, as part of formal projects performed by students of a graduate program in Business Engineering at the University of Chile, to help companies in this country to innovate in their management and improve IT support to it.

5. Conclusions and Future Work

We have shown in detail the workings of our approach for developing BOF based on BPP. This included the presentation of a realistic example framework. In particular, we have presented a working procedure that allows to incorporate domain knowledge in providing generalized solutions that can be reused and specialized for integrated business and system design in a given application domain. This also solves, 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 development costs--and the option to easily customize and optimize a solution according to the specific characteristics of a given case.

Our research is continuing in several directions. First, we are applying the full version of the example framework of this paper to the actual solution of a real life supermarket forecasting case in Chile. Numerical results of such an application will be presented in a sequel paper. Second, such framework is being extended to include cases not included in it currently, in particular, for situations where analytical methods do not work well. Thirdly, frameworks for other activities in the value chain defined in this paper--supply chain management, production and operations planning, and logistics--are being perfected. Also, we are working in the integration of these frameworks; in particular, we have developed an integrated framework--which covers the whole value chain--with practices adapted to small and medium sized companies [10]. Finally, we are perfecting the way to deliver these frameworks for practical use by using technologies such as EJB and web services. A first test of these technologies was done with the framework for small and medium-sized companies, which was developed using EJB.

References


2. A. Arredondo, Rediseño del proceso de programación de la producción en Papeles Cordillera, Professional Thesis, 2002, Industrial Engineering Department, University of Chile (Available at www.obarros.cl).


10. J. Contesse, Diseño y construcción de componentes de negocio a partir de patrones de procesos para la creación de software de apoyo a PYMES, Professional Thesis, 2003, Industrial Engineering Department, University of Chile.


12. A. Diaz, Introducción de tecnología de inteligencia de negocios al proceso de ventas del área residencial de Telefónica CTC Chile, Professional Thesis, 2002, Industrial Engineering Department, University of Chile (Available at [www.obarros.cl](http://www.obarros.cl)).


15. A. González, Diseño del procedimiento de administración de causas para el Ministerio Público, Professional Thesis, 2001, Industrial Engineering Department, University of Chile (Available at [www.obarros.cl](http://www.obarros.cl)).


18. A. Jiménez, Diseño y construcció de componentes de negocios analíticos a partir de un patrón de procesos orientado a medianas empresas, Professional Thesis, 2003, Industrial Engineering Department, University of Chile.


24. R. Vallette, Apoyo tecnológico a la administración de solicitudes de pabellón a partir de la arquitectura de un e-business, Professional Thesis, 2002, Industrial Engineering Department, University of Chile (Available at [www.obarros.cl](http://www.obarros.cl))


26. [www.bwpccoe.org](http://www.bwpccoe.org)

27. [www.ebusinessforum.com](http://www.ebusinessforum.com)


29. [www.siebel.com/bestpractices](http://www.siebel.com/bestpractices)