

1

Business process redesign



Adopting an object-oriented approach is ultimately motivated by the need to develop applications. In this chapter we will look at business applications.

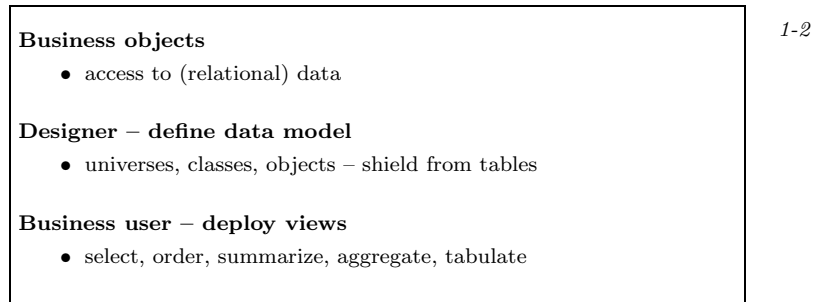
Business process redesign	11	1-1
<ul style="list-style-type: none">• business objects – the SanFrancisco framework• business process modeling – simulation• visualization support – collaboration and decision making• migrating from legacy applications – business objects		
Additional keywords and phrases: <i>business objects, business logistics, frameworks, object-oriented simulation</i>		

Slide 1-1: Business process redesign

We will start by discussing the San Francisco framework, which offers template business objects and business processes for developing (business) applications. Since IT is becoming the spine around which business is organized, we will explore methods for modeling and simulating business processes. We will then briefly describe an object-oriented simulation toolkit, and discuss support for the visualization of business processes and its potential role in collaborative decision making. Finally, we will reflect on the need to migrate from legacy applications.

1.1 Business objects – SanFrancisco framework

What are business objects? From the perspective of typical business end-users, that is accountants, engineers, managers, business objects provide access to corporate information. Traditionally, corporate information resides in legacy databases, and access means looking at tables and pasting these into documents.



Slide 1-2: Business objects

Even within this kind of limited usage, it makes sense to speak of *business objects*, as a way to shield the user from the actual structure of the underlying (relational) database and the use of query languages such as SQL to obtain the actual data, see [BO]. Business objects, as a metaphor for bringing information to the desktop, allow for accessing corporate databases in a transparent manner, and for building the tools that allow business end-users to extract information from the database in a flexible way, and to use this information for further analysis and manipulation. In [BO], a distinction is made between three types of end-users involved in the construction and use of business objects. The *designer*, who creates the objects that act as an interface to the database, the actual *end user*, who uses these objects to obtain information (and thus implicitly creates queries), and the *supervisor*, who provides users with access to the various regions or universes defined for the database. The actual *business objects* toolset, described in [BO], provides a GUI-based drawing tool to create queries by composing objects and creating relations between objects. Business objects, then, are a means to access corporate data. This is a first, but nonetheless important, step. An immediate advantage, obviously, is that business objects may be defined according to actual business needs, instead of being dictated by (relational) technology. More generally, business objects may be regarded as abstractions underlying the definition of business processes. In the following we will explore whether we can generically define business processes and whether there is a collection of abstractions that we may denote as *business objects*.

The SanFrancisco framework

The SanFrancisco framework (IBM) is an example of a framework meant to develop business applications. A business application, as we understand it here, is an application that deploys business objects to (partially) automate business processes, such as order or warehouse management.

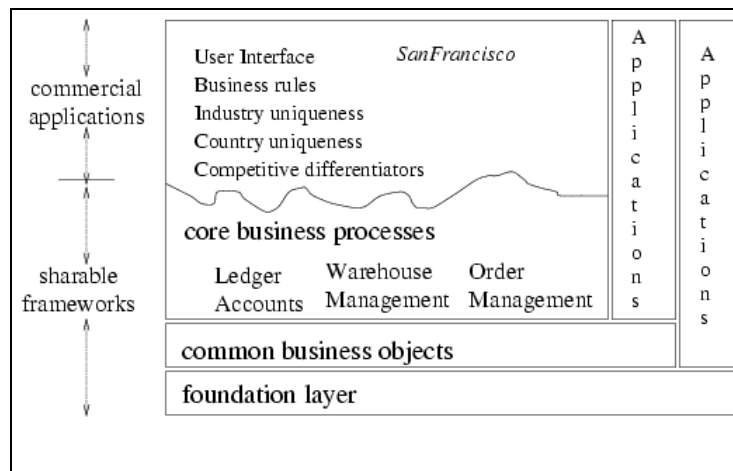
(Business) frameworks

- collection of components
- generic solution for a class of problems
- frame of mind for solving problems
- set of architectural constraints

1-3

Slide 1-3: Business frameworks

A framework is a collection of components, but may also be considered as a generic solution for a class of problems. It sets a frame of mind for solving problems and provides the means to realize solutions in software. In practice, adopting a framework means accepting a set of architectural constraints.



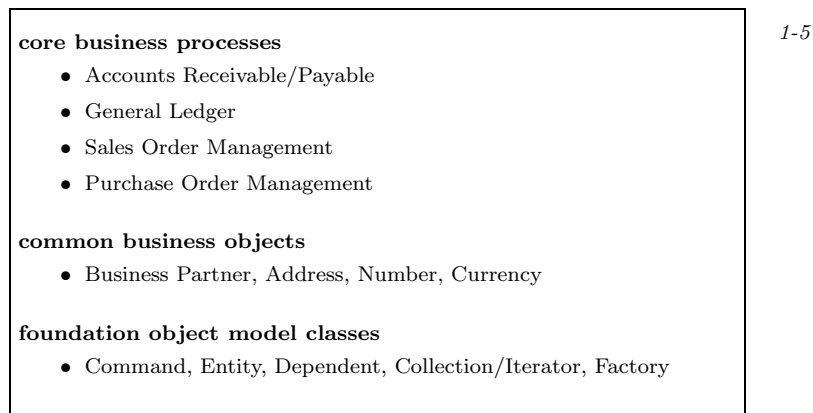
1-4

Slide 1-4: The SanFrancisco framework

As a framework, SanFrancisco aims at providing both a software solution for implementing business applications, as well as a collection of concepts or strategies to develop effective business applications. In the white paper accompanying the introduction of the SanFrancisco framework, we read that the project was started when several software vendors asked IBM to help in modernizing their application products. These vendors asked for help because there were several barriers that prevented them from modernizing their applications themselves. Barriers such as

(1) the risk in moving to new technologies (such as client/server and the Web), (2) the need to retrain their development staff to effectively use an object-oriented approach, (3) the cost of change. As the white paper states, as software developers they needed some basic infrastructure, and most companies could not afford to develop this infrastructure themselves.

The SanFrancisco framework provides such an infrastructure, and moreover, the white paper claims, an object-oriented infrastructure that provides a consistent application programming model, with many well-tested services and a collection of *core business process components* and *common business objects*.



Slide 1-5: SanFrancisco object layers

The SanFrancisco framework offers three layers of functionality, business processes, business objects, and foundation classes, each of which may be used and extended by developers to build their applications. The *process* layer itself may be regarded as a collection of frameworks, as indicated in slide ??, which build upon the business objects and foundation layers. Note that the foundation layer contains realizations of the by now familiar patterns, see chapter 2.

The SanFrancisco framework is an object-oriented framework. It allows for the classical way of extending the framework, by inheritance. As an example, think of a *Receipt* object which may contain an arbitrary number of *Purchase Order Line* instances, see slide ??. *Purchase Order Line* has an attribute *Quality inspect* which is read by *Receipt* to determine whether quality control is needed. This reflects the default (business) logic for processing orders as defined by the framework. To enhance this logic, that is to be able to execute more strict quality control, one may derive a new *Receipt* class from the old *Receipt* class and override the method by which to determine whether quality inspection is required, for example by including a check on the supplier, previous outcomes of quality control, or whether it concerns hazardous materials or high-value products. As a remark, note that this approach assumes that the business logic is to a large extent hardwired in the (structure of the) classes of the framework, whereas a decoupling of the logic and the actual processing might be preferable.