

Author: Jon Barnett  
 Date: 24 July 2002

## 1 Developing a catalogue application with FlexCorp

The first thing to understand is that we don't have a unique perspective on system analysis. The high-level requirements and recognition of the important features of the systems and processes are a necessary starting point for using the FlexCorp framework.

From our perspective, an entity is an object that is an active participant in the business process. These entities are typically identified in the system requirements statement or in a context diagram and the system context description for those who still use such analysis tools. Entities are significant objects in the system and are complex in their representation. Menus are not entities nor are category hierarchies. We would rarely find them explicitly mentioned in the system context description nor are they significantly complex in the expression of their representations. They are a means to an end in the physical system.

I still use DeMarco's DFD methodology to synthesise high level models of the system. It is quick and delivers the information in a succinct form. However, you may use any methodology with which you are comfortable to achieve our goal. Our attempt at a high level definition of the system is provided below.

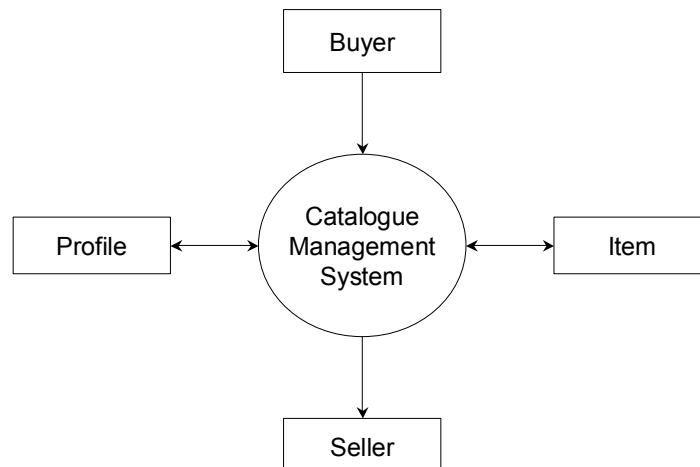
### 1.1 System context

#### 1.1.1 Aim

The system is to allow people to manage a catalogue of items so that selected others may view the contents of the catalogue.

1. All users of the system must belong to an organisation.
2. An organisation owns a catalogue item and only users that belong to that organisation may modify the contents that describe the item.
3. An organisation may only view an item if it has been given access rights to that item.
4. Items are organised hierarchically.
5. Users are assigned security profiles that control their access to system functions.
6. Security profiles are group based rather than created for each individual.
7. A user can only change an item's information if they have access to do so.

#### 1.1.2 Context diagram



## 1.2 System definitions

With this minimal definition, we are able to extract the entities and the important relationships. The FlexCorp framework does encourage a certain level of abstraction from the details of modelling an entity. The flexibility to add and remove characteristics from the entity definition allows the focus to stay firmly on the critical areas of system operation. For example, in the definition of the system operation, if the user entity did not contain an address field, it would not cause the system to fail in a critical manner. Such details of definition are for the business analyst to synthesise but are not necessary for the developer as these add no value to the understanding of the system operation.

The other operational point of relevance is the manner in which the information is delivered including the criteria for access. These are largely design choices with influence from a more detailed understanding of the user needs.

The navigation within the system is largely driven by the preferences of users and the developer and this applies to the user interface. Flexcorp provides some helper objects and helper data (templates and filters) to simplify this area of development but otherwise the navigation shell and presentation is left to the developer.

### 1.2.1 FlexCorp entity specifications

Entity	Definition	Relationships
Organisation	Represents either a purchasing or selling body	Owns items and has personnel
Users	Represent personnel belonging to an organisation	Must belong to an organisation
Profile	Represents access to system functions and system actions	Must have a security profile None
Items	Represents items provided by an organisation for other organisations to purchase	Must belong to an organisation  May be viewed by specified organisations

### 1.2.2 Modes of access

Entity	Function	Type of access	Constraints
Organisation	getOrganisations	List of all available organisations	None
	getOrganisation	Details of a single organisation	Based on organisation reference
Users	getUsers	List of all available users	None
	getUser	Details of a single user	Based on user reference
	getUserPassword	Security details for a single user	Based on user reference
Profile	getProfiles	List of all available profiles	None
	getProfile	Details of a single profile	Based on profile reference
Items	getItemsForBuyer	List of all purchasable items for a category definition (catalogue)	Based on (buyer) user reference and category definition
	getItemDetails	Details of an item (catalogue)	Based on (buyer) user reference and item reference
	getItem	Details of an item (edit)	Based on (seller) user reference and item reference

### 1.2.3 Support objects

The only mentionable objects that are required are a menu-state system and a category hierarchy system.

## 1.3 Implementation

With regard to actual implementation, we made some design decisions for our concrete delivery. These are covered in the next subsections. For the most part, the composition of entities and their characteristics (the data and the meanings of the data that define an entity) are controlled by the business analyst through the FlexCorp management system. This is the strength of FlexCorp; the ability to change the composition of an entity, to change the information that describes the entity to the presentation service and to change the manner of retrieving entity information without programmatic intervention. There are necessary programming changes when the entity changes affect control information for the application using the entity but the reduced complexity will simplify the structure of such applications and deliver clearer identification of critical maintenance areas.

### 1.3.1 Entity and entity language definitions

The first part of the implementation is for the business analyst to create the entities and their language for characteristics using the FlexCorp management system. Essentially, this consists of defining the entity names and the descriptions for the characteristics. For example, for the phone products and their technical specifications, we might have such characteristics as GPRS, IRDA, WAP and so forth. This allows a more natural way of conversing for the business analyst and other non-technical members of the project. The developer is isolated from such details and allowed to concentrate on the construction of the application and the critical control aspects. The other benefit is that we can add new language as the need arises. For example, suppose we need additional specifics for an implementation such as divisional information for a product owner. The FlexCorp framework allows such changes.

### 1.3.2 Templates

Templates define the order, manner and requirement for editable information of an entity. The presentation layer can use the information retrieved from the template to govern the display and post processing for data edits. The template is retrieved using any naming schema the developer wishes to implement. We generally apply our template naming as *application.entity.context-id*. You can use state management within the presentation layer to construct the appropriate reference.

As an example, the organisation information we construct as gathered by our business analyst is:

Label	Display	Type	Length	Height	Quality	Parsing
Full name		String	40		Mandatory	Any
Description		String	50	6	Optional	Any
Address		String	40		Mandatory	Any
Address		String	40		Optional	Any
City		String	20		Mandatory	Any
State		String	3		Mandatory	Any
Postcode		String	4		Mandatory	Integer
Country		String	25		Mandatory	Any

By using a template, a business analyst can control the information created for an organisation. The developer implements a presentation routine that processes and generates the input screen based on the information contained in the template. The template is also used to perform post-processing before the insertion of the data.

For the user we have a more interesting template as we specify the list sources for the owner organisation and the security profile. It is up to the developer to interpret this information. Note that the Owner and Profile fields are implemented from the requirements.

Label	Display	Type	Length	Height	Quality	Parsing
First name		String	40		Mandatory	Any
Last name		String	40		Optional	Any
Password		String	16		No Del Password	Any
Owner	List:Organisations	String			Mandatory	Any
Profile	List:Profiles	String			Mandatory	Any

The benefit of this is that the processing can be done in a programmatic manner and therefore is flexible to informational changes. Should the business analyst find any additional information requirements, as long as the information has no relational implications these may be introduced through the template definition without programmatic changes.

We created templates for all four entities which you may view the results of while browsing the application.

### 1.3.3 Navigation

The navigation structure we chose was a hierarchical menu based around the entities. For each entity we have submenus for create and edit. We also have top-level menu items for managing the category hierarchy and for access to catalogue views. The editing services have been constructed to allow both edit and delete. This was a design choice. These choices govern the characteristics of the security profiles. So we have a menu security mask and an edit and delete security mask. The business analyst can define the masks and if there is an increase in the menus and functions, a new mask can be quickly fabricated through the FlexCorp management system.

### 1.3.4 Functions and filters

The functions mentioned above are implemented through the FlexCorp interface as are the criteria. These can be changed as function requirements change through the lifetime of the system.

The filters are applied to the entity data once the functions retrieve the entities. Filters act to re-sequence or remove the select characteristics data of the entity for display or other use. Additionally, formats may be obtained and applied to the data after sequencing. Through the FlexCorp management interface, the filters and formats can be defined and redefined as necessary for the lifecycle of the system.

### 1.3.5 Entity accessors

For each entity type, a new EJB accessor must be built for the entity. These are straightforward to implement as there are no major deviations from the standard; such deviations occur around the entity creation phase or with additional tasks aside from the update of an entity's characteristics. All entities in this design context are name-string based, rather than being index based such as job allocation. The standard implementation source for this is copied and modified accordingly. FlexCorp supports index based entities as well, and there is a standard implementation source for this. The task takes about ten minutes to create a new entity accessor. There are 230 lines of code and the modifications involve replacing the names and a definition string.

### 1.3.6 Additional objects

We have built a menu storage service and a menu management system for navigation. As we feel these are re-usable items, we have bundled these into the FlexCorp framework. This is probably our greatest time investment in this particular implementation. We will also retain the security and web navigation framework for future projects and applications as it delivers a flexibility that complements the FlexCorp framework.

In combination with this, we devised a means of converting the hierarchy into a flat structure for the purpose of incorporating this into the item definition. The combination of these functionalities allow simple intuitive creation of hierarchical categories and marrying this with the functional requirements of classifying products.

### 1.3.7 Presentation service

The presentation service is servlet-based, this particular instance running on the Tomcat Catalina implementation. The implementation has security checks to prevent access without password verification. It is also very flexible in managing navigation and allowing customisation. The state management is embedded in the menu system and there is a central routing core that acts as a state machine, controlling the navigation. The implementation of the characteristics management in FlexCorp allows real software re-use so the input processing for all entities is handled by a single routine.

## 1.4 Results

The results of the development can be found at <http://www.pteradactyl.tzo.com/salesmanager>. The system is a full-fledged implementation using the scaling capabilities of EJB and servlet technology. The application code for the presentation service is contained in a 76 Kb WAR file. The development time was three weeks, of which two-thirds was spent on the design and development on the menu state system and security, the navigation framework and the category management.

You can use the 'Amity' login to check the system. The password is 'amity'. The login is case-sensitive so please ensure you type these as we specify here. The security profile allows you to access all areas of the application except the category hierarchy. You cannot create any new users, profiles, organisations or items. However, you can view existing data. The security profile also disables all modify and delete operations.

Some existing browseable objects are:

**Organisations:** Amity, Avnet

**Profiles:** Administrator, Operator, Demonstrator, Buyer

**Users:** jonb, Amity, Avnet, guest1

**Items:** ME45, M50, 3330, 3910, T100

This base implementation runs on a Pentium II 333 MHz with 96 Mb of RAM running Linux. The web application runs in the Tomcat Catalina servlet server and the FlexCorp system runs on the JBoss EJB server, both on this Linux system. The database is Postgresql running on a Pentium I 125 MHz with 64 Mb of RAM. The FlexCorp service also supports integration with Oracle. Porting the service to other databases depends on customer demand and the capability of the database.

## 1.5 Future Plans

We will look to develop add-ons to allow purchasing by adding robust shopping cart services, and sales management and tracking capabilities. Although we've done this before, we want to re-examine approaches that will meet the flexibility criterion we build into all our products.