A Development Process for Enterprise Information Systems Based on Automatic Generation of the Components

Adrian ALEXANDRESCU, "Ovidius" University of Constanţa,
Faculty of Matematics and Informatics

This paper contains some ideas concerning the Enterprise Information Systems (EIS) development. It combines known elements from the software engineering domain, with original elements, which the author has conceived and experimented. The author has followed two major objectives: to use a simple description for the concepts of an EIS, and to achieve a rapid and reliable EIS development process with minimal cost. The first goal was achieved defining some models, which describes the conceptual elements of the EIS domain: entities, events, actions, states and attribute-domain. The second goal is based on a predefined architectural model for the EIS, on predefined analyze and design models for the elements of the domain and finally on the automatic generation of the system components. The proposed methods do not depend on a special programming language or a data base management system. They are general and may be applied to any combination of such technologies.
Keywords: *Enterprise Information System, Conceptual modeling, Software development process, Automatic Generation of Components.*

1 The Modeling Elements for the EIS Concepts

We'll consider an EIS as an *event* processing system. The events appear, in the enterprise activity at different moments, and they are in general, descriptions of the usage and/or transformation of the enterprise resources, at a given moment. Examples of events may be the receiving of an order from a customer, for products or services delivering; the employment of a new person in the enterprise; the emitting of an invoice by a supplier, for its services done in a certain period, etc.
We'll associate to an event a synthetic description which contains identification elements, the initiator of the event etc., and also one or more detailed descriptions of the event, defined by sequences of *actions*. The events, both by synthetic description and by detailed description, refer to a set of concepts from the enterprise activity domain, which we'll name *entities*. An entity may be: a customer, a product, a supplier, an employee, etc. The entities can be things, beings or abstract notions, involved in the events. The enterprise resources are also entities.
Between the concepts presented above may exist association relations.
The state of an entity is a quantitative and qualitative evaluation of an entity at a given moment. The stock of a product at a given moment is the state of a product type entity at that moment, containing an evaluation of its quantity and value. Usually, we are interested about the state of the enterprise resources at a given moment.
The entities, the events and the states have some characteristics which we'll name *attributes*. An attribute may take values from a predefined set, which we'll name *attribute-domain.* The attributes can also represent references within associations. For example, the customer entity has a name, an unique identification code, an address, etc; the order emitting event has an identification number, an emitting date, an identification customer code, etc. ; the action associated to an emitting order event may have as attribute the product code, the measure unit, the quantity, etc.
By analogy between an event description and a natural language compound sentence, we may consider an action the equivalent of each sentence from the compound sentence, the nouns may correspond to entities and determinants may be similar to the entity and action attributes.

We define the state of the system at a moment as the set of the system entities states at that moment.
The concepts of the same type (with the same attributes) we'll group in sets. So, we'll have event-sets, action-sets, entity-sets and state-sets. For example, the trading products will form an entity-set, which we'll name Products, the employees of an enterprise form the entity-set Employees, etc. Similarly, the orders emitting from the customers we'll group into an event-set named Orders. Also, the orders content from the customers we'll group into an action-set called Orders-Content. Finally, the products stock, considered at different moments (for example at the beginning of every month) will form a state-set which we may name Stocks.
By *system conceptual elements* or shortly *system-elements* we'll understand: the entity-sets, the event-sets, the action-sets, the state-sets and the attribute-domains. The system-elements will play a very important role in the EIS developing process.

2. Predefined Analyze Models, for the System-Elements

Once identified, a system-element dispose of a predefined model for the analyze and design phases. The user will perceive a system-element by means of a predefined user-interface associated with that system-element, this meaning a predefined structure and a predefined use-case. The parameter parts of the predefined models are specified using a system-element description language (SEDL).
For example, an entity-set has the user-interface structure (I_{ES}) presented in figure 1.
The user-interface is an object, which contains data and operations, available to the user, if it has certain rights.
The entities have the *presentation attributes* A_1, A_2,…, A_p, for the user. The user-interface contains the predefined operations O_1, O_2,…, O_m. As operation examples we may have: the detailed view for the current entity, entering a new entity, edit the attributes values of an entity, etc.
The I_{ES} object contains four parts: the tabular image, the detailed image, the filtering elements and the operations.
The tabular image contains the entity-set. The columns correspond to the presentation attributes, and the rows, to the entities. The tabular image has an *entity-selector,* which indicates the *current-entity*.
The detailed image contains the presentation attributes values for the current entity. The detailed image is used when demanding operations such as: detailed viewing, enter a new entity and modifying an entity.

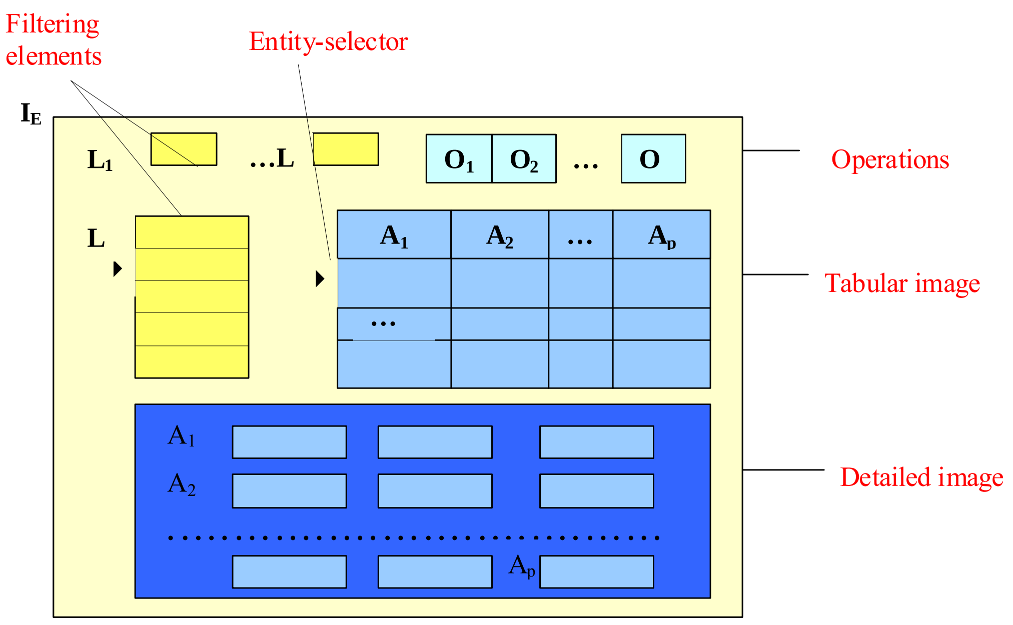
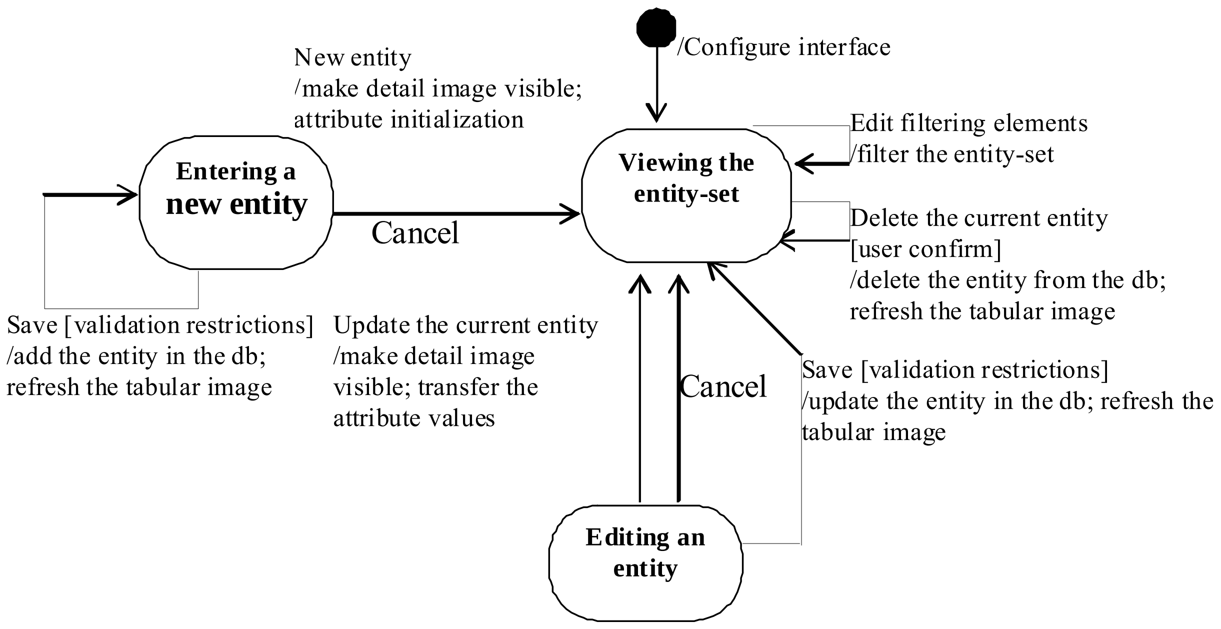


Fig.1. The structure of the entity-set user-interface, I_{ES}

The filtering elements enable the viewing of a subset of the entity-set. They correspond to some presentation attributes (filtering attributes) of the entity-set and may be shown as combo-boxes (L_i) or list-boxes (L). By user selection of some values from these lists, the tabular image will be filtered to those entities, which have the filtering attribute values equal to the selected values.
The use-case of such an user-interface may be described by a state transition diagram, as in the figure 2. There is synthetically described, the interaction between the user and the user-interface object I_{ES}, for an entity-set.

The events: new entity, delete the current entity, edit the current entity, save and cancel, are triggered by clicking the corresponding buttons of the user-interface. The diagram contains only the main elements of the interaction between the user and the interface I_{ES}. There are missing from the diagram, the detailed viewing demand, the configuration of the updating user rights, tabular image configuration commands (change columns order, resize column, hide columns etc) sorting tabular image, finding entities in the tabular image, more filtering facilities for the tabular image etc.



/Configure interface
New entity
/make detail image visible; attribute initialization
Entering a new entity
Viewing the entity-set
Cancel
Edit filtering elements
/filter the entity-set
Delete the current entity
[user confirm]
/delete the entity from the db; refresh the tabular image
Save [validation restrictions]
/add the entity in the db; refresh the tabular image
Update the current entity
/make detail image visible; transfer the attribute values
Cancel
Save [validation restrictions]
/update the entity in the db; refresh the tabular image
Editing an entity

Fig.2. The use case description for the I_{ES}

Similarly, there are defined models for the other system-elements: attribute-domain, event-set, action-set and state-set.

3. The Design of the System-elements

The design of a system-element consists in specifying its attributes and the properties of the attributes, using a system-element description language (SEDL). In general, each system-element, need one relation in a relational data base.
We'll give again as a system-element example, an entity-set, noted ES. For the persistence purpose, we'll associate a relation, R_{ES} from the data base, with the schema R_{ES} (A_1, A_2… A_n). We'll consider that the relation schema is in the third normal form. We'll name the attributes A_1, A_2… A_n *physical attributes*. Some of them will be used for the logical identification of those entities, some for the representation of the characteristics of the entities, some for referencing other entities, and another attribute might be used for referring the entities themselves.
It is possible to wish the usage, within the user-interface, of other attributes having meaning for the user in the context of the system-element. These attributes, together with the physical attributes are referred, in this article, as *presentation attributes*. They can be obtained, using *queries* expressed in SQL, which may imply more relations from the database, connected by join operators.

The tabular image from the user-interface may have as data source, the associated data base relation of that system-element, or a query object. These things, will be also specified, using the system-element description language, SEDL. SEDL enables physical and presentation attributes description for every system-element, the keys, the attributes data type, the attributes properties, etc.

4. The Automate Generation of the Components

To each system-element, corresponds a component of the final system. The automatic generation process goes from the predefined model of the user-interface of the system-element and the description of the system-element in SEDL. After the generation process there are created one or more classes which implement the user-interface for that system-element.

The author developed a prototype of the generation process, using the VBA language and the Microsoft Access. The generated components are MS Access objects, which the developer may view and even add new functionality or modify the contents of the generated objects. The main gain of the automatic generation is that the developer find a lot of work already done, in general a routine work and theoretical, without errors. This is another gain, that the automatic generated components do not require testing.

5. Using a predefined architecture for the EIS

The EIS resulting from the developing process, uses a predefined architecture organized on three layers, as in figure 3.

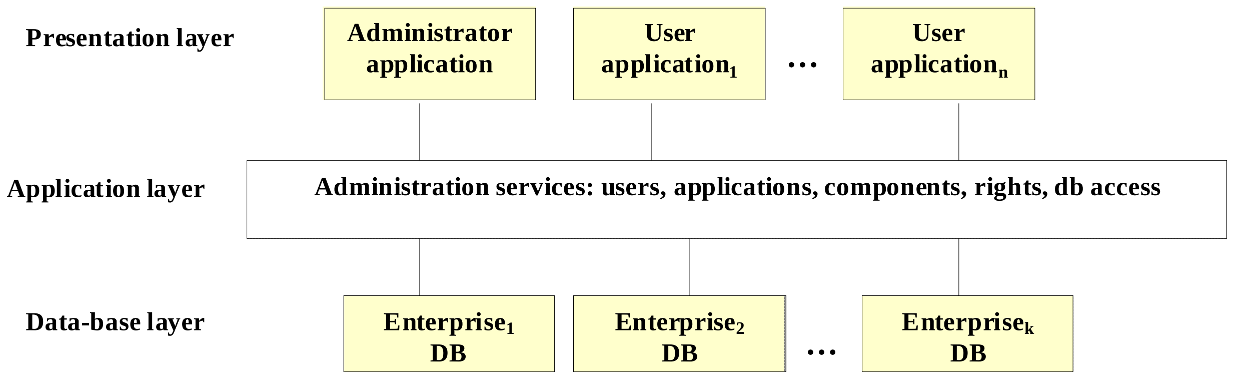


Fig.3. A predefined architecture for an EIS

The architecture presented in figure 3, is the architecture of a multi-user, multi-application and multi-enterprise information system. The application layer and the administrator application are predefined. The user applications, has a predefined kernel, to which we add a number of automatic generated components. The databases may be also automatic generated or created by the developer. The author also developed a prototype for this architecture.

References

1. D. Zaharie, I. Roşca, The Objectual Design of Information Systems, Dual Tech 2003
2. D.Oprea, D.Airinei, M.Fotache, Business Information Systems, Polirom 2002
3. L. D. Serbanati, Software Engineering - Course Notes, Ovidius University of Constanţa, 2004
4. Building Applications with Microsoft Access, Microsoft Corporation
5. Microsoft Office Visual Basic Programmer's Guide, Microsoft Corporation