Overview of Dulcian's Systems Development Approach

Dulcian's mission statement since soon after its inception was to try to create an omnibus business rules environment where all of the rules governing a system could be placed in a user-friendly repository. The data in this repository would fully specify the system. Once this specification exists, these rules can be used to either generate some or all of the system, or alternatively, the rules can be accessed at runtime.

The specification of the system rules makes no assumption about the implementation mechanism. In fact, the articulation of the system rules is independent of the implementation itself. This independence has proven itself to be a strong argument for building systems this way.

As an example, in one system, approximately 300,000 lines of PL/SQL code were generated to support reading and writing screen forms from the database. All of the rules to support this were stored in Dulcian's Business Rules Information Manager (BRIM®) repository. All of the code was generated from the repository. Well into the project development, users decided that it would be useful to provide this same functionality offline (i.e. not connected to the database). To that end, the code generator was rewritten to generate Java that would read and write from an XML source rather than PL/SQL to talk to a database. As a result, it was possible to rewrite the 300,000 lines of code with only a few weeks of effort.

Repository Structure

What does the repository to specify an entire system look like?
The repository includes four different types of rules have been identified

1. Object rules

Rules governing things of interest in the system. Object rules have two "flavors"

- Structural rules (information normally placed in data models) In BRIM®, structural rules are supported using extended UML class diagrams.
- Process rules (information normally placed in process flow or workflow diagrams) are supported in BRIM® using extended UML activity diagrams.

The object rules were the first rule types supported by BRIM®. At the time, even though it was recognized that the rule repository was incomplete, it seemed that the idea could be extended to ultimately support all of the rules in a system. This turned out to be a false assumption. There are rules in a system that do not conveniently fit into this model. Therefore, other rule type classifications were needed.

2. Object Interaction (transformation) Rules

This type of rule was needed for data migration since these rules do not apply to individual objects. This type of rule indicates that "this complex set of objects can be transformed into another set of complex objects." A different type of repository is needed to handle these rules including elements of both structure and process. A typical object interaction rule might be "loop through all of the organization's departments and child employees and then go to another database and create Organization, Person and Employment records.

The concept of object interaction is not limited to data migration. It crops up in several other places when designing and building a system. EDI applications, any time sets of information need to be transferred from one system to another, extracting an XML file to be passed to a Web Service are examples of places where object interaction rules are needed. In web applications, user transactions may need to be placed in a staging area where they pass through a process flow and are validated, prior to being migrated into the main database. Object interaction rules functionality is also useful for moving information in and out of screen forms to the database as mentioned earlier.

In BRIM®, the Mapper area supports object interaction rules.

3. Application Rules

Although default applications could be generated from the BRIM® repository to build working systems using only object and object interaction rules, the applications being created were not as user-friendly and attractive as desired. They did not nicely support the user requirements for a good user interface.

The user interface that was generated allowed the object rules to be tested but did not deliver a professional looking end result. Solving this problem involved more than making fields the right shape and size and placing them properly on the screen. A good deal of logic exists in applications that the existing repository could not represent. For example, in a purchase order system, in looking at a purchase order detail, a user might want to double click on the product name and navigate to the product maintenance screen. This type of program logic is the portion of the application that best lends itself to a repository-based approach.

The actual screen layout of the application does not lend itself well to using a repository-based approach. Using tools such as Oracle Designer makes it very difficult to design an attractive user-friendly screen by selecting various properties. Differences among screen layout mechanisms in different thin web client (JSP/HTML page) and thick client (Oracle Forms) are so radically different that it is almost impossible to successfully apply generic layout rules to both environments.

In the BRIM® environment, screen layout rules are explicitly not included in the repository. Application rules in BRIM® are supported by the Generalized Application Rule Processor (GARP™).

4. Validation Rules

Object, Object Interaction and Application rules provide sufficient richness to fully specify a system. However, another class of rule exists that require a separate repository because of the typically large number of them contained in an organization's information system. These "validation rules" include everything from the trivial "In the Employee record, Last Name is not null," and simple comparisons "Start Date is less than End Date" to complex rules spanning records in multiple tables such as "The day on any timesheet detail must be between the start date and end date of the timesheet header."

In complex order processing or HR systems where sophisticated validation is required, the number of rules to support one business function may be very large. For example, a description of the rules to support the requirement of determining whether or not a military service application was complete required a 250-page document.

Storing all of the validation rules in their own repository makes invoking these rules simpler, either within an application at runtime, or validating an entire set of records at the same time in batch mode.

Summary

The four types of rules described above make up the total BRIM® business rules application development environment. Virtually every rule required in a system can be placed into one of these four areas.

Some amount of hand coding outside of rule specification is still needed. As mentioned earlier, the BRIM® repository does not explicitly handle the screen layout of objects. Although BRIM® generates default applications, it is expected that the final layout, fonts, colors and other visual aspects of the user interface will require modification by hand.

Although it is possible to support them in the repository, some rules are better handled using some amount of hand-coding. For example, the 300,000 lines of code generated from the repository mentioned earlier, there were approximately 2,000 lines of hand-written code referenced by the rules in the repository.

It may never be possible or even desirable to completely specify 100% of a system with no hand coding at all. However, the amount of hand coding for a large, complex system can be reduced to approximately 1% using the repository-based approach described here. Systems created this way are much easier to debug and maintain over time.