Andrew Trenk
COMS 4115
Summer 2006

Diddle

A Data Definition Language for Clinical Trials Electronic Forms Systems

Introduction

A fundamental problem that is pervasive in many forms of computer applications is the need to define metadata for a set of data used by a program. Metadata is necessary in many situations, e.g. to specify constraints for the allowable values in a field, or to generate the proper SQL for the database table that will store the data.

There are many ways to define metadata, including methods such as manually adding extra validation code to a program, or by specifying rules in a simple text file. However, these solutions do not scale very well, as the specifications can easily become unmanageable as the dataset gets larger, and text files can become very difficult to parse. Additionally, these solutions also make it almost impossible for a non-programmer who is knowledgeable about the rules of the application to contribute to the specification of the metadata without the assistance of a programmer.

The goal of this language is for anyone to be able to write simple, easy-to-read scripts which define variables used in a program, and all their constraints. These scripts will then be converted into language that rest of the application is written in, merged with the application code, and be called whenever any collected data needs to be validated.

Additionally, these scripts will be able to generate the proper SQL code to create the database tables that will store the data.

Clinical Trials Data

While a metadata conversion language is useful in many applications, it is almost impossible to create a language with a completely generalized syntax that would be able to specify metadata for any type application. Thus, this language will focus solely on the specification of metadata for clinical trials systems. It would also be useful for other types of applications that have a similar data structure and flow to these systems.

Clinical trials systems mostly follow a similar structure. They each contain a number of forms that must be filled out over a period of time. Constraints for creating a form include the number of forms that may be filled out for each form type, and the time period that must elapse before another form may be created. A form contains fields that must be filled in by a user. Each field is a type of data with different constraints, e.g. a free-form text field, a date, or a checkbox. Fields may also have additional constraints, e.g. it may only be allowed to be changed if another field has a specific value.

This language will enable the constraints for both forms and data fields to be easily specified. Once the constraints are specified, they can be converted into actual code that can be used by the program for validation. Since the programs that use these validation routines will need data in a specific format, the code generated by this language must conform to a single format that can be used by the program. This language will generate

Java code that will be able to be integrated into a specific clinical trials system written in Java.

Examples

The first step to writing a script is to define the forms, along with the data fields that belong to the form (also known as the data dictionary). A sample definition of a form is as follows –

```
//Define form named Form01 with name "Registration"
Form01 'Registration'
{
     //Define the data fields
     //List the data type and the field name.
     //List additional required metadata for certain
     //form types, e.g. the range of acceptable values
     //for a choose1 data type.
     //Additional metadata can also be specified,
     //e.g. "required" for required fields.

     date FORM_DATE;
     string NAME required
     date DATE_OF_BIRTH
     choose1 GENDER ['M','F']

     //Additional constraints that the fields must conform
     //to can be specified here. An error message is
     //specified for fields that have an invalid value.

     validate()
     {
           if(DATE_OF_BIRTH.years()<18)
                 error('Patient must be 18 or older')

           if(FORM_DATE.futureDated())
                 error('Form date cannot be future dated')
     }

}
```

After a form is defined, its creation constraints can be specified. Some simple constraints to specify which forms can be created and when is as follows –

```
require 1 Form02, Form03 at Form01.FORM_DATE
date nextDate=Form01.FORM_DATE+1 weeks
allow 1 or more Form04, Form05 at nextDate
```

The first line specifies that Form02 and Form03 must both be created on the day that is specified in the FORM_DATE field in Form01. Line two creates a date variable that contains the value of a date one week after FORM_DATE. Line three allows any number of copies of Form04 or Form05 to be created at this date.

To help specify more complex constraints for form creation dates, a data type specifying a date interval is available. This allows a form to be created in a range of dates. An example declaration of an interval, which specifies a range of dates within four days in either direction of FORM_DATE is as follows –

```
interval timePeriod=+/- 4 days of Form01.FORM_DATE
```

A constraint can now be specified using this interval –

```
require 3 Form06 at timePeriod
```

It is also possible to create arrays –

```
interval dates[10]=+/- 2 weeks of nextDate per month
```

This will create an array of date intervals, the first within +/- two weeks of nextDate, and each of the following a month later than the previous. This would allow the following constraint to be specified –

```
for(date in dates)
     allow 3 Form07 at date
```

Another possibility to utilize arrays is as follows, which would specify that these forms would be required in each of the first three months –

```
require 1 Form08, Form09 at dates[0],dates[1],dates[2]
```

Additional constraints can also be specified with conditionals –

```
if(Form01.GENDER='M' or Form01.DATE_OF_BIRTH.years()>50)
     require 1 Form08 at timePeriod
```

Summary

With the proper utilization of this language, the process of specifying metadata for a clinical trials project should become vastly simpler. Additionally, it will give anyone – including people with no prior programming experience – the power to create these specifications.