Visual RAD Framework: An Engine Based RAD Tool

Victor Utomo

Program Studi Teknik Informatika STMIK PROVISI, Semarang
victor.utomo@gmail.com

Abstract

In the information era, the need of information system is enormous. Almost in every line of business, information system is in need. In order to fulfill the need, developer must be able to deliver information system fast. VRAD Framework is proposed as RAD tool which works as an engine. In this article, a visual user interface generator and a CRUD mechanism generator are proposed. Both generators generate the complete application at runtime. Currently, the framework targets Windows Form as user interface and Microsoft Access as database.

Keywords: RAD, RAD tool, engine, VRAD

1. Introduction

In the information era, the need of information system is enormous. Almost in every line of business, e.g. university, hospital, aviation, information system is in need. In order to fulfill the need, developer must be able to deliver information system fast. One of the methodologies that designated to deliver application in fast manner is rapid application development (RAD).

RAD platform currently is divided into two models. One is the engine model, in this mode, user only need to pass the parameters to engine, the engine is responsible for querying the database and processing data, and then show up in various ways. In this model, developers are not required to write code, nor have the source code. Another is source code model, this model define the operational modules through a desktop designer, and generate the framework source code. User can write and modify the code, implement business logic based on the generate code (Mao and Zhu).

This paper proposes Visual Rapid Application Development (VRAD) Framework that mixed both models. Developer creates a class, adding some objects and adding business logic. While business logic can be modified / added (source code model), complete application is generated automatically by the framework (engine model). This approach is different, e.g. from the one created by Teduh Dirgahayu that use modified SQL statement to generate code (Dirgahayu, 2011 and 2012).

VRAD Framework plans to target full application framework and visual designer. It is designed to work based on .NET 4 Framework with various platform that supported by .NET like ASP.NET, Windows Form, Windows Presentation Foundation (WPF) (Christian Nagel, 2011). However, this paper limits only in Windows Form generator using code-based interface.

The rest of the paper is structured as follows. Chapter 2 reviews the concept of Rapid Application Development. Chapter 3 shows the design and composition of VRAD. Chapter 4 gives an example on how to create a form using VRAD. Chapter 5 concludes the paper, summarizing the proposed solution and identifies future works.

2. Rapid Application Development

Rapid application development (RAD) appears to have first become topical with the publication of a text by James Martin with the same title (Martin, 1992). Martin defines the key objectives of RAD as: high quality systems, fast development and delivery and low costs. These objectives can be summed up in one sentence: the commercial need to deliver working business applications in shorter timescales and for less investment.

The following appear to be the common components of RAD approaches discussed in the literature:

1. Joint application design (JAD). RAD seems to be characterized by development teams of typically four to eight persons. Such teams are made up of both developers and users who are empowered to make design decisions.
2. Rapidity of development. RAD projects seem to be typically of relatively small-scale and of short duration. Also, two to six months is frequently discussed as being a normal work project length.
3. Clean rooms. JAD workshops are usually expected to take place away from the business and developer environments in 'clean' rooms – that is, places free from everyday work interruptions.
4. Time boxing. Project control in RAD is seen to involve scoping the project by prioritizing development and defining delivery deadlines or 'time boxes'. If project start to slip, the emphasis in RAD project is on reducing the requirements to fit the time box, not increasing the deadline.
5. Incremental prototyping. RAD is frequently discussed in terms of incremental prototyping and phased deliverables. Prototyping is essentially the process of building a system in an iterative way. RAD compresses the step-by-step development of conventional methods into an iterative process. The RAD approach thus includes developing and refining the data models, process models, and prototype in parallel using an iterative process. User requirements are refined, a solution is designed, the solution is prototyped, the prototype is reviewed, user input is provided, and the process begins again (CASEMaker Totem).

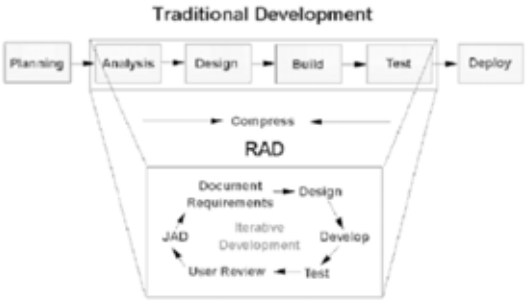

Figure 1. Iterative development in RAD

6. Rapid development tools. It is not surprising to find that modern approaches to RAD demand good support from tools for rapid developmental change. This normally means some combination of fourth generation languages (4 GLs), graphical user interface (GUI) builders, database management systems (DBMS) and computer-aided software engineering (CASE) tools.
7. High interactive – low complexity projects. Most RAD projects seem to be conducted on applications that are highly interactive, have a clearly defined user group and are not computationally complex.
8. Types of RAD projects. There generally appear to be two types of RAD projects: the intensive and the phased RAD project.

This paper puts emphasis on the rapidity of development, incremental prototyping, and rapid development tools.

The following are the advantages and disadvantages of Rapid Application Development:

1. Increase speed. As the name suggests, Rapid Application Development's primary advantage lies on application's increased development speed and decrease time to delivery.
2. Increased quality. Increased quality is a primary focus of the Rapid Application Development methodology, but the term has a different meaning than is traditionally associated with Custom Application Development. Prior to RAD, and perhaps more intuitively, quality in development was both the degree to which an application conforms to specifications and a lack of defects once the application delivered.

According to RAD, quality is defined as both the degree to which a delivered meets the needs of users as well as the degree to which a delivered system has low maintenance costs. Rapid Application Development attempts to deliver on quality through the heavy involving of users in the analysis and particularly the design stages.

3. Reduced scalability. Because RAD focuses on development of a prototype that is iteratively developed into a full system, the delivered solution may lack the scalability of a solution that was designed as a full application from the start.
4. Reduced features. Due to time boxing, where features are pushed off to later versions in favor of delivering an application in a short time frame, RAD may produce applications that are less full featured than traditionally developed applications. This concern should be addressed as soon as possible through a clear communication with the client as to what will be delivered and when (Core Partner).

3. Visual Rapid Application Development

3.1 Overview

While the framework is planned as a full application framework, this paper limits the discussion on the basic CRUD (Create, Retrieve, Update, and Delete) and UI generator. The proposed framework result is a Windows Form. The result form has two views, first view in the form of list and the second view in the form of detail or entity. For the database, using ADO.NET, VRAD supports Microsoft Access.

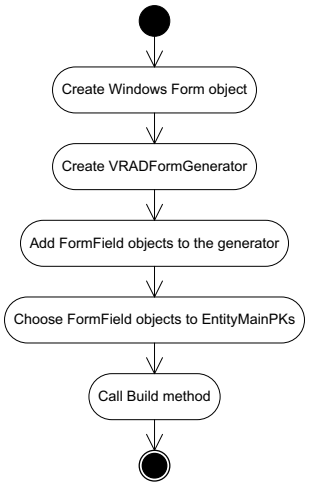

Figure 2. Activities in VRADFormGenerator

Currently the framework consists of six activities which can be seen at figure 2. First, user creates a Windows Form object that acts as the container for all components that generated by VRAD Framework. Second, user creates `RADFormGenerator` object. The object acts as the main engine that generates CRUD and UI. On its initialization, the following parameters passed into the generator:

1. The form object that created at first step.
2. Table name of the underlying table in database for the form.
3. Connection string.

Third step, user adds `FormField` objects to `RADFormGenerator` object. These `FormField` objects determine the components to add to the form. `FormField` object also determines behavior of the component. Fourth step, user adds primary key(s). Primary key(s) is composed from `FormField` objects. Fifth step, user calls `Build` method from `RADFormGenerator` that will run the actual codes that generate the user interface in the form. Sixth step, user may show the already generated form.

3.2 Properties in VRADFormGenerator

There are three important properties in `VRADFormGenerator` class, `MasterTable`, `FormFields` and `EntityMainPKs`. Figure 3 show the class diagram of `VRADFormGenerator`.

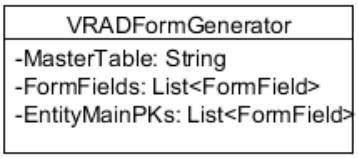

Figure 3. Class Diagram of `VRADFormGenerator`

`MasterTable` is used for indentifying which table is bound to current form. `FormFields` is a generic list of `FormField` object which defines overall behavior of the form. `EntityMainPKs`, like `FormField`, is a generic list of `FormField` object. `EntityMainPKs`' item is any `FormField` object in `FormFields`. `EntityMainPKs` defines which field is the primary key of the table.

`FormField` behaves like a table field with some additions, mainly for visual purpose. The important properties of the `FormField` class are `FieldName`, `FieldType` and `ControlType`. `FieldName` shows the corresponding field of the control in form. `FieldType` defines data type of the corresponding field of the control in form. `ControlType` defines which control is used in the form. The mandatory properties of the `FormField`, like database field, are `FieldName` and `FieldType`. By default, a `ControlType` may be defined by `FieldType`. However, `FormField` gives overload constructor which allows developer to define `ControlType` that different from the default `ControlType`.

3.3 Visual Generator

Visual components are generated under the call of Build method. The activities in Visual Generator are shown in figure 4. Before generating any component, the uniqueness of `FieldName` property at each `FormField` object is evaluated. This evaluation is needed to make sure that no more than one control can supply the value for a particular field.

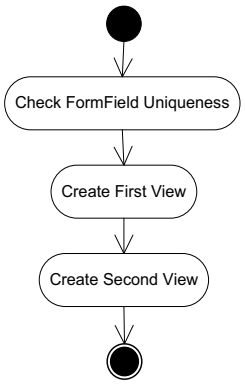

Figure 4. Visual Generator Activities

After the uniqueness evaluation, VRAD generates first view. The first view contains a list of data in tabular manner and buttons, providing add, edit and delete triggering functionality. The list contains columns based on `FormField` that meet at least one of the following criteria, the `FormField`'s `ShowOnList` property value is true or the `FormField` object is listed on `EntityMainPKs`. The contained columns' visibility depends on `FormField` object's `ShowOnList` property value. Add or edit button switch the current view into second view.

The second view contains single data in form style. Each `FormField` object is generated into a label, determined by the value of property `FieldText`, and a control. The type of generated control is determined by the value of `FieldType` property. The control bound to database field based on the value of `FieldName` property. Control type is important since it is determined the way data set into control and the way data get from control. This view contains two buttons for triggering save and close mechanism. The framework automatically detects whether save button triggers insert data or update data. Close button switch the current view into first view.

3.4 CRUD Generator

CRUD generator happens in five methods that are `FillEntity`, `FillListPanelList`, `InsertData`, `UpdateData` and `DeleteData`. The generator creates an SQL query string that serves the CRUD purpose. The SQL query is based on ANSI SQL 92 standard (www.contrib.andrew.cmu.edu).

In general, the generator takes four steps, though at each method specific step is required. Figure 5 shows the activities.

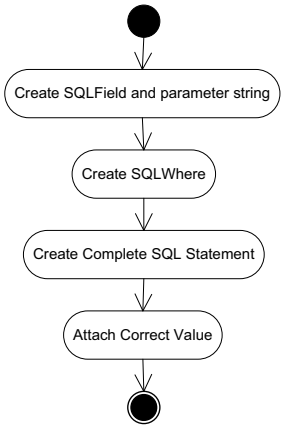

Figure 5. CRUD Generator Activities

First step, the generator scan all the objects in `FormFields` and create a text, called `SQLField`, that composed by the value of the `FieldName` property of each objects that meets the conditions. The conditions are different, depends on the corresponding SQL statement to be made. This step also creates the parameter string that matched the `SQLField` string. Second step is basically almost the same as the first step with a different condition. The second step requires a condition that the object must be contained by `EntityMainPKs`. The result of the second step, called `SQLWhere`, acts as the WHERE part of SQL statement. Third step combines the `SQLField` and `SQLWhere` into a complete SQL statement. At fourth step, the generator scans all the controls and attaches the corresponding value for each parameter in the SQL statement, identifying the control by its name.

In `FillEntity` method, at first step the SQLField contains asterisk (*) only. The generator creates SELECT statement at this method. Figure 6 shows the SELECT SQL statement.

```
SELECT * FROM [MasterTable]
WHERE [SQLWHERE]
```

Figure 6. SQL Statement in `FillEntity` Method

In `FillListPanelList` method, the generator also creates SELECT statement. In this method, second activity is skipped. Figure 7 shows the SELECT SQL statement.

```
SELECT [SQLField] FROM [MasterTable]
```

Figure 7. SQL Statement in `FillListPanelList` Method

At `InsertData` method, the generator creates an INSERT statement. At this method, first step creates two separate strings, since INSERT statement separates field list and parameter list, while the second step is not needed. Figure 8 shows the INSERT SQL statement.

```
INSERT INTO [MasterTable]([SQLField])
VALUES ([SQLField])
```

Figure 8. SQL Statement in `InsertData` Method

`UpdateData` method takes all the steps to create an UPDATE statement. Figure 9 shows the INSERT SQL statement.

```
UPDATE [MasterTable] SET [SQLField] WHERE [SQLWhere]
```

Figure 9. SQL Statement in `UpdateData` Method

`DeleteData` method skips the first step and creates a DELETE statement as the result. Figure 10 shows the INSERT SQL statement.

```
DELETE FROM [MasterTable] WHERE [SQLWhere]
```

Figure 10. SQL Statement in `DeleteData` Method

4. Example of Implementation and Result

VRAD Framework usage starts with adding VRAD Framework library. Since the library is built using .NET Framework 4.0, the application requires

at least .NET Framework 4.0. In this example, the application built using Visual Studio 2010.

A Microsoft Access Database prepared with an Employees table that has the following field, `EmployeeID`, `EmployeeName`, and `BirthDate`.

Create a Windows `Form` object and a `VRADFormGenerator` object. Supply the necessary data, like connection string and master table. Add the wanted `FormField` objects to `VRADFormGenerator` and specify which `FormField` that became the primary key. Figure 11 shows the code example.

```
      System.Windows.Forms.Form anyForm = new
Form();
      VKFramework01.VRADFormGenerator gen =
       VKFramework01.VRADFormGenerator.
       CreateForm(anyForm,
       "Employees",
       @"Provider=Microsoft.Jet.OLEDB.4.0;
       Data Source=C: \Database1.mdb;
       Persist Security Info=True");
      gen.AddFormField(
       new
VKFramework01.FormField("EmployeeID"));
      gen.AddFormField(
       new
VKFramework01.FormField("EmployeeName",
       true));
      gen.AddFormField(
       new VKFramework01.FormField("BirthDate",
       "BirthDate",

VKFramework01.VKConstant.DataTypeEnum.Date,

VKFramework01.VKConstant.ControlTypeEnum.
       DateTimePicker));
      gen.AddEntityMainPK("EmployeeID");
      gen.Build();
      anyForm.ShowDialog();
```

Figure 11. Code Example of VRAD Framework Usage

Let the `VRADFormGenerator` build the form and show the form. Figure 12 shows first view of the result form and figure 13 shows its second view.

Figure 12. First View of Example Form

Figure 13. Second View of Example Form

5. Conclusion and Further Work

This paper proposed a RAD tool in the form of engine, named VRAD. The engine generates the whole functionality at runtime. Developer needs to supply the engine with the correct parameters by code.

In current work, VRAD has the ability to generate visual user interface under Windows Form library and CRUD mechanism to Microsoft Access database.

To create the planned framework, future works is required i.e. application loader, searching mechanism, supports for option control, ability to add custom business logic and visual UI for the tool itself. The framework is also planned to support other visual library such Windows Presentation Foundation (WPF) and ASP.NET Web Form. The CRUD mechanism also needed to enhance to support databases other than Microsoft Access in a more generic manner.

Reference:

ANSI SQL 1992, www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt, accessed February 20th 2011

Rapid Application Development, Core Partner, page 2

What is Rapid Application Development?, CASEMaker Totem, page 5

Dirgahayu, Teduh, 2011, *A Framework For Rapid Development of OLTP Information Systems: Transformation of SQL Statements to Three-Tire Web Applications*, Proceeding The 1st International Conference on Information Systems for Business Competitiveness, Semarang, Indonesia

Dirgahayu, Teduh, 2012, *Pembuatan Form Masukan Aplikasi Web Secara Otomatis dari Perintah SQL Insert Termodifikasi*, KNSI

Mao, Huaqing; Zhu, Li, *Template Based Framework for Rapid Application Development Platform*

Nagel, Christian; Evjen, Bill; Glynn, Jay; Watson, Karli; Skinner, Morgan, 2010, Professional C# 4 and .NET 4, Indianapolis, Wiley Publishing, Inc.