Author: J.F.Benckhuijsen (jfbenck@users.sourceforge.net)
Version: 1.0

Roadmap and current status

Introduction

The development of the Modelbuilder application has started on Janurari 2001. After a half year of work version 0.1.0 was released. This version supported these features:

- Initial design of the whole MDK Application;
- Kiss System Initial design;
- Models: Structure sentences, Information Quadrant, Attribute Models, Reference Objects;
- Initial generation support (prototype);

In september 2001 work was continued on the Modelbuilder. A plan was created to extend the functionality of the Modelbuilder application to allow full support for the basic features the Modelbuilder requieres to generate an webapplication. These features include:

- Loading and saving of projects;
- Object Interaction Model;
- Kiss Model (Sequence Rules);
- Advanced Consistency checks infrastructure;
- Redesigned generation support and implementation (working application template);
- Initial support for the layout engine
-

This new version , 0.2.0 allows basic generation of an webapplication. The modelbuilder application can however be extended in various ways. Some of these features are needed improve the modelling process.

The development of the MDK Application will probably continue. Because there are so many ways in which we can extend the modelbuilder, we'll have to construct a roadmap to organise the flow of the project. In the past months we've though about the ways the modelbuilder could be extended. First i'll review all of these options, discuss their purpose, advantages and disadvantages. Then i'll discuss our choice of the subset of options which can be implemented in the next half year.

Proposed features

There are a number of features which can be implemented to complete the whole modelbuilder application. Some of these are just fancy extras, others belong to the core functionality of the program.

1. Make Models aware of eachother

Feature type: improvement

Library: Kiss Models / Kiss Model Components

Description: As described in the Kiss Models Library the document, each of the models *should* communicate with the Kiss System to update the system's status whenever the model is changed or to update it's own status whenever the system has changed. The first part of the communication is implemented in all models. The second part of the communication isn't always implemented however. This means the models aren't updated when any of the other models active is changed. Off course it's confusing for the user to see two models which aren't in sync. These models have to be rewritten or adapted to include this feature:

- Structure Sentences Model
- Information Quadrant
- Attribute Model
- Reference Object Model

Problems: Implementing these features into the models will require extensize knowledge of the application, especially the design of the Kiss System library and the Kiss Models. For new developers this could prove to difficult because they aren't familiar with the application.

2. Hierarchy model

Feature type: extension

Library: Kiss Models

Description: The hierarchy model is just another model of the Kiss Modelling Language. The ultimate goal would be to support all models the Kiss offers.

Problems: This feature requieres some basic knowledge of the Kiss System and the Modelviewer component. No new features have to be added to the Kiss System, so this project can be done by one person without disturbing any other part of the process.

3. Classes in the Object Interaction Model

Feature type: extension

Library: Kiss System

Description: In KISS it's possible to set classes for objects. The exact meaning of these classes has to be investigated. Also implications for the generation process are unknown. Besides that models have to be adapted to support these classes.

Problems: Change(extension) of the Kiss System is requiered. Besides that, some models may have to be adjusted to support classes. Also there may be some changes in the generator.

4. Activity support in the Kiss System

Feature type: extension

Library: Kiss System

Description: Activities are a special type of actions within a system. The current version of the Kiss System doesn't really support Activities. Also it's not yet clear what differences exist between the generation of actions or activities.

Problems: Basic knowledge of the system is requiered. Also good understanding of KISS is requiered. An oppurtunty is the help offered by some (former) employees of Kiss BV to clarify the concepts of Kiss.

5. Type library support for reference objects and quantities

Feature type: extension

Library: Kiss System

Description: Often an company (or another user) creates a set of application to support it's business. In most cases these applications share are common set of datatypes, like customer information or branch specific types. Currently we have to rewrite all these types every time we create an application. We would like to offer the user a possibility to store a set of types in a separate library, which can be saved for use in other projects.

Problems: The Kiss System will be changed because of this, however this will probably only result in an extension, not in a change.

6. Full consistency check support

Feature type: improvement

Library: Kiss Rules

Description: The Kiss Rules library consists out of two types of checks: the first are the design time consistency checks, which are implemented for about 75%. They are constantly active, but don't guarantee a system which is completely consistent. The other part are the before-generation consistency checks, they are applied before generation and guarantee a fully consistent system. They are not implemented at the moment.

Problems: Extensive knowledge about KISS is requiered to ensure a 100% consistent model.

7. Improved implementation of the models GUI

Feature type: improvement

Library: Kiss Model Components

Description: Currently all models are hardcoded into the user interface. We would like to create a more modular design in which models can be easily added without having to change the UI (much).

Problems: This would require a rewrite of some models and their User Interfaces.

8. Improved error handling in the models, generator and file IO

Feature type: improvement

Library: All

Description:

Problems:

9. Advanced layout engine for the models

Feature type: improvement

Library: Modelviewer

Description: The current Modelviewer component currently supports a very basic implementation of a layout engine. The responsibility of this engine is to calculate the optimal layout of all the elements which have to be drawn. The application will probably be easier to use when the user doesn't have to layout all shapes manually.

Problems: Although this offers a nice view to the user, it's may prove a rather difficult problem to calculate the optimal (or even sub-optimal) layout. It could take a lot of time to create such an engine.

10. Decision tables for business rules

Feature type: extension

Library: All

Description: In every business certain rules apply, we may want to allow a user to store these business rules and generate code from these rules.

Problems: This will requiere investigation in the way rules can be stored. Currently it's unknown to us if KISS supports any of this.

11. Version management of projects

Feature type: extension

Library: Kiss System, Kiss Generator

Description: No application is ever ready after the first development session. It's also possible a application will change after a few months or years. We want to provide the user a tool to access it's old data.

Problems: There are various approaches to this problem. For example we might provide a migration tool, or allow different versions of an object to coexist. Also this may require some redesign of the Kiss System depending on the chosen approach.

12. Theme support for the generated user interface

Feature type: improvement

Library: Kiss Generator

Description: Every type of application has it's own style of User Interface. The type of user interface chosen depends on the type of application which will be generated. We would like to offer the user a selection of user interface types and generate the selected.

Problems: *none*

13. Simple reporting engine for the generator

Feature type: extension

Library: Kiss Generator

Description: The currently generated application lack any reporting. Data can be entered, actions can be preformed, but meaningfull reports can only be created by a user. We would like to offer the user a possibility to create it's own custom reports.

Problems: We would have to think of a way to save the reports themselves, also we'll have to come up with a way to store the reports within a project.

14. Report generation toolkit & runtime generation

Feature type: extension

Library: Kiss Generator

Description: Often reports will have to be created after the webapplication is generated and completed. To support this process we want to create a toolkit of (php)functions to support the developer in the creation of new custom reports. We may even want to support a runtime version of the report creator, although this may prove difficult with because the application is run on the web.

Problems: We'll have to create a reporting toolkit which is both flexible and powerfull and still easy-to-use.

15. Direct upload using FTP or FrontPage Extensions

Feature type: extension

Library: Kiss Generator

Description: Currently all generated files are saved on the local harddisc of the developer. It would be convient for the user to allow the application to be uploaded directly without having to use a separate program.

Problems: *none*

16. Support for additional databases(Oracle, MSSQL, Sybase, etc.) and languages (ASP(.Net), C#, J2EE, Perl, Python, etc.)

Feature type: extension

Library: Kiss Generator

Description: Currently the Kiss Generator produces an webapplication which uses Php and Mysql to run. An extension could be to support other databases or even other languages, so the user can select his or her favourite language to code or extend the generated application.

Problems: This requieres great knowledge about the way webapplications work, knowledge about the database or lanuage to support and some basic knowledge of the Kiss System, from which the information about the application is retrieved. An advantage is that the current Php/Mysql implementation can be thought of as an example implementation.

18. Porting the application to Linux

Feature type: extension

Library: Kiss GUI / Kiss Generator

Description: Linux is a new emerging platform. Support of this platform offers a fairly large group of developers and users which could contribute to the application.

Problems: Extensive knowledge about porting applications and how to program in linux is needed to port all parts of the Modelbuilder to linux. Large parts of the program won't be any problem, because the are coded in standard C++. Most work will be to port the GUI of the application to linux.

Features proposed for version 0.3.0

These are the features we selected to be implemented in the next half year. Our goal was when we selected these features was to create a basic set of functionality to call our application mature. After these implementations we may consider the product to be in a 1.0 stage, meaning we have can release our first complete application. This doesn't mean the application isn't complete or working properly.

- Making models aware of eachother (1)
- Hierarchy model (2)
- Classes in the object interaction model (3)
- Full consistency check support (6)
- Improved implmentation of the models gui (7)
- Advanced layout engine support (9)

The features 1 and 7 are two parts of the same process, which is to complete the design as described in the developers documentation of the Kiss Models Library. Off course it's natural to implement both features, because the effects of 1 aren't visible without 7 and 7 depends on 1.

Feature number 2 is a simple extension, which provides additional information about a system, without any problems or depencies on any other running projects.

To relieve the user of the resposibility of checking the whole system for consistency we'll have to implement feature 6. Checking a system by hand can be quite hard in large systems, and one could easily forget certain checks.

Besides that, to make the user interface more attractive for the user we've selected feature number 9 to improve the display of the various models. This project can also run separate from the rest of the projects, so managing the project will become easier.