Model refactoring within a Sequencer

TOMAŽ KOS[1], TOMAŽ KOSAR[2], JURE KNEZ[1], MARJAN MERNIK[2]
[1]DEWESoft d.o.o.
Gabersko 11a, 1420 Trbovlje
SLOVENIA
{tomaz.kos, jure.knez}@dewesoft.si http://www.dewesoft.si
[2]Faculty of Electrical Engineering and Computer Science, University of Maribor
Smetanova ulica 17, 2000 Maribor
SLOVENIA
{tomaz.kosar, marjan.mernik}@uni-mb.si

Abstract: - The development of measurement procedures within the DEWESoft measurement system is done in a visual manner using a domain-specific modelling language Sequencer. Although, engineers use just the abstractions from the measurement domain without any junction with the programming code, additional tools that add to their comprehension are more than welcome. Therefore, a tool was developed for the refactoring of domain-specific models. This tool enables model restructuring, without any affect on the measurement's behaviour. This paper introduces some of the refactoring methods supported in the Sequencer, as well as the refactoring tool itself. Moreover, some advantages and disadvantages are provided and experience is given of using the refactoring tool.

Key-Words: - domain-specific modelling language, model refactoring, model testing, measurement system

1 Introduction

Refactoring is a process of improving a programming code's quality without changing its behaviour [1]. This technique was first proposed by Opdyke as a methodology for restructuring programs [2]. Refactoring can be an efficient and effective way of improving the design of a software code that can be done manually or by a suitable tool. Supporters of extreme programming classify code refactoring as part of the software development cycle [3].

In addition to refactoring the programming code, there is also the term model refactoring [4]. Such a methodology transforms the model rather than the programming code. Similarly to refactoring the program code, transformation of the model does not change the behaviour of the program/model.

Tools that support model refactoring can be found in literature and in practice [5]. Biermann et al. presents a tool, based on an existing C-SAW tool, which uses a generic transformation of models [6]. The disadvantages of these tools are that they are inflexible and difficult to integrate within an existing product.

A model refactoring at a higher-level of abstraction is new within the research area. This refactoring is becoming a desirable feature for improving models using transformations based on domain-specific modelling languages (DSMLs) [7]. This paper presents methods for model refactoring in the field of measurement systems, as well as the refactoring tool integrated within the DEWESoft package [8].

The paper is divided as follows. Section 2 introduces the background of the presented tool for creating measurement systems. Section 3 provides details about domain-specific model refactoring. Section 4 presents those refactoring methods incorporated within the Sequencer. The testing of model refactoring is presented in Section 5. A discussion follows in Section 6 and finally, Section 7 summarizes the main features of model refactoring for measurement systems, and provides the concluding remarks.

2 DEWESoft measurement package

The DEWESoft software package (current version 7.0.4) is a product from the company with the same name. Software enables data acquisition, data processing, data analyzing, and data storage, as used in the automotive, aerospace, construction, electronics, and even in the space industry. It represents an intermediate stage between those general measurement systems (e.g. oscilloscope, spectrum analyzer) that are usually very difficult for the user, and those measurement systems specially adapted for specific devices. The DEWESoft software package can capture different data, such as analogue and digital signals, CAN data, GPS data, and many more. This tool also supports a wide-range of graphical displays (Figure 1) and exports data in various formats such as text, Excel, FlexPro, Matlab, etc.

Despite these features that allow easier usage, the measurements are very demanding and require precise definitions of the measurement procedure. In order to improve flexibility, productivity, and also to only stress the important details of the measurement, DEWESoft software enables the development of measurement procedures using a modelling tool.

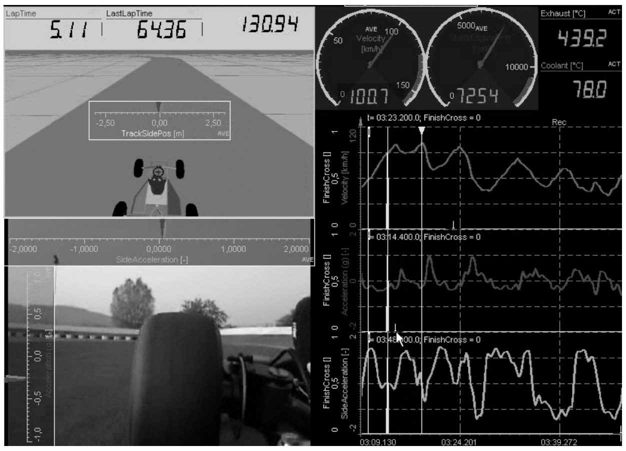


Fig. 1: DEWESoft measurement software

2.1 Modelling Tool

Modelling tool is used for developing measurement procedures by using a DSML called a Sequencer [9]. Basically, DSMLs are very suitable for the construction of complex systems with high-level constructs [10]. These languages have mainly been developed for shifting programming from programming engineers to the domain experts [11]. Domain experts, who have the skill necessary for the problem domain but no formal computer knowledge, can write their own domain-specific models in order to solve a specific need within their domains. In the presented case, the domain expert can very easily, without any knowledge of programming, develop measurement procedures that can be further executed on a measurement system.

A domain expert can create a measurement procedure using predefined modelling blocks which represent actions on the measurement system. Execution starts with the initial modelling block and continues to the next modelling block connected to the initial one. Modelling blocks also contain local and global variables (that represent channels). Their purpose is to store specific values within the measurements. In order to have better control over the model's extent, a custom block has been introduced that embodies several modelling blocks within a single one.

Figure 2 shows the modelling environment. Domain-specific modelling blocks can be found on the left-side of the tool. On the right-side there is a working panel where domain experts can model a specific measurement procedure.

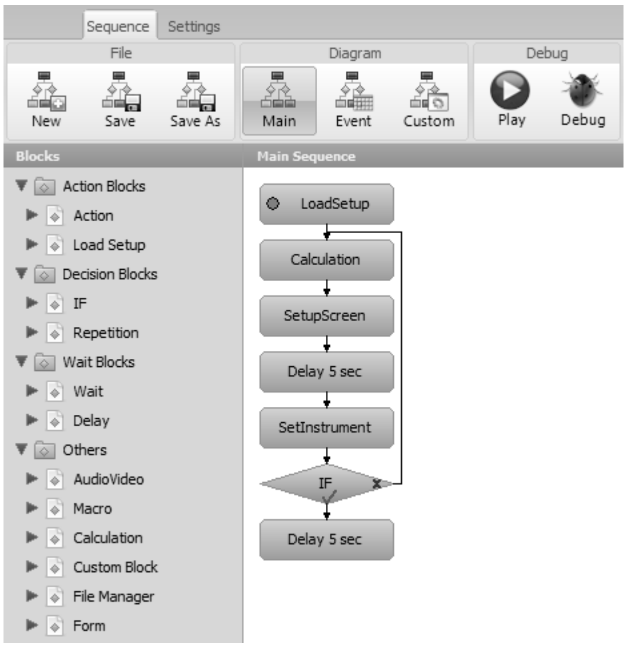


Fig. 2: Measurement procedure in the Sequencer

3 Domain-Specific Model Refactoring

Model refactoring improves the quality of the model in the fashion of small sequential steps in which each step changes the internal structure. However, the external behaviour of the measurement procedure should not be changed during these steps.

3.1 Procedure of refactoring

With the help of the refactoring tool in the Sequencer, the user can change parts of the model as follows:

- Find the place in the model where refactoring is needed.
- If the test cases exist, run them and record the results if desired.
- Select the refactoring method.
- Follow the steps of model refactoring.
- After refactoring, run the test cases again and compare the results with the results before refactoring.
- If necessary change the interface of the testing model.
- After that, re-run the test cases.
- If test results do not report any errors and the results are the same as before the transformation, model refactoring is completed.

As stated above, the refactoring starts at those places that »call for change«. An example of model's critical part is represented by those duplicated parts of a model where identical functionality is implemented in different positions. Another example of a critical part is represented by those long and unclear models that can be divided into smaller parts. The final example of a critical part is represented by incorrect denominations in the model (modelling blocks, parameters, etc.) as a result of poor knowledge from the problem domain.

4 Refactoring methods

DEWESoft has implemented several methods by which end-users can use the modelling tool more easily and efficiently. Currently, the Sequencer supports the following refactoring methods:

- extract custom block,
- rename custom block,
- add parameter,
- delete parameter,
- rename parameter,
- rename local variables,
- rename device name.

A few of them are described in the following subsections.

4.1 “Extract Custom Block” Method

The "Extract Method" is one of the more frequently used methods in refactoring [1]. In the Sequencer tool it is called the "Extract Custom Block". It is a useful aid when the model is becoming too large. For this reason, part of the model, more specifically some modelling blocks, are combined into a (sub)model.

For various reasons it is recommended that a new model should be short and logical. Firstly, it increases the reusability when the model is completely independent. Secondly, it raises the level of abstraction, and consequently, an easier understanding of the extracted blocks. Moreover, concentration is also aimed at naming a new model where the correlation between the name and the model’s semantics should be expressed unambiguously.

Figure 3 shows an example of model refactoring using the “Extract Custom Block”. Firstly, the modelling blocks (on the left-hand side of Figure 3) are selected, and then refactoring method is chosen. On the right-hand side there is the result – the redesigned model that adds a new abstract block containing modelling blocks.

The operation behind this method can be described in the following order. Firstly, a new nested model is created. Then, a name is given and the belonging properties are defined (e.g. model type, identifier). Next, the selected blocks are copied from the original model into a new one. Afterwards all the input and output connections of the selected block are searched to help determine the later input and output connections of the new block. When the search is finished, the selected blocks are removed from the original model and a new custom block is inserted in the former position of the removed blocks. Finally, connections are established with the custom block.

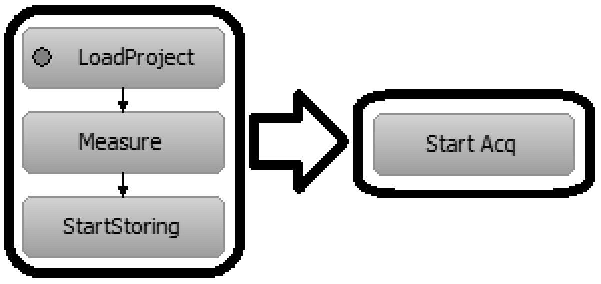


Fig. 3: An example of refactoring a DSML model

4.2 Rename Device Name

The Sequencer enables the communication and control of different hardware devices. These devices monitor and control various measuring instruments. Sometimes, the end-user needs to replace or modify an existing communication interface with a new one. In this case, the end-user needs to manually change all references to this device within the model. For this reason, a refactoring method has been developed that easily changes the name of the device and all the references within the model.

Figure 4 shows the basic concept of connectivity between the models and their hardware components. An individual device communicates with the hardware via different physical and data interfaces such as Ethernet, UART, CAN, etc. These interfaces are used in DSML models (sequences) for controlling different devices. The sequences refer to the device via an identification represented by a unique device name.

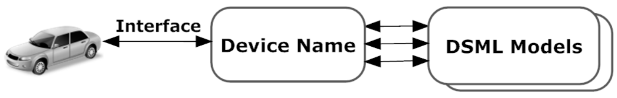


Fig. 4: The concept of connectivity models and hardware components

The basic principle of this refactoring method works with relatively simplicity. Firstly, the uniqueness of the device name is checked. Followed by a review of all models referred to as the (old) type devices. When a reference is found, the old name is replaced with the new device name. Finally, it is recommended that the user runs test cases over the modified model to check its behaviour.

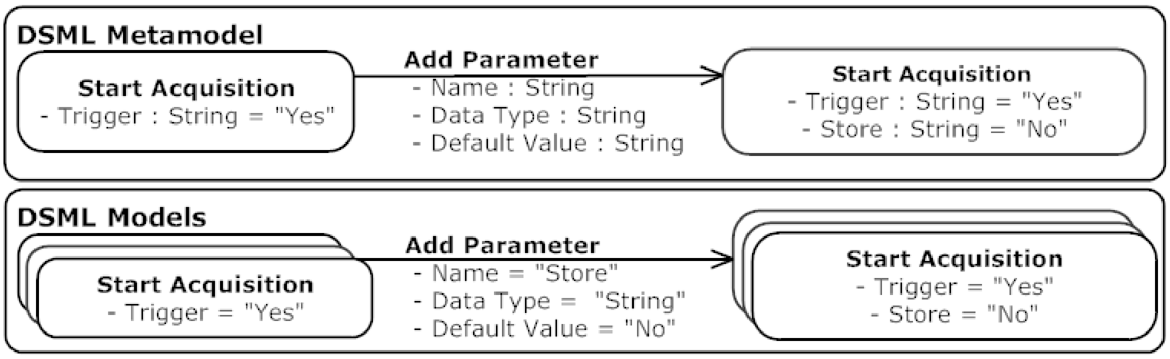


Fig. 5: Added parameters in the custom block

4.3 Add / Remove Parameters

A model can become unmanageable when it has a lot of modelling blocks. For this reason, the modelling tool allows for the construction of a custom block through which the modelling blocks can be combined into one and then refer to them in the remaining part of the model. These blocks can contain parameters for use within the model as local variables. Sometimes it is desirable to add or remove parameters to improve the design of the software. Since these operations may require some unnecessary effort, this tool allows for the effective option of adding or removing parameters in a simple way, without any major effort.

This refactoring method, the purpose of which is to add input parameters to the model, works in such a way that the name, data type, and default value of the new parameter are first defined. Then the uniqueness of the name is examined and if the verification is successful, a new parameter can be added to the custom block. Afterwards, searching for references in all the modelling blocks is done. When any reference is found, a new parameter is added to the modelling block, which then is set to the default value defined previously. At the end verification follows and the testing of the transformed model.

Figure 5 shows a practical example of adding parameters through model refactoring. On the left side is an old model which would be transformed with a new function Add Parameter. For custom block “StartAcquisition”, we added the new attribute with name “Store”, which data type is »String« and has a default value »No« which means that data acquisition does not begin by saving.

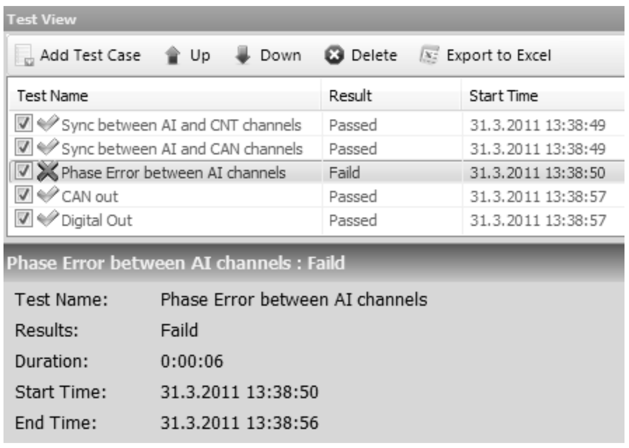
5 Model Testing

As mentioned before, it is necessary to check the results of the measurement system. Tests ensure the proper functioning of the measurement procedure, because the behaviour of the model should not have changed.

In general-purpose languages such tests are usually executed using unit testing [12]. This technique is based on the idea of testing small parts of a program before and after changes, and then comparing the results. A similar testing tool for testing the models has been developed within the Sequencer [13]. This tool provides an infrastructure for testing the measurement systems as well as the models. The test cases are modelled in the same way as the measurement procedure. The only difference is that within the Sequencer a new block for detecting a correct operation is incorporated within the model (e.g. “Assert” block).

If the test cases return unequal behaviour, it is necessary to change the new model. When the model has been changed, the user needs to re-run the test cases. This procedure should be repeated until the user obtains the same testing results as he/she had before refactoring.

Figure 6 shows an example of the results from a testing tool. The successful and failed test cases after refactoring can be observed in this figure. Moreover, an error message can be seen that can help to locate the error.



Test Name	Result	Start Time
☑ ✓ Sync between AI and CNT channels	Passed	31.3.2011 13:38:49
☑ ✓ Sync between AI and CAN channels	Passed	31.3.2011 13:38:49
☑ ✗ Phase Error between AI channels	Faild	31.3.2011 13:38:50
☑ ✓ CAN out	Passed	31.3.2011 13:38:57
☑ ✓ Digital Out	Passed	31.3.2011 13:38:57

Phase Error between AI channels : Faild

Test Name:	Phase Error between AI channels
Results:	Faild
Duration:	0:00:06
Start Time:	31.3.2011 13:38:50
End Time:	31.3.2011 13:38:56

Fig. 6: Unit testing for DSML models

6 Discussion

In practise, many software environments have automated refactoring support [14]. The usage of automated refactoring has been shown to be effective in many cases [15]. Even in the Sequencer's refactoring tool this is no exception. The difference is that the Sequencer tool uses refactoring within the specific domain, and these new methods have been implemented for this reason. Practise has shown that automated refactoring within the presented domain can be four times quicker than manual refactoring. Three different domain refactoring methods are analyzed in Table 1 using a new tool with manual refactoring. The efficiency was measured by the number of interactions (e.g. clicks, entries) for each method. In Table 1, it can be observed that the method "Extract custom block" using manual refactoring needs 15 interactions, whilst the automated refactoring tool needs only 4. Note that the N in Table 1 means the number of blocks that depend on the method within the model.

Table 1: Comparison of Automated / Manual Refactoring

Method	Manual Refactoring	Automated Refactoring
Extract custom block	15	4
Rename device name	3 + 3 * N	2
Add parameter	5 + 4 * N	5

The development of a refactoring tool and its supporting tools brings both advantages and disadvantages. At the beginning, it is necessary to invest in developing the tool. Most of the implementation time has been spent on integration of refactoring in the existing tool, whereas implementation of the transformation was not an issue. For example, the method "Extract custom block" has only 125 lines of code written in the Delphi programming language.

7 Conclusion

This paper presented model refactoring within a DSML Sequencer. The tool is included in the DEWESoft software used in industry, where measurements in the form of capturing and data processing are desirable. With this modelling tool, the user can quickly and easily refactor models and the model refactoring can be done in a visual way. This paper, also presented a few different types of model refactoring.

In the future we want to extend the functionality of the modelling tools to domain-specific language (textual language) in order to support users by using the same functionality.

Acknowledgements

This work was partly sponsored by the European Social Foundation.

References:

[1] Fowler M., Beck K., Brant J., Opdyke W., Roberts D., Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999.

[2] Opdyke W. F., Refactoring: A Program Restructuring Aid in Designing Object-Oriented Application Frameword, PhD Thesis, University of Illinois at Urbana-Champaign, 1992.

[3] Succi G., Marchesi M., Extreme Programming Examined, Addison-Wesley, 2001.

[4] Sunye G., Pollet D., LeTraon Y., Jezequel J.-M. Refactoring UML models. In Proc. Int'l Conf. UML 2001, Vol. 2185, pp. 134–138. Springer-Verlag, 2001.

[5] Biermann E., Ehrig K., Köhler C., Taentzer G., Weiss E., Graphical of In-Place Transformations in the Eclipse Modeling Framework, MoDELS'06, 2006, pp. 425-439.

[6] Zhang J., Lin Y., Gray J., Generic and Domain-Specific Model Refactoring using a Model Transformation Engine, Volume II of Research and Practice in Software Engineering, Springer, 2005, pp. 199-218.

[7] Kelly S., Tolvanen J.-P., Domain-Specific Modeling: Enabling Full Code Generation, Wiley-IEEE Computer Society Press, 2008.

[8] DEWESoft: http://www.dewesoft.si

[9] Kos T., Kosar T., Mernik M.: Development of Data Acquisition Systems by Using a Domain-Specific Modeling Language, Accepted for publication in Computers in Industry, 2011.

[10] Mernik M., Heering J., Sloane A., When and how to develop domain-specific languages, ACM Computing Surveys, Vol. 37, No. 4, 2005, pp. 316-344.

[11] Kos T., Kosar T., Knez J., Mernik M.: From DCOM interfaces to domain-specific modeling language: A case study on the Sequencer, Computer Science and Information Systems, Vol. 8, No. 2, 2011, pp. 361-378.

[12] DUnit: http://dunit.sourceforge.net

[13] Kos T., Kosar T., Mernik M., Knez J.: SeTT: Testing-tool for Measurement System DEWESoft, Recent Researches in Economics, WSEAS Press, 2011, pp. 111-115.

[14] Embarcadero Delphi: http://www.embarcadero.com/products/delphi

[15] Mealy E., Carrington D., Strooper P., Wyeth P.: Improving Usability of Software Refactoring Tools, in Proc. of ASWEC '07. IEEE, 2007.