Behavioral Reflection

Yashasvi Appilla Chakravarthi
Department of Software Engineering
The University Of Auckland
38 Princes Street, Auckland 1020, New Zealand
yapp001@ec.auckland.ac.nz

ABSTRACT

With a rapid advance in computer technology, user interfaces are one of the most important aspects in the development of software architecture today. As the complexity of the software products is increasing day by day, it is important that the users find the system easy to use and convenient. One of the new software mechanisms that evolved due to the demands on software development is a dynamic approach called reflection. This paper discusses how reflection can be used in the development of user interfaces and how the usability of software systems can be improved. Reflection is a mechanism that supports meta-computations in the program, which allow the program to adapt itself during run-time. This concept has been widely used in the programming side of the software development. However, due to the increasing demand of usable and easy to use systems, reflection is being applied in the world of user interfaces.

1 Introduction

Reflection is a software mechanism that supports computations of computations i.e. meta-computations in a program which allow the program to modify its structure or behavior during run-time. One of the applications of reflection in the software development is in the field of user interfaces. This mechanism can be used to provide feedback to the user based on the current behavior or state of the program. It allows the users to understand the software better and use it efficiently.

This paper discusses the applications of reflection in software development in the field of user interfaces. In Section 2, the relation between reflection and user interfaces is defined. This section also explains how reflection can be used in this area. Section 3 discusses the different types of reflection that exist and Section 4 explains how reflection is implemented in the software. Section 5 describes the different ways in which reflection can be used for extending user interfaces and Section 6 explains some examples of how reflection has been used. Finally, the paper describes some of the future work in Section 7 and concludes with Section 8.

2 Reflection in User Interfaces

Reflection allows a program to modify itself during run-time by using meta-computations or meta-data. This concept is used similarly in the field of HCI i.e. user interfaces, where the meta-data is used to identify the current state of the program and enable the user interface to modify the program. Thus, the concept of reflection has spread from the programming aspect of the software field to the area of user interfaces, with a slight modification to the mechanism.

3 Types of reflection

Based on the type of runtime mechanism, there are two common types of reflection [4].

3.1 Structural reflection

In this type of reflection, the reificiations of the program are based on the structure of the program itself and its respective data. The program is able to read and modify its own structure, and hence enables users to adapt the structure to their own needs.

3.2 Behavioral reflection

The reifications in this type of reflection are based on the behavior of the program during runtime. The program can read information about its current state and modify its behavior at runtime.

These types of reflection provide a useful mechanism for adapting the software during runtime, and providing a dynamic user interface. Thus, the operations in this mechanism are split into two types: introspection and intercession.

The introspection operation allows the program to read information about a specific aspect of itself based on the type of reflection that takes place. Thus, structural introspection allows the user interface to present application data along with their data structure, while behavioral introspection allows the user interface to acquire information on how the system interacts with the user.

The intercession operation allows the program to modify a specific aspect of it, thereby, allowing the user interfaces access to either the structure or the behavior of the program depending on the type of reflection. One of the uses of structural reflection is in the field of generic templates, while behavioral reflection can be used for providing feedback to the user's actions.

4 How does reflection work

The software architecture consists of two levels of computations – the base computations and the meta computations [4]. The metalevel is represented by metaobjects or metadata. These levels are connected such that a modification in one level triggers the modification of the computations in the other.

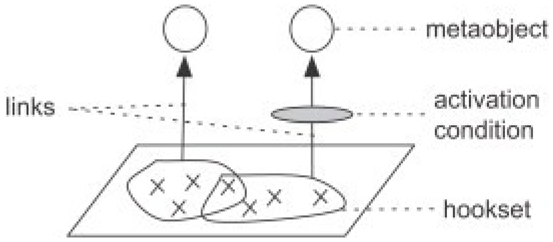


Figure 1. The model of the reflection mechanism – hooksets and metaobjects

These metaobjects are linked to reifications of the program such as its behavior or its structure. This allows the software to adapt its specific aspects during run-time without enforcing any static modification. A model that defines behavioral reflection is based on the concept of metaobjects which are linked to hooksets which represent certain execution points in the program. These metalinks are defined by several attributes and conditions which are dynamically evaluated.

During the run-time of the program, specific events trigger the modification of metadata. These events are then processed by the metalinks to select the hookset that should be used in order to invoke the new reification. Once the required link is found, the reification takes place and the program adapts during run-time.

5 Uses of Reflection in User Interfaces

There are a number of applications of reflection in the field of user interfaces. Some of them have been listed and described in [1].

5.1 Master Instances

Master instances represent the functionality of their normal instances i.e. any change in the formatting of the master instance propagates the same change to all the instances of that master instance. In this concept, the metadata as well as the data are represented by a single data model. The metadata is also displayed in the same format as the main data and can be modified too.

The main factor in this type of functionality is the degree of freedom that the user has to modify the metadata. Usually, not all the metadata is available for modification to the user.

One of the main drawbacks of this type of application is that the master instances are file-specific i.e. they act as templates only for their own file and cannot be spanned across a directory of files.

5.2 Generic User Interfaces

One of the most useful applications that reflection is useful for is in the field of generic user interfaces. With an advance in software technology, there are many programs that can use data in different formats. These programs can use these different data formats using a single data model using reflection.

They provide a generic user interface that can accept almost any type of data set, due to the underlying data model. This allows the program to accept data of one type and use the reflection mechanism to process this data along with a data set of another type.

Thus, this allows the programs to accept a wide range of data sets without needing to manually create data models for each type of dataset.

5.3 Dynamic Help for User Interfaces

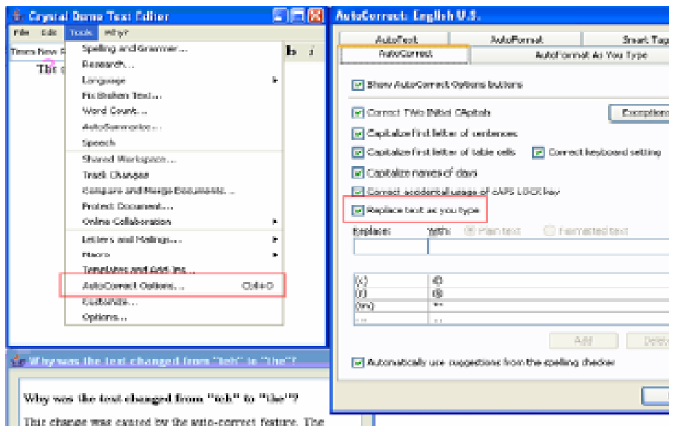
A major field of software development where reflection is playing a very important role is dynamically generated help for user interfaces. Due to the advance in software technology, the software products that are being developed are very complicated and hence, it is required that the users are able to use the software without any difficulties.

From [3], some of the qualities that the help document should provide are consistency, navigability, completeness, relevance, conciseness, coherence, fidelity and reuse.

The user interface acquires information about its current state and any other attributes present [2, 7]. Then, the help document dynamically builds questions or tips based on the actions just performed or on the current state of the program (see Figure 2). Sometimes, the document also responds to the cursor location, a key stroke etc.

Once the help comments are generated, the user interface displays the reason why a specific event happened and how the event can be rectified. For example, the user can enquire why a particular button is disabled and what is required to enable it.

When reflection was first used for generating help dynamically, textual help documents were produced. The questions or tips that were displayed were listed in a textual format either in a menu task bar or as a separate document. Gradually, dynamically generated animated help was implemented. This enabled the users to understand the help menus more clearly and made it easier for them to use the systems.



☑ Show AutoCorrect Options buttons
☑ Correct TWo INitial CApitals
☑ Capitalize first letter of sentences
☑ Capitalize first letter of table cells ☑ Correct keyboard setting
☑ Capitalize names of days
☑ Correct accidental usage of cAPS LOCK key
☑ Replace text as you type
☑ Automatically use suggestions from the spelling checker

Why was the text changed from "teh" to "the"?

This change was caused by the auto-correct feature. The

Figure 2. An example of a dynamically generated help menu based on actions that just took place

5.4 Other application areas

There are several other application areas where reflection can be used in the software development. Some of them are plug-in architectures, direct data access and document-oriented user interfaces [1].

Plug-in architectures allow the user to add additional functionality to a system, which changes the behavior of the program during run-time. This uses structural and behavioral intercessions, and shows a little bit of behavioral introspections.

Direct data access allows the user to directly access the data model and different types of data structures, which will allow the program to accept these data structures too. This type of mechanism can be handled by the reflection process.

Metadata is normally stored in auxiliary dialogues which have different life scopes with different variations. In document-oriented user interfaces, the metadata and the data is stored together as a whole document. This gives the user an idea of the scope of the metadata in the document and allows the metadata to be integrated with the data model. Thus, reflection can be used for this purpose of producing document-oriented user interfaces.

6 Applications using reflection in user interfaces

Section 5 describes a number of application areas where reflection can be used in user interfaces to make the system more user-friendly. This section discusses some of the applications and user interfaces that use reflection.

One of the major advances in user interfaces is the automatic generation of context-sensitive dynamic help.

6.1 Dynamic Help Generation

In the early days of software development, the help manual was hard-coded within the software code before its compilation. Thus, the user would be able to view the same help context repeatedly irrespective of the actions that they performed. As the user interfaces got more complicated, advances were made in displaying help tutorials. Automated help manuals were implemented which displayed information to the user related to the actions that had just been performed.

In the earlier versions of the dynamic help generation, the widgets in the user interface consisted of pre-conditions and post-conditions [6]. These conditions needed to be fulfilled in order to make the widgets enabled or visible. Whenever the widget was disabled, the help menu would display to the user that some of the conditions for that particular widget were not met. Moreover, the help menu displayed the conditions that needed to be set in order to make the widget enabled. However, these versions had a few drawbacks that rendered these versions inefficient. They were based on an assumption that each widget was used independently, whereas there were several cases where a combination of widgets needed to be activated to perform a specific task. This would have made it cumbersome to load all the widgets into the help menu. Also, due to the large number of widgets, it was not efficient to store a group of condition constraints for each widget to generate the help menu. Thus, newer user interface help generators evolved. One of these help generators was HelpTalk which was created in the UIDE (User Interface Design Environment) [2].

6.1.1 HelpTalk – UIDE

HelpTalk is an automated help generator in UIDE which clearly separates help access and help generation as two different mechanisms. UIDE separates its application into two separate models – the application and the interface models. The application model can have a number of interface models linked to it. Each of these models consists of their respective actions and objects. These actions and objects are linked to one another through action-object mappings (see Figure 3).

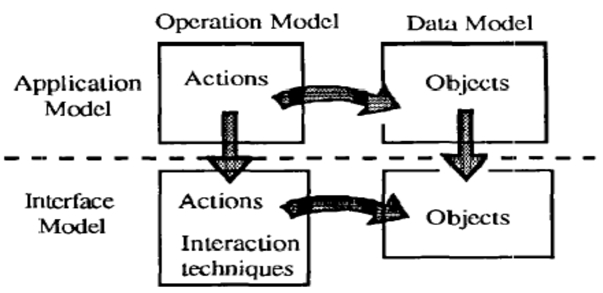


Figure 3. Application and Interface models in UIDE

These models are represented by their respective blackboards during the runtime of the UIDE. It also consists of a User Interface Controller (UIC) which processes the user's interactions, invokes the application action and displays dialogues in order.

HelpTalk has access to the UIDE's entire knowledge base and produces its comments using the state that the blackboards are in. Once the help is generated, it displays an animation of how to perform the specific task.

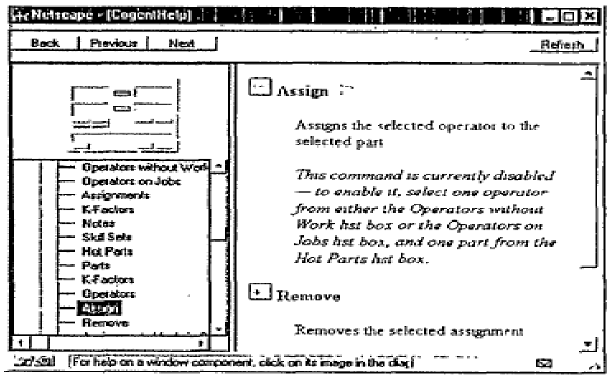
Using the knowledge base, HelpTalk generates the textual help by explaining the reasons why the interface is in that particular state. The reasons are phrased based on the models present in the UIDE's knowledge base. Then, it generates an animation that displays the procedures required to complete the textual help displayed.

There are a number of help generators that use a different mechanism to that of HelpTalk. One of these generators is CogentHelp.

6.1.2. CogentHelp

CogentHelp is a prototype tool that generates dynamic help for user interfaces built using Java Abstract Windowing Toolkit. It uses a different mechanism for creating help documents using snippets and servers [3].

CogentHelp accepts a set of "human-written" snippets as an input and attaches these snippets to their respective widgets in the user interface. When the help system for a specific widget is generated, the snippets for the specific widget are linked together. The help system consists of different views including a thumbnail and a tree view (see Figure 4).



Assigns the selected operator to the selected part

This command is currently disabled — to enable it, select one operator from either the Operators without Work list box or the Operators on Jobs list box, and one part from the Hot Parts list box.

Removes the selected assignment

[For help on a window component, click on its image in the diag]

Figure 4. CogentHelp consists of different views

The help topics are delivered through an HTTP server via a Java Servlet API. When a help request is made, the help server is loaded through a particular URL. This URL encodes the current state of the system, some parameters and attributes as well as the "snippets" for the corresponding widgets. The generated HTML forms can be used by the program authors to edit snippets using "frames" which allow the authors to add in missing keywords in the snippets.

The help authors generate exemplars which are frameworks that support different text generation methodologies. These exemplars follow object-oriented design and can inherit from one another. They have been implemented in the system to make use of Java's object-oriented design and to help the authors. Once the exemplars are prepared, the help documents are generated through the server.

6.1.3 SmartAide

SmartAide is another help system which takes the automated help generation to the next level [5]. Apart from displaying step-by-step textual instructions to the user, it also executes the action sequence of the task being described and changes the state of the interface.

When a user requests for help, an AI planning system generates an action sequence which are designed to execute within the user's workspace. It changes the state of the user interface and completes the task that the user desires.

Thus, there are a number of advantages over the other help generation methods. First, the action sequences are executed by changing the state of the system and the help authors do not have to worry about the intermediate states. Also, the action sequences that are executed can be tailor-made to suit the user's preferences.

From Section 6.1, it can be seen that there are a few applications that use reflection for developing user interfaces by creating help manuals. Another example for this is the CRYSTAL framework which is similar in structure to the HelpTalk framework [7].

6.2 Operating Systems

There are a number of examples where reflection has been used in other software fields such as operating systems [1]. Unix-style operating systems maintain a particular approach known as the "all-file" approach. In this approach, all the objects in the system are stored as files, including metadata. Thus, this is the case of structural introspection and intercession where the metadata and the data share the same data model. This enables the files to be modified similarly. Later versions of these operating systems are even storing windows as files as well.

In the MS operating system, the registry directory service also uses structural reflection.

One of the major applications of reflection in MS Office is the use of master instances. As discussed in Section 5.1, these master instances can be modified to alter all the instances of the master instances.

6.3 META-Case tools

META-Case tools provide the users the ability to modify the metadata and specify the functionality of the underlying models. A user has to define a model using a diagram and specify the model elements as well. Since the user has direct access to the metadata and the functionality of the model, these tools use structural reflection in their implementations.

7 Summary and Conclusion

This paper discusses the role of reflection in user interfaces in the field of HCI. Reflection has been used in the programming aspect of computer technology, but has been used for user interfaces recently. Some of the applications that reflection is useful for has been listed, along with examples in the software field where reflection has already been used. These examples have been explained briefly to give an overview of how reflection is implemented in these systems.

In conclusion, reflection is a powerful mechanism that can be used in user interfaces to improve the usability of systems as these systems are becoming more complicated day by day. It can be used to generate help documents or to process different types of data structures. Even though reflection has not been broadly used for this purpose, it has a very important role to play in the field of HCI.

8 Future Works

Even though reflection is used in the generation of the help documents, these documents only last for the session that they have been used for. One of the next steps would be to consider using reflection to store these help documents as files which can be used in later sessions. Another step would be to look into the use of reflection for other aspects of the software development such as debugging and so on. One of the final steps would be to conduct a research on the completion of many of the current application frameworks that use reflection and conclude whether reflection has been successfully used in all the fields of the user interface.

9 References

1. Lutteroth, C., Weber, G. (2007), Reflection as a principle for better usability. In *Proceedings of the Australian Software Engineering Conference, ASWEC,* Art. No. 4159682, pages 297-306

http://ieeexplore.ieee.org.ezproxy.auckland.ac.nz/iel5/4159639/4159640/04159682.pdf?tp=&arnumber=4159682&isnumber=4159640

2. Sukaviriya,P.N., Muthukumarasamy, J., Spaans, A. and de Graaff, H.J.J. (1994) Automatic generation of textual, audio, and animated help in UIDE: the user interface design. In *AVI'94: Proceedings of the workshop on Advanced visual inter-face*s, pages 44–52, New York, NY, USA, ACMPress.

http://delivery.acm.org.ezproxy.auckland.ac.nz/10.1145/200000/192322/p44-sukaviriya.pdf?key1=192322&key2=9342819021&coll=ACM&dl=ACM&CFID=65452701&CFTOKEN=34290955

3. Caldwell, D.E. and White, M.. (1997) Cogenthelp: a tool for authoring dynamically generated help for java guis. In *SIGDOC'97: Proceedings of the 15th annual international conference on Computer documentatio*n, pages 17–22, New York, NY, USA. ACM Press.

http://delivery.acm.org.ezproxy.auckland.ac.nz/10.1145/270000/263371/p17-caldwell.pdf?key1=263371&key2=0093719021&coll=ACM&dl=ACM&CFID=65440510&CFTOKEN=94513471

4. Caromel, D., Cointe, P., Noye, J., Tanter, E. (2003), Partial behavioral reflection: spatial and temporal selection of reification. In *OOPSLA'03 Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, Vol. 38, Issue 11, ACM Press.

http://delivery.acm.org.ezproxy.auckland.ac.nz/10.1145/950000/949309/p27-tanter.pdf?key1=949309&key2=1453819021&coll=ACM&dl=ACM&CFID=65452701&CFTOKEN=34290955

5. Ramachandran, A., Young, R.M. (2005), Providing intelligent help across applications in dynamic user and environment contexts. In IUI '05*Proceedings of the 10th international conference on Intelligent user interface, ACM Press.*

http://delivery.acm.org.ezproxy.auckland.ac.nz/10.1145/1050000/1040893/p269-ramachandran.pdf?key1=1040893&key2=8183819021&coll=ACM&dl=ACM&CFID=65452701&CFTOKEN=34290955

6. Foley, J.D., Gieskens, D.F. (1992), Controlling user interface objects through pre- and post conditions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 189-194, Monterey, California, USA.

http://delivery.acm.org.ezproxy.auckland.ac.nz/10.1145/150000/142787/p189-gieskens.pdf?key1=142787&key2=6424819021&coll=ACM&dl=ACM&CFID=65452701&CFTOKEN=34290955

7. Chau, D.H., Ko, A.J., Myers, B.A., Weitzman, D.A. (2006), Answering why and why not questions in user

interfaces. In *CHI'06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, Montreal, Quebec, Canada. ACM Press

http://delivery.acm.org.ezproxy.auckland.ac.nz/10.1145/1130000/1124832/p397-myers.pdf?key1=1124832&key2=6934819021&coll=ACM&dl=ACM&CFID=65452701&CFTOKEN=34290955