CHAPTER 5

Evaluation and Conclusion

n this chapter, the strength and the limitations of UMLCASE and as well as the metaCASE tool, **MetaEdit, will be described in the first part, while the conclusion of the research will be given in** he second part of this chapter.

.1 System Evaluation

The evaluation of UMLCASE is divided into two parts. The first part will discuss on the strength f UMLCASE and the following part will discuss on the limitations of UMLCASE and MetaEdit.

.1.1 System Strength

The strength of UMLCASE are given as follows:

Most of the UML notation, syntax and semantics are fulfilled.

UMLCASE is developed according to the notation, syntax and semantics of UML. This is to nsure that it is able to operate exactly like the Rational Rose UML CASE tool.

UMLCASE acts as a modeling tool to support system analysis and design.

his CASE tool is developed via the meta-modeling approach and can be used to model the cenario or the problem domain of a new system.

UMLCASE can be used to support almost all the techniques of UML using the meta-modeling pproach.

Dne of the objectives is to develop a CASE tool for UML by using the metaCASE tool, Aetaedit. All the eight techniques of UML have been successfully developed and run under the Aetaedit environment.

UMLCASE supports code and report generation.

All the models or diagrams that are drawn using UMLCASE can be generated into source code nd reports. Source code can be generated into four types of languages: C++, Smalltalk, Java and Delphi. A source code or report might not be a complete program but it can define all headers or nterfaces of the system, the attributes, and operations of classes as well as their visibility. In other words, the system developer does not need to start coding or writing reports from scratch. n MetaEdit 1.2, each technique can produce report output but the source code has to be written nanually.

UMLCASE provides user-friendly graphical representations.

The shapes are designed according to UML notations in order to avoid confusion and enable the ıser to use the tool in their analysis and design. In the MetaEdit 1.2 environment, the objects and ypes of relationships for each technique can be accessed from the *Model menu.* As for Aetaedit+ environment, the user will be provided with a palette that contains the object and the elationship icons. The palette is capable of assisting the analysis and design process.

UMLCASE provides model integration.

The extended version of UML techniques supported by UMLCASE using the MetaEdit+ can be ntegrated and allows users to access the techniques simultaneously. The integration techniques nclude explosions, decomposition, reuse of objects and sharing individual properties, such as *ame,* that can be applied to some other techniques.

.1.2 Limitations

JMLCASE has fulfilled the objectives and the scope of the project. However, there are some mitations in UMLCASE which are related to the limitations of the features of MetaEdit.

.1.2.1 Limitations of the UML Techniques Supported by UMLCASE

The limitations of UMLCASE are as follows:

- The techniques developed in MetaEdit 1.2 cannot be integrated. In other words, the objects or relationships that are defined can not be reused in other techniques. This feature is only supported by MetaEdit+.
- UMLCASE cannot imply the full syntax of UML since MetaEdit does not support the features such as the name of a class role in a collaboration diagram, as follows:

ClassRolename : Classifiername

- In the Deployment Diagram, the user cannot draw instance objects such as components or interfaces, directly in a node. The objects can only be created separately and dragged them manually into the nodes.

- In the Class Diagram, the visibility of the class, such as public, private or protected will not be displayed in the model. The user would have to fill in all the fields in the class form so that all hidden information can be listed in the report output.

For the Class Diagram, the code generation cannot generate the generalization relationship, but the code can be created using the report tool.

5.1.2.2 Limitations of MetaEdit

The limitations of the MetaEdit have caused a few problems during the development process. The limitations are identified as follows:

1. MetaEdit stores all the design data in a special memory location, called dynamic memory. As a result, it increases the efficiency of the memory management. However, it causes some limitations due to the fixed size of the memory. The default size of the dynamic memory is optimized to be 512 KB and sometimes it does not allow the user to move the object or copy the whole diagram due to insufficient memory.

2. MetaEdit 1.2 uses a pointer from the object pointer to allocate new objects and relationships. The limitation is related to the number of objects. Each object and relationship consume many pointers. As such, all the pictures must not be kept open or iconized.

3. In MetaEdit 1.2, the diagrams are modeled separately using the new techniques of UML. As such, the object or relationship that has already been defined cannot be reused. For instance, in the Use Case diagram, the actors, use cases and relationships cannot be recalled in other techniques. In other words, MetaEdit 1.2 only supports one modeling technique at a time.

4. MetaEdit 1.2 does not support certain syntax of UML such as, the name of object cannot be underlined or its font style cannot be applied.

5. MetaEdit 1.2 only provides a report that can list the objects or relationships.

2 Conclusion

ne objectives of the research have been achieved and have led to the development of MLCASE via the meta-modeling approach.

2.1 Future Enhancement

ture enhancement can be done to make UMLCASE more robust and ease of use. There are veral enhancements that could extend the usability of this CASE tool.

Extending the ability of UMLCASE

ne techniques can be developed in a more interactive environment such as adding the ultimedia features like icons or audio to guide the user in using the CASE tool.

Providing an interactive interface

future, the techniques could be enhanced to provide a clear guide or instruction to ease the rning and using the software. Each button could be incorporated a text box to describe the nction of the button.

Providing a more consistent presentation of notations

future, UMLCASE could be enhanced to support all the notations, syntax and semantics of ML. This is to ensure all the techniques to have consistent presentations.

2.2 Overall Conclusion

sed on the analysis and development phase, it can be concluded that the meta-modeling proach had proved its capability in defining other meta-models such as UML. A comparison s been made between the Booch method, OMT and OOSE in Chapter 2. These three ethods have been unified to form to the so-called UML. This comparison is made to identify e features of the three methods that are applied in UML. UML has provided eight techniques,

vhich are *Class Diagram, Use Case Diagram, Sequence Diagram, Collaboration Diagram, tate Diagram, Activity Diagram, Component Diagram and Deployment Diagram.*

The meta-model of all these techniques are defined using the OPRR (Object, Property, elationship and Role) concept and are extended using GOPRR (Graph, Object, Property, elationship and Role) concept. The first phase of this project is to develop the techniques sing MetaEdit Personal 1.2. In this phase, the techniques are modeled using the concepts of PRR. All objects, properties, roles and relationships of every technique such as the Use Case iagram must be identified. After all the techniques have been defined, the Method Definition ol is used to check their consistency. The **.mof file* is created and is ready for testing after ompilation.

n the second phase of the project, the techniques are extended using MetaEdit+ 2.5. In this ase, the meta-model of the UML techniques are developed using the concepts of GOPRR. In e MetaEdit+ environment, each model element is defined separately by a different tool. The raph tool is then used to combine the model elements into one graph. The main enhancement the second phase is the integration of all these techniques. In other words, all the elements hich have been defined separately can be combined in one graph and they can also be applied other graphs. The properties can also be shared or reused in other techniques. Even though l the techniques have been successfully developed and implemented through object-oriented nalysis and design, it can be further enhanced by adding the features that are not supported by etaEdit 1.2 and MetaEdit+. Finally, this project has successfully fulfilled the project's scope d objectives.