ABC TO METACASE TECHNOLOGY

MetaCase
Ylistönmäentie 31
FI–40500 Jyväskylä, Finland

Phone   +358 400 648 606
Fax   +358 420 648 606

E-mail: info@metacase.com
WWW: http://www.metacase.com

ABC TO METACASE TECHNOLOGY

Abstract
This article forms a short introduction to metaCASE technology. First, we describe the main advantage of metaCASE tools when compared to CASE tools. Second, we explain the basic principles behind metaCASE tools and illustrate how they work. MetaCASE tool functionality is then illustrated using the MetaEdit+® metaCASE tool. The article concludes by presenting a number of different application domains on using metaCASE technology.

1   WHY METACASE TECHNOLOGY?

Traditional CASE tools permit the user only the use of a certain fixed method. A CASE tool automates the use of specific modeling concepts, rules, diagramming notations and possibly more. For example, most CASE tools for object-oriented modeling are heavily based on the UML method. A method also dictates other CASE tool functions, such as how models can be made, checked and analyzed, and how code can be generated. For example, a tool can generate CORBA IDL definitions only if the modeling language can adequately specify and analyze CORBA compliant interfaces. If the tool (and method) does not generate them, it offers very little, if any, support for your work on interface design and implementation.

When using methods developers often face similar difficulties. They can not specify the domain and system under development adequately because the method does not provide concepts or notations for the task at hand. End-users may find the models difficult to read and understand because they are unfamiliar with the modeling concepts. Typically they also find it difficult to map the concepts and semantics used in the models to their application domain. After creating the models, which fail even to illustrate the application domain adequately, the tool does not provide the necessary reports nor does it generate the required code.

Sounds familiar? Don't worry, you're not alone. According to empirical studies[1,2], over 50% of developers modify methods or even develop their own methods. The reason is obvious: software development projects and organizations differ significantly from one another and evolve over time. One fixed method simply cannot work for all of them. Clearly, web-applications are developed differently from mobile phones, geographical information systems or logistic business processes. In

addition, product frameworks, components, programming languages, and database solutions, as well as the experience, structure and culture of the organization, all influence the method needs. Standard methods do not sufficiently recognize these situational needs and thus need to be customized. For example, all recent conferences on UML have been largely composed of presentations that promote extensions to UML!

The ideal solution is to customize the methods to the application domain and to the user's needs. For example, inventory systems are tailored for company-specific needs because inventories are managed differently in different companies and in different industries. However, until recently, reliable and easy method customization for software and system development has not been possible. Making design sketches with a pen and paper you may easily overcome the limitations of a method, but without any tool support. Using a CASE tool such customization is not possible because the tool dictates how you can design. This is true not only for CASE tools, but also for business process modeling tools, workflow modeling tools and design tools for information system architecture.

What is needed then is the ability to easily capture the specifications of any method and then to generate CASE tools automatically from these specifications. Later when the situation in the application domain evolves and the development environment changes you may incrementally update the method support in your CASE tool. This is exactly what metaCASE technology offers.

2   HOW DO METACASE TOOLS WORK?

Traditional CASE tools are based on a two-level architecture: system designs are stored into a repository, whose schema is programmed and compiled into the CASE tool. This "hard-coded" part defines what kind of models can be made and how they can be analyzed. Most importantly, only the tool vendor can modify the method, because it is fixed in the code.

MetaCASE technology removes this limitation by providing flexible methods. This is achieved by adding one level above the method level. This is illustrated on the top-right side of Figure 1.

```
 ┌─────────────────────────────────────┐
 │ MetaCASE tool                        │
 │  ┌──────────────┐                    │
 │  │ Metamodeling │                    │
 │  │ language     │                    │
 │  └──────────────┘                    │
 │         │                            │
 │         ▼                            │
 │  ┌──────────────┐                    │
 │  │ Any method   │                    │
 │  │ (metamodel)  │                    │
 │  └──────────────┘                    │
 │         │                            │
 │         ▼                            │
 │  ┌──────────────┐                    │
 │  │ Models       │                    │
 │  └──────────────┘                    │
 └─────────────────────────────────────┘
```
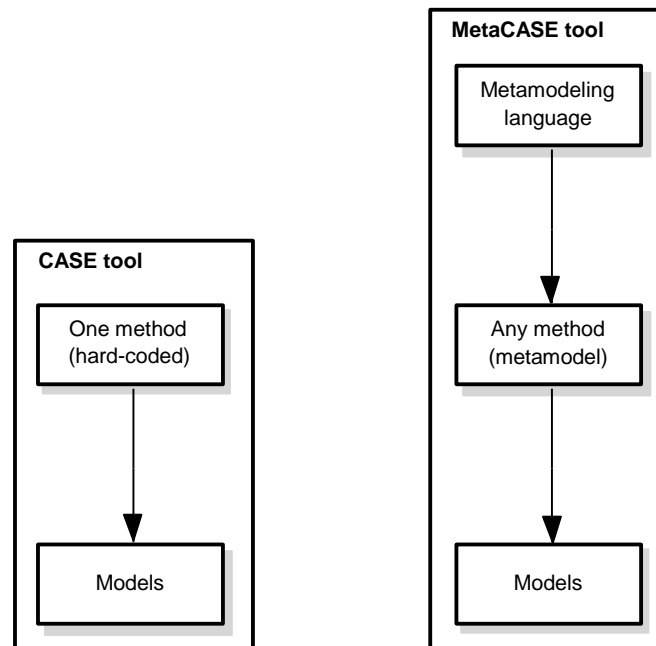
Figure 1. CASE tool versus metaCASE tool.

MetaCASE tools are based on a three-level architecture. The lowest, the model level, is similar to that of CASE tools. It includes system designs as models.

The middle level contains a model of the method, i.e. a metamodel. A metamodel includes the concepts, rules and diagramming notations of a given method. For example, a metamodel may specify concepts like a 'class' and an 'inheritance', how they are related, and how they are represented. However, instead of being embedded in code in the tool, as in a fixed CASE tool, the method is stored as data in the repository. The use of metamodels has recently become more popular. Many method books now include metamodels of their method, and several important innovations, such as XMI, are metamodel-based.

Unlike a CASE tool, a metaCASE tool allows the user to modify the metamodel. Hence, metaCASE is based on the flexibility of the method specifications. This is achieved by having a third, higher level that includes the metamodeling language for specifying methods. This level is the "hard-coded" part of the metaCASE software. A short example of metamodeling is shown in Section 4, which demonstrates metaCASE technology in practice.

All the three levels are tightly related: a model is based on a metamodel, which in turn is based on a metamodeling language. Clearly, no modeling is possible without some sort of metamodel. This dependency structure is similar to that between objects, classes, and metaclasses in some object-oriented programming languages.

3   A CASE OF METACASE: METAEDIT+

To illustrate the ideas of metaCASE technology we can look at an example of a metaCASE tool. MetaEdit+[3] is a multi-user and multi-platform environment that supports both system development and method development simultaneously.

As a CASE tool, MetaEdit+ forms a versatile and powerful multi-tool environment, which enables flexible creation, maintenance, manipulation, retrieval and representation of design information among multiple developers. Figure 2 illustrates some parts of typical CASE tool functionality: an editor for class modeling, a piece of generated code and a dialog for editing parameters in designs. We will use a class diagram here as an example because it is a well-known and documented modeling technique for describing code of object-oriented programming.
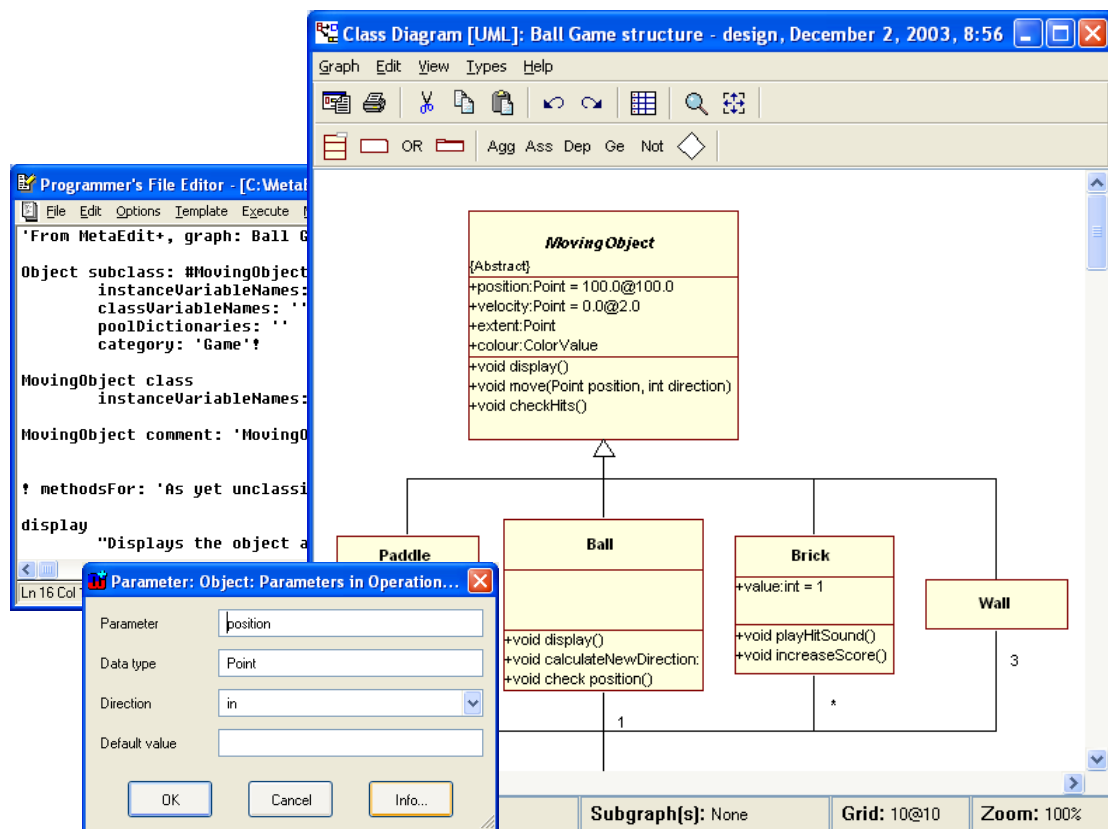

Figure 2 A screenshot of CASE functionality.

Figure 3 illustrates some of these functions by specifying a concept 'Class' with its properties, symbols and rules. Therefore, Figure 3 shows part of the metamodel whereas the Figure 2 shows a model based on class diagram.

Definition of a concept Class

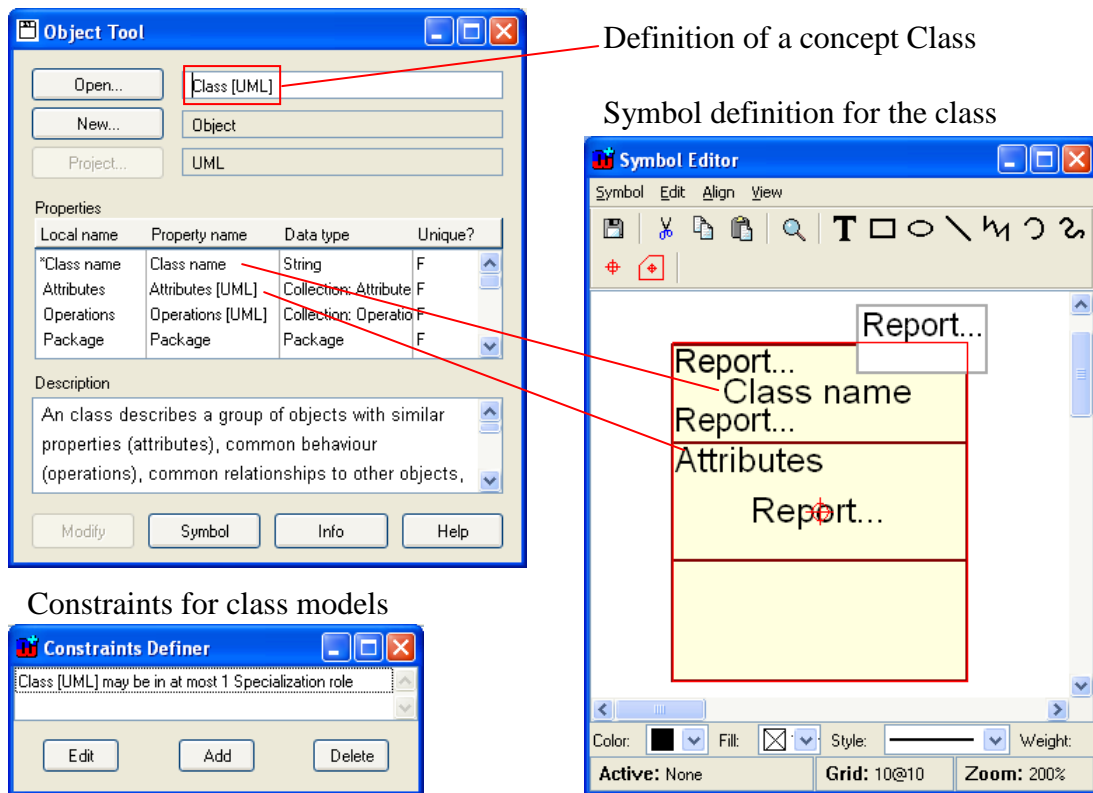Symbol definition for the class

Constraints for class models

Figure 3 A screenshot of metaCASE functionality.

The top left window is an Object Tool, currently showing the definition of the concept 'Class'. According to the definition, a 'Class' has a number of properties, such as a name and a collection of attributes. These properties are also shown in the notation defined with a Symbol Editor. The 'Class name' is displayed in the top compartment of the rectangle and 'Attributes' in the middle compartment.

Method definitions require a bit more than just a set of concepts and their symbols. For this purpose MetaEdit+ provides tools for specifying relationships between the concepts and setting different kind of rules. Figure 3 shows some examples of such rules for a metamodel of class model: a rule stating that 'Class may be in at most 1 specialization role' defines a single inheritance constraint.

As for other metaCASE functionality, MetaEdit+ provides modeling tools, a repository and basic reports automatically based on the metamodel. There is no programming, no compiling, no extra text files for menu definitions and no configuration files to be shared to users. The generated CASE tool can be used immediately in design work. MetaEdit+ shares the method definition automatically for developers and updates the existing models according to the changed metamodel. This makes CASE tool implementation fast and very cost-effective. For example, a person

who knows their application domain can specify a metamodel for it (and also a complete CASE tool for modeling the application domain) within a few hours. A method and its tool support can later be extended by improving the modeling language, making code generators and by linking the design results to or from other programs (simulators, documentation tools, compilers, debuggers etc).

4 CONCLUDING REMARKS

Tool customization is not a new idea: many application domains, such as ERP software, apply it. In the CASE domain it has not been used so far because of the high costs of customization. MetaCASE technology provides a major change. It allows users to quickly and easily define methods fitting their developers' needs. The result is a set of automatically generated CASE tools with the modeling techniques, checking rules, analyzers, code generators and documenting reports. This revolutionizes system development: the user dictates the tool support, not vice versa.

As an example of metaCASE technology we briefly presented MetaEdit+. This metaCASE tool has been used to implement more than 100 modeling techniques and dozens of report and code generators.

Although metaCASE technology is quite new, several large and mission critical systems have already been made using customized methods and metaCASE tools. Nokia has customized methods for developing network management systems and also develops its mobile phones using metaCASE tools; Deloitte & Touche used a metaCASE tool to implement tool support for their OO-method; and the development of the Eurofighter relies on metaCASE technology. These are only a few references of metaCASE technology use. Contact us for more information on how you too can reap the benefits in your business.

---

[1] Russo, N., Wynekoop, J., Walz, D., The use and adaptation of system development methodologies. *Procs of International Conference of IRMA*, Atlanta, May 21-14, 1995.
[2] Hardy, C., Thompson, J., Edwards, H., The use, limitations and customisation of structured systems development methods in the UK. *Information and Software Technology*, 37 (9), 1995.
[3] MetaCase, *MetaEdit+ 4.0: User's Guide*, 2005.

The trademarks, product and corporate names appearing in this article are the property of their respective owner companies.