Integrating CASE Tools with Knowledge-Base by Object Orientation

Fuyau Lin
Dept. of Computer Engineering
Santa Clara University
Santa Clara, CA 95053

Rob Langsner
Dept. of Computer Science
University of Nevada
Reno, NV 89557

Abstract
Knowledge-based systems are a branch of artificial intelligence that deals with the processing of knowledge. Integrating CASE, Knowledge-base, and Object Orientation is very a worthwhile approach. By providing the developer both conventional and knowledge based approaches to application development, object-oriented repository techniques for CASE tools can be applied to the specific task at hand. Object Orientation facilitates the reuse of design as well as programs. Additionally, a knowledge based component can contribute in assisting the analyst in making the right choice. This paper discusses the issues, directions, and strategies of the integration.

1 Introduction
The major goals of Object Orientation are to improve developer productivity by increasing software extensibility and reusability and to control the complexity and cost of software maintenance. When object oriented programming is used, the analysis and design phase of software development is more closely linked with the implementation phase [15, 20, 2, 11, 15].

Knowledge Base Systems (KBS or Expert System) are primarily designed for logic-intensive application [4, 16, 9]. The term rule-based system [9] describes a paradigm in which discrete pieces of knowledge are expressed in terms of an expression that states: when some condition is true, then some conclusion can be drawn or some action performed.

Computer Aided Software Engineering (CASE) adds engineering discipline to the software development process. When carefully applied, CASE tools strengths come partly from the automation of modeling data and process; the typical suite of CASE tools reflects these requirements [16]. In general, CASE tools include [18]:

- Graphic Analysis and Design Tools for drawing and for detecting incomplete, syntactically incorrect or inconsistent specifications.
- Screen and Report Painter for creating system specifications and simple prototypes.
- Repositories or Encyclopedias for storing and managing application and enterprise specifications.
- Code Generators to automatically generate executable code from specifications in the repository. Documentation generators produce technical and user documentation from the repository-based specification as well.

In this paper, we examine the integration issues and the methods to resolve the integration problem. Section 2 and 3 give a brief review of the level of integration and the problems. Section 4 discusses various requirements for integration and possible directions. Section 5 sketches how to integrate rule objects into CASE tools. Sections 6 gives a short summary.

2 Level of Integration
CASE needs to integrate with KBS on the application, tool, methodology, and meta-data levels. Before we discuss the problems of the integration, let us examine the level of integration required [7, 19, 11].

- Application Integration: CASE needs to address the full spectrum of applications, from completely data or logic driven to heterogeneous applications involving subsystems which fall into both categories. These applications will mix knowledge-based and high level language or code generator components.
- Methodology Integration: A KBS should be able to be designed using the more popular structured or object-oriented techniques, with some enhancements. These enhancements should include the notational constructs to support object-oriented analysis, design, and implementation.
- Tool Integration: Complex KBS have evolved with sophisticated Object Oriented components. In order to design a KBS with CASE tools, structured methods will need to be enhanced to support object oriented analysis, design, and code generation.
There are many issues surrounding achieving the integration which includes:

- **Meta-Data Integration:** It follows that CASE repositories will need to become capable of handling KBS rules as well as other objects which may be unique to KBS. The combination of using the object-oriented approach to analysis and design, coupled with the power and maintainability of rules and inferencing, all brought together under the CASE umbrella, provides a powerful development framework.

3 The Issues

There are many issues surrounding achieving the integration previously discussed. The issues cover a broad spectrum which includes:

- **Overcoming the Commercialism**
- **Structured Methods are not Object-Oriented**
- **Slow Adoption of OO by Industry Leader**
- **Lack of MIS Acceptance of KBS**
- **Lack of Pressure on Vendors from Users**
- **Pure Object Orientation vs. KBS Object Orientation**
- **Potential for Dramatic Market Changes**

The integration is difficult because the Big Three are each coming from a slightly different background and therefore are not synchronized due to this. They are all heading in the same general direction, just not together.

4 Requirements and Strategies

We describe the general requirement for the integration as follows:

- The Knowledge-base must be executable.
- The CASE tool must generate interfaces of different components automatically and correctly for the target environment.
- Object orientation needs object-oriented notation, layering, advanced features, and model checks as well as seamless integration of object-oriented analysis, design, implementation, and testing.

4.1 Integration Strategies

It would seem that the knowledge is merely the semantic interpretation of what is in the data structures called classes, instances, and rules. It is conceptually equivalent to the data that is in the data structures called tables. The CASE tool could provide intelligent help if it had a knowledge base pertinent to the domain of selecting data or knowledge-based approach to solve the type of problem being analyzed and designed.

The following steps provide one possible scenario as the steps that may be taken by the industry while moving towards the complete integration of CASE, Object Orientation, and Knowledge-Based Systems.

- **Step 1 – Repository Interface:** The first step towards integration will be sophisticated conversion tools designed to construct skeleton knowledge bases from the high level design information stored in the CASE repository. AD/Cycle is a very good example of this approach [12]. The repository interface tool would be available to transfer relevant design information from the CASE repository to the knowledge base, automatically generating the skeleton knowledge base.

- **Step 2 – Methodology Integration:** This process will be marked by a completely integrated development methodology. The designer will model data and knowledge structures side by side and may not necessarily need to make a distinction. One single methodology accommodates the entire development process. An integrated CASE/KBS framework supports this methodology through a single tool, although complete tool integration has not occurred yet. The repository has both KBS and traditional CASE information stored indistinguishably, and use throughout the entire life cycle.

- **Step 3 – Meta-Data Integration:** Also known as Co-ordinated Development, this step will offer a more integrated development environment. While KBS and CASE tools will remain separate, they will interact easily with one another. Further, they will provide robust support of a single integrated methodology. Version control and dependency information will be synchronized across KBS and CASE generated components [1].

- **Step 4 – Tool Integration:** Complete integration occurs when there is significant integration of methodologies, tools, and repositories. Newer tools are providing object-oriented extensions to diagraming tools and are also providing new diagraming tools which have object-oriented analysis purposes. The notational constructs range from standard techniques, as found in existing tools, to those newer patented notational constructs, e.g. Booch notation [3].

- **Step 5 – Reusable Application Models:** There have been many attempts to accomplish this and there are currently a number of database products on the market that support such a notion. It is a natural extension of the CASE tool. The repository is extended to handle objects of larger granularity. The size of these can range from the entity to a complete piece of functionality, such as programs, which handle general tasks like scheduling, monitoring, transaction processing, configurations, and etc. Any time a problem becomes well understood, and the solution lends itself to a knowledge-based approach, a new application model can be added to the repository providing reusability of not just functions and objects but also of complete solutions to problems.

- **Step 6 – Intelligent CASE Tool:** This step is marked...
by the CASE tool using advanced techniques for supporting the various steps in the construction of a program. The case-based reasoning technique currently being developed in the Expert System industry is a likely candidate for supporting decision making during analysis and design. The CASE tool is, then, an intelligent agent that helps you model your problem, and by searching its memory, finds similar problems and recommends solutions - and then implements them on request.

5 Integrating Rule-Object into CASE Tools

This section briefly summarizes two approaches to integrating knowledge base rules into CASE tools [19]. They are (1) extending the mini-spec, and (2) adding a new diagraming tool specifically for rules. There may also be other approaches.

We first discuss properties of rules briefly. This discussion leads to a set of requirements for the integration. Finally, with this background, we can discuss the approaches to integration, including the benefits and drawbacks of each.

5.1 Overview of Rule Properties

In order to integrate the KBS rules into the CASE framework, the properties of rules must be accounted for so that code generation can occur in a seamless manner. Some properties of rules that should be considered for inclusion include:

- Forward/Backward/Multi-directional provides information to the inference engine indicating which inference strategy to employ the rule in.
- Cost Property provides information to the inference engine regarding an estimate of the expense for trying a particular rule.
- Priority provides the inference engine with a sequence or ordering in which the rules should be evaluated for application. The user may know a priori which rules are most likely to be successful.
- Rule Body provides the data conditions which must be present for the rule to be applicable as well as the instructions for executing the rule once it has been determined that it should be applied.
- On-data vs. On-request provides the knowledge of whether the rule should be applied whenever a specific piece of data becomes known, changes or meets some criteria vs. only applying the rule at the specific request of the implementer. The On-data specification provides the ability of the rules to behave like active objects - the rule knows when to apply itself.
- Rule Scope provides an understanding of the appropriate time, during the overall processing, to consider applying the rule. Often in a knowledge base there is a notion of current scope of control. This is somewhat similar in a conventional language to executing only those functions within a specific control block.

The requirements for integrating rule objects into CASE tools should consider the following aspects: store all rule properties, check logic for consistency, generate missing rules, eliminate redundancy, and provide rule interpreter, cross referencing, and language templates.

5.2 Enhancing Standard CASE Tools

We would enhance notation or functionality for supporting rules in the repository by extending the mini-spec action diagraming tool.

The mini-spec is a diagraming tool attached to sequential processes in dataflow diagrams. You are typically allowed to specify procedural action associated with a dataflow bubble inside this tool. Since rules can be viewed as procedural, this becomes a natural place to introduce them into the analysis processes. There are two sub-strategies for implementing rules into the mini-spec:

- Having each mini-spec correspond to a single rule.
- Allowing rules to be written into a single mini-spec.

5.3 Adding a new Diagraming Tool

This approach consists of providing a new diagraming tool which allows rules to be supported using mathematical foundations of decision table (matrix arithmetic) such that the automation requirements can be met. The most likely candidate for this comes from Decision Management Science and it is the concept of a decision table. This structure uses a tabular format to map the logic behind a decision making process. Complex judgements can be viewed in a format which is very easy to comprehend.

5.4 Discussion

- Mini-Spec Advantage: Enhancing the mini-spec action diagramer with properties that provide the information required for a KBS to interpret the rules is an attractive option. Meeting this requirement is difficult in that many KBS use different strategies for interpreting rules. A general set of properties could be included and then any code generation would simply map available information into the target Expert System environment.
- Mini-Spec Disadvantage: There are some obvious flaws in this approach. The primary one is that rules are typically grouped into related sets, all of which are tried for given set of data. Even if a mini-spec allowed you to enter a, possibly sequential, listing of rules - this is not a very user friendly way to view this
data. Some form of abstraction would be needed in
order to get the global view of the rule base. If there
was the restriction that only a single rule could be
represented per mini-spec, then it would become dif-
ficult to analyze and access the rules for cohesiveness,
not to mention test them.

- **New Diagraming Tool Advantage:** It should be rela-
tively easy to automate this type of diagraming tool
and construct an interpreter. The interpreter gives
the analyst a chance to see the rule set in action be-
fore generating the code. This is a significant benefit
when one considers the fact that the KBS compo-
nent is likely to be embedded into calls from conven-
tional code, which must go through the time consum-
ing compilation process.

- **New Diagraming Tool Disadvantage:** The first draw-
back is that the new tool would be a departure from
conventional SA/SD. It should be noted, however,
that the rule constructs themselves are significantly
different than any constructs found in the conven-
tional diagraming tools - so possibly adding this new
tool is justified. Naturally, many opponents will argue
against having to learn a new tool.

6 Summary

Integrating CASE, Knowledge-base, and Object Orienta-
tion is very worthwhile approach. By providing the devel-
oper both conventional and knowledge based approaches
to application development, object-oriented repository for
CASE tools can be applied to the specific task at hand.
Object Orientation permits the reuse of design as well as
programs. Additionally, a knowledge based components
can contribute to assisting the analyst in making the right
choice. The problem of integrating these technologies re-
quires industry leaders to adopt solutions which can be
placed in the public domain. If this placement is not done,
the pace of integration will slow. Major CASE or I-CASE
vendors must integrate these frameworks and then they
must supply some growth path for users to effectively take
advantage of the integration. A methodology is essential
here and users, hence the integrated methodology, will not
succeed without one.

References

[1] B. Alabiso, "Transforming of Data Flow Analysis
Models to Object Oriented Design," OOPSLA'88

[2] T. J. Biggerstaff, C. Richter, "Reusability Frame-
work, Assessment, and Directions," IEEE Software,


of KBS and CASE," Software Engineering Strategies,
Nov. 1991.

[5] P. Coad, E. Yourdon, "Object-Oriented Analysis," Ob-

[6] P. Coad, E. Yourdon, "Object-Oriented Design," Ob-

Reusing Abstract Data Types," Proceeding of the 9th

[8] E. Freberger, "Will Object Technology Replace

quision for Knowledge-Based Systems," Academic

[10] K. Gardner, "Designing Knowledge Systems with Ob-


[12] D. M. Hembry, "Knowledge-Based Systems in the

[13] R. E. Johnson, V. F. Russo, "Reusing Object-
Oriented Designs," University of Illinois Tech. Report,


[15] B. Meyer, "Object-Oriented Software Construction,
" Series in Computer Science, Prentice Hall Interna-


olution: Should Structured Methods be Objectified?"

[18] R. S. Pressman, "Software Engineering: A Practi-

[19] D. R. Rahn, "Integrating CASE, Expert Systems and
Object Orientation," Research Paper for COEN 698,
Santa Clara University, 1991.

[20] R. Wiener, L. J. Pinson, An Introduction to Object-
Oriented Programming and C++, Addison Wesley,

[21] J. Williams, "The Object Paradigm Requires Object-
Oriented methods and Tools," Object Magazine,