Using Paradigm Plus CASE Tool in a Fusion-based Application Development Project

Bajgoric, N.
Boğaziçi University, Department of Industrial Engineering
80815 - Bebek, İstanbul, Türkiye
nijaz@boun.edu.tr

Draman, M.
Boğaziçi University, Department of Industrial Engineering
80815 - Bebek, İstanbul, Türkiye
draman@boun.edu.tr

Altınel, I.K.
Boğaziçi University, Department of Industrial Engineering
80815 - Bebek, İstanbul, Türkiye
altinel@boun.edu.tr

Ünal, A.T.
Boğaziçi University, Department of Industrial Engineering
80815 - Bebek, İstanbul, Türkiye
unaltam@boun.edu.tr

Abstract

This paper presents some experiences in using Fusion as an object-oriented development method and Paradigm Plus as a CASE Tool supporting Fusion steps in an application development project. The paper aims to identify some questionable semantic aspects of Fusion and areas of confusion in using both Fusion and Paradigm Plus.

1. Introduction

In 1997, a group of faculty members and researchers from Industrial Engineering Department at Boğaziçi University started a project called AREMOS* (A REfinery MOdeling System). AREMOS presents itself as a prototype interactive visual modeling tool for building a visual presentation of a complex process industry - a refinery, with all relevant unit-specific production data and parameters. Without any intervention by the user, this visual model would automatically create a hidden linear programming model for optimizing production decision variables.

A project team decided to adopt the following application development framework:

- Object-oriented approach as a dominant paradigm in contemporary software engineering
- Fusion as an application development method which supports object-oriented approach
- Platinum's Paradigm Plus as a CASE tool which supports Fusion method
- HP SoftBench as an integrated development environment on UNIX platform
- Visual Edge's UIM/X as a tool for GUI design on UNIX platform
- HP C++ as an implementation language on UNIX platform
- C++ Builder as a RAD tool for PC-version of the application
- Borland C++ as an implementation language on PC platform
- CPLEX as a solver

* Research supported by Boğaziçi University Research Fund, Scientific and Technical Research Council of Turkey TUBITAK-MISAG, and hardware and software donation by Hewlett-Packard.

2. Fusion

In this section, we describe some problems we encountered while working with Fusion, in order to identify shortcomings and inconsistencies:

- *Requirements.* Fusion has no requirements phase. Paradigm Plus supports the Fusion method extended to include requirements capture and the Jacobson Use Case Model (1). Since we started working on requirements phase before selecting a CASE tool, in our project we have used standard document describing the requirements against the new system.
- *Object Model.* The notion of Object Model is not consistent with its meaning because it is not based on objects. Rather, since it represents a system structure by means of entities, attributes, and relationships, it would be more appropriate to call it "Entity-Relationship (ER) model" or "Class Model".
- *Object Model vs. System Object Model.* Confusion exists in relations between Object Model and System Object Model and representation of external real-world objects that remain outside the system boundary. The confusion arises when there are both agents in the environment and classes inside the system that reflect the external agents.
- *Relationships.* From ER perspective, a standard relationship (association type of relationship) can also be used in modeling the containment by using "contains" or "is contained by". No explanation is given which approach is more suitable. Another question that arises here is how to represent the "many-to-many" containment relationship in which more than one parent (superclass) may have one or more children (subclasses). It seems that Fusion uses aggregation in two different manners interchangeably:
    - as a way of capturing the containment type of relationship
    - as a way of "grouping" entities/classes in Object Diagrams
- *Interactions between Agents and System.* All kinds of interactions between Agents and the System in Object Model are represented with standard association type of relationships, even though they are not so. They should not be treated as relationships since they are in fact System Operations. The same analogy applies also in case of the actions performed by System to Agents (System Events).
- *Readiness of Life Cycle model.* After expressing all sequencing constraints in Life Cycle model, its readiness becomes a problem. Therefore a graphical representation of Life Cycle would be more useful.
- *Timeline Diagrams (Scenario Diagrams).* There is no explanation on how to show alternative communications.
- *Operational Model.* Showing the optional events in the Send clause of the Operational Model is not described.
- *Object Deletion.* Deletion of objects is not treated in the operation model nor in Object Interaction Graphs (OIGs).

Many of the problems we have encountered and listed above come from incomplete definitions. As far as we could see in a document entitled "Engineering Process Summary" which is announced as Fusion 2 (2), most of these problems will be addressed in the next version.

3. Paradigm Plus

Platinum's Paradigm Plus (3) is a very comprehensive CASE tool since it includes support for almost all well-known object-oriented (OO) methods (Rumbaugh/OMT method, Booch OOAD method, UML method, Fusion method, Martin/Odell OOIE method, Shlaer/Mellor OOA method, Coad/Yourdon OOAD method, OOCL method. In addition, Paradigm Plus provides some method-independent features like: Project Model, Use Case Model and Architecture model.

In the section that follows, we try to identify some problems and inconsistencies in Paradigm Plus' Fusion support. They are listed according to different aspects covering specific models (diagrams) in Fusion-related phases.

3.1. Object Diagrams

While working with Object Diagrams, the following problems occurred:

- Cardinality, aggregation names and roles in aggregation type of relationships are not done automatically upon creating socket relationships (primitive text string is used instead)
- There is no propagation of aggregation cardinality and aggregation roles from Object Diagram (text string); anything different than default (one to many) must be done through Table Editor
- Difference between “many”, “optional” and “one or more”, or “zero or more” multiplicity annotations is not clear
- Links representing the generalization type of relationship get broken when some changes are done either on parent or child classes
- Multiplicity annotations in ternary relationships (relationship between class A and class B through a ternary relationship X, which is represented as a new class in Paradigm Plus, becomes a relationship between A and X, and B and X.
- There is no table list by aggregation name and association name

3.2. Life Cycle

Life Cycle is supported only through metaclass (Object Browser) and it is an ordinary text-based document, with no propagation with Scenario Diagrams and Operation Model.

3.3. Scenario Diagrams

Scenario Diagrams have no propagation with Life Cycle and that part of Object Diagram which defines Agents. Also, repetition of system operations and system events are not supported (dotted portions of time line diagrams).

3.4. Operation Model

Operation Model is represented by an ordinary NotePad document and there is no propagation with Object Interaction Graphs.

3.5. Object Interaction Graphs

Propagation of objects in object repository while defining objects creates some problems in object definition. For example, if an object is defined as a controller in one OIG, then if it is used in another OIG as a collaborator, it is again shown as a controller. The same thing occurs when a new

object is created and when the same message name is used but with different parameters and sequence numbers. One of the main shortcomings of Paradigm Plus when OIGs are considered is in very poor support for method descriptions (again simple text mode). Also, there is no propagation between method descriptions and Class Descriptions.

3.6. Visibility Graphs

In Visibility Graphs the term of binding is considered as a type of visibility (reference lifetime according to Fusion) along with permanent and dynamic link.

3.7. Class Descriptions and Code Generation

Paradigm Plus supports creation of Class Descriptions and automatic code generation. Class description is created in design phase for each class taking info from the following models: Object Models, Object Interaction Graphs and Visibility Graphs. But, while generating class description Paradigm Plus takes info only from Object Diagrams. Even, it does not use all Object Model information, e.g. object-valued attributes that come from association relationships.

Paradigm Plus converts all aggregation relationships directly into object-valued attributes in class description, without considering visibility info. According to Fusion, "… Object attributes are extracted from the visibility graph for the class" (5, pp. 89). "… The reader may be surprised that Pump does not have Gun and Holster components, despite their appearing on the object model in section 6-3.1. This illustrates that aggregation on the object model is a matter of descriptive convenience rather than physical containment. Only the structure necessary to implement the references on the visibility graph appears in the class description…"(5, pp. 172). Apart from non considering visibility info, Paradigm Plus' class descriptions do not contain association relationships. This is not a big problem, because an automatic code generation uses both object-valued attributes, coming from aggregation and association relationships.

4. Conclusions

Although we appreciate Fusion as an excellent application development method, through our work we identified some difficulties in analysis and design. As for Paradigm Plus CASE tool, although we have not evaluated Paradigm Plus with respect to its support for other OO methods, but in Fusion's case, our experience shows that the main shortcomings are as follows:

- Poor propagation between analysis and design
- Incomplete code generation; it uses information from Object Diagrams only
- Poor support for method descriptions in OIGs; no propagation with Code Generation

However, we have used P+ extensively, especially for project development visualization and project documentation.

References

[1] Object-Oriented Software Engineering (OOSE) by Jacobson et. al.
http://wwwis.cs.utwente.nl:8080/dmrg/OODOC/oodoc/oo-12.html
[2] PLATINUM Paradigm Plus, Methods Manual, PLATINUM Technology, 1996
[3] Engineering Process Summary (Fusion 2), Hewlett-Packard Company, January 1998
[4] Object-Oriented Analysis and Design Methods
http://wwwis.cs.utwente.nl:8080/dmrg/OODOC/oodoc/oo-Contents.html
[5] Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F., Jeremaes, P., Object-Oriented Development The Fusion Method, Prentice Hall, 1994