MSc Advanced Software Engineering
Thesis Proposal • Ralph Skinner • 06169236
Supervisor: Dr. Joseph R. Kiniry

An Integrated Development Environment For Business Object Notation

1 Overview

Business Object Notation (BON) is a method for modelling object-oriented software with a supporting textual and graphical notation [1]. BON builds upon the benefits of general object-oriented design to increase software productivity by focusing on three aspects of software engineering: seamlessness (from design to implementation), reversibility (synchronicity between the design and the implementation), and software contracting (increasing quality by adding semantic specifications to classes). The BON graphical format is fundamental to communicating views of the software model more effectively to individuals, and in particular, the ability to give a global (if simplified) view of a complex model is seen an essential feature that cannot be provided by text alone. This makes BON an ideal candidate for CASE tool support, and the goal of this project is to create an IDE for BON development with support for the textual and graphical aspects of the language.

2 Related Work

Previous work with regard to CASE tools for BON are the BON Case Tool [2], and the BON Development Tool (BDT) [3]. Whilst the BON Case Tool does allow some visual editing, the functionality falls short of what is alluded to in [1]. For example, it lacks the capability to show diagrams at different levels of detail, with classes and clusters expanded or compressed. We aim to provide this kind of visual flexibility in a very seamless way, allowing the user to reveal more or less detail about the model by zooming in or out of the view. For the BDT, no software could be located, and neither the BDT or the Bon Case Tool projects appears to be in active development.

3 Core Features

The following is a list of specific functionality the IDE should provide.

- **Project View –** A view that allows the user to see all the files in the current project, add, remove, or rename files, and open any file in the editor by double-clicking on it.
- **Class View –** A view that lists the classes in the project in a hierarchical 'tree', with the functionality to select any class and go directly to it's definition in the BON code.
- **Syntax Highlighting –** Colour the BON code in the editor to highlight keywords, comments, class names, etc.
- **Auto Completion –** When typing object names in the editor, prompt and allow the user to select from a list of possible completions, i.e. class features.
- **Diagram Generation –** Draw and layout the BON code as a diagram, in particular, display the static model that shows clusters, classes, class features, and the relationships between them, such as inheritance or aggregation. The diagram should be interactive and allow the user to filter the content by zooming into specific clusters or classes, and to adjust the level of detail shown, e.g. display the expanded or compressed form of a class. Diagram elements will be designed to maximize readability, but will generally follow the conventions shown in [1]. The diagram view will be persistent, i.e. the user-adjusted zoom level and detail settings will be preserved between editing sessions.
- **Visual Editing –** Allow the user to edit elements of the project by modifying the diagram directly. For example, classes can be added by drag-dropping a class icon from a palette of diagram elements. The IDE would then automatically add the required BON code.

4 Optional Features

- **Error Highlighting –** Underline non-compiling sections of code with red 'squiggly lines' to highlight problems.
- **Interface with other BON tools, compiler, etc. –** Provide menu items or buttons to trigger the execution of other BON tools, such as a BON compiler.
- **BON Process Support –** Integrate other elements of the BON approach into the IDE to provide broad support for the entire BON process 'under one roof'. This could include, but is not limited to, support for system, cluster and class charts, dynamic charts, dynamic diagrams, and creation charts.

5 Planned Approach

The BON IDE will be developed in Java as an Eclipse Plug-In. The high-level idea is to design a meta-model for BON, and implement it using the Eclipse Modelling Framework (EMF). The meta-model will be a precise definition of the constructs and rules for creating BON models, and will capture the syntactic constraints that all valid BON models must obey [4]. Designing the meta-model is a good starting point because it will demand a good understanding of BON in general, and it will provide a representation of BON models that is independent of the type of notation (i.e. textual/graphical). This in turn should allow a certain level of meta-model based conformance checking [5]. The EMF meta-model can then be used as input to the Eclipse Graphical Modelling Framework (GMF). It should be possible to leverage the code generation capabilities of GMF to significantly advance the development of the BON IDE. The intention is also to integrate existing work by others on syntax highlighting [6] into the BON IDE.

References

[1] K. Walden and J. M. Nerson, *Seamless Object-oriented Software Architecture: Analysis and Design of Reliable Systems*. United Kingdom: Prentice-Hall, 1994.

[2] J. Lancaric, "The homepage of the bon case tool." `http://www.cse.yorku.ca/~eiffel/bon_case_tool/`.

[3] A. Taleghani and J. Ostroff, "Bon development tool," 2004.

[4] R. F. Paige and J. S. Ostroff, "Precise and formal metamodeling with the business object notation and pvs," 2000.

[5] R. F. Paige and J. S. Ostroff, "Metamodel-based model conformance and multiview consistency checking," 2007.

[6] F. Fairmichael, "The BONc Tool homepage." `http://kind.ucd.ie/products/opensource/BONc/`.