

Source Code Translation

Everyone who writes computer software eventually faces the requirement of converting a large code base from one programming language to another. That requirement is sometimes driven by the need to port software between operating systems, to a new programming language or development system or from one computer, operating system, phone or gadget to another.

Source code conversion is inherently a tedious, error-prone and labor intensive process. While there is no magic button to turn old code into new code, there are tools that can dramatically reduce the time and cost of a conversion project. While a development project includes testing and other activities, this discussion will focus on the code translation process.

Project Plan

The first step in any successful project is to understand what you are starting from, where you are going and then develop a plan to get there. For a code conversion project, you want to identify:

- Old and New Code Design and Architecture
- Foundational Differences Between Old and New System
- Programming Language Syntax and Semantic Differences
- Desired Feature Changes

Think about the design and architecture of the old and new code base. Is the code written in a procedural language like C, object-oriented language like Java or a mix of both? Is the new and old code a single program, collection of programs, libraries and components or even a multi-threaded system? A big architectural change like procedural C to object-oriented Java takes more effort than say C++ to Java since both share an object-oriented design.

Think about the foundational differences in the old and new system. Software usually sits on top of an infra-structure of hardware, operating system services, third-party components and code libraries. Identify core resources that change in the old and new system and how it impacts the translated software.

Think about the old and new language and its supporting environment. Programming languages have both semantic and syntactic differences. A lot of search, replace and grunt-work can usually take care of syntactic differences, but if the old system relies heavily on semantic features like language generics, automatic garbage collection, etc. that aren't available or easily emulated in the new programming environment, than design changes may be required.

Think about the feature changes needed in the new system. Don't waste time converting features that are no longer relevant or cost-justified or leave out features that are required to make the new system work in its new environment.

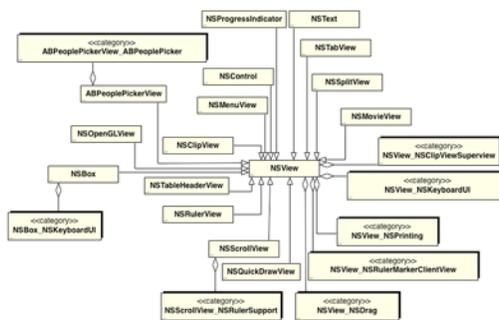
Now that you understand the scope of the project, where you are coming from and where you are going, we can discuss some tools that automate and accelerate many of the steps in the translation process.

The translation process consists of:

- Create Design of Old Code
- Modify Design for New Code
- Generate New Code Structure
- Translate Function Logic

Create Design of Old Code

The design of the old code base can be graphically represented as structure charts for procedural code like C, Basic or Pascal, UML class diagrams for object-oriented code like C++, Java or Delphi or data models for SQL database-centric code. These diagrams show the static structure of the code with objects on the diagram linked directly to the associated code fragment.



Class Diagram for Objective-C code with Attributes and Operations Suppressed

Automation Tools

WinA&D is a tool that runs on Windows and can represent the architecture and design of any software system. It supports procedural designs, object-oriented design, database-centric design, multi-threaded software, desktop applications, embedded systems, cloud based services or any mixture of the above. MacA&D runs on Mac OS X will similar capabilities.



If you already use WinA&D or MacA&D to design and generate code for your old software system, you already have and understand the current design. If not, WinTranslator on Windows or MacTranslator on Mac can be used to generate diagrams with dictionary information from your current code base.

WinTranslator scans source code files stored in a collection of folders to extract design information to a set of text files. Those files are imported into an empty WinA&D project to populate the data dictionary and automatically generate a set of graphic diagrams. This automated process can usually be done in minutes.

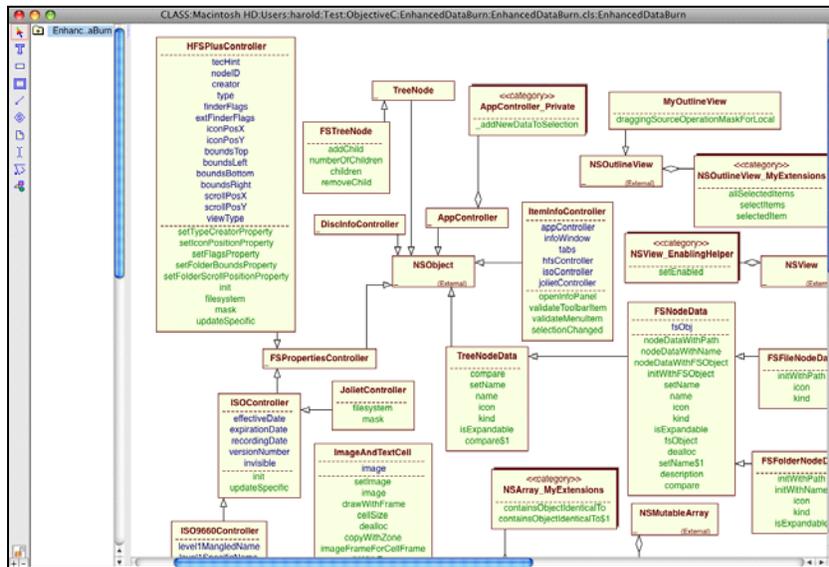


Consider a translation project with Objective-C source code being converted to Java code using MacA&D and MacTranslator on a Mac OS X computer. The concepts, tools and approach are essentially the same for other languages or when using WinA&D and WinTranslator on Windows. For example, perhaps C# code on Windows will be converted to Objective-C code on Mac.

Class Diagram

A screen shot of the Class window, Dictionary window and Code Browser window illustrates how Objective-C code is represented within the MacA&D modeling tool. A named box represents each class. The class name is red, attribute names are blue and operation names are green. Some class are shown with a normal presentation that shows all attributes and operations, while others are shown with a suppressed presentation that only shows the class name. Various relationship lines connect related classes.

A project may have one class diagram, many diagrams within one class document or hundreds of separate diagrams and documents. The UML notation includes many diagram types that are all supported by MacA&D, but for the purpose of this discussion on code translation, only the class diagram will be used.



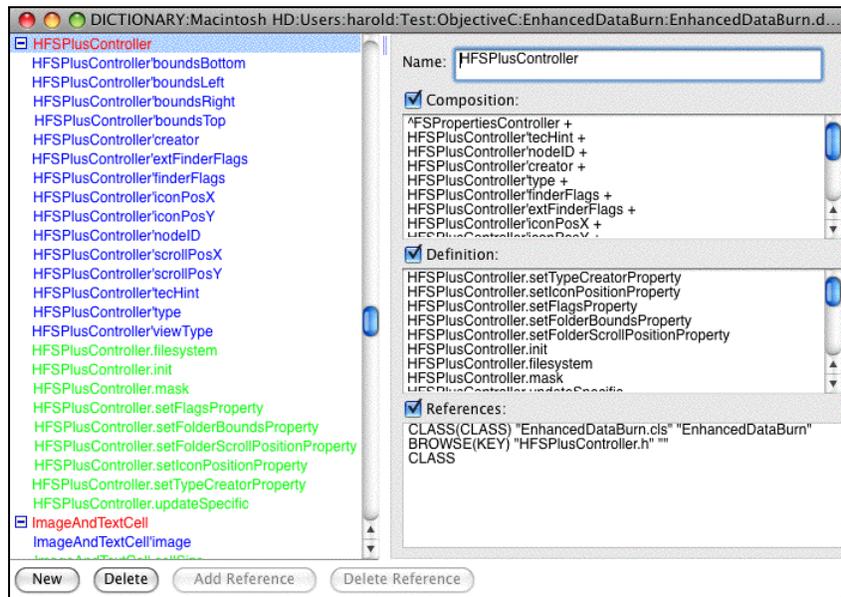
Class Diagram Created from Objective-C Source Code

Data Dictionary

Each object on a class diagram has an associated dictionary entry. In addition to a class type entry, each attribute and each operation of the class also has an associated dictionary entry.

The dictionary entry holds detailed information associated with that class. A class diagram is a visual presentation of that dictionary information. A MacA&D project may have many diagram, code, specification and other document types that are all logically linked together through one data dictionary.

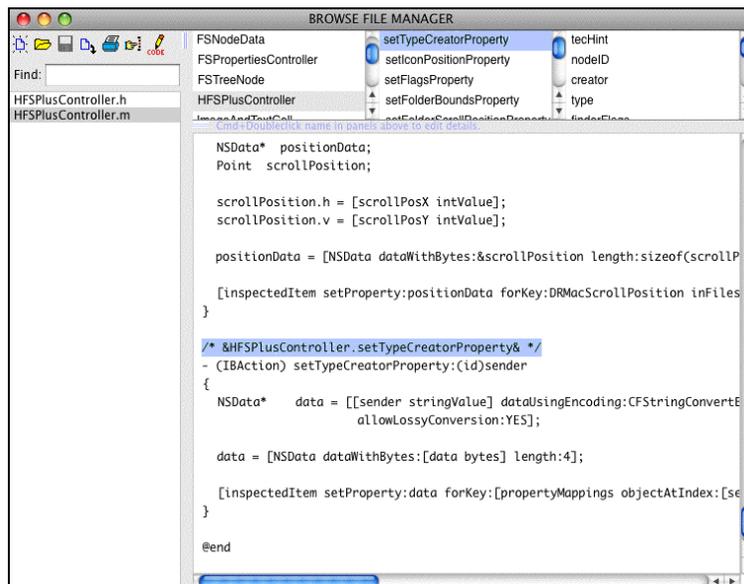
The Dictionary window shows entry names on the left and the details of a specific selected entry in the Name, Composition, Definition and References fields on the right. Red, blue and green entry names indicate entries of type class, attribute and operation, respectively.



Data Dictionary for Objective-C Project

Code Browser

Each selected object on the diagram is linked to an associated code file stored in one of many code folders. A developer can select a class object of interest, click to the Browse window to view, edit or navigate through the code files.



Code Browser Shows Objective-C Code Linked to Class Object on Diagram

By understanding the main steps required to generate graphic diagrams from code, the software designer will be able to customize the process to fit the specific needs of a project.



Use MacTranslator's Reengineer Project dialog to identify the source code folders, choose the Objective-C programming language and select the type of design data to collect. MacTranslator then scans the code files and produces one Dictionary.rp file for all the code files in each code folder. The top toolbar in MacTranslator shows the file path of each code file being processed. A Dictionary.List file is generated for the project that includes the full file path of the individual Dictionary.rp files.

Dictionary List File Format

The Dictionary.rp file is a plain text file using a simple data format to describe each dictionary entry. A software developer can view the Dictionary.rp file with a text editor to understand the kind of collected information. These files are imported into the MacA&D modeling tool to populate the Dictionary document for a project.

Remember how each class object has an entry in the dictionary for the class itself and for each of its attributes and operations. Within the Dictionary.rp text file, each entry is represented with four fields of information:

- Name
- Composition
- Definition
- References

The right-hand side of the Dictionary window shows the Name, Composition, Definition and References field for a selected entry. This directly corresponds to the simple format used by data in the Dictionary.rp file. MacTranslator generated the text fragment from the Init operation of the ApplicationController class.

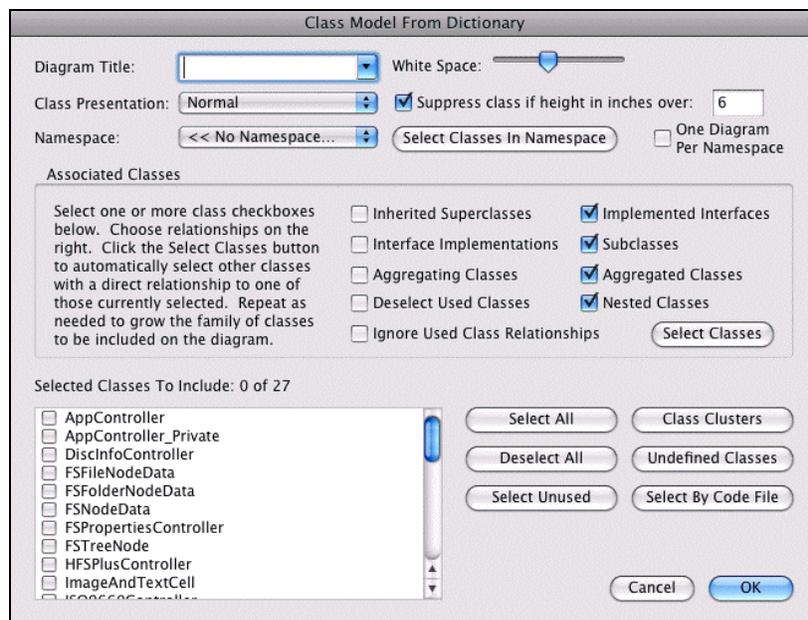
```
%NAME ApplicationController.init
%COMPOSITION
%DEFINITION
#GeneralDetails
Describe:
##
#ObjectiveCDetails
DataType: - (id)
Arguments:
Access: private
##
%REFERENCES
OPERATION
BROWSE(KEY) "AppController.m" ""
```

The Name of the dictionary entry takes the form Class.Operation and Composition field is empty. The Definition field holds a GeneralDetails section of data and an ObjectiveCDetails section of data. If this entry was generated from Java source code, that section would be named JavaDetails. Finally, notice the entry type "Operation" and code file named "AppController.m" in the References field.

Diagrams from Dictionary List

Use the New Project dialog in MacA&D to create a new project folder that holds a class diagram document and associated dictionary document. Import the Dictionary.List file into the Dictionary using the Import Dictionary command. Data from each Dictionary.rp file referenced from the Dictionary.List file is imported into the Dictionary window.

Once the Dictionary is populated, select the Class window and choose the Generate Class Model command to generate one or more class diagrams from the dictionary information.



Generate Class Diagram from Dictionary

There are many options to choose from when generating class diagrams from dictionary information including the specific classes to include on each diagram, the types of relationships to show between classes and the amount of white space between classes.

If a class has many attributes and operations, displaying it on the diagram with a normal presentation takes a lot of diagram space. Each class can also be shown with a suppressed presentation (all class members hidden) or a customized presentation (only selected class members visible).

The Document Defaults dialog for a class document has additional customization options. For example, you can choose which class members (attributes, properties, operations, events) to show in each class box, the color convention to use, and how much detail (data type, arguments, etc.) to present for each class member.

Modify Design for New Code

Class diagrams enable the designer to see the structure of the existing source code. Class objects are linked to associated source code files making it easy to explore the big picture and drill down to the specific code details.

To see the code for a selected class object, click the Code button in the tool bar. MacA&D presents the associated code file in the Browse window. The language-independent Browse window is used to browse or edit object-oriented source code.

Class objects on a diagram have associated dictionary entries with the same name as the class itself. MacA&D links selected diagram objects to associated code files using the file name referenced in the dictionary entry. MacA&D knows the source code file can be located in either the main project folder or any one of the namespace folders defined in the Document Defaults dialog of the Dictionary document. MacTranslator automatically collects the folder paths when scanning source code.

Design changes are often needed to fit the new target language or environment. Using the class model editor in MacA&D, new classes, attributes or operations can be added, modified or deleted.

MacTranslator captured Objective C details like data types, arguments, etc., however, Java specific details are needed before new Java code can be generated. Since the syntax, standard data types, etc. are different for each language, this step requires human editing. Java language specific details can be edited from the Attribute Details or Operation Details dialog by first selecting Java from the Language menu.

After editing Java code specific details like data types and function arguments select a Java class on any diagram and click the Generate Code button to generate a Java code file.



Translate Function Logic

MacTranslator has an option to simplify the language translation process by creating the Java details section of each dictionary entry from a cloned copy of the Objective-C details. From the Details dialog, select "Clone Language Details for" checkbox and select Java from the language popup menu before processing the source code.

For a Java project, MacA&D generates the class declarations with an empty function frame to hold the user written function logic code. To assist the programmer in a code translation project, MacTranslator can capture the code of each class operation into a Notes section of the class operation dictionary entry.

Within the MacA&D class model editor, the captured code can be viewed or edited within the Notes panel of the Operation Details dialog. During Java code generation, MacA&D can emit this Notes section into the body of each generated class operation.

The automated process of capturing function code with MacTranslator, editing it with MacA&D and generating it into the new Java code can save the programmer a significant amount of time. The code within each function can now be human translated from Objective-C to Java from either the MacA&D code editor or the Java development environment.

Project Summary

Translating source code from one programming language or target environment to another is normally a very labor intensive, error-prone process. Automated tools can eliminate much of the grunt work by capturing the current design and enabling the designer to modify the design to fit the new environment. The tool generates code files with the new class structure and helps the programmer translate the code for each function.