# Avoiding Pitfalls when Porting to Mac OS X

A case study of the Macromedia Authorware 7 conversion to Mac OS X from Mac OS 9.

By François Breton



**Integration New Media, Inc.**
1425 West René-Levesque Blvd.,
Suite 906
Montreal (Quebec) H3G 1T7
Canada

Tel.: +1 514 871-1333
Fax: +1 514 871 9251

www.IntegrationNewMedia.com

Over my years working with Integration New Media (INM) I have had the opportunity do a lot of cross-platform development as well as other types of consulting interventions helping clients on a variety of projects. Sometimes at INM, we develop clients' ideas from scratch while other times we help them through rough spots. Recently, however, quite a few projects have required conversions to Mac OS X® . This paper was therefore written as a basic primer for those who are faced with this task for the first time. It is supported by what we consider a good standard case study project in which INM ported Macromedia® Authorware® 7 to Mac OS X.

It is useful to point out that the work involved in such a project is often underestimated (this was not the case for Macromedia, though) and it is worth taking the time to acquire a basic working knowledge before delving into such seemingly simple projects.

## Getting Started: Carbon vs Cocoa

First let's make the statement that there are two basic paths from which to choose at the very onset of such a project.

**Option A**
Port the existing code by running it through standard and documented steps beginning with Carbon Dater.

**Option B**
Rewrite the code entirely using Cocoa® so that the architecture and all of the features become Mac OS X native.

The decision is essentially business-based. The best way to illustrate this is to put it on a continuum where cost and ease of update and future release are at each end of a vector.

To simplify the differences we can state that using Cocoa is usually considered a more complete but more expensive proposition while taking **Option A** can produce a less robust end-product but at substantially lower cost. But this is a simplistic view of the situation.

As with anything else, many shades of grey come into play in the decision. For one thing, one may apply a combination of both methods. For another (which requires a certain level of expertise in the matter) it is possible to push the Carbon Dater approach by spending calculated efforts reworking the code so that the final product will be just as robust while still keeping the costs in the lower

tier for such conversions. Because of the nature of our expertise and of the nature of the code we analyzed for the Authorware 7 conversion, we suggested and eventually did select **Option A** with Macromedia. We determined that it would provide the necessary robustness while making sure the final product would keep its planned pricing for the consumer.

This paper therefore covers porting using the first approach. In order to make the rest of this reading as easy as possible, I have broken it down into **six key** issues developers should address to avoid major pitfalls when porting to Mac OS X.

## Six Key Issues

### 1. Carbon Date

The very first step is to use Carbon Dater (developed and marketed by Apple®) to get a sense for the scale of the work involved in the conversion. This application analyzes the older OS' libraries and code for compatibility with OS X. The output is an HTML report customized for your application that lists function calls and code usage that need to be adapted in order to be compatible with Mac OS X and/or Mac OS 9®. This process is fairly straightforward but does not constitute a unique step in the preparation phase.

### 2. Analyze Code

The key point here, is to know that Carbon Dating does not provide a complete assessment of the code. In fact there are a few very important elements it cannot capture as it analyzes code. Two of which are, 68K-based executable files (they are too old for the tool) and resource based code or code that is generated at run-time.

You therefore have to perform your own analysis of the code and its structure to get a sense of how old it is (including how many Mac OS versions it has served in the past).

At this point, I strongly suggest that you discuss with your client the conversion route they wish to take. Doing this may provide you the opportunity to overhaul some aspects of the code that will reduce the complexity of the final conversion as well as cleanup the program in general.

In the case of Authorware 7, the code was mature and the architecture complicated. As with almost all long standing products, it contained code that was too old to be directly ported to OS X. To simplify the conversion to OS X

and future platforms, we recommended that the code be rendered completely consistent with OS X as opposed to simply adding OS X to the mix of previous platforms in the existing code. This overhaul produced a much more robust final product while allowing us to work faster. Also, Authorware 7 will now be much easier to port to the next versions of the Mac OS. A win-win for everyone.

### 3. Convert Code

This is a very key phase which, depending upon how old the code to be converted is, requires a somewhat intuitive but necessary approach.

First, let's state that there is not a simple direct conversion path from 68K-specific code to Mac OS X. So if you are working with 68K code the way to do this is in two stages:

1) Port from 68K-specific code to Power PC
2) Do the same thing from Power PC to OS X

Doing this allows you to take advantage of two standard and well-documented migration paths. The other option, the 68K to Mac OS X direct conversion, is currently under-documented and bears an important amount of risk we just don't recommend taking.

Also make sure you pick the right compiler so that you don't inadvertently add complexity to your work. As an example, in the case of Authorware 7 we used CodeWarrior™ instead of Apple's now-obsolete MPW®. This rendered the process fast, accurate and efficient. Another alternative would have been to use Apple's new Project Builder, but at the time of the development of this project, Project Builder couldn't handle Carbon libraries. Since the release of Panther™, it can, and would have been a suitable alternative to CodeWarrior.

### 4. Clean Code

Over time and through numerous conversions, certain files become obsolete or unnecessary. Remove them! Precious time could be saved by identifying and removing unused code before porting.

In the Authorware 7 project, INM was able to clean out over 30 files – saving valuable development time and increasing the efficiency of the software's operation. We also took this opportunity to, here again, reorganize the remaining files in such a way that it will be much easier for Macromedia to update its software in the future. Some are truly simple things such as putting all the Xtras into one single directory.

### 5. Re-write Shared Memory Code

While previous versions of Mac OS made use of shared memory, Mac OS X now supports a direct interprocess communication strategy. It is substantially different from the previous OS memory management scheme in that it doesn't use a shared memory buffer anymore but instead it communicates directly with the OS X operating system. This new approach requires some adaptation by the developer but in the end, it is a much more efficient way to work. This is what we did when we overhauled the scripts in Authorware 7 that enable the Netscape® plug-in.

**Note:**
If the application you are porting calls upon shared memory, there is an high risk of memory loss once ported to OS X, if you haven't converted it first.

### 6. Capitalize on OS X

As is usually the case when new platforms are released, new OS features become available that can replace some of the code you are trying to port. Often times, although not always, these newly built-in features constitute a better alternative and allow developers to simplify and solidify parts of their program.

OS X is not an exception to this rule. In fact, many of the new built-in functionalities have helped us to substantially streamline development time for our clients. Here again, the Authorware project is a good example of how this can be useful.

After assessing the existing code, we found that the authoring tool had its own imaging capabilities. We evaluated the overall situation and concluded that instead of updating and porting this section of the code, the final product would be better served if we simply used the QuickTime® imaging utility. Since Mac OS version 7.1.1, QuickTime has been part of the standard OS installation, and is defacto on virtually all end-users systems. The decision to use QuickTime meant Authorware 7 would be able to take advantage of the most recent and proven imaging technology for the Mac with minimal porting and development.

## Conclusion

As you can see, porting is not the straightforward process it may initially appear. The operative word here is structure: structure in the code but also, in your approach. If you are careful in your analysis and you are rigorous in your treatment of the code with a strong focus on re-organizing and only re-using when appropriate, major pitfalls will be avoided. Experienced developers who will do this for the first time mainly have to become familiar with the new built-in features of OS X to accomplish this task.

However, if you're dealing with a time sensitive project that may have code as old as 68K, you're probably better off using experienced outside help because hitting rough spots is almost inevitable, especially the first few times you port. It's the type of project that can quickly become a case of a bunch of details that creep-up and grind you to a halt.

*François Breton has a B.Sc. in Computer Science and over 7 years experience in the industry. As head R&D project manager for Integration New Media (INM) in Montreal, Canada, François played a key role as INM pioneered the development of cross-platform software components for multimedia applications.*

## For more information:

**On Macromedia Authorware 7**
visit: http://www.macromedia.com/software/authorware/

**On Apple Carbon Dater**
visit: http://developer.apple.com/carbon/dater.html

**On Mac OS X**
visit: http://www.apple.com/macosx/

**On INM**
visit http://www.IntegrationNewMedia.com/