

Using a Decision Tree to Navigate Complex APIs

by Colin Gray - May 26, 2005

First published on Artima Developer.

Reprinted with permission.

Summary

As software products become more complex, developers must invest increasing amounts of time to learn about new API and product capabilities. Traditional documentation tools were not designed to help navigate that increasingly complex landscape. This article reports on an interactive documentation navigator developed by Business Objects, S.A., for their Crystal Reports and enterprise reporting-related products. The documentation navigator tool not only presents API components and describes how to use classes and methods of an API, but also guides a developer through the decisions of what API elements to use to accomplish a given task.

In his autobiographical travelogue *Wind, Sand and Stars* [see [Resources](#)], Antoine de Saint-Exupéry sums up the beauty of airplane design: "In design, you have reached perfection not when there is nothing more to add, but when there is nothing left to take away." An aviation pioneer and globetrotter long before the jet age, he extols us that "He who would travel happily, travels light."

As developers, we can easily feel with Saint-Exupéry: While he had to find his way in the complexity of a cockpit, our job is to navigate around increasingly complex APIs to get our job done. We would all rather travel light, but we also must deploy complex software solutions to help our organizations stay competitive. Those solutions, in turn, require that we make use of components and APIs that present increasingly more features. With more features, we have more packages, classes, methods, and configuration options to keep up with.

Increased complexity confronts us not only during development time, but also in the process of evaluating new APIs and software products. It used to be that we could simply download a single package from a vendor's Web site, install it, fire up some examples, have a look at the APIs—and pretty much be done with the evaluation even before our cup of coffee grows cold.

That is no longer the case. Just a cursory look at the latest J2SE, J2EE, or .NET platform APIs can put us at unease. Evaluating such products and APIs might take days or even weeks of trying out different features and examining different API subsets. That evaluation requires increasingly significant investment from both potential users and software vendors, be they commercial outfits or open-

source projects. The voluminous documentation of open-source projects, such as Hibernate, Apache Jakarta, or NetBeans illustrates that taming the complexity beast is no privilege reserved for for-profit software vendors.

One solution to the problem of evaluating and learning about increasingly complex APIs comes—not surprisingly—in the form of software. While traditional documentation tools are still lagging in helping to cope with complexity, a few efforts under way aim to produce the kinds of interactive documentation that can ease someone into an APIs complex innards. This article provides a brief review of a few approaches, including an interactive web application that uses a decision tree to help users navigate complex APIs.

Taming complexity in Java APIs

A very simple example of such an effort provides a graphical overview of the hugely complex J2SE API. While it's easy to mistake it for cute marketing graphics, the J2SE Platform Overview is actually an image map that gives a birds-eye view of different parts of the J2SE 1.5 API [see [Resources](#)]. Clicking on any area of this image takes you to a Web page for the appropriate API. Each of those API areas provides a brief overview, and then gives URLs to the relevant Javadoc documentation. While simple, the image map-based documentation overview helps a developer understand what's available in a complex API bundle.

Still, just knowing what's available does not immediately help a developer understand how to use an API's classes and methods. Explaining that detail has traditionally been the task of a documentation tool, such as Javadoc [see [Resources](#)]. While Javadoc has evolved over the years, its basic paradigm of generating documentation from source code hasn't changed. With an API as large as the JDK 1.5 APIs, simply presenting classes and methods in an HTML page becomes limiting: There are just too many classes and methods. It's similar to the classic problem of not being able to see the forest for the trees. Although you can see individual trees clearly, by reading the documentation of individual classes, there are so many of them that it is difficult to get the big picture.

The sheer number of HTML pages that make up the J2SE 1.5 Javadoc presents the same problem search engines try to solve. An obvious next step for Javadoc would be to provide a search capability. That capability is already a part of JavaHelp, and it might serve developers as well as it serves an application's end-users, JavaHelp's typical audience [see [Resources](#)].

A few open source projects are already addressing this need, such as the very simple Javadoc- Search tool hosted on SourceForge [see [Resources](#)]. A more interesting project is run by the JavaLobby folks at jdocs.com [see [Resources](#)]. They have compiled a list of so far 132 Java APIs, and provide textual searching

across those JavaDoc files. In addition, they also provide a way for developers to post and read comments about those APIs.

While the Jdocs site brings us one step closer to showing how to use certain APIs, it still does not help us decide what API elements to use to solve a given problem. That guidance is traditionally reserved for API tutorials that do not normally form part of a traditional API documentation set. Moreover, tutorials are by very nature selective in what aspects of an API they cover.

Another approach to guiding a developer through a complex API was recently implemented by Business Objects, S.A., the maker of Crystal Reports. Business Objects has seen its product offerings, which include APIs, grow in number and complexity over the years. To help developers decide which of its API elements to use to solve a given problem, Business Objects recently created an interactive *object model decision tree* web application [see [Resources](#)]. This application guides developers through Business Object's APIs using DHTML and JavaScript, and is available on Business Objects' Developer Zone website.

Using a Decision Tree to Navigate Complex APIs

by Colin Gray
May 26, 2005

An interactive object model decision tree

The [interactive decision tree application](#) walks developers through a series of questions related to application requirements. It asks a developer the level of reporting functionality required in an application. Each question guides the developer to a new level within the object model hierarchy.

As a developer works her way down the decision tree, each API box in the path represents an API that should be considered for the desired functionality. Color-coding, consisting of dark and light blue, indicates the products that include the referenced API. [For the differences between Business Objects products and APIs, see the [sidebar](#)].

The user interface is dynamic: If a user clicks on an API or question box, the right panel shows related instructions and information. The right panel also contains an access point for detailed navigation of the API. Each object in the API is also dynamic—a single click reveals related help on each object.

For example, say you are responsible for a web application and you have a new requirement to allow users of your web app to export documents to different

formats, such as Adobe PDF or Microsoft Word. You are looking around for tools to help you implement that enhancement, and are completely unfamiliar with Crystal Reports' API. How would you figure out what classes, properties, and methods in Crystal Report's API offer the report exporting functionality?

Using the interactive object model decision tree, you first answer a few simple questions, then select documentation subsets that become visible as you navigate the decision tree. Figure 1 provides a thumbnail view of the decision tree user interface.

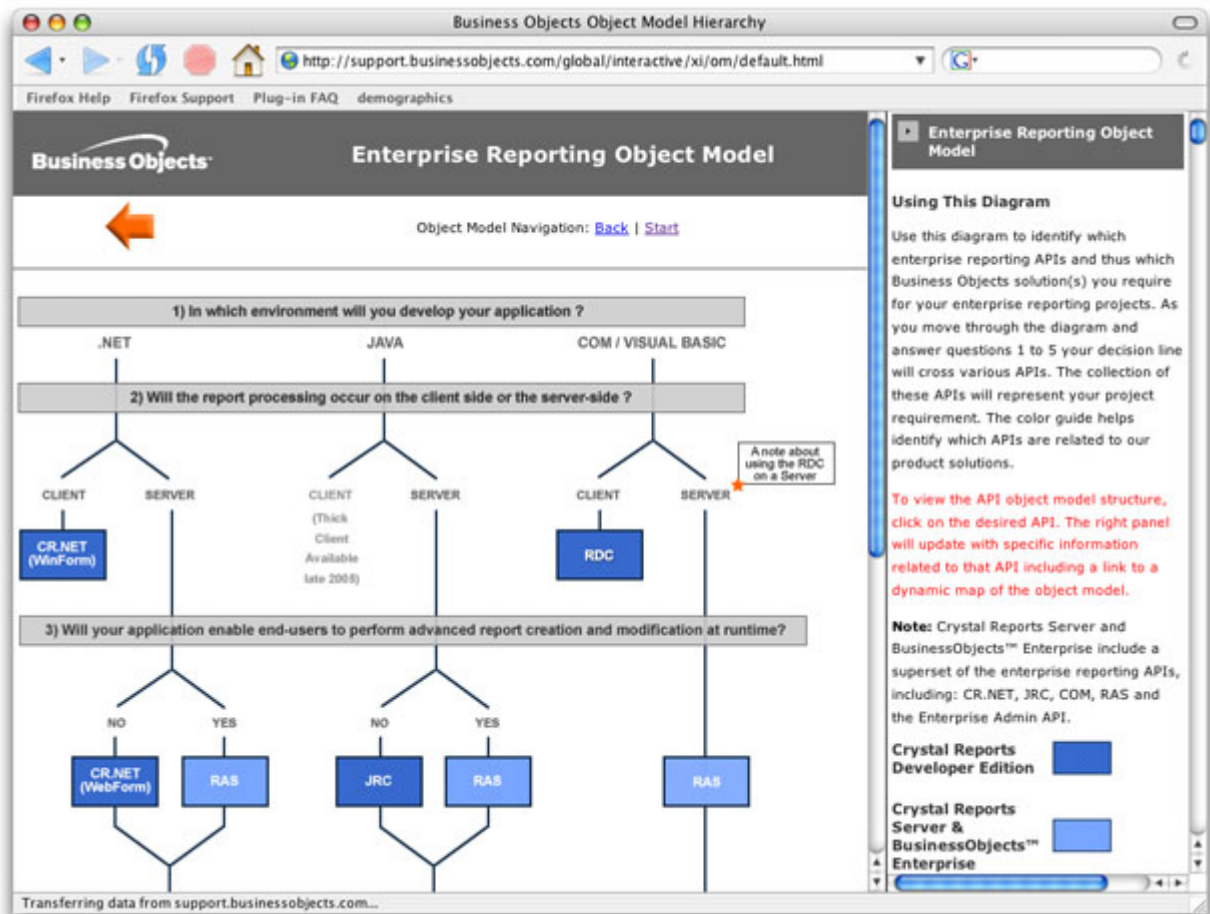


Figure 1. The Crystal Reports object model decision tree.

Finding the report exporting API is just a few answers away:

1. *In which environment will you develop your application?*

The options here are Java, .NET, and COM/VB. For this example, say you follow the .NET path.

2. *Will the report processing occur on the client side or the server- side?*

Since you're building a web app, the processing will occur on the server side. You follow the "SERVER" branch.

3. *Will your application enable end-users to perform advanced report creation and modification at runtime?*

You're not sure what "advanced" means, but you suspect you want to provide advanced report creation, so you choose the "Y" branch. (When in doubt, you can click on the question's text box to display a detailed explanation of the question.)

4. *Will you be incorporating reporting services such as report security, scheduling, and management into your application?*

For this app, you don't provide such features. Since your choice is NO, you've come to the end of the decision tree.

As mentioned previously, each blue box that you traverse as you navigate the decision tree is an API you should consider. In this example, the YES branch in question 3 took you to a box labeled "RAS," and the NO branch in question 4 took you to a box labeled CR.NET. You can now start exploring these two APIs to see if they will help you accomplish your goals.

If you click on the RAS box, you'll learn in the right hand frame that the RAS API provides access to the .RPT report object model. Following a brief overview of the RAS API in that frame, you'll see a thumbnail view of the RAS API diagram, as shown in figure 2.

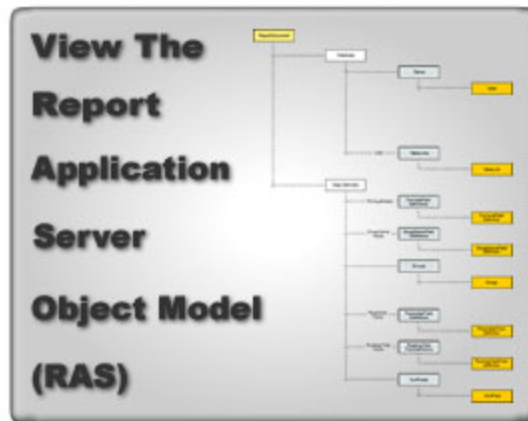


Figure 2. Thumbnail of RAS API diagram .

Clicking on that thumbnail now replaces the main screen section with detailed information about the RAS API and SDK.

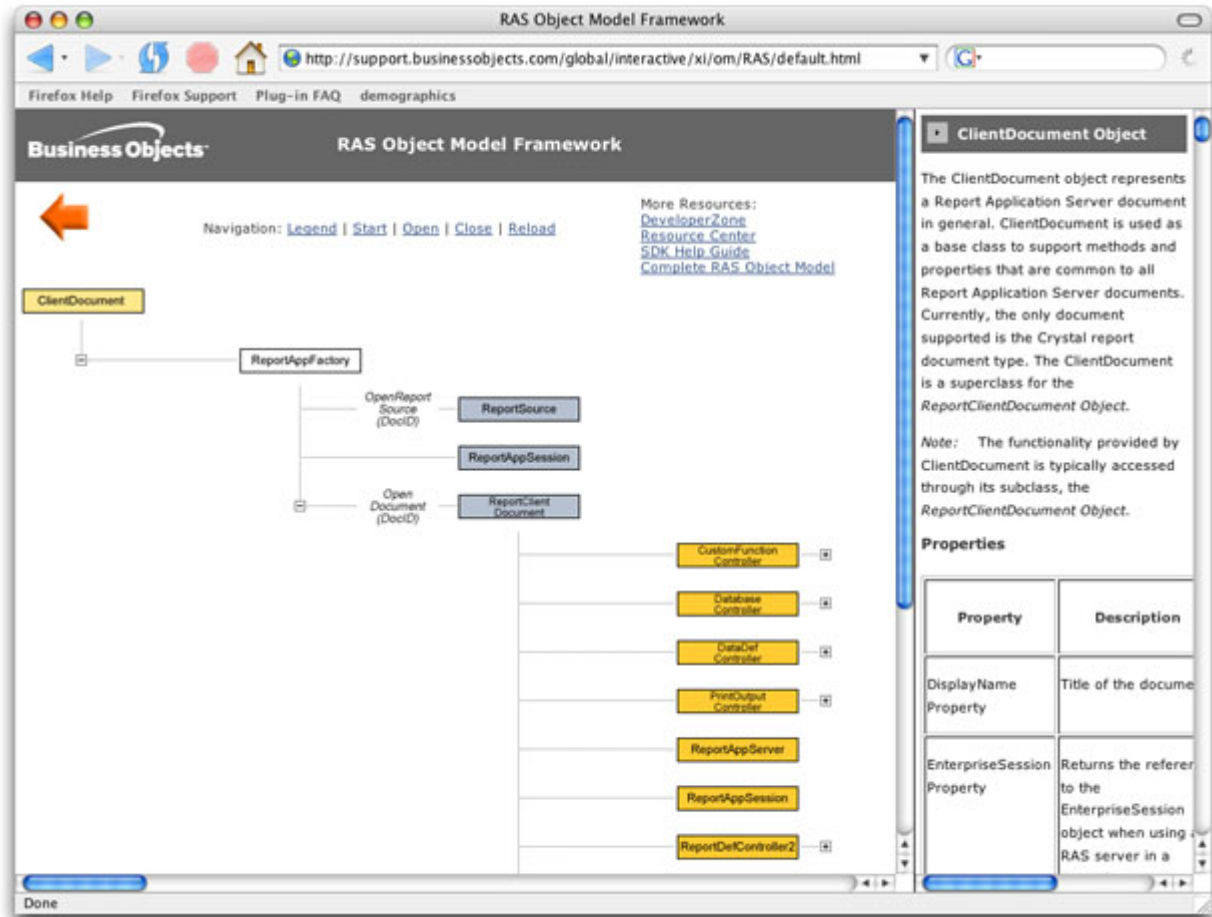


Figure 3. A detailed view the RAS API.

The object model diagram itself is dynamic HTML that presents a collapsible tree. Clicking on the Open label expands the tree's first-level branches. With the top-level objects in full view, the **PrintOutputController** object stands out as a possible candidate to control report exporting. Clicking on a box representing that object reveals the object's further details, including that "PrintOutputController is used to export reports to a specific format, such as Excel, RTF, or PDF and to modify various formatting options." The detailed object view also shows the object's method, including **Export ()** that performs report exporting.

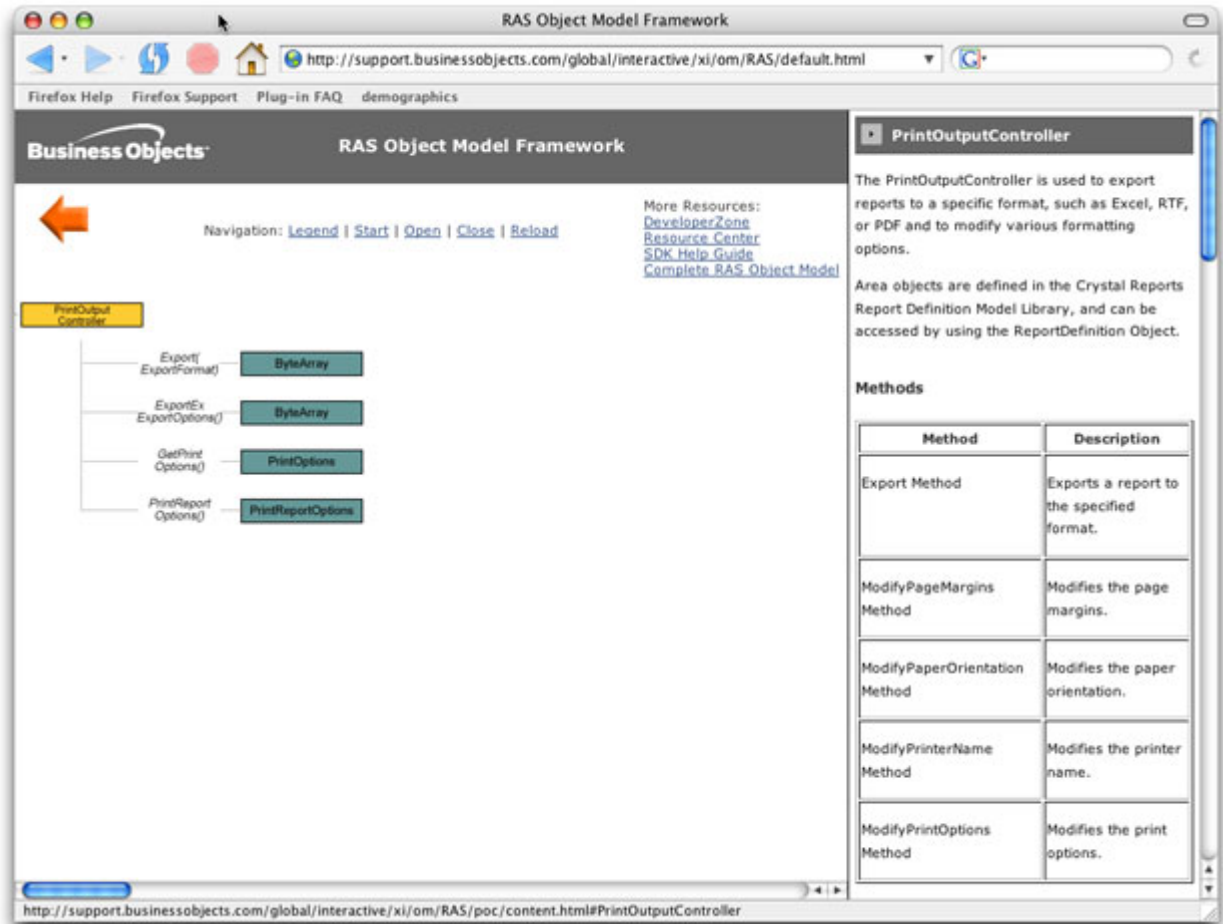


Figure 4. Detailed view of the `PrintOutputController` API.

Summary

Organizations are finding it increasingly necessary to deploy more complex software solutions just to stay competitive. More capable software presents new complexity to be tamed. That increasing complexity challenges developers, marketers, and users alike. Developers must struggle to keep things as simple as possible, to keep the code understandable and manageable. Marketers must figure out how to explain complex products to potential customers. Users must invest effort to determine which complex product best fits their needs.

Documentation tools must become more interactive to help developers tame the burden that comes with increased choices. Recent efforts to make JavaDoc more developer-friendly, and the dynamic HTML object model developed by Business Objects is an example of such interactive documentation. While traveling light is not always an option, such interactive documentation tools can make a developer's life happier.

The Crystal Reports interactive object model decision tree had, at the time of this writing, live on the Business Objects Developer Zone for two months. Sixty percent of the developers who visited the Developer Zone over the first month viewed the decision tree. Based on a poll conducted with over 700 users of the decision tree, almost 70% indicated that the tool had either provided guidance related to API use with their reporting projects, or that they expect it will do so in their future projects.

Resources

To access the live decision tree application please visit:

<http://support.businessobjects.com/global/interactive/xi/om/default.html>

To access the Crystal Reports Developer Zone visit:

http://www.businessobjects.com/products/dev_zone/default.asp

Writing JavaDoc comments:

<http://java.sun.com/j2se/javadoc/writingdoccomments/>

An open-source JavaDoc search tool:

<http://sourceforge.net/projects/javadoc-search/>

JavaLobby's collection of annotated, searchable JavaDocs:

<http://www.jdocs.com>

Wind, Sand, and Stars, by Antoine de Saint-Exupéry

<http://www.amazon.com/exec/obidos/ASIN/0151970874/>

About the author

Colin Gray is the America's developer marketing manager for Business Objects. Colin has been in technical sales, product management and developer marketing roles with Business Objects (Crystal Decisions pre-acquisition) for 16 years. Over this tenure one of the most daunting communication hurdles has been helping developers map their project requirements to the solutions that Business Objects offers. This goal gave birth to The Interactive Object Model Tree.

Sidebar: Crystal choices

Crystal Reports is a banded-style report writer for accessing and formatting data into actionable information. Over the years, the Crystal Reports solution has become more complex. Users can choose to design reports within one of three editions of Crystal Reports, and manage those reports with one of two deployment options. Crystal reports customers are developers and business

managers that need to understand how the various Crystal Reports reporting options relate to their existing requirements.

The first decision for a prospective user is to choose the right Crystal Reports edition. While each edition shares the same report designer, editions vary in their abilities to connect to data and how they integrate reports into applications.

The most basic edition is the Standard edition that allows connecting to data from PC databases, such as MS Access, Pervasive, or Paradox. If a Crystal Reports user is interested in accessing SQL-level data as well, she would select the Professional edition. Finally, a Developer edition is available for users interested in integrating reports into applications, and is the only one to include such integration APIs.

For out-of-the-box management and secure delivery of reports to departments or entire organizations, Business Objects offers two additional products that act as a framework around the report files. Both are built on the same enterprise reporting platform. The small business, or "light," edition is called Crystal Reports Server, and the enterprise edition is named BusinessObjects Enterprise.

Developers wanting to integrate reporting into their applications can use various Crystal Reports APIs. APIs exist for Java, .NET, VB platforms. Some of these APIs allow report integration without having to write reporting code. For example, a developer could embed one of three report component engines into an application, and use the related APIs to pass that engine instructions at runtime, instead of having to code the report processing logic of an application.

If a fuller reporting solution is desired—with a shared portal for report access, security and report object management, for instance—the enterprise reporting services of Crystal Reports Server or BusinessObjects Enterprise can be employed. Embedding the enterprise reporting services via an API means that those more complex reporting services need not be created via code, but instead can be integrated with the core application and accessed via the API.

Business Objects owns the following U.S. patents, which may cover products that are offered and licensed by Business Objects: 5,555,403; 6,247,008 B1; 6,578,027 B2; 6,490,593; and 6,289,352. Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Enterprise, Crystal Analysis, WebIntelligence, RapidMarts, and BusinessQuery are trademarks or registered trademarks of Business Objects SA or its affiliated companies in the United States and other countries. All other names mentioned herein may be trademarks of their respective owners. Copyright © 2005 Business Objects. All rights reserved.