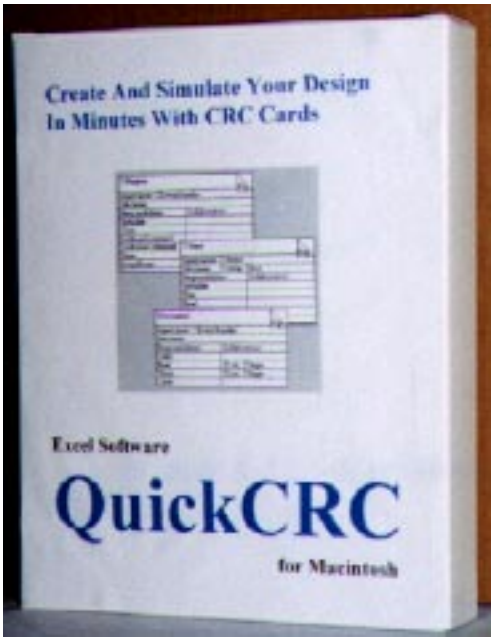
QuickCRC for Macintosh



Introduction

QuickCRC is a software design tool for object-oriented development projects. Its intuitive interface style and support for responsibility driven design using simple CRC cards makes it very quick and easy to apply to real projects without formal method or tool training. QuickCRC runs on Macintosh, Solaris, HP-UX, Windows 95 and Windows NT and design documents are binary compatible between computers.

QuickCRC design documents can be exported to or imported from a dictionary text file. This allows a designer to automatically move QuickCRC designs to MacA&D or WinA&D for detailed design and code generation or use MacTranslator or WinTranslator to automatically generate QuickCRC cards from existing source code.

QuickCRC is used to discover objects and related properties in the early phase of an object-oriented development project. A CRC card as shown below, represents an object class and its properties. Throughout this tutorial the terms card, class and object are often used interchangeably although technically they are not exactly the same thing. A class is a type from which an object is instantiated and a card is a collection of information about a class.

TWindow	Flip
Superclasses: TEventHandler	
Subclasses:	
Responsibilities:	Collaborators:
Initialize	
Close	
DoMenuCommand	
DoMouseCommand	TWindow
Draw	TWindow, TPalette, TS
SetupMenus	

Front Side of CRC Card

The card named TWindow for the class it represents, currently has one superclass named TEventHandler and no subclasses. This class has several responsibilities which represent the functions the object must perform for itself and to service the requests of other objects. This card must collaborate with other cards to accomplish some of its responsibilities, for example, TWindow's Draw responsibility collaborates with TWindow, TPalette and etc.

The back side of a CRC card includes its description and a list of its attributes representing the data the class must keep track of. CRC cards are created by clicking on a diagram with the Card tool and typing data into the Property dialog presented. Information on a card can easily be changed by double-clicking it with the Selection arrow to access its Property dialog. Click the flip tab to make the front or back side of the card visible.

TWindow	Flip
Description:	
this class implements a window through which a d	
Attributes:	
fDocument	
fPalette	

B*ack Side of CRC Card*

To discover the data each class knows about and the responsibilities it must perform, scenarios are created. Create a scenario by clicking on the diagram with the Scenario tool and typing information into a Property dialog. A scenario is a group of steps outlining the interaction between a group of classes to implement a mechanism in the design.

For example, the Read Document scenario is shown below. TDocument is a class that holds data about a list of shapes read from disk into memory. This scenario has three steps starting with any client class calling the Read responsibility of TDocument. TDocument then uses a sequence of the Initialize and Read responsibilities of TShape to create shape objects and read in their data from disk.

Read Document		
read document's data from disk into memory		
Client:	Server:	Responsibility:
AnyClient	TDocument	Read
TDocument	TShape	Initialize
TDocument	TShape	Read
TDocument	TList	Insert

Scenario

During the early design process, developers usually focus on the normal interactions between objects when creating scenarios. Special situations often arise due to error conditions that must be handled, but usually these can be deferred until detailed design. If an error condition is significant to the overall design, it can be dealt with as a separate scenario.

Scenarios can also refer to other scenarios. For example, the Open Document scenario shown below uses two other scenarios, Initialize Document and Read Document (illustrated above).

Open Document		
open an existing document by creating it and reading it		
Client:	Server:	Responsibility:
TApplication	TDocument	Open
TApplication	TDocument	Initialize Document
TApplication	TDocument	Read Document

Scenario Using Other Scenarios

Designing an object-oriented system using cards and scenarios is an iterative refinement process. To exercise the part of your design expressed by a group of cards and scenarios, select a scenario and click the Simulate button.

The best way to learn QuickCRC and start using CRC cards on your project is to step through an illustrated tutorial. Refer to the SimpApp tutorial below.

SimpApp Tutorial

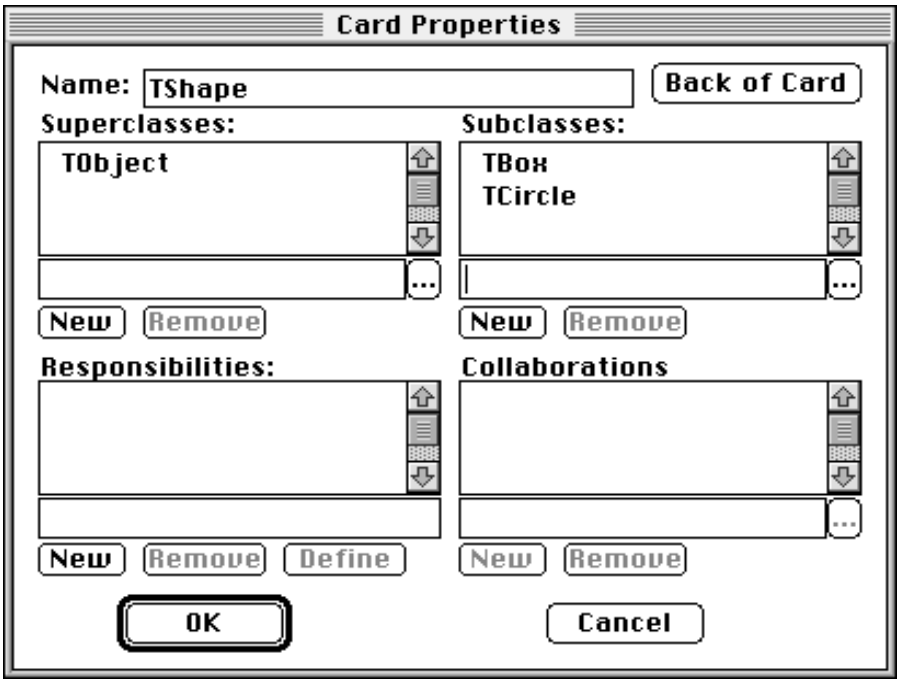
In this tutorial, the user will model a software application called SimpApp. SimpApp allows the user to edit a diagram document using a tool palette for drawing box and circle shapes. The user will create classes, establish relationships, assign responsibilities and attributes, define and simulate scenarios, verify the model, arrange and list cards and scenarios and finally export design information to MacA&D.

1. Create CRC Document

A CRC project is usually performed by an individual or small, core group of designers during the early phase of an object-oriented development project. Double-click the QuickCRC icon to start the application.

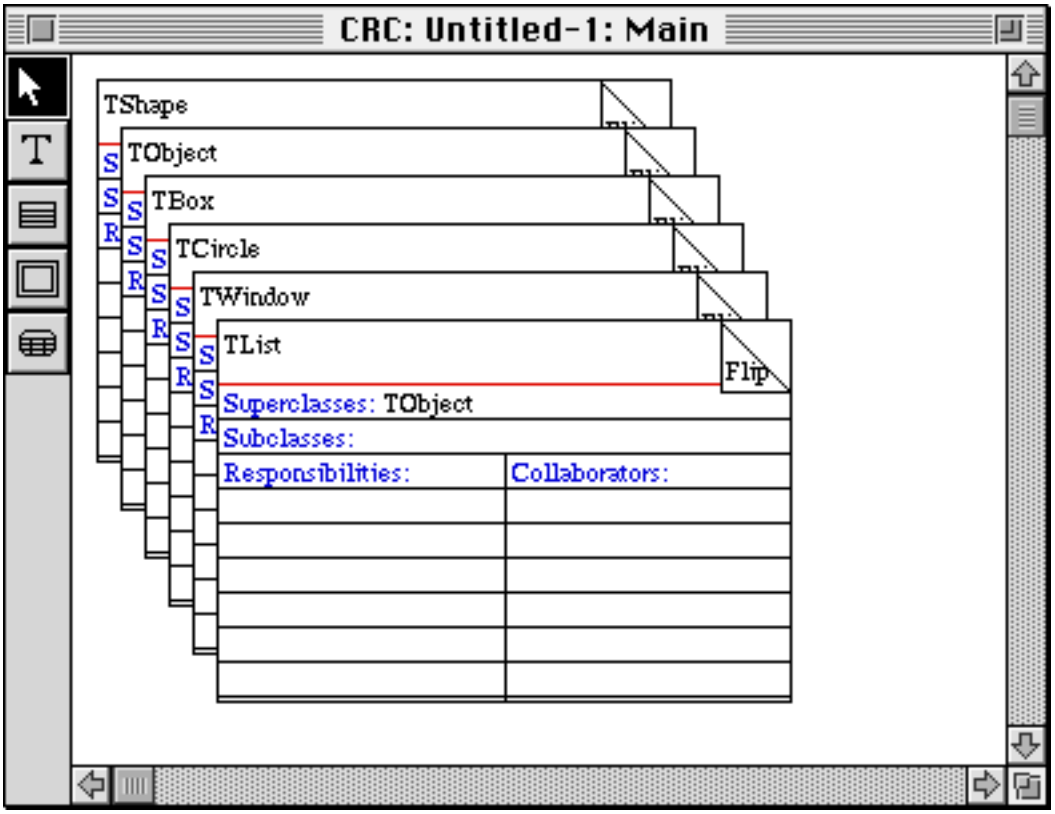
From the File menu, choose the New->CRC command. A new unnamed document is created with one empty diagram titled Main. Notice that the tool palette at the left side of the window contains a Selection Arrow, Caption tool, Card tool, Subdiagram tool and Scenario tool.

Pick the Card tool and click the diagram to define the TShape class. Fill in the dialog as show here to indicate that TShape has a superclass called TObject and two subclasses, TBox and TCircle. To add an item to a listbox, type its name and then click the New button below the Edit field to add it or press the Enter key (not the Return key).



Card Property Dialog

Now click the OK button to dismiss the Property dialog. Each class we referenced that doesn't exist yet will be added and the corresponding superclass and subclass relationships filled in. Use the Card tool to define two more classes TList and TWindow each derived from the superclass TObject. The arrangement of your cards may appear somewhat different. We will discuss later how QuickCRC can arrange cards for you to highlight various relationships in the design.



CRC Window

Obvious relationships can be added at the beginning of a card session. However, if designers are uncertain as to whether or not a relationship exists, it is better to keep classes separate, and see if a superclass or collaboration relationship arises out of the project scenarios. Assuming a relationship too early may force a particular decision and bias the distribution of responsibilities. The emphasis and strength of the CRC Card technique and responsibility design in general lies in deriving the behavior of the class and not the structure.

2. Assign Responsibilities

Once a set of classes are defined, behaviors can be assigned that will provide the functions of the application. Responsibilities that are derived from the requirements or that are obvious from the name of the class can be listed before any scenario execution commences. When in doubt, only add new responsibilities to a class when a design scenario dictates its need, otherwise cards will contain responsibilities that are unnecessary to solve the problems at hand.

 Using the Selection Arrow, locate and double-click on the TShape card. You may need to move other cards out of the way to see it, by positioning over a card, pressing the mouse button and dragging. To show this card on top of others in the diagram, select it and choose command Object->Bring To Front from the Edit menu. This class and its subclasses have some fairly obvious responsibilities like Initialize to create it, Free to dispose of its memory, Read to load its data from disk into memory, Write to save it to disk and Draw to illustrate it on the diagram. Type each responsibility into the Edit field and press the Enter key to add it. Other responsibilities may be discovered later when we create scenarios to work through the mechanisms in our design.

Once responsibilities have been added to the list, you can define a short description of each. Select the responsibility name in the list and click the Define button or simply double-click the name. Use the Definition dialog that is presented to rename or change the description of each responsibility as indicated below. Notice that some responsibilities of TShape are just virtual placeholders for the actual responsibilities to be added later by subclasses TBox and TCircle.

```
Initialize - virtual function allocates shape memory

Free - release memory allocated to shape

Read - virtual function to read shape

Write - virtual function to write shape

Draw - virtual function to draw shape
```

3. Add Attributes

Attributes of classes may also be identified early in a CRC session. Often, nouns that are not classes but rather characteristics of classes are best represented as attributes. Attributes can be assigned to classes as they are discovered, but should be done in moderation and only when it becomes apparent that the class must know that information.

Use the Card Property dialog to add attributes fPosition, fType and fSelected. This information is added to the back side of the card using the Back of Card button in the dialog. You can also add descriptions for each attribute or rename it by double-clicking its name in the attribute list to access the Definition dialog. Alternatively, you can just type into the Description field at the right of a selected attribute.

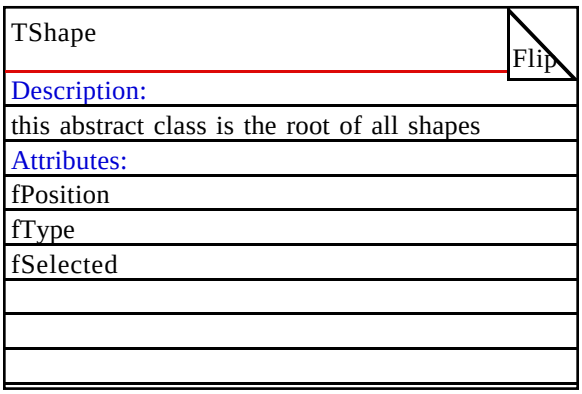
```
fPosition - center point of shape

fType - identifies shape as BoxType or CircleType

fSelected - boolean set true if shape selected
```

The Description field at the top of the back side of a card can be used to describe the card itself. Click the Back of Card button in the card's Property dialog. Enter "this abstract class is the root of all shapes" into the field. Click the Front of Card button, then click OK button to dismiss the Property dialog.

Using the Selection Arrow you can alternate between showing the front or back side of a card on the diagram by clicking on the Flip tab.



Back Side of TShape Card

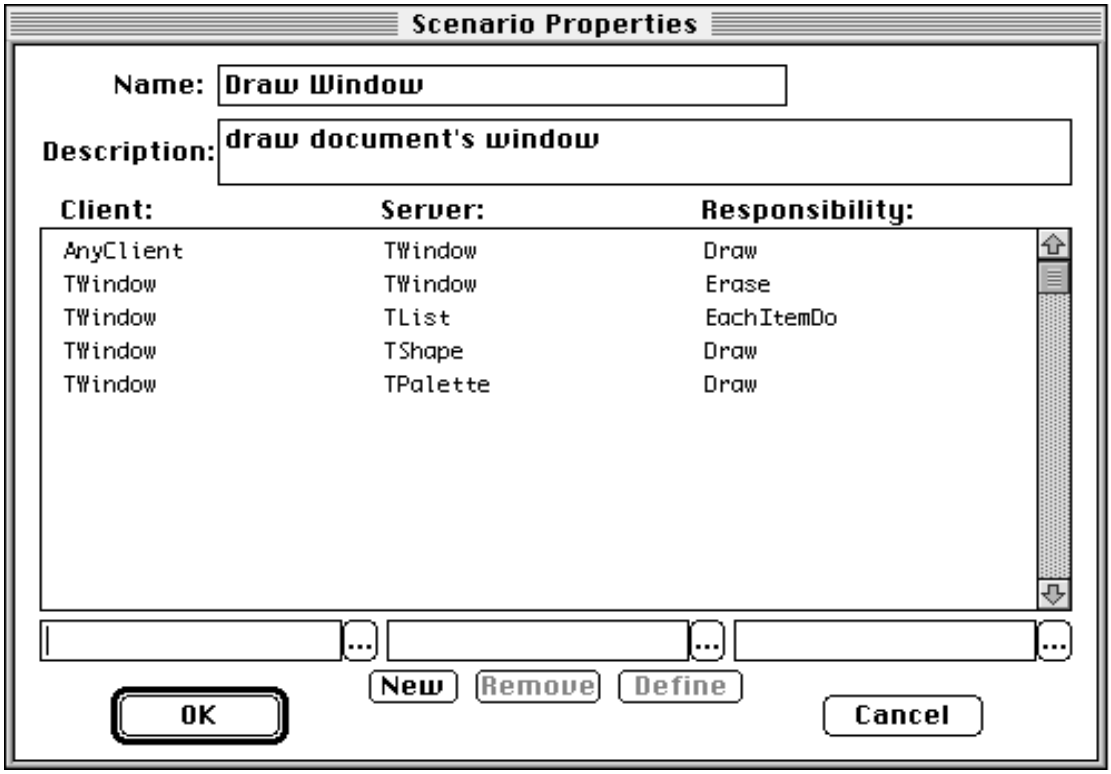
4. Define a Scenario

A scenario describes a sequence of steps to define a mechanism in the design using the responsibilities of a group of collaborating classes. We assign responsibilities to classes by simulating how the system responds to external events.

Scenarios are detailed examples of the functions in a system, where each function refers to visible, testable behavior. The scenario describes what happens in the system from a high-level, user point of view. The goal of walking through scenarios is to discover new required classes, responsibilities and collaborations or to locate existing items that are now redundant.

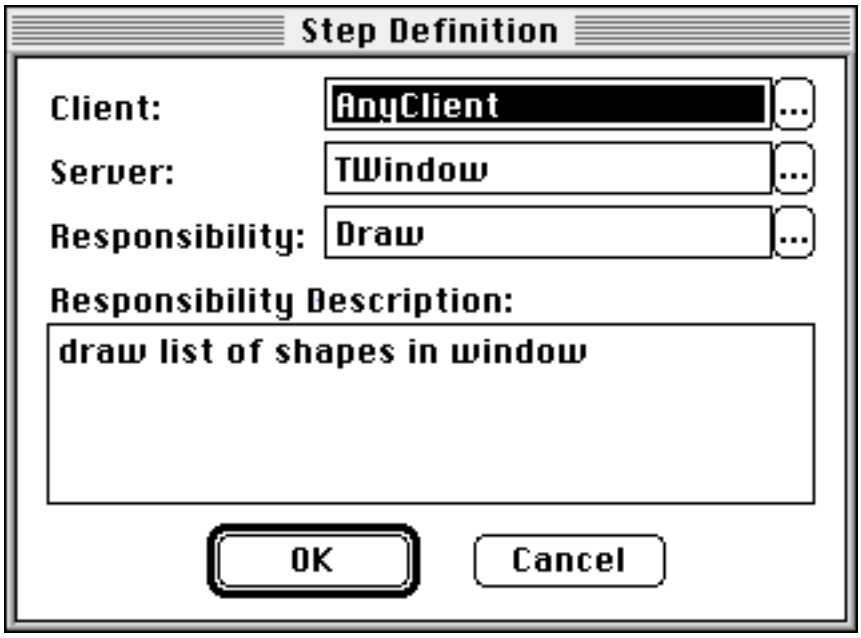
 With the Scenario tool, click to create the Draw Window scenario.
Name this scenario Draw Window and describe it by typing “draw document’s window” into the Edit field. The TWindow class draws itself by looping through a list of shapes in the document drawing each one and then drawing the tool palette.

Using the Edit fields at the bottom of the Scenario Property dialog you can add a step to the scenario by selecting the Client, Server and Responsibility names with the Pick List button at the right of each edit field. After all the fields have been filled in, click the New button to add that step. For example, select AnyClient to indicate that we are not sure yet which class will initiate this scenario. Select TWindow as the server class. The Responsibility’s Pick List dialog will show all the responsibilities of the server class TWindow, but since none are defined yet, just type the name Draw into the Edit field. Click the New button to add this step to the scenario.



Scenario Property Dialog

If you make a mistake or need to change any step, double-click on that line in the list of steps to access the Step Definition dialog shown below for the first step. This dialog also allows you to describe any new responsibilities you’ve created.



Step Definition Dialog

Add the remaining steps to the scenario as indicated in the Property dialog above. The new responsibilities Draw and Erase are added and defined for TWindow.

```
Draw - draw list of shapes in window

Erase - clear contents of the window before drawing it
```

The new responsibility EachItemDo is added for TList.

```
EachItemDo - this boolean function is true if another item in
list and returns pointer to item.
```

The existing Draw responsibility is selected for TShape.

The server class TPalette and its Draw responsibility don't exist yet in our design so just type the name in now. Once the step is added, use the Step Definition dialog to describe it "draw the tool palette". Your scenario is now done so click the OK button. Cards are added or updated for you to reflect the editing changes you've made.

Draw Window		
draw document's window		
Client:	Server:	Responsibility:
AnyClient	TWindow	Draw
TWindow	TWindow	Erase
TWindow	TList	EachItemDo
TWindow	TShape	Draw
TWindow	TPalette	Draw

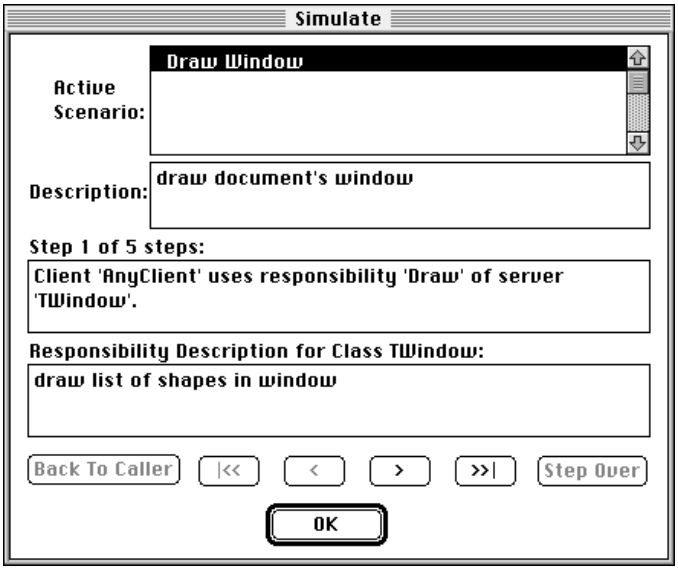
Scenario Object on Diagram

5. Simulate the Scenario

We have already stepped through a scenario and made some decisions. However, the process will need to be repeated at later stages, to confirm that the design still works as it evolves and more detail is added.


Select the Draw Window scenario and click the Simulate button. The Simulate dialog is presented allowing you to step through the scenario and get a clear picture of what actions take place at each step. Later in this tutorial we will explore a more complex example where one scenario can use other scenarios to complete its job.

Click the small arrow buttons at the bottom of the dialog to step to the beginning, step backwards, step forwards or step to the end of a scenario.



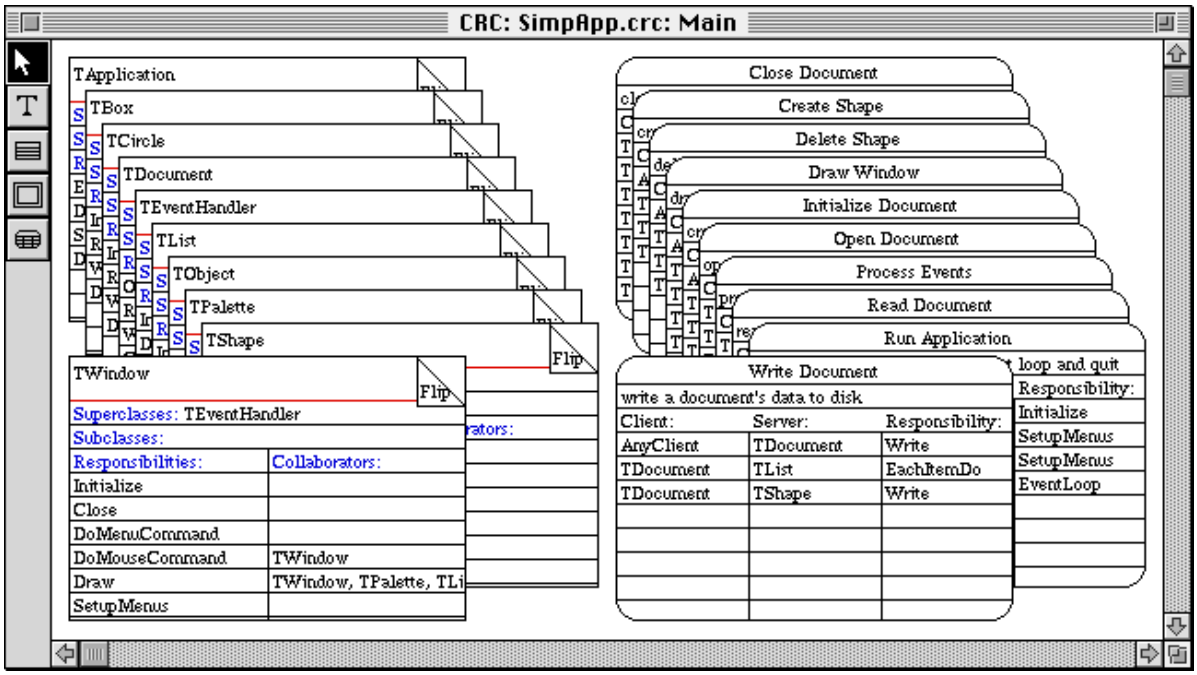
Simulate Dialog

6. Complete the Design

You've now created the basic components of a QuickCRC design, its cards and scenarios. We will now explore some of the other capabilities of the tool using a more complete design for SimpApp that has already been built for you. Click the close box in your CRC window to discard the document without saving.

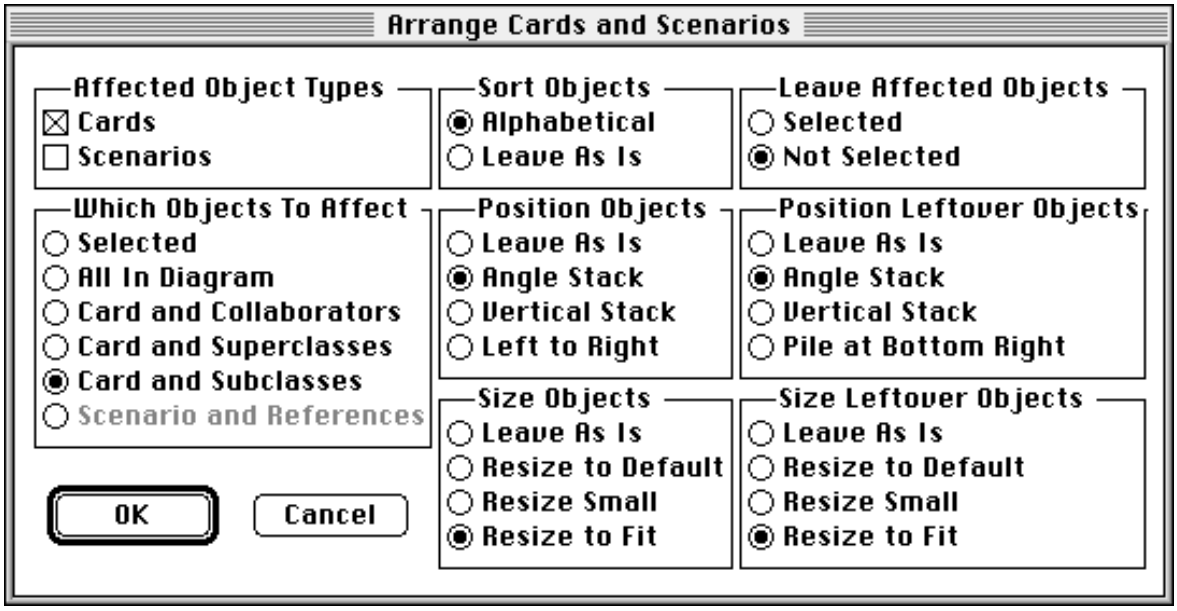
Click the Open File button and double-click the document SimpApp.crc to display the more complete CRC design. Notice that the TWindow card has some collaborating objects listed for its Draw responsibility. Collaborating objects are added to a card by selecting a responsibility in the Card Property dialog, typing a collaborating object name into the Collaboration Edit field and clicking the New button to add it. Alternatively, you can use the Pick List button to select from existing object names.

You may notice that the diagram shows all cards stacked on the left in alphabetical order and scenarios stacked on the right. Objects on the diagram can easily be arranged by positioning the Selection Arrow over the object's name, pressing the mouse button and dragging the object. To change the front to back ordering, click to select an object and use the Bring To Front or Send To Back commands.



SimpApp Design Showing Cards and Scenarios

It is often useful to rearrange cards in various ways to explore relationships between them. As an example, select the single card, TShape and choose the Arrange Cards and Scenarios command from the Option menu and fill in the dialog exactly as shown here. The TShape card and each subclass are positioned on the left of the diagram and everything else gets stacked on the right side of the diagram.

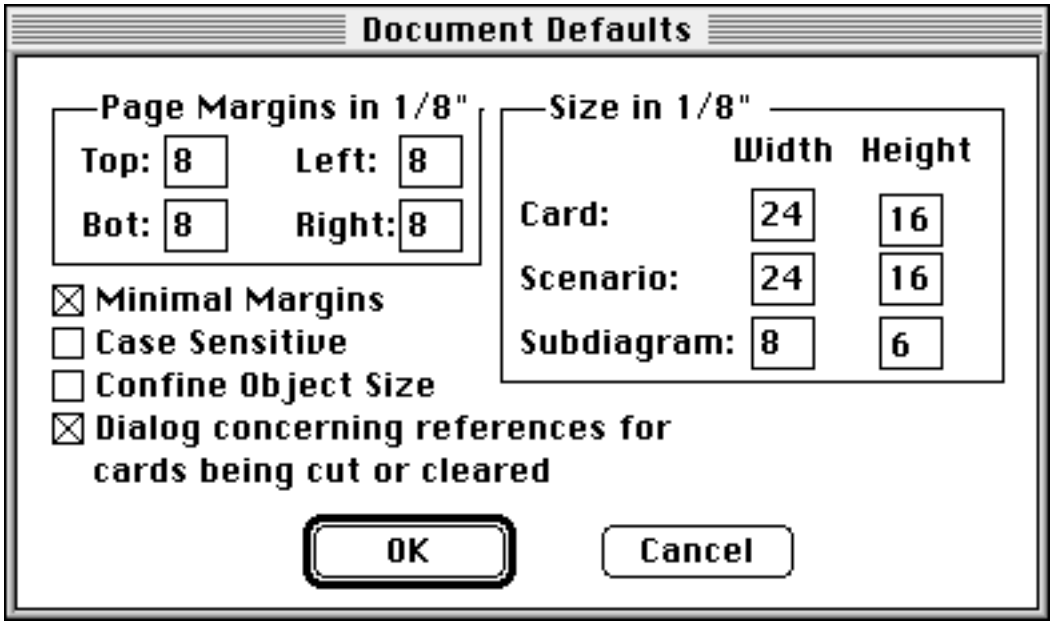


Arrange Cards and Scenarios Dialog

To fill in this dialog work from the top left corner to the bottom right indicating which objects you want to affect, how to size and position them and what to do about the remaining leftover objects. This command does not modify or remove any information in your design, it simply rearranges the visual presentation of objects on the diagram to help you focus attention on specific issues.

As the options on this dialog reveal, QuickCRC gives you the flexibility to easily arrange, size or position the objects on your diagram. Some of the choices on the Arrange Cards and Scenarios dialog are affected by options you've selected on the Document Defaults dialog for your CRC document.

Choose the Document Defaults command now to show the dialog. The default width and height of each newly created card, scenario and subdiagram object can be specified here. The Confine Object Size checkbox allows you to force all objects to the default size and prevent them from being individually resized. When this option is checked, some of the resizing options on the Arrange Cards and Scenarios dialog are disabled.



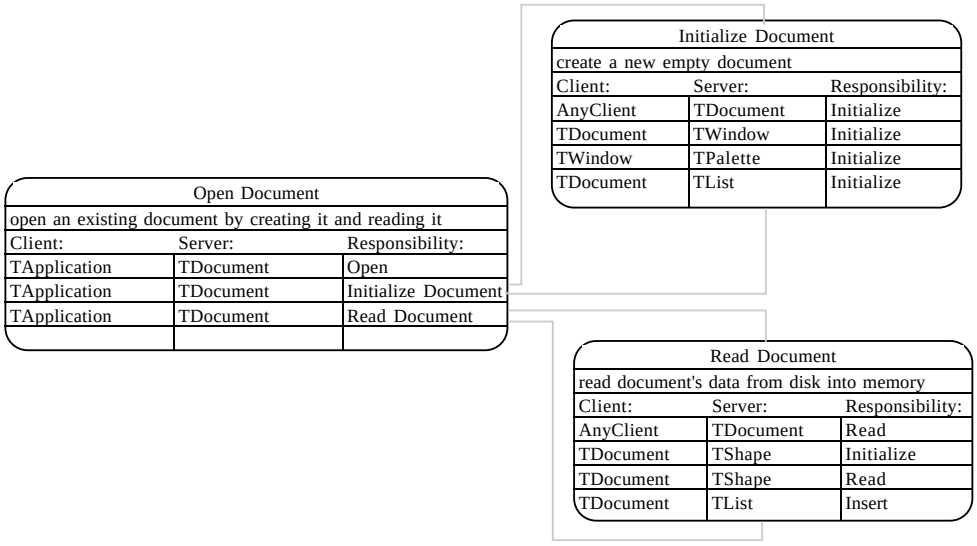
Document Defaults Dialog

7. Simulate With Subscenarios

As mentioned earlier some scenarios can use other subscenarios to get their work done. Initialize Document and Read Document are both subscenarios related to the server class TDocument. For example, the Open Document scenario references the Initialize Document subscenario in step 2, by specifying its server class TDocument in the server field and its scenario name in the Responsibility field. The first step in the Initialize Document subscenario uses TDocument as the server class name.

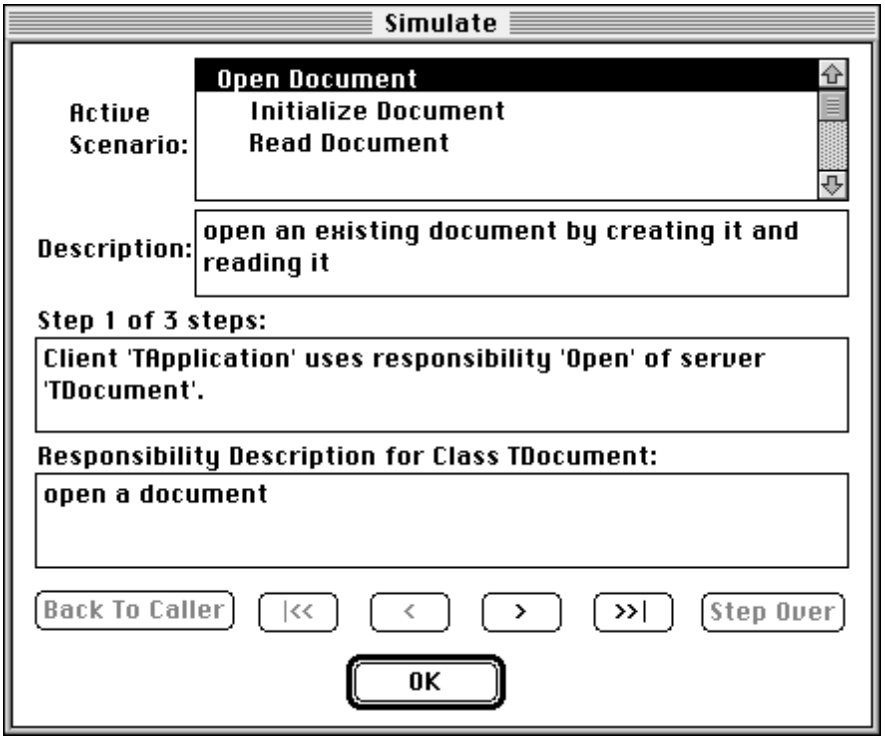
Open Document		
open an existing document by creating it and reading it		
Client:	Server:	Responsibility:
TApplication	TDocument	Open
TApplication	TDocument	Initialize Document
TApplication	TDocument	Read Document

Scenario Using Other Subscenarios



Path Through Subscenarios

 Select the Open Document scenario and click the Simulate button.



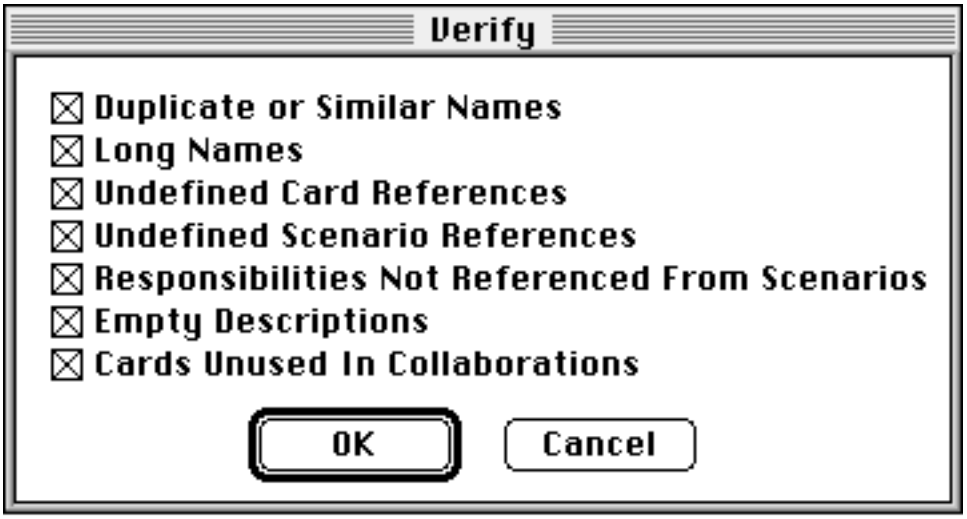
Simulate Dialog

 Within the Simulate dialog, click the Step Forward button to step into and step through each scenario. Notice that the Active Scenario list at the top shows your position within a hierarchical stack of scenarios. This list helps you keep your bearings in a complex simulation and even allows you to change to a different spot in the simulation path by clicking an item in the list.

8. Verify Your Work

Creating and simulating scenarios will help verify that your CRC design is correct and complete. The Verify CRC command also has several error checks to help locate problems in your design. It presents the following dialog allowing individual checks to be selectively turned on or off. Depending on the specific characteristics of the model, some messages may not necessarily indicate errors, but often they will reveal inconsistent or incomplete areas of the model.

The Duplicate or Similar Names check can detect class, attribute or scenario names that are identical once spaces are stripped from the name. Using the same name for different things will likely be problematic. It might be acceptable, however, to have the same attribute name in different classes, as in this model where fDocument refers to a TDocument object from the TApplication and TWindow classes.



Verify CRC Dialog

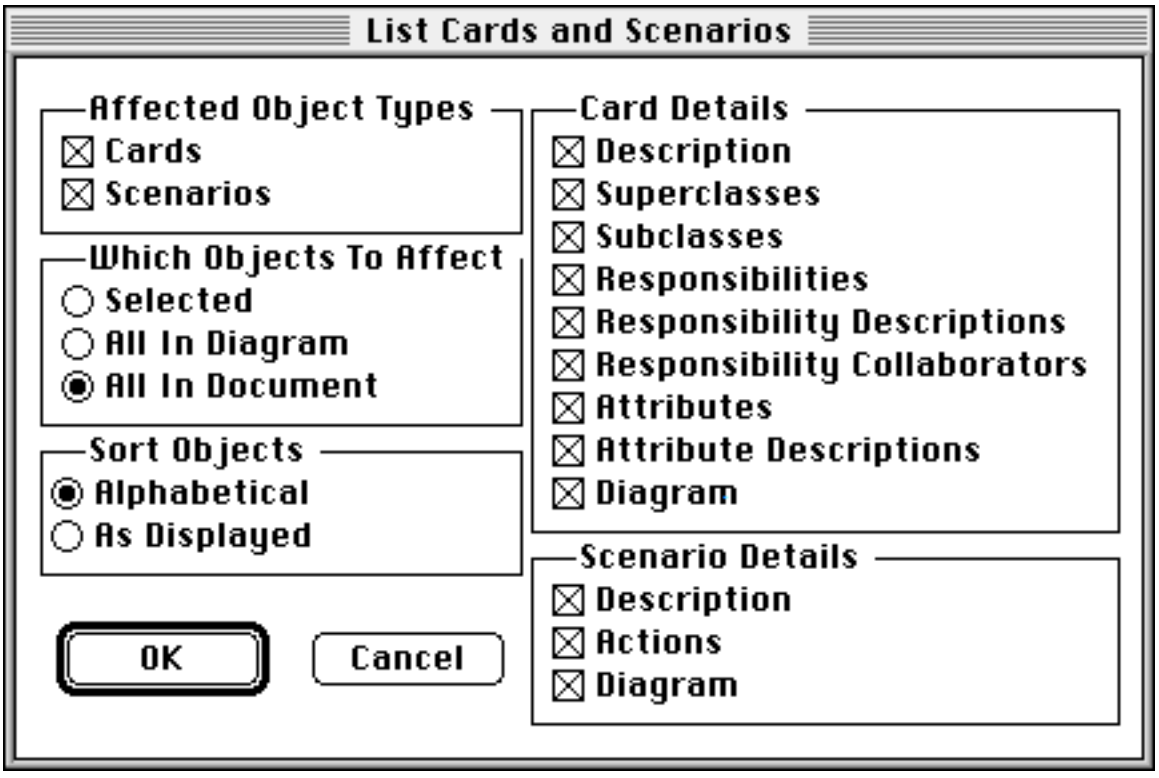
The Long Names check will detect class attribute or class operation names that exceed 40 characters and would be truncated when generating a dictionary entry list for use by MacA&D.

The Undefined Card References and Undefined Scenario References checks can locate references to classes, responsibilities or scenarios that were never created, have been deleted or were renamed.

The Responsibilities Not Referenced From Scenarios check often reveals responsibilities that are not needed or have not been completely exercised from the existing scenarios. Likewise the Cards Unused In Collaborations check often reveals unneeded classes or incomplete modeling. The error messages that remain in our model are okay, however, since they refer to the TBox and TCircle subclasses of the TShape class which itself has been exercised by our scenarios.

9. List Cards and Scenarios

On the Report menu, the List Cards and Scenarios command presents a dialog for listing specific information about cards and scenarios to a text window. This is often the most useful approach for getting information about your model into a format to print, review or distribute. The report can also be used as a coding specification.



List Cards and Scenarios Dialog

The Report window to which reports are generated can then be saved to disk and used by other word processing applications or printed using the Print command. By the way, you may notice that if you open multiple text windows the first one is named Report and each additional text window is named Text. The window named Report is special because that is where your reports list out to or from which information is imported.

The Export Dictionary Entries command puts information about your design into a format that can be used by MacA&D. Likewise, information from MacA&D or MacTranslator can be imported using the Import Dictionary Entries command.

10. Quit and Close Documents

This completes the tutorial. You can now close the documents you've created without saving changes. This tutorial has touched upon many of the features of QuickCRC.

Other features you may want to explore include subdiagrams and external agents. Separate diagrams are often useful for partitioning your model into subject areas each on its own diagram within the same CRC document. The Diagram Manager will then be useful for navigating or changing information about each diagram. External agents are used to identify users or external interfaces to the system being defined. External agents are defined using the External Agents command on the Option menu and can be used in place of class names when defining cards and scenarios.

Choose Quit from the File menu if you want to quit QuickCRC.

For information on QuickCRC and related products, contact Excel Software by phone at 515-752-5359, by email at info@excelsoftware.com or on the web http://www.excelsoftware.com.

Excel Software

Computer Aided Software Engineering