Contents

The following Help Topics are available:

About This Release
Getting Started
Glossary
Process Overview
Step-By-Step Procedures
License Information
Registration Form

For Help on Help, Press F1

About This Release

This is the first release for a Class-Responsibility-Collaboration (CRC) Object-Oriented Design Tool, CRC Tool 1.0. This release has the basic functionality to do responsibility driven design as described in *Designing Object-Oriented Software* by Wirfs-Brock, Wilkerson and Wiener. New features will be incorporated in future releases. For example, collaboration diagrams drawn from the specification may be incorporated in a future release.

This program is offered as shareware. This program is not freeware. Pricing for the CRC Tool 1.0 is $40.00 for an individual license. A corporate or site license is $200.00.
See the License Agreement for details.
See the Registration Form to register.

Contact Stan Mitchell with input regarding problems or requests for this program at:
e-mail: stanm@users.compumedia.com.
US mail: Stan Mitchell
3107 Smithers Ave S
Renton, WA 98055

Known problems:

- an anemic help file
- inconsistent keyboard accelerators in dialogs
- not all controls in dialogs have accelerators
- the only way to select which subsystems or classes to print is by page number
- redraw of views is not optimized
- there is no way to get from Responsibility View to System View and vice versa via the keyboard

Features to be added in future releases:

- page numbers on print out
- commands to expand and contract the hierarchical tree in the System View list
- collaboration graphs
- undo delete
- exception handling
- save views and window status for a design on close
- merge input design files
- option to display WYSIWYG size cards with scrollbars in card views
- consistency check; e.g. contracts have responsibilities, contracts have clients, abstract classes don't inherit from concrete classes, etc.
- selection of classes and subsystems for printing of cards
- hypertext design document in HTML output format
- hypertext design document in winhelp RTF format
- drag 'n drop between all views
- add font selection, card size, line height, and print backs option to print preview toolbar
- write design to a database
- provide scripting language to extract information from design
- print hierarchy and collaboration graphs
- zoom on hierarchy and collaboration graph views
- code generation for Smalltalk and C++

(Comments on the prioritization of features are welcome)

Getting Started

CRC Tool consists of five views:

1. A responsibility list view
2. A subsystem-class hierarchy list view
3. A class card view
4. A subsystem card view
5. A hierarchy graph view

Each view has a current selection. When a selection has been made, the right mouse button may be used to select a command; or a command from the menubar or toolbar may be selected.

Double-clicking on an item invokes the default action (where implemented).

- The Responsibility View invokes the Responsibility Property Sheet when a responsibility is double-clicked.
- The System View displays the selected item's view when it is double-clicked.

Double-clicking in the card views depends upon which line is selected.

- The first three lines of the Class Card View apply to the class and the Class Property Sheet is invoked. The numbered lines are the contracts; and the contract edit dialog is invoked. The remaining lines are the responsibilities and the responsibility edit dialog is invoked.
- The first line of the Subsystem Card View applies to the subsystem and the subsystem edit dialog is invoked. The remaining lines are the contracts; and the contract edit dialog is invoked.
- The Hierarchy Graph View does not support double-clicking at this time.

Follow the Step-by-Step Procedures to create your design.

Glossary

Most of these definitions were taken from the book *Designing Object-Oriented Software* by Wirfs-Brock, Wilkerson and Wiener. Refer to the book for more information on each topic.

Abstract Class
Attribute
Browser View
Class
Class Card View
Class Property Sheet
Client
Collaboration
Concrete Class
Contract
Hierarchy
Hierarchy Graph View
Inheritance
Instance
Multiple Inheritance
Object
Protocol
Parameter
Private Responsibility
Responsibility
Responsibility Property Sheet
Responsibility View
Server
Subclass
Subsystem
Subsystem Card View
System View
Superclass

Abstract Class

Abstract classes exist merely so that behavior common to a variety of classes can be factored out into one common location, where it can be defined once and reused again and again. Abstract classes are not intended to produce instances of themselves.

Attribute

Attributes are variables defined for a responsibility.

Browser View

The Browser View is a split view which consists of the Responsibility View and System View.

Class

A class is a generic specification for an arbitrary number of similar objects. It is a template for a specific kind of object. Classes allow us to describe in one place the generic behavior of a set of objects, and then to create objects that behave in that manner when we need them.

Class Card View

A class card view is a graphical representation of an index card that holds all the pertinent information about a class, one class per card. The card holds the class name, superclasses, subclasses, contracts, responsibilities and collaborations. It also provides flag indicating if the class is abstract or concrete.

Class Property Sheet

A Class Property Sheet is a tabbed dialog box that provides all the necessary fields related to a class for edit. It includes the class' name, the subsystem it is assigned to, a description, an abstract flag, and any superclasses, contracts, and responsibilities associated with it.

Client

A client is the object that makes the request in a collaboration.

Collaboration

Collaborations represent requests from a client to a server in fulfillment of a client responsibility. We say that an object collaborates with another if, to fulfill a responsibility, it needs to send the other object any messages. A single collaboration flows in one direction - representing a request from the client to the server.

Concrete Class

Concrete subclasses inherit the behavior of their abstract superclasses and add other abilities unique to their purpose. They may need to redefine the default implementations of their abstract superclasses, if any, in order to behave in a way meaningful to the application of which they are a part. These fully implemented classes create instances of themselves to do the useful work in a system.

Contract

A contract between two classes represents a list of services an instance of one class can request from an instance of the other class. The object that makes the request is the client, and the object that receives the request and thereupon provides the service is the server. A service can be either the performance of some action or the return of some information. All of the services listed in a particular contract are the responsibilities of the server for that contract.

Hierarchy

A hierarchy is an inheritance relationship between related classes. The relationships include classes, superclasses and subclasses.

Hierarchy Graph View

A hierarchy graph view presents a graphical representation of the inheritance relationships between related classes.

Inheritance

Inheritance is the ability of one class to define the behavior and data structure of its instances as a superset of the definition of another class or classes.

Instance

An instance of a class is an object that behaves in a manner specified by a class.

Multiple Inheritance

Multiple inheritance is the ability of a class to inherit behavior from several superclasses.

Object

An object is an instance of a class.

Protocol

Protocols are member functions; things specifically asked of a class.

Parameter

Parameters are arguments to protocols.

Private Responsibility

Private responsibilities represent behavior a class must have, but which cannot be requested by other objects.

Responsibility

Responsibilities include two key items: the knowledge an object maintains, and the actions an object can perform. The responsibilities of an object are all the services it provides for all of the contracts it supports. A service can be either the performance of some action, or the return of some information. Express responsibilities in general terms and try to keep all of a class's responsibilities at the same conceptual level.

Responsibility Property Sheet

A Responsibility Property Sheet is a tabbed dialog box that provides all the necessary fields related to a responsibility for edit. It includes the responsibility's name, the class it is assigned to, any contracts, collaborations, protocols and attributes associated with it.

Responsibility View

A Responsibility View is a view located in the Browser which is separated from the System View by a splitter bar. The Responsibility View provides a list of the responsibilities for the system. If a responsibility in the list has been associated with a class, the class name will be shown in parentheses after the responsibility name.

Server

A server is the object that receives the request in a collaboration and thereupon provides the service.

Subclass

A subclass is a class that inherits behavior from another class. A subclass usually adds its own behavior to define its own unique kind of class.

Subsystem

A subsystem is a group of classes, or groups of classes and other subsystems, that collaborate among themselves to support a set of contracts.

Subsystem Card View

A subsystem card view is a graphical representation of an index card that identifies subsystems. The cards include the subsystem name, a short description of the overall purpose, contracts required by clients external to the subsystem, and beside each contract, the delegation to the internal class or subsystem that actually supports the contract.

System View

The System View is a view located in the Browser which is separated from the Responsibility View by a splitter bar. The System View displays an outline form of the various components of a design. The components include subsystems, classes, and hierarchy graphs. Special icons reflect each of the component types. The System View provides a means of viewing the design structure, as well as modifying the structure by editing and deleting components.

Superclass

A superclass is a class from which specific behavior is inherited.

The Process

The following summary of the CRC process was taken from *Designing Object-Oriented Software* by Wirfs-Brock, Wilkerson and Wiener. Refer to the book for more information. Notes in '( )'s are related to CRC Tool.

Exploratory Phase:

1. Read and understand the specification.
2. As you follow the steps below, walk through various scenarios to explore possibilities. Record the results on design cards. (Record the results on design cards; or record on design cards and transcribe into CRC Tool; or just record in CRC Tool.)

Classes

3. Extract noun phrases from the specification and build a list.
4. Look for nouns that may be hidden (for example, by the use of the passive voice), and add them to the list.
5. Identify candidate classes from the noun phrases by applying the following guidelines:
   - Model physical objects.
   - Model conceptual entities.
   - Use a single term for each concept.
   - Be wary of the use of adjectives.
   - Model categories of objects.
   - Model external interfaces.
   - Model the values of an object's attributes.
6. Identify candidates for abstract superclasses by grouping classes that share common attributes.
7. Use categories to look for classes that may be missing.
8. Write a short statement of the purpose of the class.

Responsibilities

9. Find responsibilities using the following guidelines:
   - Recall the purpose of each class, as implied by its name and specified in the statement of purpose.
   - Extract responsibilities from the specification by looking for actions and information.
   - Identify responsibilities implied by the relationships between classes.
10. Assign responsibilities to classes using the following guidelines:
    - Evenly distribute system intelligence.
    - State responsibilities as generally as possible.
    - Keep behavior with related information.
    - Keep information about one thing in one place.
    - Share responsibilities among related classes.
11. Find additional responsibilities by looking for relationships between classes.
    - Use "is-kind-of" relationships to find inheritance relationships.
    - Use "is-analogous-to" relationships to find missing superclasses.
    - Use "is-part-of" relationships to find other missing classes.

Collaborations

12. Find and list collaborations by examining the responsibilities associated with classes. Ask:
    - With whom does this class need to collaborate to fulfill its responsibilities?
    - Who needs to make use of the responsibilities defined for this class?
13. Identify additional collaborations by looking for these relationships between classes:
    - the "is-part-of" relationship,
    - the "has-knowledge-of" relationship, and

- the "depends-upon" relationship.

14. Discard classes if no classes collaborate with them, and they collaborate with no other classes.

Analysis Phase:

Hierarchies

15. Build hierarchy graphs that illustrate the inheritance relationships between classes.
16. Identify which classes are abstract and which are concrete.
17. Draw Venn diagrams representing the responsibilities shared between classes.
18. Construct class hierarchies using the following guidelines:
    - Model a "kind-of" hierarchy.
    - Factor common responsibilities as high as possible.
    - Make sure that abstract classes do not inherit from concrete classes.
    - Eliminate classes that do not add functionality.
19. Construct the contracts defined by each class using the following guidelines:
    - Group responsibilities that are used by the same clients.
    - Maximize the cohesiveness of classes.
    - Minimize the number of contracts per class.

Subsystems

20. Draw a complete collaborations graph of your system.
21. Identify possible subsystems within your design. Look for frequent and complex collaborations. Name the subsystems.
    - Classes in a subsystem should collaborate to support a small and strongly cohesive set of responsibilities.
    - Classes within a subsystem should be strongly interdependent.
22. Simplify the collaborations *between* and *within* subsystems.
    - Minimize the number of collaborations a class has with other classes or subsystems.
    - Minimize the number of classes and subsystems to which a subsystem delegates.
    - Minimize the number of different contracts supported by a class or a subsystem.

Protocols

23. Construct the protocols for each class. Refine responsibilities into sets of signatures that maximize the usefulness of classes.
    - Use a single name for each conceptual operation, wherever it is found in the system.
    - Associate a single conceptual operation with each method name.
    - If classes fulfill the same specific responsibility, make this explicit in the inheritance hierarchy.
    - Make signatures generally useful.
    - Provide default values for as many parameters as reasonable.
24. Write a design specification for each class. (automated by CRC Tool)
25. Write a design specification for each subsystem. (automated by CRC Tool)
26. Write a design specification for each contract. (automated by CRC Tool)

Step-By-Step Procedures

- Create Classes
- Create Responsibilities
- Assign Responsibilities to Classes
- Create Hierarchy
- Re-evaluate Responsibility to Class Assignments
- Create Collaborations
- Create Hierarchy Graphs
- Create Abstract Classes
- Create Contracts
- Create Subsystems
- Assign Protocols and Attributes
- Print Cards
- Create and Print Design Document

License Information
CRC Tool 1.0 - Shareware Evaluation License

LICENSE AGREEMENT

Please carefully read the following terms and conditions. Continued use of *CRC Tool* constitutes your acceptance of these terms and conditions and your agreement to abide by them.

CRC Tool is a "shareware program" and is provided at no charge to the user for an evaluation period of 30 days. If you find this program useful and find that you are using *CRC Tool* after the 30 day evaluation period, please register the product (see registration details below). *CRC Tool* may not be modified in any way, and should be distributed with all supplied files in its original archive format: CRCTOL16.ZIP or CRCTOL32.ZIP

The 30-Day Free Evaluation License is a legal agreement between you, the end user, and Ixtlan Software. By using *CRC Tool*, you are agreeing to be bound by the terms of this Agreement. If you do not agree to the terms of this Agreement please discontinue using *CRC Tool*.

Business & Government Site License
After the 30 day Evaluation Period, any corporation, institution, government agency or business wishing to continue use of *CRC Tool* in the course of its internal business is required to purchase a *CRC Tool* registration or site license. Any individual wishing to use *CRC Tool* within a corporation, institution, government agency or business must purchase a *CRC Tool* registration or site license. The site license is provided for those who want to *CRC Tool* on multiple computers.

Upgrade Policy
Registered users of *CRC Tool* will be entitled to all upgrades of *CRC Tool* at no additional charge.

IXTLAN SOFTWARE'S SHAREWARE DISTRIBUTION POLICY

The essence of shareware software is to provide users with an opportunity to evaluate quality software before buying it, thereby keeping prices low while still providing developers with an incentive to continue development. Shareware is a distribution method, not a type of software. Copyright laws apply to registered Shareware just the same as they do for commercial software.

CRC Tool as Shareware
CRC Tool is distributed under the "Try Before You Buy" Shareware marketing concept. *CRC Tool* is a FULLY FUNCTIONAL program and no features, commands or functions have been disabled or crippled in any way. *CRC Tool* is NOT free software, and we strongly encourage you to register your copy. With registration you will be entitled to FREE version upgrades.

Distribution Policy
A number of Shareware publishers, having attained a reasonable degree of success, have "graduated" away from the Shareware marketing and distribution concept, often citing the low rate of user registration as an obstacle to economic growth. Despite the obvious economic risks of publishing quality software under the Shareware concept, and due mainly to a strong belief in the future and sheer power

of the "super data-highway" as a medium for publishing and distributing software, Ixtlan Software has deliberately embraced the Shareware distribution policy.

How this policy benefits you
Ixtlan Software's distribution policy directly benefits you, the end-user, as you are able to fully evaluate *CRC Tool* BEFORE spending a cent.

REGISTRATION

Registering *CRC Tool* licenses you to use the product after the 30 day evaluation period. Registered users of *CRC Tool* will receive a serial number with which to license *CRC Tool* in their name. A two-tiered pricing structure exists for registration: $40 (US) for a single licensed copy of *CRC Tool*, $200 (US) for a site license. Prices and terms are subject to change without notice.

COPYRIGHT NOTICE

DISCLAIMER

Ixtlan Software makes no warranty of any kind, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to this software and accompanying documentation.

IN NO EVENT SHALL IXTLAN SOFTWARE BE LIABLE FOR ANY DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THIS PROGRAM, EVEN IF IXTLAN SOFTWARE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TRADEMARKS

Any product or brand names mentioned in this document are trademarks or registered trademarks of their respective owners.

Registration Form

Print this page and fill out to license CRC Tool. Send it together with your check for $40 for an individual license, or $200 for a site license, to:
Stan Mitchell
3107 Smithers Ave S
Renton, WA 98055

Name:
_____

Company:
_____

e-mail address:
_____

Address:
_____

_____

_____

Check here for a site license: _____ Amount enclosed: _____

A company name is required for a site license.
A registration ID will be sent to you for you to use to register the program. If an e-mail address is provided the ID will be sent via e-mail.

Class Dialog

Use the class dialog to create a new class.

The name field specifies the class name and must be unique.

The subsystem field identifies the subsystem theclass belongs to.

The abstract button, if checked, indicates the class is an abstract class.

The description field allows you to document the purpose of the class. This description is printed on the back of a class card if the print option to print backs is selected.

Select Create Subsystem if you need to specify a subsystem which does not exist as yet.

Select OK to accept the new class and terminate class creation.

Select Cancel to abort the new class and terminate class creation.

Select Another to accept the new class and prepare to create another class.

Browser Window

The browser window consists of two parts separated by a splitter bar. The top half holds the Responsibility View. The bottom half holds the System View.

A selected line in each view is highlighted. A line in the System View may be selected by clicking on the line with the left or right mouse button. If the right mouse button is used a pop-up menu of commands will be displayed. When a line is highlighted it is the target for the Edit or Delete command.
The Responsibility View allows multiple selection. If multiple responsibilities are selected only the Delete command is enabled.

Class Card Window

A class card window holds a Class Card View. The top line of the view displays the class name and the abstract/concrete class indicator. The second line displays the class' superclasses. The third line displays the class' subclasses. The remaining lines display the class' responsibilities and contracts. Contracts are numbered. If a responsibility has been assigned to a contract it will be displayed indented beneath the contract. A responsibility which is not assigned to a contract is a private responsibility. A responsibility's collaborations are displayed on the right side of the card on the same line.

The selected line is displayed highlighted. A line may be selected by clicking on the line with the left or right mouse button. If the right mouse button is used a pop-up menu of commands will be displayed. The Edit or Delete commands will then target the object on the selected line. Alternatively, the up and down arrow keys may be used to move the current selection up and down. Pressing the enter key will display the pop-up menu of commands.

Subsystem Card Window

A subsystem card window holds a Subsystem Card View. The top line of the view displays the subsystem name. The remaining lines display the contracts which are exported by the subsystem. That is those contracts which have clients outside of the subsystem. The server class for the contract is displayed to the right of the contract.

The selected line is displayed highlighted. A line may be selected by clicking on the line with the left or right mouse button. If the right mouse button is used a pop-up menu of commands will be displayed. The Edit or Delete commands will then target the object on the selected line. Alternatively, the up and down arrow keys may be used to move the current selection up and down. Pressing the enter key will display the pop-up menu of commands.

Hierarchy Graph Window

A hierarchy graph window holds a Hierarchy Graph View. Classes are displayed in rectangles showing their inheritance relationships. Abstract classes display a black triangle in the upper left hand corner.

The selected class is indicated by thick bordered rectangle. To select a class in the view click the left mouse button in the class rectangle. The right mouse button may then be used to display a pop-up menu of commands.