APPENDIX A

Architecture Example

Clipboard Architecture

The following example represents the clipboard architecture for an operating system such as Windows95. First we present a set of use cases, we then construct a set of test cases. This specification is at the API level and does not consider GUI-related issues such as how the material to be cut or copied is selected by the user.

Actor Model

There are two actors:

1. Application User – The user of any application that uses the clipboard can move material from one place to another.
2. Application Programmer – The programmer must ensure that the application can properly receive information from the clipboard and send information to the clipboard.

Use Cases

Use Case # 1.0	Name: User moves material	
User: Application User	Frequency:	On demand
	Criticality:	High

Use: The application user cuts material from one place in a document, repositions the cursor and pastes the material into this new location.

Use Case # 1.1	Name: User moves material

within a single document
User: Application User Frequency: On demand
Criticality: High
Use: The application user cuts material from one place in a document, repositions the cursor to another position in the same document and pastes the material into this new location.

Use Case # 1.2 Name: User moves material from one document to another
User: Application User Frequency: On demand
Criticality: High
Use: The application user cuts material from one place in a document, repositions the cursor to a position in another document and pastes the material into this new location.

Use Case # 2.0 Name: User copies material
User: Application User Frequency: On demand
Criticality: High
Use: The application user copies material that occurs at one place in a document, repositions the cursor and pastes the material into this new location.

Use Case # 2.1 Name: User copies material within a single document
User: Application User Frequency: On demand
Criticality: High
Use: The application user copies material that occurs at one place in a document, repositions the cursor within the document and pastes the material into this new location.

Use Case # 2.2 Name: User copies material from one document to another
User: Application User Frequency: On demand
Criticality: High
Use: The application user copies material that occurs at one place in a document, repositions the cursor in the second document and pastes the material into this new location.

Use Case # 3.0 Name: User removes material
User: Application User Frequency: On demand
Criticality: High
Use: The application user cuts material from one place in a document. That material is not used anywhere else in the document.

Use Case # 4.0 Name: User uses material from the clipboard repeatedly
User: Application User Frequency: On demand
Criticality: High
Use: The application user pastes material from the clipboard after it has been pasted previously.

SubUse Case # S1.0 Name: Material is written to the clipboard
User: Cut Use Case, Copy Use Case Frequency: Whenever Cut or Copy Use case is used
Criticality: High
Use: The application sends a block of material to the clipboard object.

Test Cases

Test Case # 1.1
Test Scenario: The user starts application one. The user cuts material from application one and then pastes the material back into application one.
Expected result: The application contains the same material after the operation as it did before the operation.

Test Case # 1.2
Test Scenario: The user starts application one and application two. The user cuts material from application one and then pastes the material into application two.
Expected result: Application one has less material than before the operation and application two has more material.

Test Case # 2.1
Test Scenario: The user starts application one. The user copies material from application one and then pastes the material back into application one.
Expected result: Application one contains two copies of the same material.

Test Case # 2.2
Test Scenario: The user starts application one and application two. The user copies material from application one and then pastes the material into application two.
Expected result: Application one has the same material as before the operation and application two has additional material that matches the material that was copied from application one.

Test Case # 3.1
Test Scenario: The user starts application one. The user cuts material from application one and then closes the application.
Expected result: The application contains less material after the operation than it did before the operation.

Test Case # 4.1
Test Scenario: The user starts application one. The user cuts material from application one, pastes it into application one, and then pastes it into application one again.
Expected result: The application contains two copies of the material that was cut originally.

Test Case # 4.2

Test Scenario: The user starts application one and application two. The user cuts material from application one, pastes it into application one, and then pastes it into application two.
Expected result: The applications each contain a copy of the material that was cut originally.

Test Case # 4.3
Test Scenario: The user starts application one and application two. The user cuts material from application one, pastes it into application two, and then pastes it into application one.
Expected result: The applications each contain a copy of the material that was cut originally.

Rapide Implementation

Rapide is an architectural description language developed at Stanford University. The following example illustrates how the clipboard architecture, specified in Appendix TBA, is implemented. The example then illustrates how these test cases are executed.

Rapide Implementation

```
-- A model of a simple clipboard architecture



---------Application Type -----
type Application is interface
action in CutCommand();
action in CopyCommand();
action in PasteCommand();
action out CuttoClipboard(n:integer);
action out CopytoClipboard(n:integer);
action out PastefromClipboard(ap:Application);
provides
function print();
function getContents() return Integer;
```

```
function setContents(n:Integer);
end Application;


module newApplication(n:integer) return Application is
  contents:var Integer:=n;
  function print() is begin
                IO.CPut("Contents of Application:");
                IO.CPut(Image(Integer,$contents));
           end print;
  function getContents() return Integer is begin return $contents; end;
  function setContents(n:Integer) is begin contents :=n; end;
parallel
when (?i:integer)CutCommand() do
        print();
        CuttoClipboard($contents);
        contents := 1;    ----------------------This is the problem line-------------
        print();
end when;

when (?i:integer)CopyCommand() do
        CopytoClipboard($contents);
        print();
end when;

when (?i:integer)PasteCommand() do
        PastefromClipboard(self);
        print();
end when;
end newApplication;


---------Clipboard Type -----
type Clipboard is interface
action in Receive(n:integer);
action in Send(ap:Application);
provides
function print();
end Clipboard;
```

```
module newClipboard() return Clipboard is
 store:var Integer:=0;
 function print() is begin
                                        IO.CPut("Contents of Clipboard");
                                        IO.CPut(Image(Integer,$store));
                            end print;
parallel
when (?i:Integer)Receive(?i) do
             store := ?i;
end when;

when (?a:Application)Send(?a) do
             ?a.setContents($store);
end when;
end newClipboard;

-------------User Type----------------
type User is interface
function UserCuts(ap:Application);
function UserCopies(ap:Application);
function UserPaste(ap:Application);
function print(ap:Application);
function script();
function last();
end user;

module newUser(ap1:Application; ap2:Application) return User is
application1:var Application := ap1;
application2:var Application := ap2;
function print(ap:Application) is begin
                                                     IO.CPut("User sees");
                                                     IO.CPut(Image(Integer,ap.getContents()));
                                                     end print;
function last() is begin
                            print($application1);
                            print($application2);
                          end last;

----Script is the implementation of the test case
function script() is begin
```

```
                UserCuts($application1);
                UserPaste($application2);
        end script;



function UserCuts(ap:Application) is begin ap.CutCommand(); end;
function UserCopies(ap:Application) is begin ap.CopyCommand(); end;
function UserPaste(ap:Application) is begin ap.PasteCommand(); end;
initial
        script();
end newUser;


-------------------Architecture definition-----------------


architecture clip4() return root is
Ap1:Application is newApplication(8);
Ap2:Application is newApplication(7);
C:Clipboard is newClipboard();
TheUser:User is newUser(Ap1,Ap2);



connect
(?i:integer)Ap1.CuttoClipboard(?i) to C.Receive(?i);
(?a:Application)Ap1.PastefromClipboard(?a) to C.Send(?a);
(?i:integer)Ap2.CuttoClipboard(?i) to C.Receive(?i);
(?a:Application)Ap2.PastefromClipboard(?a) to C.Send(?a);


end architecture clip4;
```