6 System Specification

6.1 Purpose

This specification states the requirements for a real-time digital signal processing application for x86 architecture PCs. The requirements stated serve as a basis for the acceptance procedure of this system. This section of the document is also intended as a starting point for the design phase and is structured according to the IEEE-830 Recommended Practice for Software Requirements (IEEE, 1998).

6.2 Specification Overview

Section 6.4 ~ 6.7 of this document describes the system in general terms and is intended as an all encompassing description of the system and its facets as they are to be in the final product.

Section 7 of this document serves as a reference point for defining the specific methods to be used to provide each expected function of the system, to expose any irregularities in the product description, and to reveal any possible useful features that may be enabled in the system without adding monolithically to the scope of the project.

Functions in this section are categorised according to system state and are detailed in section 7.2.

6.3 Overall Description

6.4.1 Product Perspective

The application will allow a user to turn a personal computer into a real-time digital-effects processor. It will allow the creation and custom-reconfiguration of any one of the stored effect-types. These may then be chained together to provide the user with a set of effects that can be run simultaneously in the order in which they are defined.

The system will provide the user with the functionality to create an 'effects-bank', in which many sets of chained effects may be stored for live use and fast switching while doing so.

An API will also be made available as the primary point from which all effects available in the system will be derived.

The system will interface with the ASIO specification for direct communication with the sound-device on the host computer.

6.4.2 Product Functions

The application will at any point in time be running in one of 4 possible states:

- Ready state, where no specific action has been selected to be performed.
- Effect designer, in which effects can be loaded, saved, configured, chained, and tested - both individually and as part of a chain.
- Effects bank builder, an interface to allow a user to specify which chains of effects to add to a bank and in which order the effect-bank is specified.
- Run processor, to initiate and run the sound signal processing with a chosen bank of effects.

6.5 User Characteristics

Typical users are expected to be in the following categories:

- Professional sound/recording engineers,
- Independent musicians/artists,
- 3rd party developers,
- Combinations thereof.

6.6 Constraints

All deliverables as far as the design model must be complete and satisfy the functional specification by the point of implementation.

In order that completion of each system will support the development of the next, implementation of the system must follow the order:

- Effect Chain Designer
- Effect Bank Designer
- Effect-Unit API
- Live Signal Processor

Black box-testing will occur during the development of each system to ensure that the software is performing the duties expected of it and again in a final pass after the unit tests for each sub-system have been implemented and proven to pass.

6.7 Assumptions and Dependencies

Development will take place on a Windows platform using an x86 architecture PC. The ASIO standards will apply for the hardware interface by way of the Asio4All project as this will provide professional level, low latency sound sampling and will also render the product capable of supporting professional grade hardware if it exists on the host system.

Development will be carried out using the .NET framework from Microsoft and the languages C# and (managed) C++. C# will be used for implementing the User Interfaces, the Effect-Chain Designer and the Effect-Bank Designer because of its ease of use and prototyping speed. Managed C++ will be used for the Live Signal Processor system and the API because of the reduced latency that it provides when using polymorphic type resolution.

7. Specific Requirements

7.1 Functional Requirements

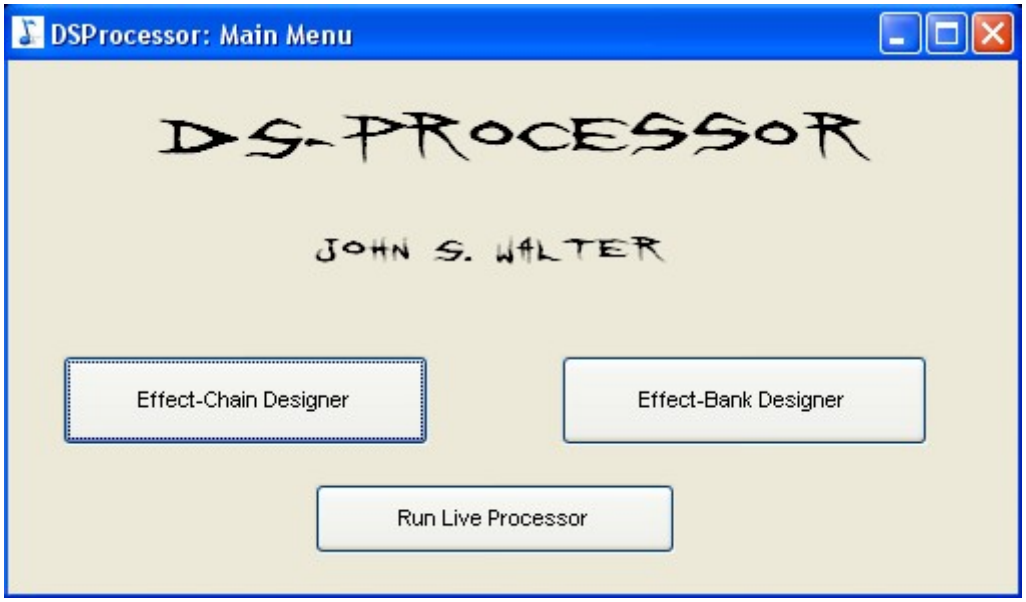
State Type	No.	Description
Main Menu (no action)	FR-001	The system will allow the user to navigate to the "Effect-Chain Designer" sub-system
	FR-002	The system will allow the user to navigate to the "Effect-Bank Designer" sub-system
	FR-003	The system will allow the user to navigate to the "Live Signal Processor" sub-system
Effect-chain designer	FR-004	The system will allow the user to create a new, unsaved, empty effect-chain
	FR-005	The system will allow the user to add a customisable version of any of the available effects to an effect chain
	FR-006	The system will allow the user to reorder the effects in a chain
	FR-007	The system will allow the user to alter the configuration of any of the effects in a chain
	FR-008	The system will allow the user to test the chain configurations being worked on at any one time
	FR-009	The system will allow the user to set any sub-set of the effects in a chain to inactive (bypassed) for testing purposes
	FR-010	The system will allow the user to discard changes made to the current chain and revert to its saved state
	FR-011	The system will allow the user to save the current chain being worked on
	FR-012	The system will allow a user to open a saved chain for editing
	FR-013	The system will allow a user to remove an instance of an effect module from a chain
	FR-014	The system will allow the user to exit the effect-chain designer state
Effect-bank builder	FR-015	The system will allow the user to create a new, unsaved effect bank
	FR-016	The system will allow the user to add any of the saved effect-chains to the bank
	FR-017	The system will allow the user to save the bank currently being worked on
	FR-018	The system will allow the user to open any of the previously saved banks
	FR-019	The system will allow the user to reorder the chains in the bank
	FR-020	The system will allow the user to remove an instance of an effect-chain from the bank currently being worked on.
	FR-021	The system will allow the user to exit the effect-bank builder state

State Type	No.	Description
Live signal processor	FR-022	The system will allow the user to select one of the available ASIO drivers on the host system
	FR-023	The system will allow the user to select one of the effect-banks for live processing
	FR-024	The system will allow the user to start and stop the processor at will
	FR-025	The system will allow the user to switch between the loaded effect-chains in the bank while processing is taking place
	FR-026	The system will allow the user to exit the live processor state

7.2 Graphical User Interfaces

The interface designs were built as development for each part of the system took place using the form builder facility of Visual Studio. The designs were considered thoroughly during development and were deemed to be appropriate for the functional scope of this prototype.

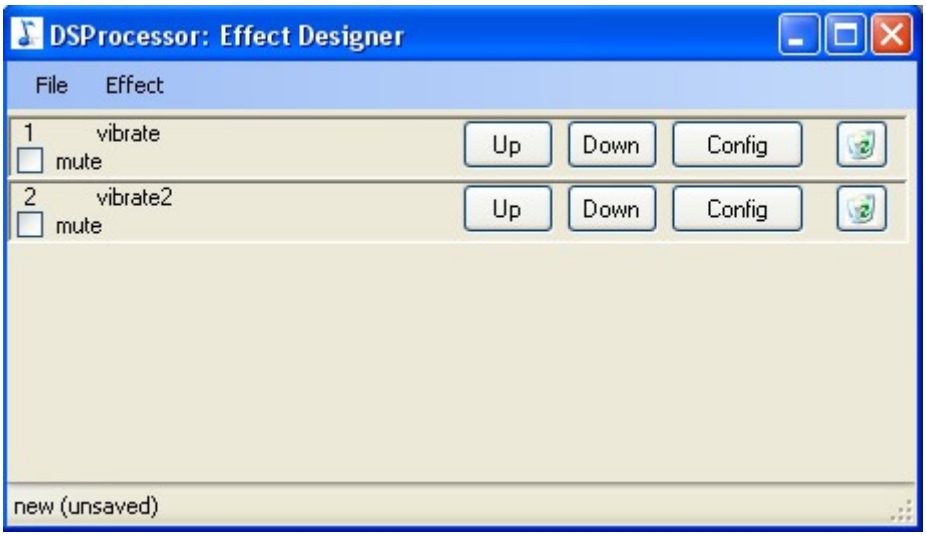
7.2.1 Main Menu



As can be seen, the main menu screen provides access to the three functional systems highlighted by functional requirements [FR-001, FR-002, FR-003]. When any one of these is invoked, the main menu is removed from view and only returns to the user when the particular sub-system is closed.

In order to load the relevant part of the application, the button clicks open the controller for the main menu, which like all controllers in this application is a singleton instance. The controller for the main menu then invokes the controller for the requested sub-system and calls it into view. If it happens that it cannot display then an error is displayed and the user is returned to the main menu.

7.2.2 Effect-Chain Designer



The effect chain designer provides the user with the facilities to create chains of effect modules. Each one that is added to a chain is represented by a custom UI component, the component is labelled with the name of the effect, and the location it occupies in the list on the form. The "Up" and "Down" buttons shift the location of each effect up or down by one place [FR-006], unless of course it is at the extent of the list and cannot be moved another place further.

The "Config" button on each effect unit will open the effect configuration dialogue [FR-007] in section 7.2.4 (below), and the button holding the recycle-bin icon will remove the effect-unit from the chain [FR-013]. The check-box for "mute" will set the list item as inactive so that it will not be loaded when the menu option for "Test Chain" is selected [FR-009].

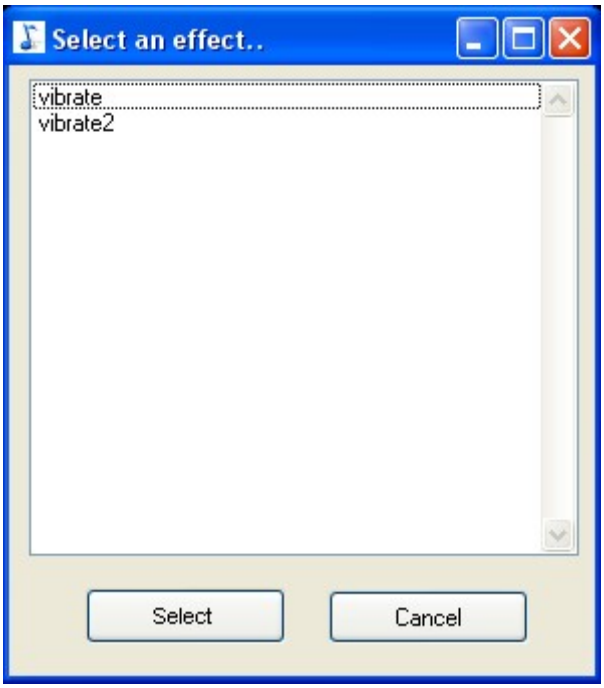
The "File" menu provides commands to clear the existing chain & create a new one [FR-004], open a saved one [FR-012, FR-010], or "Exit" [FR-014]. On each action the user is prompted as to whether he/she is happy to discard the changes if there are any. The underlying system controls this information, the EffectChainBuilder class of the Manager package holds the instance that is represented on this form and is also responsible for handling all of the required actions, which are invoked by the controller classes. The EffectChain class of the Entity package holds a Boolean flag that can be accessed by the manager classes in order to ascertain whether or not the current chain has unsaved changes. Thus if any of the methods that alter the structure of the current effect-chain entity are invoked and complete successfully, the flag that marks the chain as altered is set to true and the controller class that handled the user's request updates with the new information stored in the EffectChain object.

If the current edit has been saved previously then the label on the bottom left corner of the form will update with the name of it, if not then the word "new" is displayed, and for any chain that is in edit, if there exists unsaved changes on it then the word "(unsaved)" will be displayed after the name of the chain (as can be seen above).

The file menu also provides the functions for saving the current edit [FR-011], if the chain has no name already then the "Save" option will redirect to "Save As" otherwise the action overwrites the one already stored. For any editable chain the "Save As" option will always provide the dialogue in section 7.2.3 below.

The "Effect" menu item provides further options for "Test Chain", and "Add Effect" module. "Test Chain" will load the items in the current edit window which are not marked as mute and fire off the live processor system in order for the user to try out different arrangements [FR-008]. The "Add Effect" option will open the dialog in 7.2.3 which provides the user with a list of all the available sound processing units [FR-005].

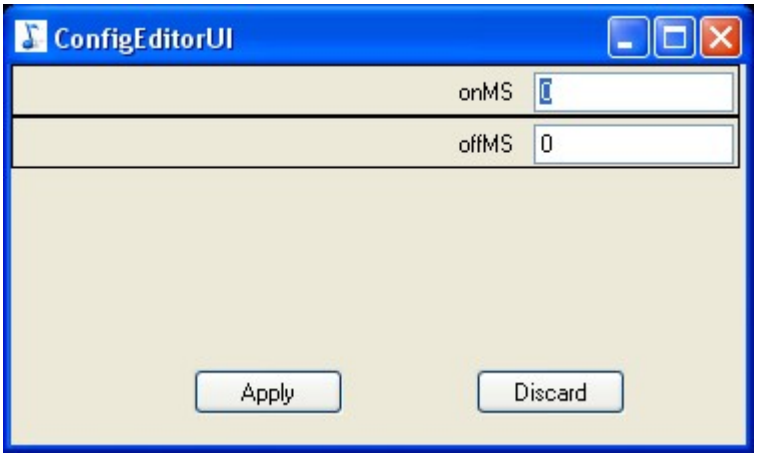
7.2.3 Effect Selector



When the user selects the option to add an effect to the current chain in edit, this is the dialogue that the user is presented with. The list set contained within is sorted into alphabetical order and lists the effect modules according to the name associated with each one. The actual file access functionality is contained within the EffectPrototypeStorage class of the Storage package. This class parses all of the effect modules on instantiation and loads the configuration file for each in order to build its list. The individual configuration files are parsed by the ConfigGenerator class of the Storage.XML package.

When an effect is selected from this list it is appended on to the end of the chain currently being worked on. The Cancel button closes the window and performs no further actions.

7.2.4 Effect Configuration Dialog

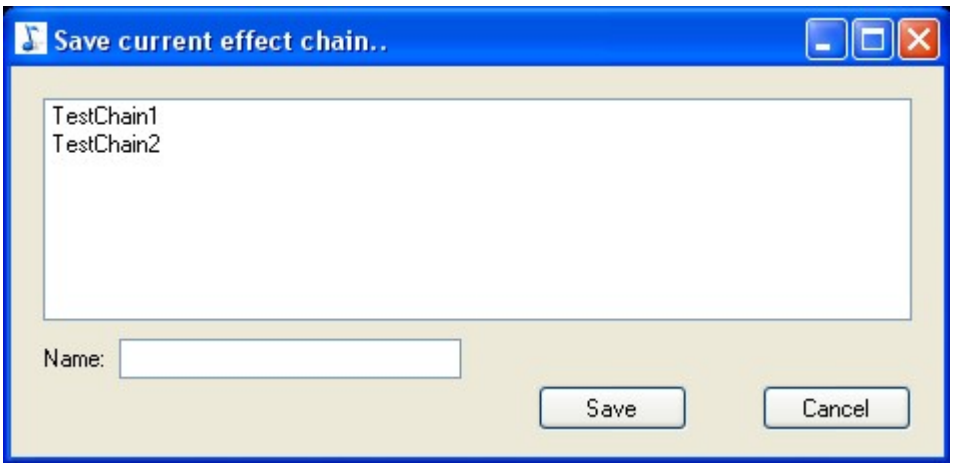


The effect-configuration dialogue consists of a list of custom UI components. They are much the same in structure to the ones representing an instance of an effect module except they contain less child entities. The configuration objects in this system consist of a list of key-value pairs. It was not considered necessary to use anything more complicated than this because in a digital system, every running variable is essentially just a value identified by a location.

The list of configuration attributes is generated from the contents of the prototype configuration file for each effect, which is stored in the zip file with the compiled effect-unit class.

The “Apply” button commits the changes to the current chain in edit.

7.2.5 Save Effect-Chain Dialog

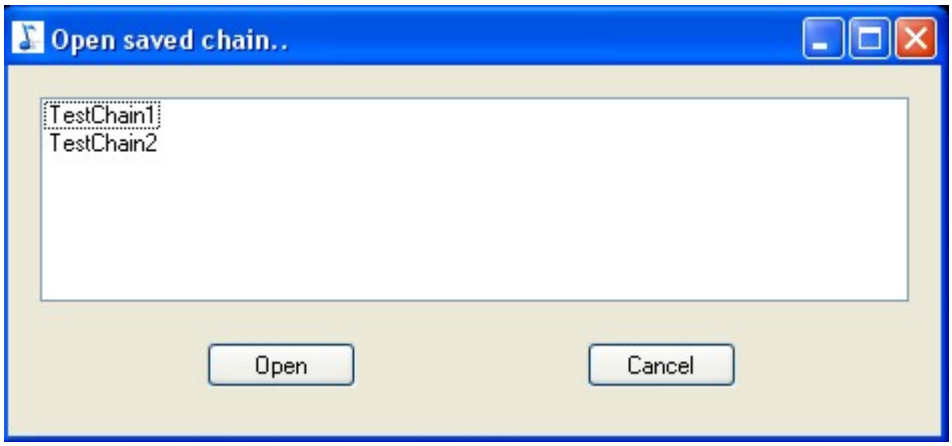


It was decided not to use the default operating system file dialogues because of the self-contained nature of the application. All of the entities when stored are saved in a particular location, with a particular extension, whereas the default file operation dialogues allow the user too much scope for losing or corrupting the file.

Any name entered in to the “Name” will be applied to the new file, unless of course it already exists, in which case the user will be informed of this fact and given the opportunity to cancel the action and save the previous version. In the same manner, if any of the entries in the dialog are selected by the user, then this name is applied to the chain and the user prompted to make certain of their decision.

In order to carry out the save operation in this example, the controller object invokes one of the save methods in the EffectChainBuilder object, which calls the “Save” method in the EffectChainStorage object.

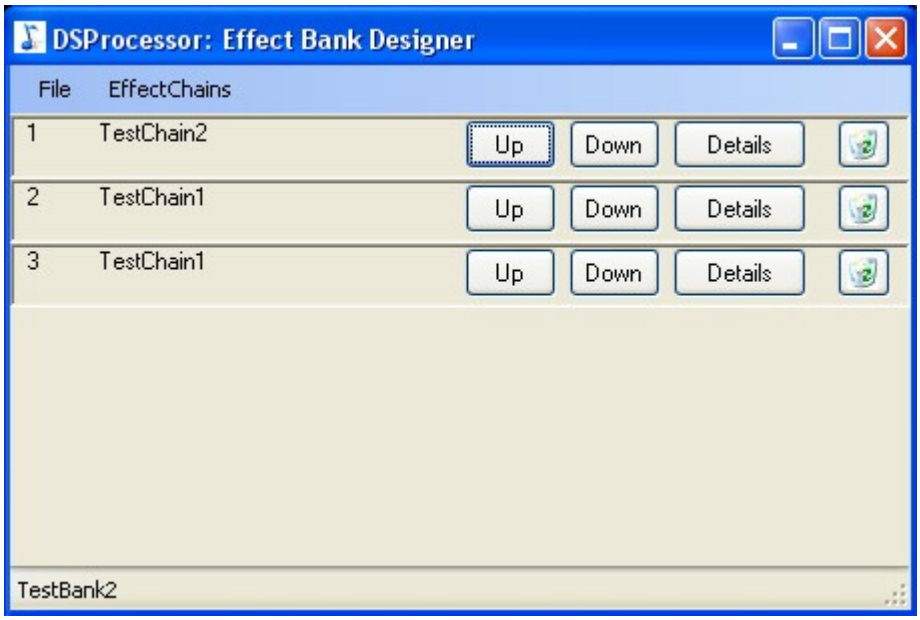
7.2.6 Open Effect-Chain Dialog



This dialog is populated in the same manner as the save dialog above, the EffectChainStorage class of the Storage package parses the files in a particular location and generates the list.

The “Open” button retrieves the chain from the file system and applies it to the current instance of the EffectChainBuilder class.

7.2.7 Effect-Bank Designer



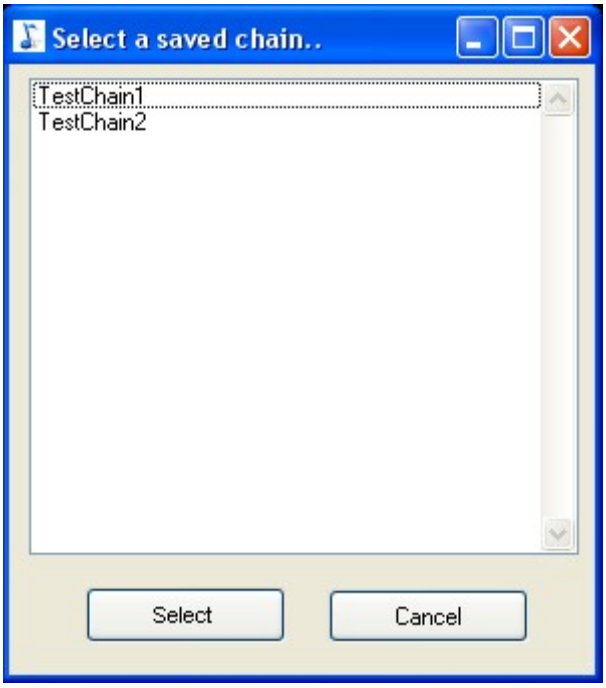
As can be seen from the diagrams above, the functionality available for constructing effect bank is very similar to that provided for the effect-chains. The "File" menu is mostly identical except for the fact that the functions therein operate on effect-banks and not effect-chains. It will allow a user to create a new effect-bank for working on [FR-015], open a saved effect-bank [FR-018], and save the current effect-bank [FR-017] either under a new name or using the one already assigned to it.

The effect-chains menu contains just one item at the time of this writing and that is to enable the user to add an instance of a saved effect chain to the current bank [FR-016]. Selecting this will open the dialogue in section 7.2.8.

The visual representations of the effect-chains are custom UI controls with slightly less complex make-up from the ones in the effect-chain designer. The four buttons allow the user to shift the chains up and down in the list [FR-019], remove any one of them [FR-020], or view details of the particular entity each one represents (which is not part of the requirements but is useful to help identify what each one is).

The exit option in the "File" menu will return the user to the main menu [FR-021].

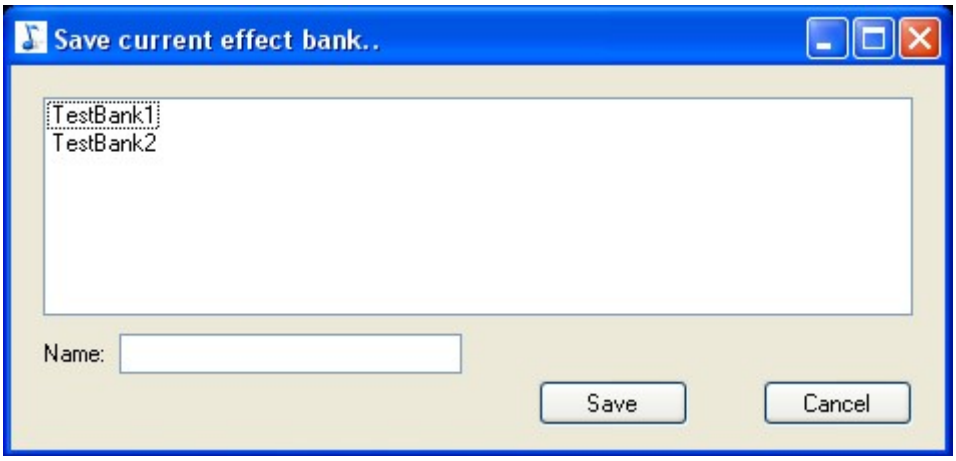
7.2.8 Effect-Chain Selector



When the user selects to add one of the stored chains to the current effect bank, this is the dialogue that enables this to take place. The process used to populate the list is the same as that for the "Effect-selector" dialogue – see section 7.2.3. The list is generated by the ChainPrototypeManager class of the Manager package, using the list of prototypes it retrieves from the EffectChainStorage class of the Storage package.

The list is constructed of facade objects that represent the underlying list of functionally complete objects. When an item is highlighted and selected using the "Select" button, it is appended on to the end of the list for the current bank in edit.

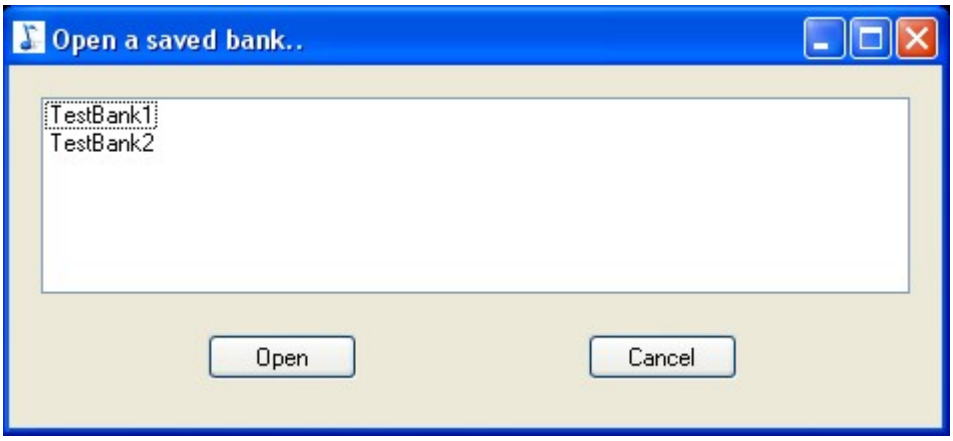
7.2.9 Save Effect-Bank Dialog



To save an effect bank, the process is identical to that for "Save Effect-Chain" – see section 7.2.5. The use of custom user interfaces instead of the default file system dialogs for the operating system means that it can be set to work in the same way as the previous example, except using a different storage location.

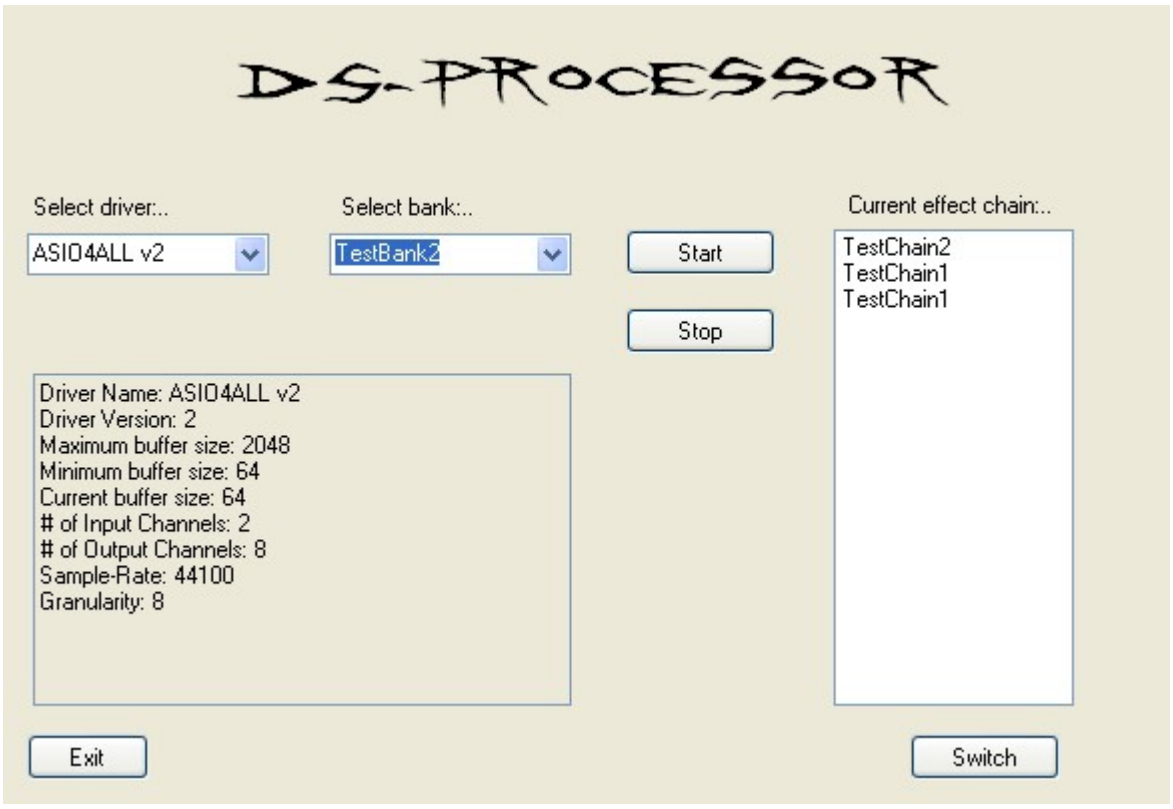
In order to carry out the save operation in this example, the controller object invokes one of the save methods in the EffectBankBuilder object, which calls the "Save" method in the EffectBankStorage object.

7.2.10 Open Effect-Bank Dialog



As with the example above for "Save Effect-bank", this process uses the same custom user interfaces as the "Open Effect-chain" process in section 7.2.6 above. Again, the only differences are the classes that carry out the actions that populate the list and the storage locations.

7.2.11 Live Processor UI



The final step in the overall process is to run the system for live signal processing. In order for this product to function the host system must have an ASIO driver installed. All of the available drivers are listed in the first dropdown select-box [FR-022]. Once one has been selected, the information about it is displayed in the text area above the "Exit" button and the select-box for the banks is then enabled. The bank-select box is populated using all of the available banks in the system [FR-023], the reason this was not developed into a more user-friendly process is due to time constraints, this part of the system was the last to be developed and time was running short so in order to meet the requirements the design was simplified.

The processor can be started and stopped by the user at will [FR-024] using the "Start" & "Stop" buttons respectively, and when the processor is in motion, the "Switch" button will allow the user to flip processing of

the buffers to the next chain of effects in the list [FR-025]. Finally, the "Exit" button closes the interface and returns the user to the main menu [FR-026].

7.1.1 Software interfaces (API Design)

One of the fundamental points that was designed into this system is the use of an extensible API for implementing modules that can be loaded into effect-chains. The reason for this is in order that anyone with programming knowledge would be allowed the opportunity to experiment with applying custom methods of signal transformations to the sample buffers. The API classes that enable this can be found in Appendix C: Code Excerpts.

The chains of effects are made possible by the use of the decorator pattern, the enabling factor of which is the ability of the classes involved to contain instantiations of one another by the use of a base class that declares the containing and container type. In the instance of this project, that type is named EffectBase and is contained in the Processor package/namespace. The EffectBase class is programmed in (managed) C++ and as such is a purely abstract type as is recognised in other object oriented languages.

This base class makes available already: the declaration for the contained object, the 'property' for setting that object and the "Process" function for passing in the sample buffers. Each of these is publicly accessible as they are all required by external classes of objects in order to set the chains up and run the processing algorithms.

Use of the API itself is in overriding the "Setup" method, which is required by the developers in order to access the configuration properties sent in, and in overriding the "Transform" method. The "Transform" method can only be accessed by the class itself, and is invoked by the "Process" method for each module before the buffer is passed on to the next in the chain (as long as one exists).

As the base class for this element of the system is programmed in (managed) C++, this also provides the ability for a developer to create the custom effect modules in the range of languages supported by the Microsoft Visual Studio Suite of tools. Because some of the compiled languages are faster than others for carrying out complex mathematical functions it would be advisable to keep with C++ for this part of the system.

7.1.2 Hardware interfaces

This application requires direct communications with the sound hardware on the host system. For ordinary applications running on a MS-Windows platform the standard release of a WDM driver for the hardware is appropriate. In this case however it is not, this is discussed in section 8.

In order to access this hardware in a low-latency manner an ASIO driver must be present on the system. ASIO (Audio Stream Input/Output) is a standard created by Steinberg who are professional music equipment designers, manufacturers and distributors. For this project I have imported some classes written by one Rob Philpott who provided the code included with this as an example of accessing said driver(s) attached to a posting on the CodeProject website (Philpott, 2008).

Following the example provided has enabled this project to produce the system detailed above. What it does is to transpose a COM compatible interface onto the driver object which is retrieved using the drivers registry-key identifier and the "CoCreateInstance" method from the libraries included with the Windows operating systems. This COM wrapped driver object is then contained in another class which attaches an event-handler to it, and this is the object that is loaded into the ProcessHarness class when the user selects the various items from the user interface.

9 Results

The result of this project is that the specified functionality has been met and covered with slight alterations. Some may argue that in certain respects the project has suffered from "scope slip". However the changes made to the requirements have only involved minor alterations to the core functionality: for the inclusion of stereo channels, copy & pasting of effect unit configurations, and test-running effect-chains. The author feels entirely justified in the refactoring of the requirements based on the reasons above.

The original intention was to implement the project and then use it to demonstrate the transformation of sound signals. While this is still the intention for demonstrating the application, it was not appropriate to include that in the scope of the project because the project is a software development task and not a purely DSP related investigation. That said however, the system has been tested with very simple examples and has been proven to work as expected.

As can be seen in appendix G, prior to the implementation of this product latency tests were carried out to assess the suitability of different languages to the solution of the problem. The result was that providing the application is run on a reasonably powerful PC, the use of Java would render the processing of signal transformations at twice the speed to using VC# or VC++.

The results show that the use of a less powerful machine has the effect of slowing down the Java execution. This is caused by the overhead of the Java hosting environment on the PC involved. When the host PC has the resources available, the overhead is catered for and the application is able to run at its own pace.

The system requires the use of the API in order for the transformation algorithm to be implemented. When this is compiled it is moved into a Zip file and a file named config.xml added to the same file. The config.xml file follows the structure detailed in appendix H. This configuration file is used when a user requires that an effect be added to the current chain in edit. The user may alter the configuration values and these are then saved as part of the chain in the xml file that represents it. The original configuration file is not altered so a developer may apply default values to the entries within.

The effect banks are constructed from sets of the effect chain file contents and are stored in xml files also as lists of chains with the values inside. The live signal processor uses the effect-bank xml files to locate the effect objects and instantiate them using the polymorphic type EffectBase. Each EffectBase object is passed an EffectConfiguration object that it uses in the "Setup" method which is then used to set the internal values for the effect. When the live signal processor is put into action, the channels are read from the ASIO driver and passed to the chain of effects. As mentioned, this is working as expected and is featured in the black box tests in Appendix D.

10 Recommendations

Wave Generation

Implementation of sound-wave generation classes with variable parameter sets, possibly allowing the use of other wave generator instantiations for some/all the parameters.

This may then require alterations to the structure of the effect-configuration design in order to accommodate the use of the wave as a source for the parameters required during processing.

GUI Advancement

In order to incorporate stereo effect building it would be a good idea to re-implement the front end with a work-bench style approach. In this way it would be possible to display the effect configurations that the user is currently working on with associations that depict where the signal is splitting, where it is being recombined, and where each channel is going to for the next stage of processing.

Another useful attribute may be to allow the user to define custom configuration editing forms. This would allow the implementer to show attributes in graphical form using graph-like displays.

Incorporation of Nyquist

As has been done with Audacity, provide a wrapper to use to load Nyquist modules for use. The only downside to this is the addition of a layer of message passing created by adding the wrapper.

VST wrapper class

Use of an effect-module implementation to access a specified VST module and use it through the "process" function of the effect, again this has the downside of the layer of message passing but the benefit of allowing users to incorporate modules that they would have potentially paid a lot of money for.

Implementation in stand-alone OS (*NIX)

This would also serve as an interesting opportunity to look into the Linux/Unix kernel and operating system. To implement this application hosted on a minimal set of OS components in an attempt to lower latency by stripping away all unnecessary software and processes.

A use for this would be in developing RISC machines for running the application and providing the modules as rack-mounted units with specialised hardware for handling large amounts of gain that result from combining signals during processing.

HAL update functionality

Implementation of functionality on the UI that would allow the user to alter the sample rate, bit-rate, and buffer size without interacting with the driver layer directly, this has scope issues however, especially when considering the previous suggestion for future work.

On-the-fly configuration attribute alteration

Allow the user to make alterations to the configuration of any of the effects while audio-stream processing is taking place.

This would require 3rd party developers to implement an update function that would update internal parameters used by the effects based on events received.

Development of USB peripherals for application control

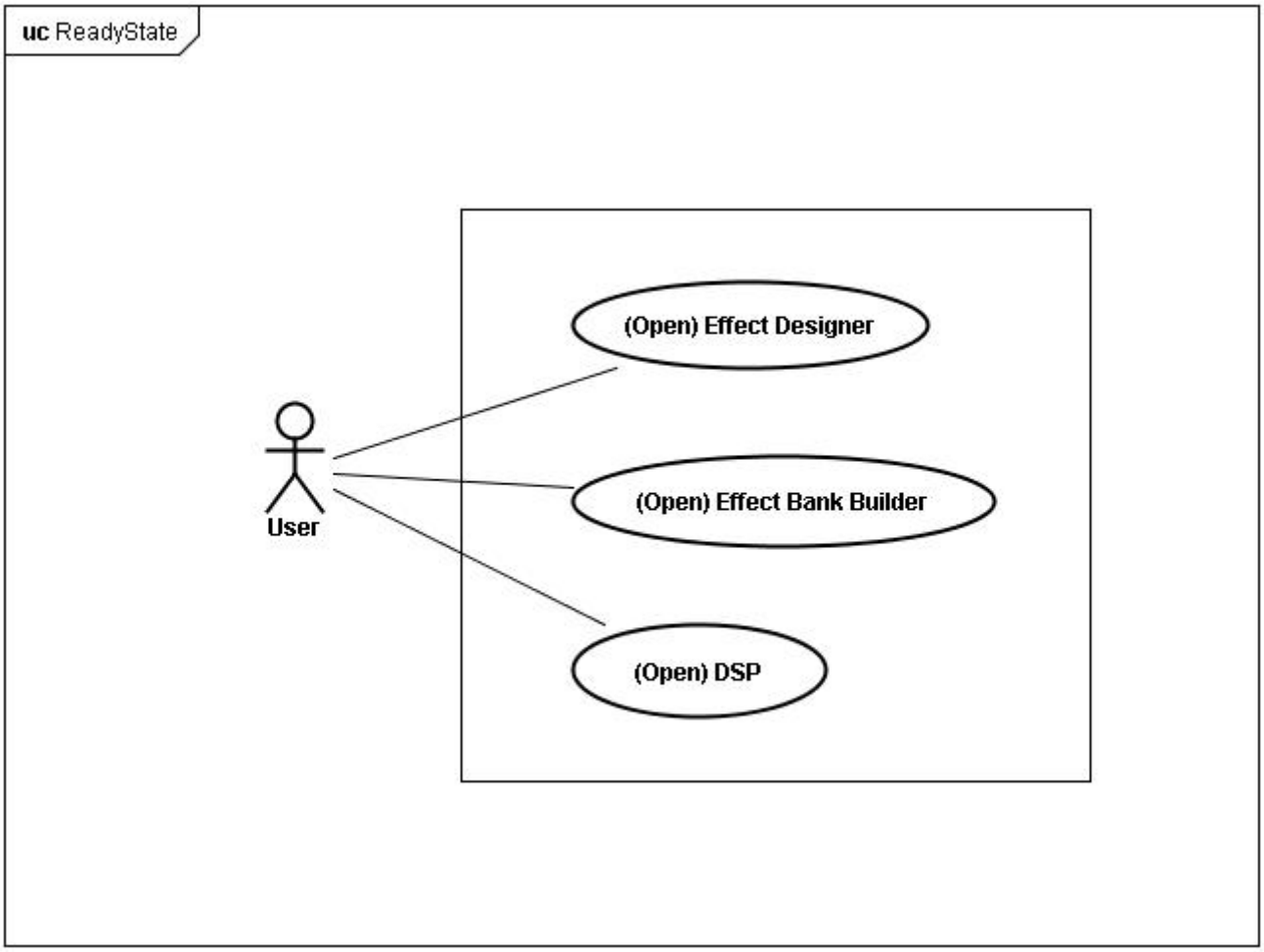
External control units for manipulating the state of the application. By providing a set of foot switches the user would be able to assign events to them such as to step up or down through the effect-chains in a bank, or to alter configuration settings on the effects. A potentiometer style-control (as used on whammy and wah-wah pedals) could also be provided for the same purpose but to function dynamically as opposed to in a stepping fashion.

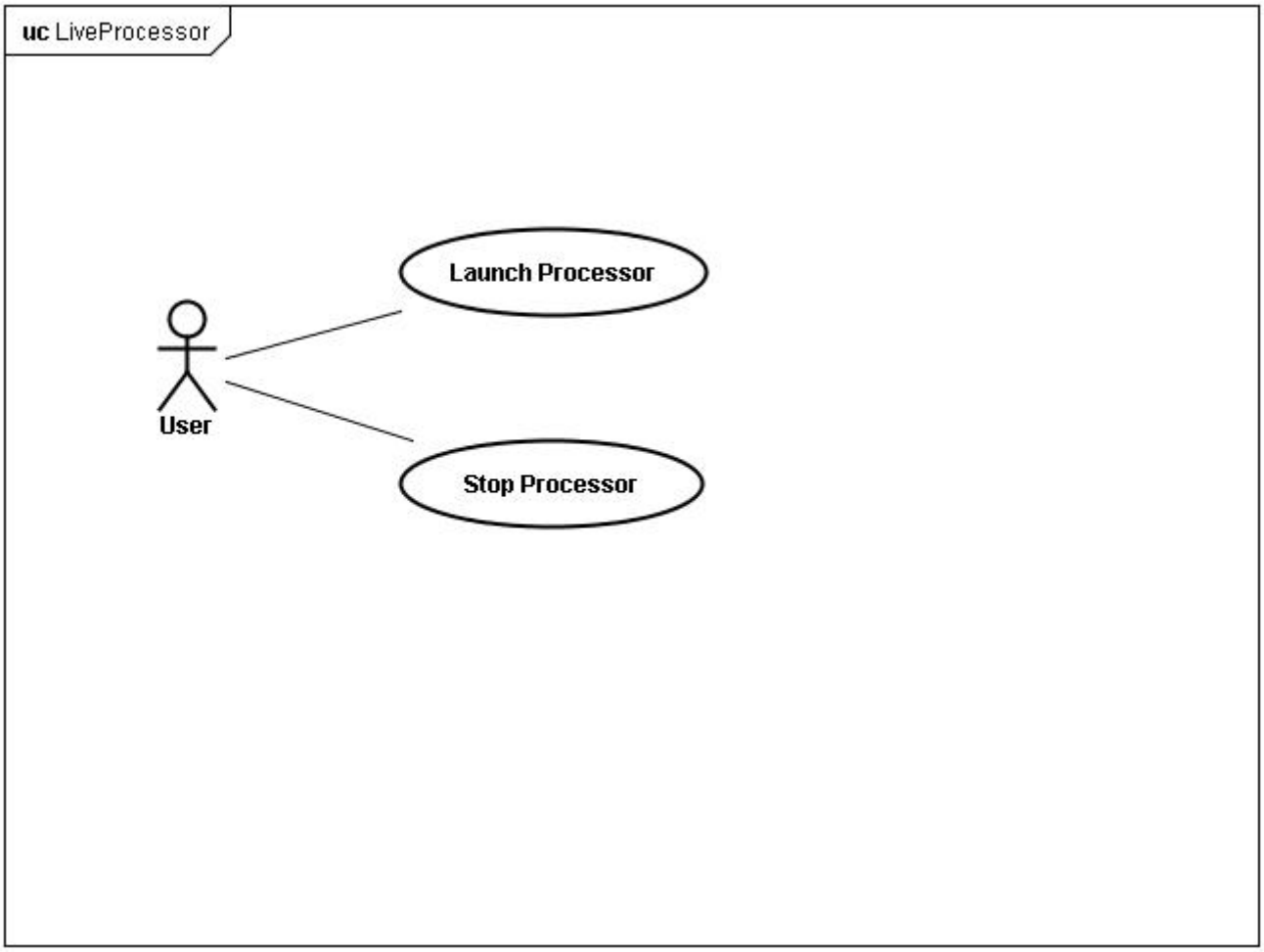
This would involve the development of drivers for each platform the application runs on, and incorporating into the application the ability to assign configuration attributes to handles to switches via the driver.

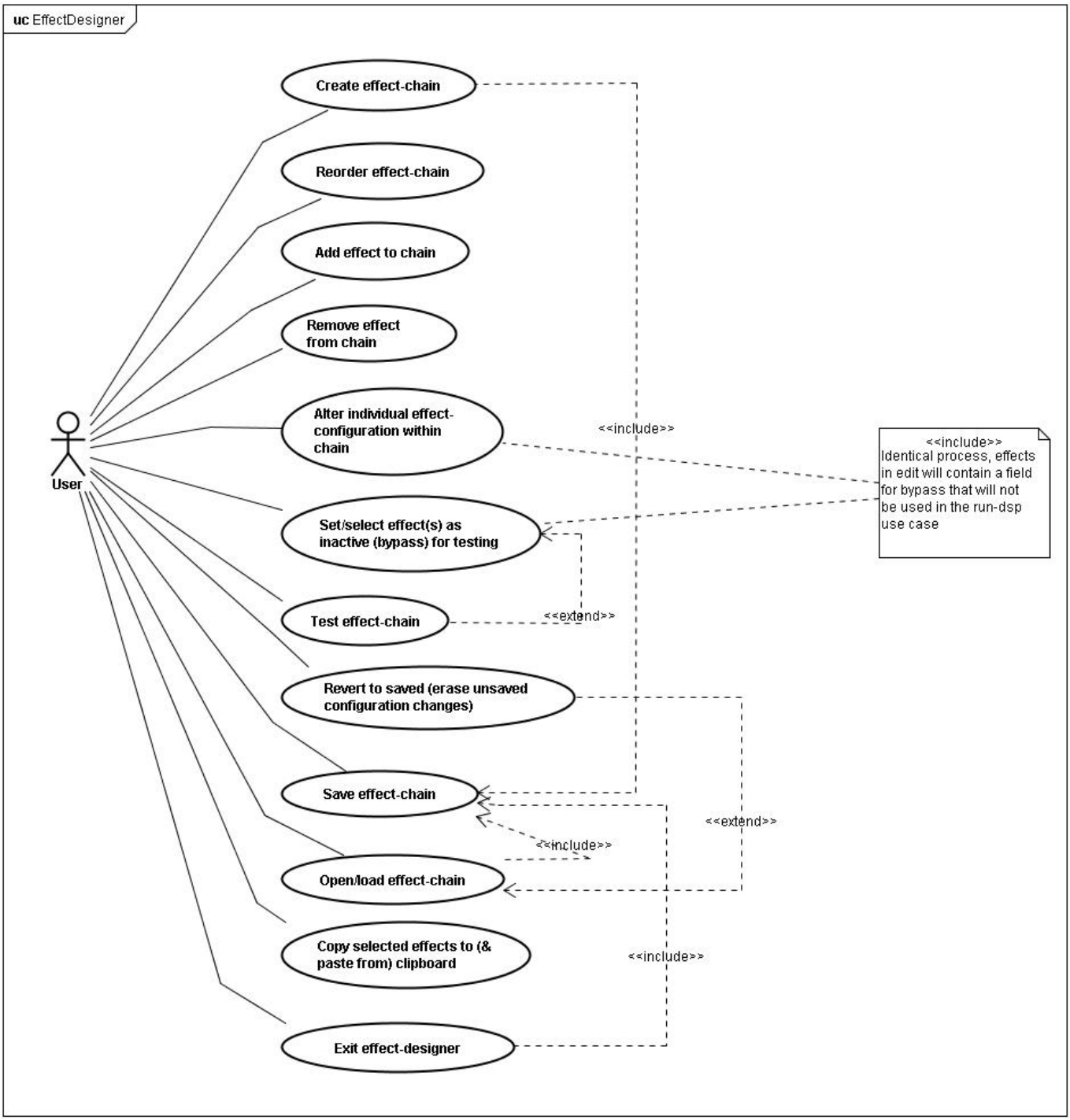
Sample accuracy

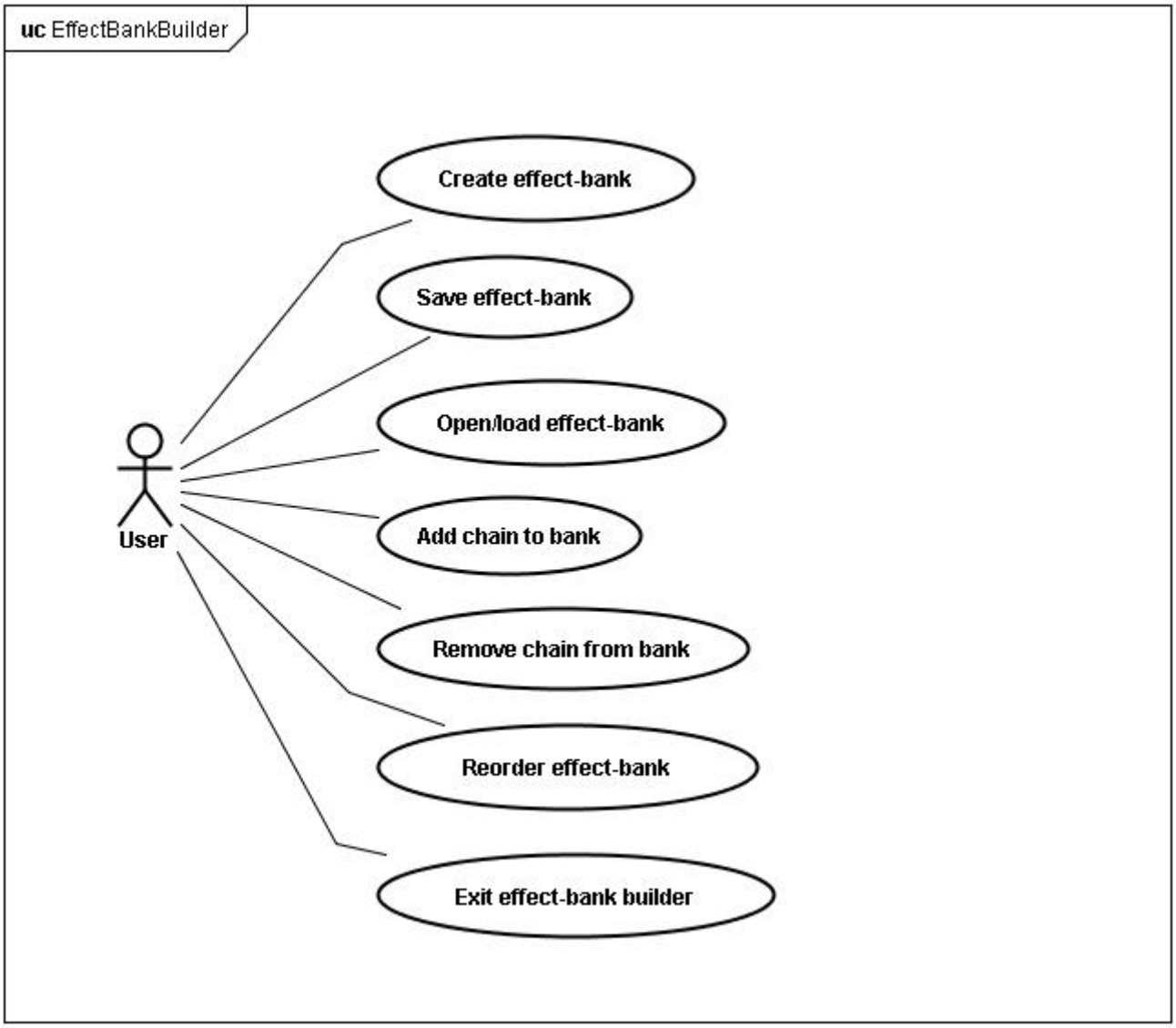
Reimplement the code for the channel buffers using double values instead of floats by default. The benefit of this is in the increased accuracy and thus lower levels of quantisation. This is arguable however as the added precision will add to the calculation latency so tests would be required to provide a basis for a recommendation.

APPENDIX B: System Design


uc ReadyState
(Open) Effect Designer
User
(Open) Effect Bank Builder
(Open) DSP

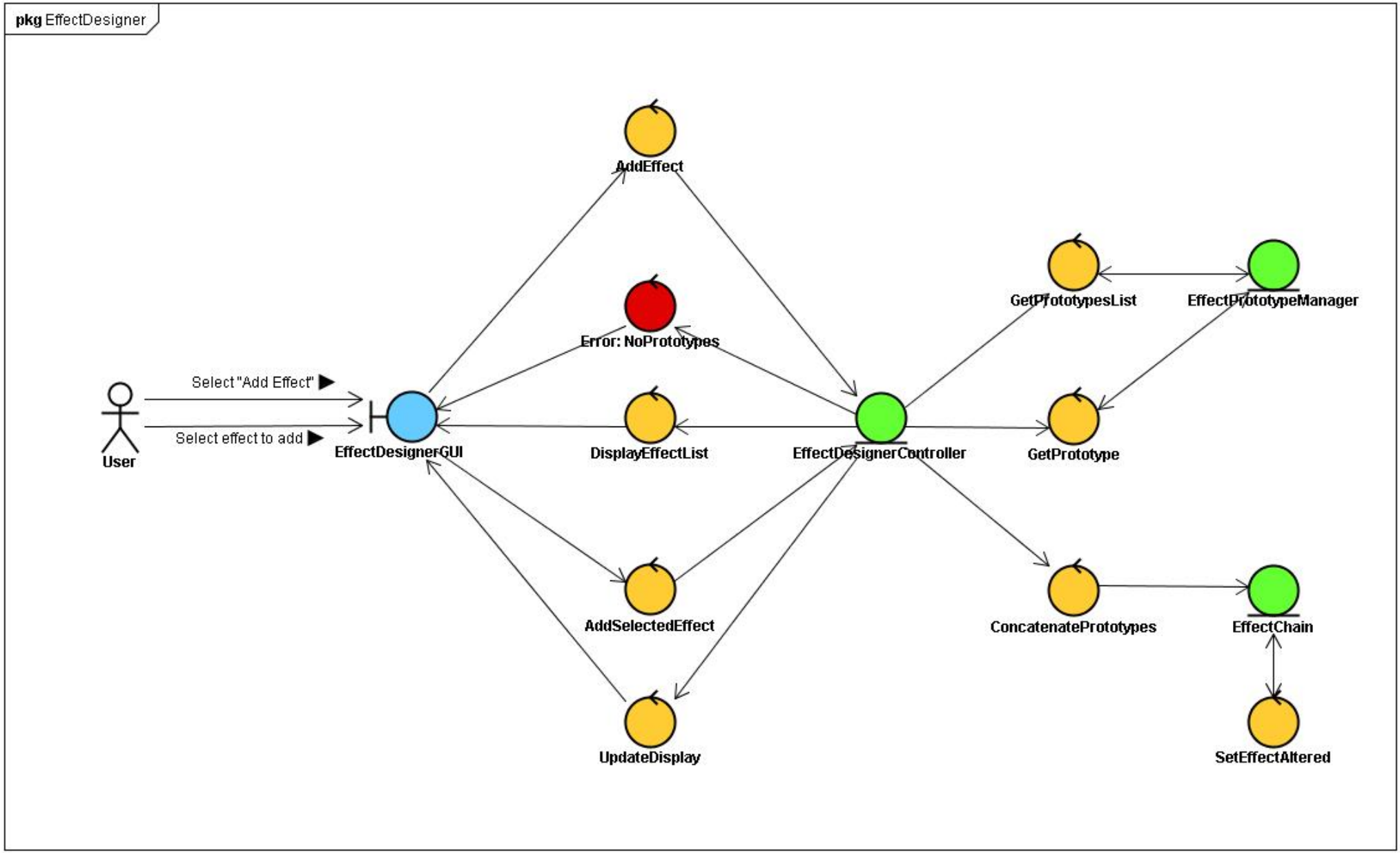

uc LiveProcessor
Launch Processor
User
Stop Processor

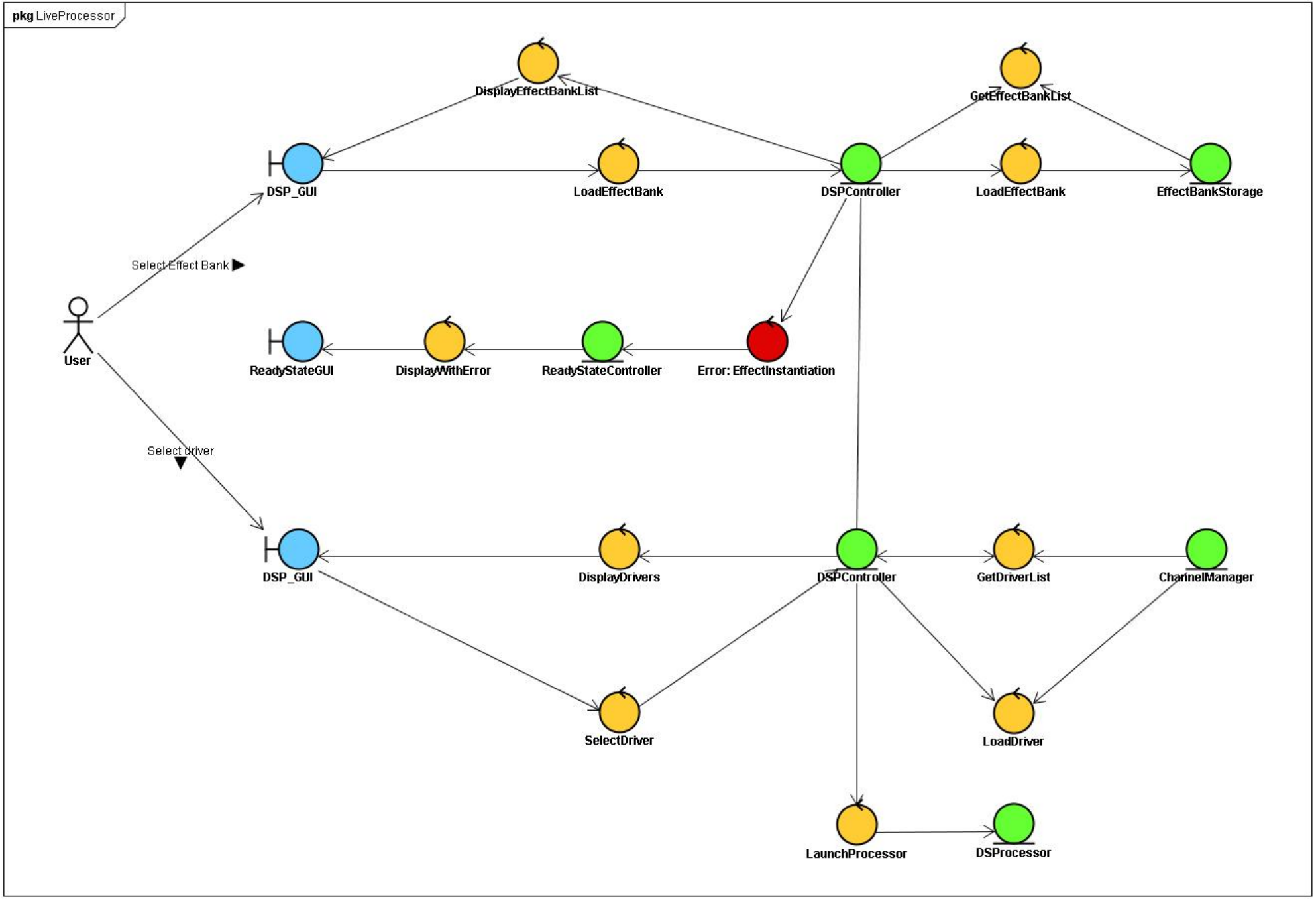

uc EffectDesigner
Create effect-chain
Reorder effect-chain
Add effect to chain
Remove effect from chain
Alter individual effect-configuration within chain
Set/select effect(s) as inactive (bypass) for testing
Test effect-chain
Revert to saved (erase unsaved configuration changes)
Save effect-chain
Open/load effect-chain
Copy selected effects to (& paste from) clipboard
Exit effect-designer
User
<<include>>
<<include>>
Identical process, effects in edit will contain a field for bypass that will not be used in the run-dsp use case
<<extend>>
<<extend>>
<<include>>
<<include>>

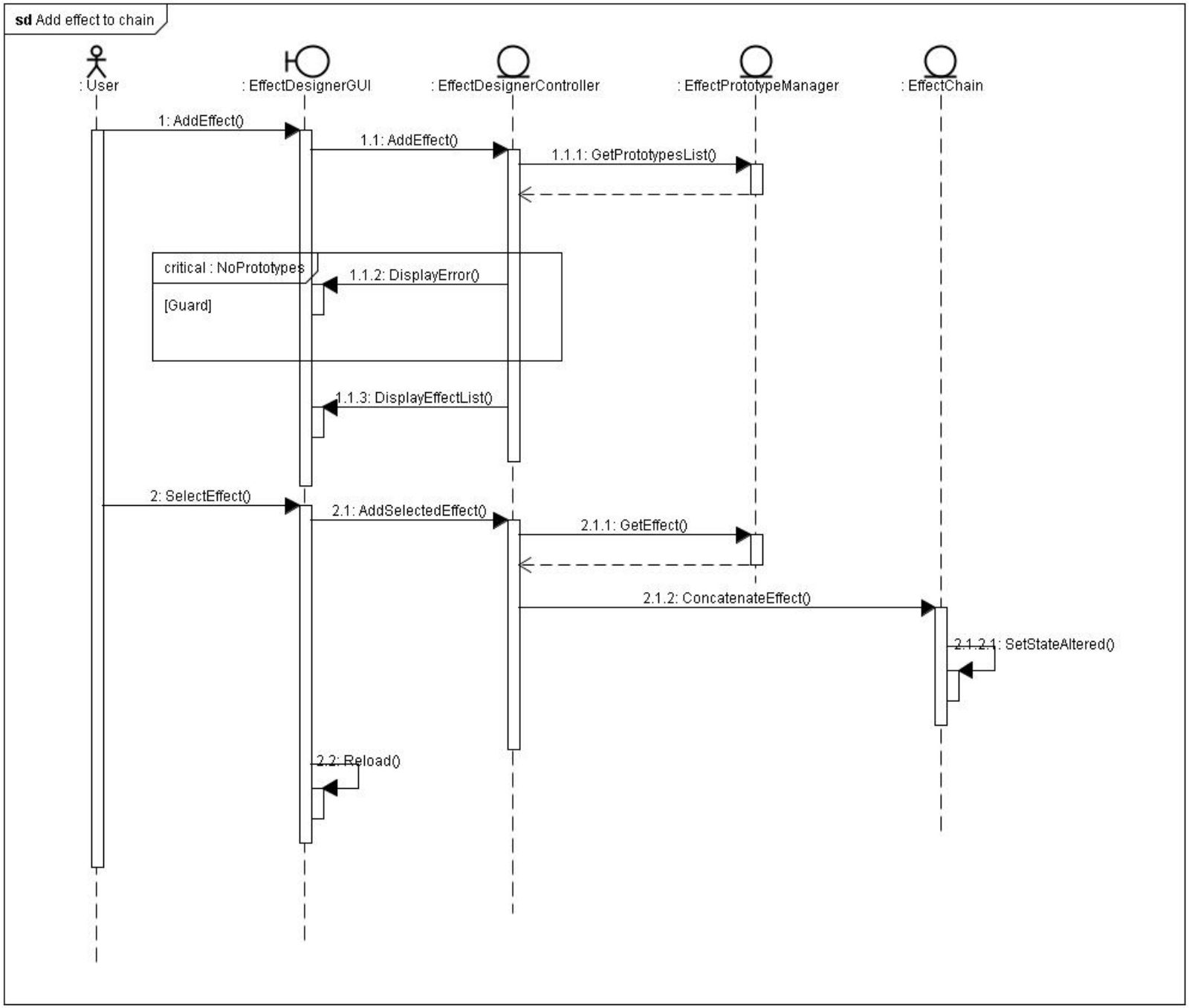

uc EffectBankBuilder
Create effect-bank
Save effect-bank
Open/load effect-bank
Add chain to bank
Remove chain from bank
Reorder effect-bank
Exit effect-bank builder
User

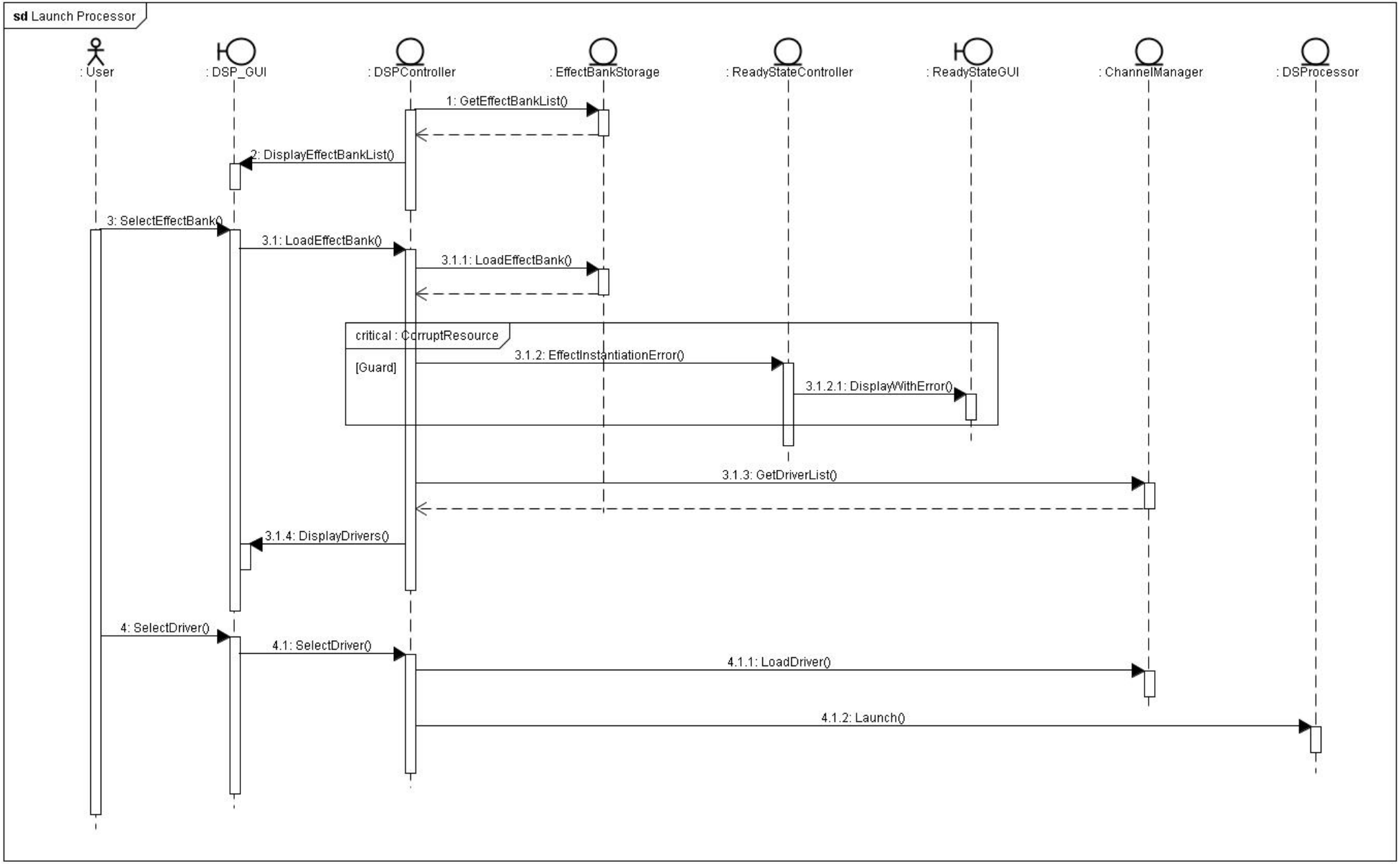
	Use Case: Add effect to chain		
step	User action	System Response	Alternative Actions ref
1	Select "Add-Effect" option		
2		Provide list of available unconfigured prototypes	#1
3	Select effect to add		#2
4		Concatenate effect onto chain	
5		Mark chain as changed (unsaved)	
	Alternative courses of action		
#1	**No prototype effects are available**		
step	User action	System Response	
1		Display error	
2	Acknowledge error		
3		Return to effect-designer interface	
#2	**User cancels action**		
step	User action	System Response	
1	Select "Cancel" option		
2		Return to effect-designer interface	

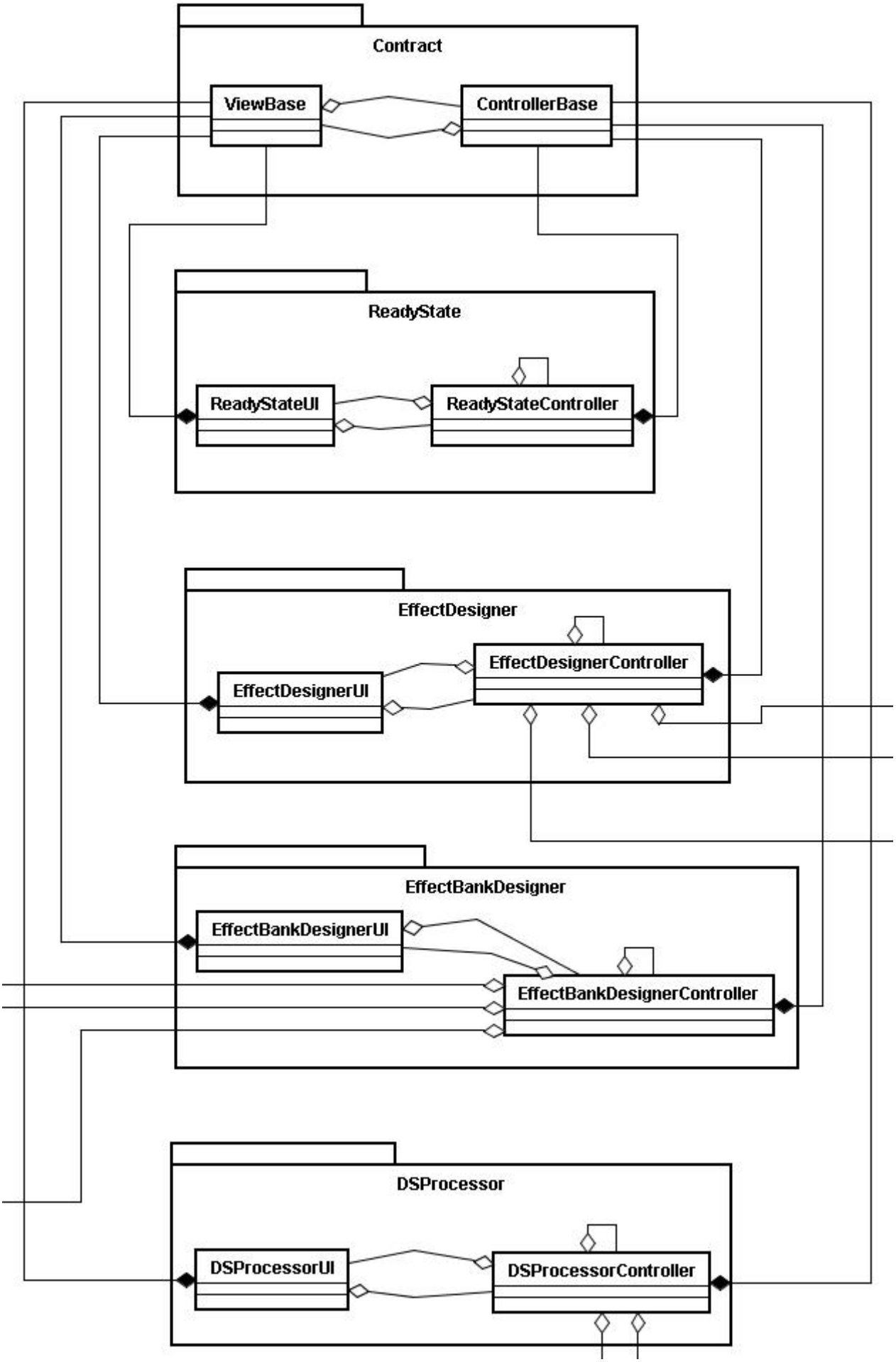
	Use Case: Run processor		
step	**User action**	System Response	Alternative Actions ref
1	**Select to run the effect-processor system proper**		
2		Check for ASIO drivers installed on the system	
3		Display a list of available effect-banks to be loaded for use	#1
4	**Select a bank of effects for the system to load**		#2
5		Load effects bank into memory	#3
6		Display a list of available ASIO drivers	
7	**Select ASIO driver to be used by the system**		
8		Load drivers for system usage	
9		Run DSP execution	
10	**Select "Exit" option**		
11		Return to Ready-state interface	
	Alternative courses of action		
#1	**No ASIO drivers are available**		
#2	**No effect-banks have been created and stored**		
#3	**Error occurs while loading effects-bank**		
step	**User action**	System Response	
1		Display error	
2	**Acknowledge error**		
3		Return to Ready-state interface	
	Use Case: Run processor		

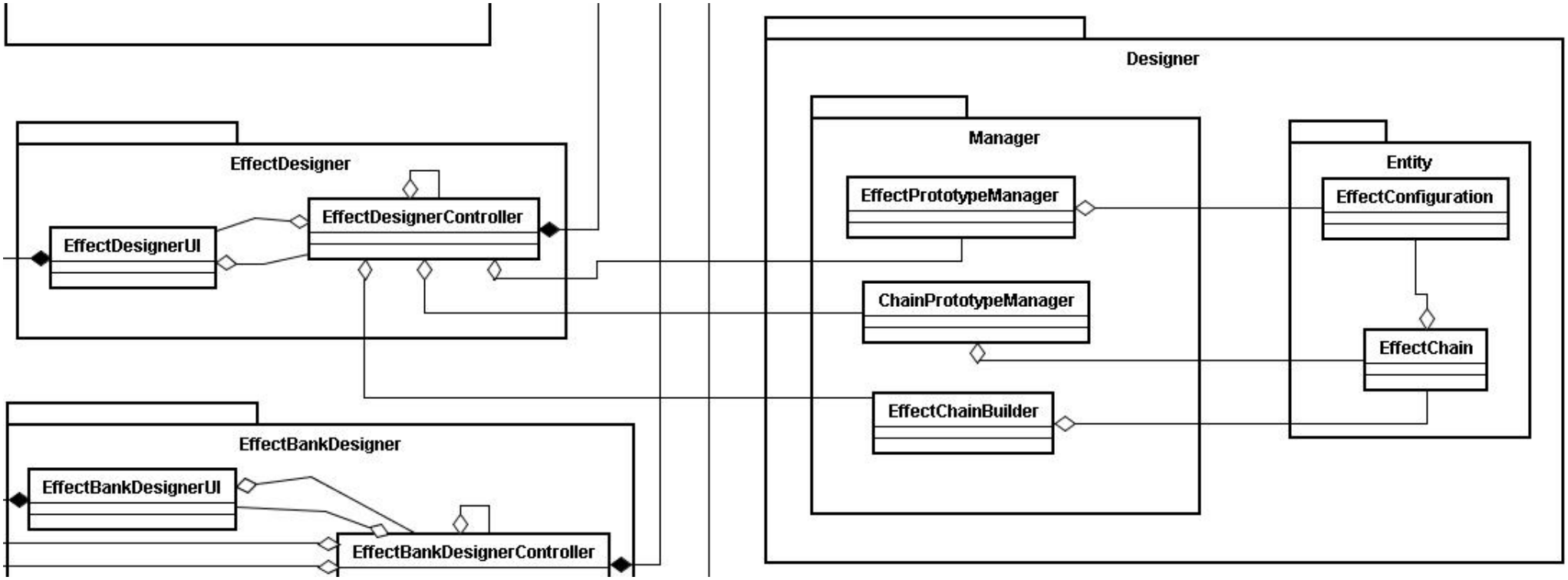

pkg EffectDesigner
AddEffect
Error: NoPrototypes
GetPrototypesList
EffectPrototypeManager
Select "Add Effect"
Select effect to add
User
EffectDesignerGUI
DisplayEffectList
EffectDesignerController
GetPrototype
AddSelectedEffect
ConcatenatePrototypes
EffectChain
UpdateDisplay
SetEffectAltered

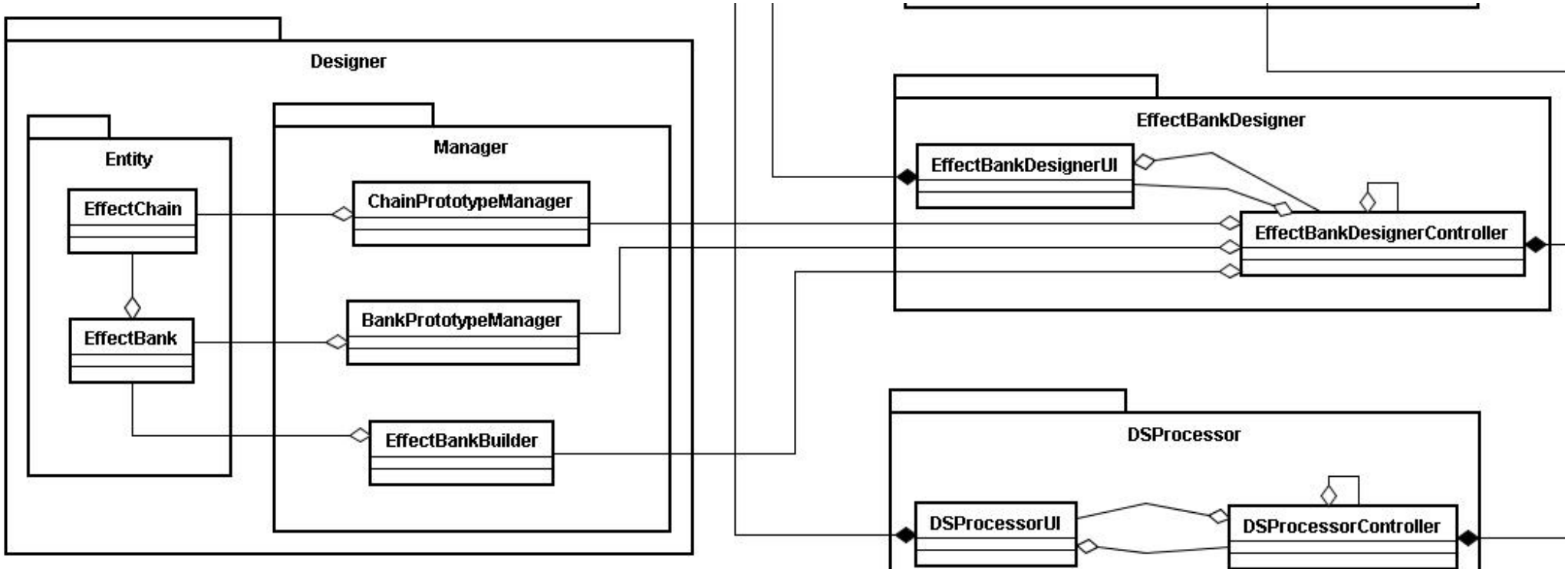

pkg LiveProcessor
DisplayEffectBankList
GetEffectBankList
DSP_GUI
LoadEffectBank
DSPController
LoadEffectBank
EffectBankStorage
Select Effect Bank
User
ReadyStateGUI
DisplayWithError
ReadyStateController
Error: EffectInstantiation
Select driver
DSP_GUI
DisplayDrivers
DSPController
GetDriverList
ChannelManager
SelectDriver
LoadDriver
LaunchProcessor
DSProcessor

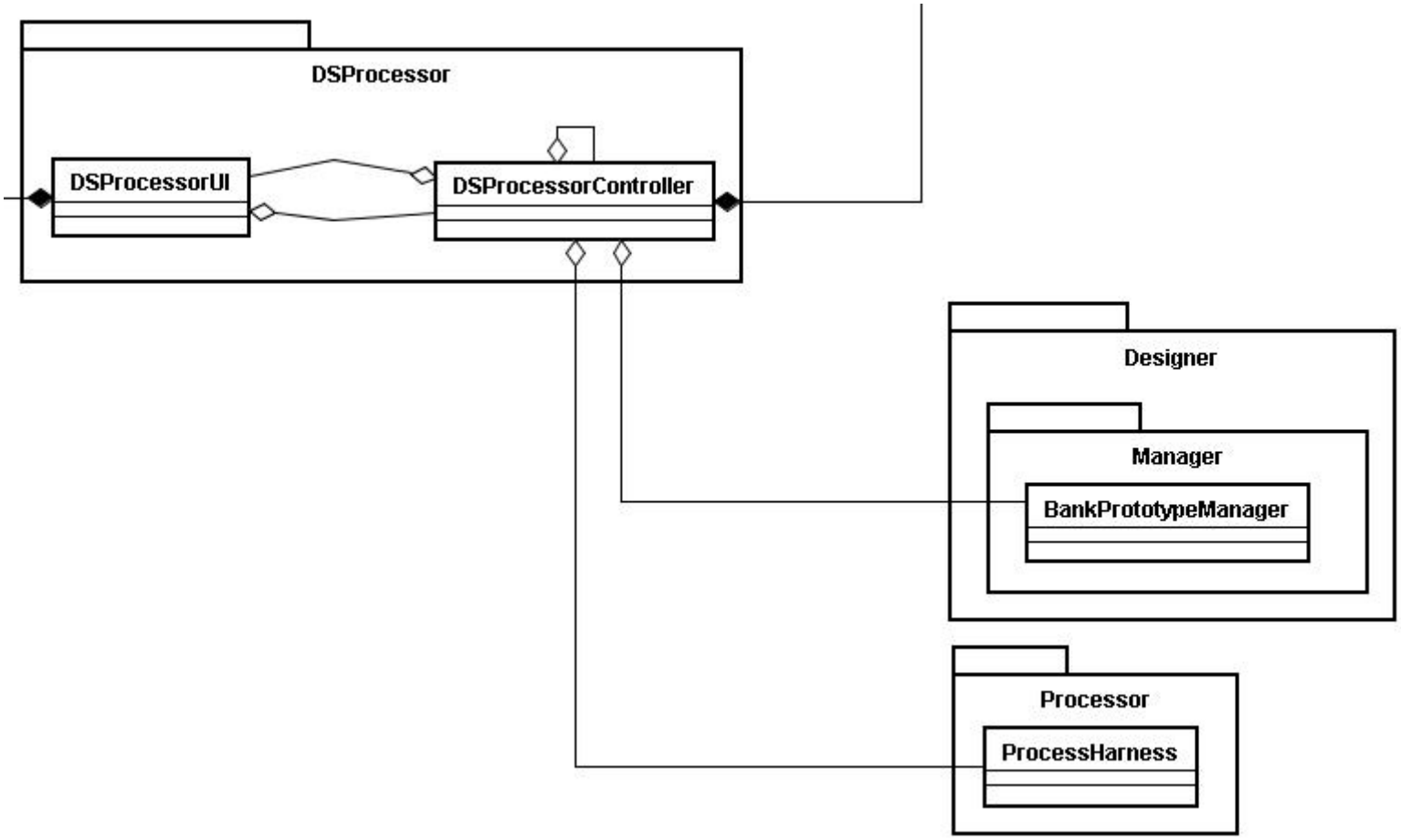

sd Add effect to chain
: User
: EffectDesignerGUI
: EffectDesignerController
: EffectPrototypeManager
: EffectChain
1: AddEffect()
1.1: AddEffect()
1.1.1: GetPrototypesList()
critical : NoPrototypes
[Guard]
1.1.2: DisplayError()
1.1.3: DisplayEffectList()
2: SelectEffect()
2.1: AddSelectedEffect()
2.1.1: GetEffect()
2.1.2: ConcatenateEffect()
2.1.2.1: SetStateAltered()
2.2: Reload()

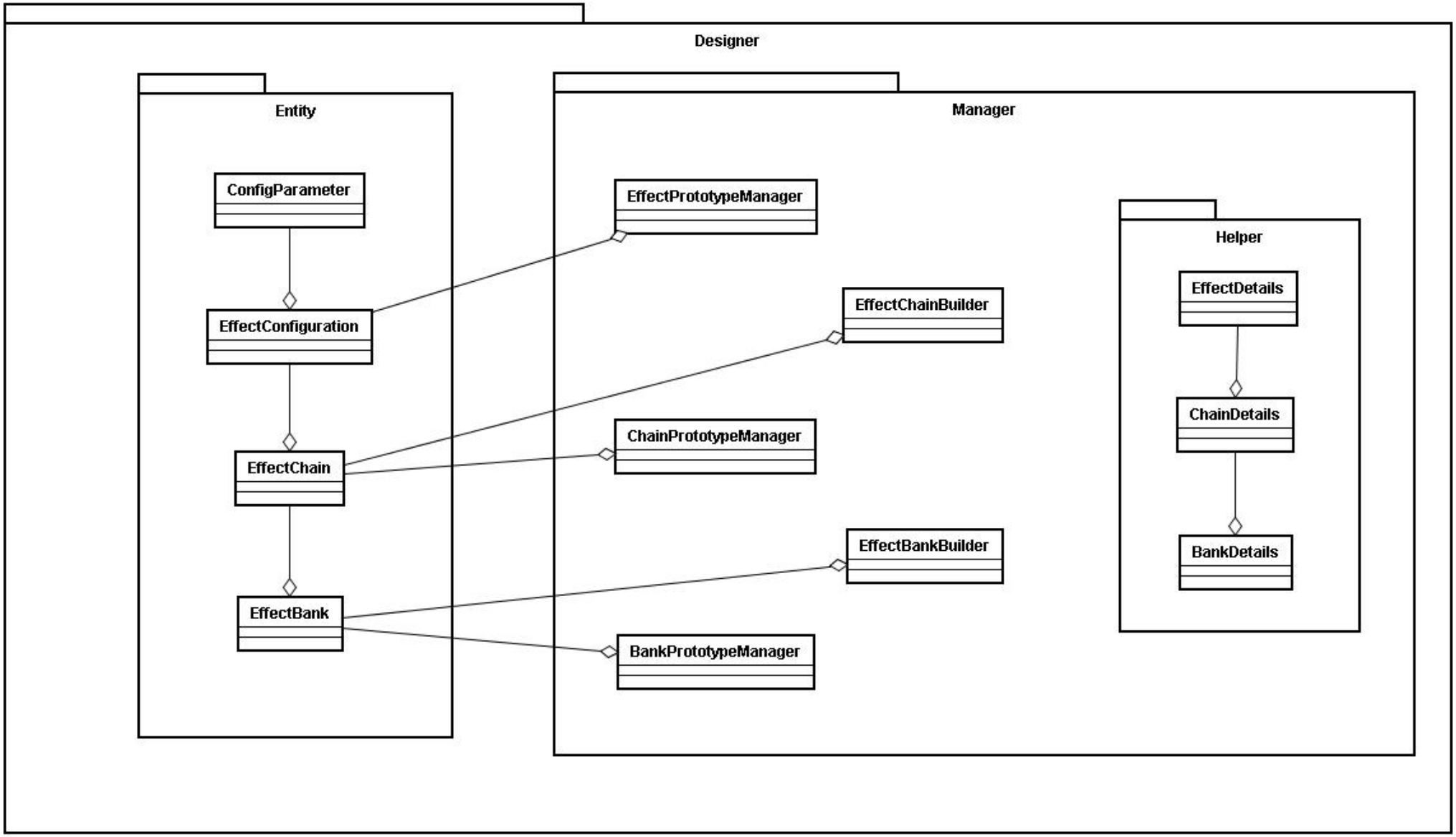

sd Launch Processor
: User
: DSP_GUI
: DSPController
: EffectBankStorage
: ReadyStateController
: ReadyStateGUI
: ChannelManager
: DSProcessor
1: GetEffectBankList()
2: DisplayEffectBankList()
3: SelectEffectBank()
3.1: LoadEffectBank()
3.1.1: LoadEffectBank()
critical : CorruptResource
[Guard]
3.1.2: EffectInstantiationError()
3.1.2.1: DisplayWithError()
3.1.3: GetDriverList()
3.1.4: DisplayDrivers()
4: SelectDriver()
4.1: SelectDriver()
4.1.1: LoadDriver()
4.1.2: Launch()

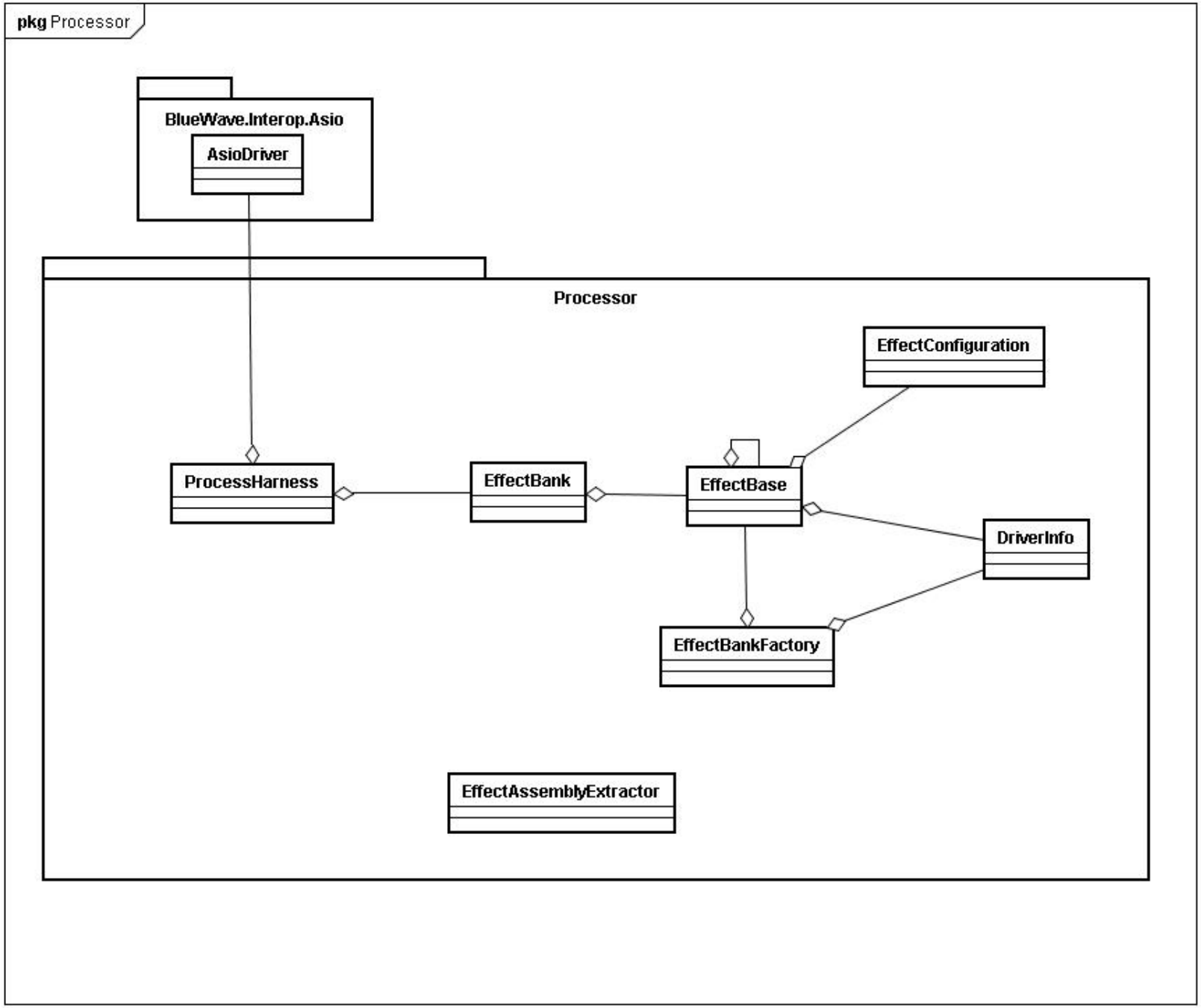

Contract
ViewBase
ControllerBase
ReadyState
ReadyStateUI
ReadyStateController
EffectDesigner
EffectDesignerController
EffectDesignerUI
EffectBankDesigner
EffectBankDesignerUI
EffectBankDesignerController
DSProcessor
DSProcessorUI
DSProcessorController


Designer
Manager
EffectPrototypeManager
ChainPrototypeManager
EffectChainBuilder
Entity
EffectConfiguration
EffectChain
EffectDesigner
EffectDesignerUI
EffectDesignerController
EffectBankDesigner
EffectBankDesignerUI
EffectBankDesignerController


Designer
Entity
EffectChain
EffectBank
Manager
ChainPrototypeManager
BankPrototypeManager
EffectBankBuilder
EffectBankDesigner
EffectBankDesignerUI
EffectBankDesignerController
DSProcessor
DSProcessorUI
DSProcessorController


DSProcessor
DSProcessorUI
DSProcessorController
Designer
Manager
BankPrototypeManager
Processor
ProcessHarness


Designer
Entity
ConfigParameter
EffectConfiguration
EffectChain
EffectBank
Manager
EffectPrototypeManager
EffectChainBuilder
ChainPrototypeManager
EffectBankBuilder
BankPrototypeManager
Helper
EffectDetails
ChainDetails
BankDetails


pkg Processor
BlueWave.Interop.Asio
AsioDriver
Processor
EffectConfiguration
ProcessHarness
EffectBank
EffectBase
DriverInfo
EffectBankFactory
EffectAssemblyExtractor

APPENDIX D: Black Box Testing

Test No.	Test Description	Expected Result	Actual Result
1	Main Menu: Effect Chain Designer button	Closes main menu, opens Effect-Chain designer window	as expected
2	Main Menu: Effect Bank Designer button	Closes main menu, opens Effect-bank designer window	as expected
3	Main Menu: Run live processor button	Closes main menu, opens live processor interface	as expected
4	Effect-Chain designer: Add effect with no chain loaded	not allowed	as expected
5	Effect-Chain designer: File->new with no chain loaded	form displays new (unsaved)	as expected
6	Effect-Chain designer: File->new with chain loaded, no unsaved changes	form displays new (unsaved)	as expected
7	Effect-Chain designer: File->new with chain loaded, with unsaved changes	prompt for confirmation to discard the current chain	as expected
8	Effect-Chain designer: Add effect with a chain loaded	selection box appears displaying the effects stored in the system	as expected
9	Effect-Chain designer: Select Effect from list	selected item appears as a new entry on the end of the chain, (unsaved) appears for effect-chains that were saved previously	as expected
10	Effect-Chain designer: Effect-module: shift up	The item in question is raised to one point higher in the chain, (unsaved) appears for effect-chains that were saved previously	as expected
11	Effect-Chain designer: Effect-module: shift down	The item in question is lowered to one point lower in the chain, (unsaved) appears for effect-chains that were saved previously	as expected

12	Effect-module: edit configuration	configuration screen for the selected item is displayed on the screen, (unsaved) appears for effect-chains that were saved previously	as expected
13	Configuration: apply changes	changes are persisted to the configuration of the object in question, (unsaved) appears for effect-chains that were saved previously	as expected
14	Effect module: remove from chain	the selected item is removed frmo the chain, (unsaved) appears for effect-chains that were saved previously	as expected
15	Effect-Chain designer: File->save for previously unsaved chain	"Save As.." dialogue appears	as expected
16	Effect-Chain designer: File->save as for previously unsaved chain	"Save As.." dialogue appears	as expected
17	Effect-Chain designer: File->save for previously saved chain	alterations are persisted, any (unsaved) marker disappears	as expected
18	Effect-Chain designer: File->save as for previously saved chain	"Save As.." dialogue appears	as expected
19	Effect-Chain designer: Save as: select item from list	Item appears in Name field	as expected
20	Effect-Chain designer: Save chain using existing name	Confirmation requested	as expected
21	Effect-Chain designer: Confirmation granted	Chain is persisted, (unsaved) removed from chain status if present	as expected
22	Effect-Chain designer: Confirmation cancelled	User returns to save-as dialogue	as expected
23	Effect-Chain designer: Save chain using unique name	Chain is persisted, (unsaved) removed from chain status if present	as expected
24	Effect-Chain designer: File->open selected	Open-chain dialogue appears	as expected
25	Effect-Chain designer: open selected chain	The content of the selected ityem is displayed on the screen	as expected

26	Effect-Chain designer: Exit with unsaved changes	prompt for confirmation	as expected
27	Effect-Chain designer: cancel confirmation	user returns to current edit	as expected
28	Effect-Chain designer: confirm action	changes are discarded, user is returned to the main menu	as expected
29	Effect-Chain designer: exit without unsaved changes	user returns to main menu	as expected
30	Effect-Bank designer: Add chain with no bank loaded	not allowed	as expected
31	Effect-Bank designer: File->new with no bank loaded	form displays new (unsaved)	as expected
32	Effect-Bank designer: File->new with bank loaded, no unsaved changes	form displays new (unsaved)	as expected
33	Effect-Bank designer: File->new with bank loaded, with unsaved changes	prompt for confirmation to discard the current bank	as expected
34	Effect-Bank designer: Add chain with a bank loaded	selection box appears displaying the chains stored in the system	as expected
35	Effect-Bank designer: Select chain from list	selected item appears as a new entry on the end of the bank, (unsaved) appears for effect-banks that were saved previously	as expected
36	Effect-Bank designer: Effect-chain: shift up	The item in question is raised to one point higher in the bank list, (unsaved) appears for effect-banks that were saved previously	as expected
37	Effect-Bank designer: Effect-chain: shift down	The item in question is lowered to one point lower in the bank list, (unsaved) appears for effect-banks that were saved previously	as expected
38	Effect-chain item: view details	details screen is displayed showing the user the structure of the chain	as expected

39	Effect chain: remove from bank	the selected item is removed from the bank, (unsaved) appears for effect-banks that were saved previously	as expected
40	Effect-Bank designer: File->save for previously unsaved bank	"Save As.." dialogue appears	as expected
41	Effect-Bank designer: File->save as for previously unsaved bank	"Save As.." dialogue appears	as expected
42	Effect-Bank designer: File->save for previously saved bank	alterations are persisted, any (unsaved) marker disappears	as expected
43	Effect-Bank designer: File->save as for previously saved bank	"Save As.." dialogue appears	as expected
44	Effect-Bank designer: Save as: select item from list	Item appears in Name field	as expected
45	Effect-Bank designer: Save bank using existing name	Confirmation requested	as expected
46	Effect-Bank designer: Confirmation granted	Bank is persisted, (unsaved) removed from bank status if present	as expected
47	Effect-Bank designer: Confirmation cancelled	User returns to save-as dialogue	as expected
48	Effect-Bank designer: Save bank using unique name	Bank is persisted, (unsaved) removed from bank status if present	as expected
49	Effect-Bank designer: File->open selected	Open-bank dialogue appears	as expected
50	Effect-Bank designer: open selected bank	The content of the selected ityem is displayed on the screen	as expected
51	Effect-Bank designer: Exit with unsaved changes	prompt for confirmation	as expected
52	Effect-Bank designer: cancel confirmation	user returns to current edit	as expected
53	Effect-Bank designer: confirm action	changes are discarded, user is returned to the main menu	as expected
54	Effect-Bank designer: exit without unsaved changes	user returns to main menu	as expected
55	Live Processor: start processor with no driver selected	Display error	as expected
56	Live Processor: start processor with a driver and no bank selected	Display error	as expected

57	Live Processor: select driver	display driver details in the text-panel, bring up driver interface	as expected
58	Live processor: select bank	display list of all members of the selected bank in the list box	as expected
59	Live processor: start processor with bank and driver selected	present the transformed sound signals through the selected sound channels	as expected
60	Live processor: switch active chain	highlight the next member of the list (or the first in the liust if the current is the last), present the new transformed sound signals through the selected sound channels	as expected
61	Live processor: stop processing	cut all sound channels	as expected
62	Live processor: exit while processing is taking place	release the driver instance, return the user to the main menu	as expected
63	Live processor: exit cleanly	release the driver instance, return the user to the main menu	as expected
64	Main menu: exit application	close application instance, remove main menu UI from screen, close application processes	as expected

APPENDIX G: Latency Test Results

Test Setup

Application

Effect instances in chain: 2
Number of delayed repeats: 5
All to begin replaying within: 500ms
Buffer size: 128 samples
Sample rate: 44,000 hz
Data type: *float* (to simulate 16 bit)

Time measured on # of buffers required for processing 1 second of audio data.

Hardware Setup #1

Processor: x86 AMD Athlon 998Mhz
Memory: 256 mb

Results #1

Language / Compiler	Approximate Average Latency (ms)
Java	30
C#	30
C++	22

Hardware Setup #2

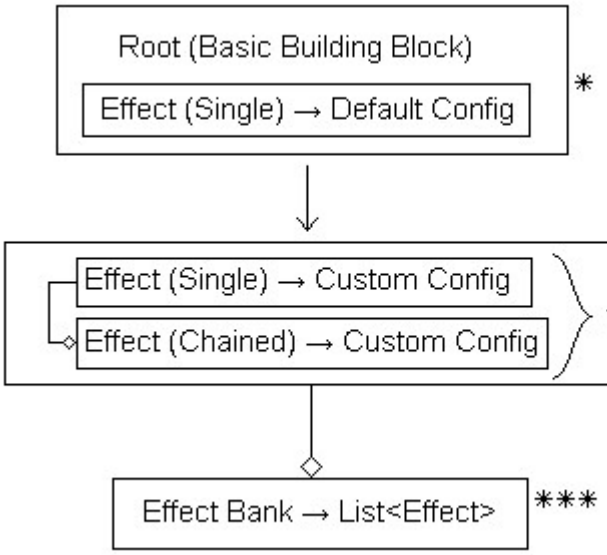
Processor: x86 AMD Athlon 1808Mhz
Memory: 1536 mb

Results #2

Language / Compiler	Approximate Average Latency (ms)
Java	12
C#	30
C++	22

APPENDIX H: File Structures

System storage mechanism design



FileSystem

* /DefaultPrototypes/*.zip
Effect_x.zip → Effect_x.dll
→ Effect_x.xml

** /EffectChains/*.xml
Chain_A.xml
Chain_B.xml

*** /EffectBanks/*.xml
EffectBank_A.xml
EffectBank_B.xml

Effect configurations, housed in config.xml (part of teh .zip file containing the compiled effect unit class):

```
<effect name="effect_a">
     <param>
          <name>ConfigOption_A</name>
          <value>100</value>
     </param>
     <param>
          <name>ConfigOption_B</name>
          <value>200</value>
     </param>
     <param>
          <name>ConfigOption_C</name>
          <value>150</value>
     </param>
</effect>
```

EffectChains:

```
     <effectChain name="effectChain_a">
          <effect name="effect_a">
               <param>
                    <name>ConfigOption_A</name>
                    <value>222</value>
               </param>
               <param>
                    <name>ConfigOption_B</name>
                    <value>456</value>
               </param>
               <param>
                    <name>ConfigOption_C</name>
                    <value>802</value>
               </param>
          </effect>
          <effect name="effect_b">
               <param>
                    <name>ConfigOption_A</name>
                    <value>1</value>
               </param>
               <param>
                    <name>ConfigOption_B</name>
                    <value>2</value>
               </param>
               <param>
                    <name>ConfigOption_C</name>
                    <value>3</value>
               </param>
          </effect>
     </effectChain>
```

EffectBanks:

```
<effectBank name="effectBank_a">
     <effectChain name="effectChain_a">
          <effect name="effect_a">
               <param>
                    <name>ConfigOption_A</name>
                    <value>222</value>
               </param>
               <param>
                    <name>ConfigOption_B</name>
                    <value>456</value>
               </param>
               <param>
                    <name>ConfigOption_C</name>
                    <value>802</value>
               </param>
          </effect>
          <effect name="effect_b">
               <param>
                    <name>ConfigOption_A</name>
                    <value>1</value>
```

```
            </param>
            <param>
                <name>ConfigOption_B</name>
                <value>2</value>
            </param>
            <param>
                <name>ConfigOption_C</name>
                <value>3</value>
            </param>
        </effect>
    </effectChain>
    <effectChain name="effectChain_b">
        <effect name="effect_c">
            <param>
                <name>ConfigOption_A</name>
                <value>222</value>
            </param>
            <param>
                <name>ConfigOption_B</name>
                <value>456</value>
            </param>
            <param>
                <name>ConfigOption_C</name>
                <value>802</value>
            </param>
        </effect>
        <effect name="effect_a">
            <param>
                <name>ConfigOption_A</name>
                <value>1</value>
            </param>
            <param>
                <name>ConfigOption_B</name>
                <value>2</value>
            </param>
            <param>
                <name>ConfigOption_C</name>
                <value>3</value>
            </param>
        </effect>
    </effectChain>
</effectBank>
```