If you use the software design pattern, reusing good design and architecture become easy. The pattern helps to choose a design for reuse and to exclude interrupted design for reuse. Also, the pattern can improve the maintenance and the documentation, and we can correct specifications of the class and define interaction between objects & intention for design. If you use Pattern Add-In of StarUML, it will guarantee to maximize the productivity and quality because the design pattern is easily and quickly adapted to user's "better" design.

Index of Pattern Add-In Help

- 1. Basic Concepts
- 2. Pattern Add-In Overview
- 3. Using Pattern Add-In
- 4. <u>FAQ</u>

This chapter introduces the basic concepts required for an effective use of Pattern Add-In. Included are a brief introduction to patterns and the list of predefined patterns.

- Introduction to Patterns
- Pattern Definition List

What is a Design Pattern?

- 1. A design pattern refers to "an editable object or class that is mutually shared for solving general design problems in specific situations."
 - Well-designed object-oriented structures are those that contain many
- 2. patterns. In software development, design patterns are put together as solution packages by expert designers to help other developers.

A design pattern is an abstraction of the main elements in a common design 3. structure useful for making reusable object-oriented designs.

A design pattern identifies and defines roles for the classes and their

4. instances participating in the pattern, defining relations and assigning responsibilities among them.

Pattern Add-In contains definitions of 23 GoF design patterns and 3 EJB patterns.

GoF Design Pattern List:

- 1. **Abstract Factory**: Provides an interface for organizing a group of many related or independent objects without defining sub-classes.
- 2. **Adapter**: Operates different classes of interface together.

Bridge: Separates the implementation and abstraction concepts. By means 3. of it, implementation can undergo various changes as an abstraction concept, and the interface also can be diverse, independent from implementation.

Builder: Separates classes defining methods for creating complex objects 4. and expressing them. Provides the same construction process even for different expressions.

- 5. **Chain of Responsibility**: Removes the coupling between message-sending objects and message-processing objects.
- 6. **Command**: Treats a request as an object. Also treats other requests as objects, relaying them as parameters to clients.
- 7. **Composite** : Allows the client to treat individual objects and composite objects alike.
- 8. **Decorator**: Provides a more flexible way of adding functions than creating a sub-class. Allows dynamic addition of new services in an object.

Facade: Defines a comprehensive concept interface for easy use of a

- 9. subsystem.
- **Factory Method**: Used to define an interface for creating an object while 10. deferring to the sub-class the responsibility of determining which class will have the new instance.
- 11. **Flyweight**: Introduces the concept of sharing for effective use of large-scale delicate objects.
- 12. **Interpreter**: Used for expressing a frequently recurring problem in a simple sentence.
- 13. **Iterator**: Provides a way of accessing composite object elements in sequence without having to open their internal expression formats.
- 14. **Mediator**: Allows an independent object to manage object references rather than related objects defining their references to each other.
- **Memento**: Allows restoration of an object to its saved state by identifying 15. and expressing the internal state of the object while not violating encapsulation.
- **Observer**: Notifies a change of state for one object to all other objects in a 16. one-to-many relationship and facilitates automatic modifications if necessary.
- 17. **Prototype**: States the type of object to create by using a sample instance, and uses the sample to create the object.
- 18. **Proxy**: Creates bridging objects to access other objects.
- 19. **Singleton**: Provides a single unified access to a class if it can have only one instance.

- 20. **State**: Allows an object to modify behaviors according to its internal state. The object appears to change the class.
- 21. **Strategy**: Can modify various client-independent algorithms. Clients do not need to be modified in any way even when the algorithms are modified.
- **Template Method**: The sub-class of a class can redefine algorithm process stages without changing the algorithm structure.
- 23. **Visitor**: Allows definition of a new operation, without changing the class for the element to be processed.

EJB Pattern List:

- 1. **Entity EJB**: Provides expression of business domain data as an object.
- 2. **MessageDriven EJB**: Provides a means to receive and process asynchronous JMS messages from other applications or clients.
- 3. **Session EJB**: Provides implementation of the presentation layer for a business function.

☑Note

- The user may define new patterns and apply them to the tool. For methods of defining new patterns, see the StarUML(tm) technical documentation.
- For detailed descriptions on patterns, see the pattern manuals.

This chapter contains a general overview of Pattern Add-In: main functions and configurations.

- Pattern Add-In Functions
- Pattern Add-In Configurations

Pattern Add-In provides the following function.

Design Pattern Application: Pattern Add-In can use pattern templates to create new classes or members. Templates can also be applied to existing classes and members. Once Pattern Add-In is installed, 23 GoF design patterns and 3 EJB patterns are available.

☑Note

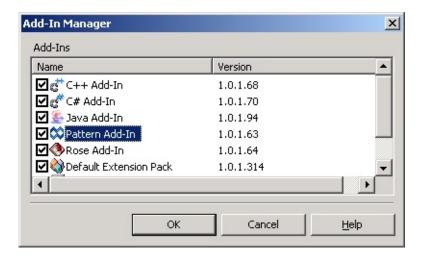
Creating and Applying User-Defined Design Patterns: This function allows the user to create new patterns and apply them to the tool. This

function allows definition and sharing of new design patterns in a project development team.

Once Pattern Add-In is installed, it is enabled for use in StarUML(tm) by default. Installed Add-Ins can be enabled or disabled through Add-In Manager in StarUML(tm). If an Add-In is disabled, no main menu and popup menu items related to it are displayed, and no StarUML(tm) events are relayed to it.

Procedure for Enabling Pattern Add-In:

- 1. Select the **[Tools] -> [Add-In Manager...]** menu in StarUML(tm).
- 2. At the Add-In Manager dialog box, check the "Pattern Add-In" checkbox in the Add-In list.



3. Click the **[OK]** button to close the dialog box.

Procedure for Disabling Pattern Add-In:

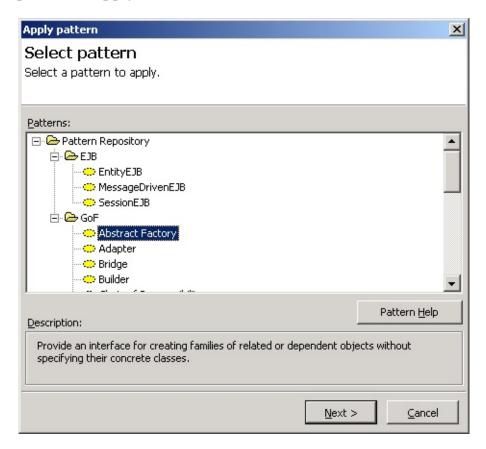
- 1. Select the **[Tools] -> [Add-In Manager...]** menu in StarUML(tm).
- 2. At the Add-In Manager dialog box, uncheck the "Pattern Add-In" checkbox in the Add-In list.
- 3. Click the **[OK]** button to close the dialog box.

This chapter describes how to use Pattern Add-In.

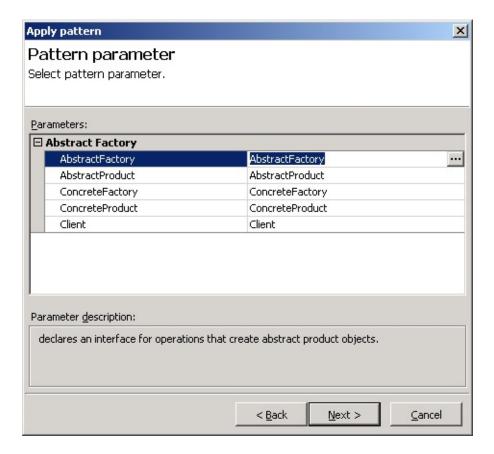
• Applying Pattern

Procedure for Applying Pattern:

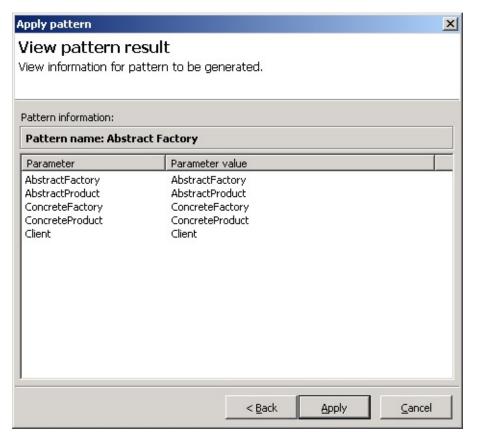
- 1. Select the **[Tools] -> [Apply Pattern...]** menu.
- 2. At the Select Pattern Type page in the Apply Pattern dialog box, select a pattern to apply from the "Patterns" list and click the **[Next]** button.



3. At the Pattern Parameter page, enter a value for each element in "Parameters" and click the **[Next]** button.



4. Verify the "Pattern Information" at the View pattern result page.



5. Click the **[Apply]** button to apply the selected pattern to the tool.

☑Note

- 'Class Diagram', 'Use Case Diagram', 'Component Diagram', or 'Deployment Diagram' must be activated in order to apply design patterns.
- Select a pattern at the **[Select Pattern Type]** page and click the **[Pattern Help]** button to see help for the selected pattern.

FAQ Top Previous

The following are frequently asked questions and answers for Pattern Add-In.

1. <u>Does Pattern Add-In only have 23 GoF design patterns and 3 EJB patterns?</u>

Only one parameter value is generated in the view even though I entered two 2.

or more parameter values in the Pattern Parameters dialog.

3. The **[Tools]->[Apply Pattern ...]** menu cannot be run.

1. Does Pattern Add-In only have 23 GoF design patterns and 3 EJB patterns?

Pattern Add-In contains definitions of 23 GoF design patterns and 3 EJB patterns. The user may also define and use new patterns. For procedures for defining new patterns, see the StarUML(tm) technical documentation.

2. Only one parameter value is generated in the view even though I entered two or more parameter values in the Pattern Parameters dialog.

There can be one or more parameter values depending on the pattern being applied. If more than one value is entered, the view is generated with the last parameter name and value only. However, it can be verified through the model information that all the elements of the parameter values are generated. Drag the parameter values from the model element to the view as needed.

3. The [Tools]->[Apply Pattern ...] menu cannot be run.

See the section on Pattern Add-In configurations. Class, Use Case,

Component, or Deployment diagram must be activated in order to apply patterns to the tool.