Reusable Software Assets

Jonathan I. Maletic, Ph. D.

<SDML>
Department of Computer Science
Kent State University

What is Reuse

- Black box Reuse –
  - reuse of an asset as is
  - Plug and play
  - No modification of asset is needed.
- Glass box Reuse –
  - modification of asset is needed in order to utilize it for the specific problem.

Reuse Process

- Find the proper asset
- Understand it
- Modify it
- Integration and testing

Reusable Software

- Consists of not only of source code, but includes a wide variety of software-related products and concepts.
- These concepts are at varying degrees of abstraction.
- Software asset = item that costs money to build, store, and train others to use properly.

Types of Assets:

- Architectures [Monroe97] and Architectural Styles [Shaw96, Monroe97]
- Idioms [Coplien97]
- Design Patterns [Gamma95]
- Pattern Languages [Kerth97]
- Frameworks [Johnson97, Schmid97]
- Components [Kerth97]
- Objects/Classes
- Kits/Libraries
- Domain Models

Architectures

- Software Architecture involves the description of elements from which systems are build, interactions among those elements, patterns that guide their composition, and constraints on these patterns [Shaw96].
- Sometimes architecture and design are equated. Although an architecture may represent a design, not all designs are also architectures.
- Domain specific. High level of abstraction.

Architectural Styles (1)

- Define a family of architectures.
- Define a *vocabulary* of components, *connector* (interactions among the components) types, and a set of *constraints* on how they can be combined.
- Slightly domain specific. High level of abstraction.
- Example: client-server, pipe-and-filter, blackboard architectures.

Architectural Styles (2)

- The term *architectural styles* is often used interchangeably with *architectural patterns*, or *architectural idioms*.
- The exact definition of a style is an active research issue and debate.

Idioms

- Typical styles or methods about methods which are used to build a software systems
- A philosophy of use
- Domain independent
- High level of abstraction
- Examples:
  - Coding styles
  - GUI look and feel

Frameworks (1)

- A set of reusable classes or components used to develop a specific type of software system or subsystem. High level definitions (design patterns) of the way the components interact are also contained.
- Reusable designs of all or part of system.
- Are actual programs.
- A framework's purpose is to provide a system/application skeleton that developers can customize.

Frameworks (2)

- Domain specific. Contains elements of high and low level of abstraction.
- Types of frameworks:
  - Domain Specific - Accounting framework
  - Generic GUI Framework for information visualization

Design Patterns

- Description of methods (communicating objects and classes) that can be customized to solve a general design (recurring) problem in a particular context.
- Design patterns vary in their granularity and level of abstraction.
- Patterns are classified by their *purpose* (creational, structural, behavioral) and *scope* (class, object).

Design Patterns - examples

- Factory method (creational, class) – allows a class to defer instantiation to subclasses.
- Facade (structural, object) – provides a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.
- Iterator (behavioral, object) - provides a way to access an aggregation of objects or elements.
- Could be both domain dependent and independent. High level of abstraction.

Pattern Languages

- A system of patterns organized in a structure that guides the pattern's application.
- Simply put, pattern languages are collections of related patterns.
- Pattern languages order the high (meta) level problem solving processes.
- Domain dependent. High level of abstraction.

Components

- A reusable concrete (implemented) piece of software (program) that is concise with respect to problem type.
- Usually, a component provides a particular function or group of related functions.
- Black box component - no modification required, typically parameterizable.
- White box component - modification required to solve problem at hand.
- Domain dependent. Low level of abstraction.

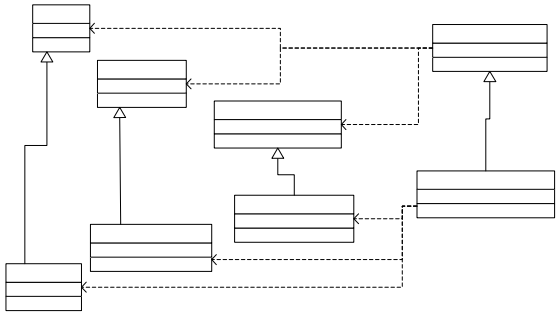
Kits/Libraries

- A set of useful routines, classes, functions.
- Domain dependent. Low level of abstraction.
- Examples:
  - STL (Standard Template Library)
  - Math library
- They are used, but not really reused.

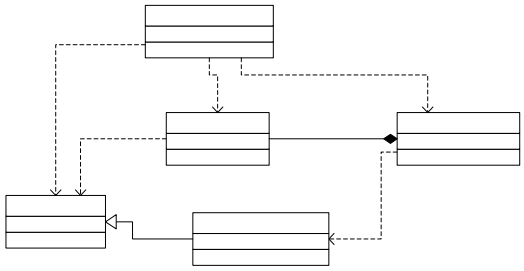
The Relationships

- Use standard UML descriptions and relationships
  - Dependency
  - Aggregation
  - Association
  - Inheritance and instantiation

Architectures



Design Patterns



KSU J. Maletic 19

Architectural Style

Idioms

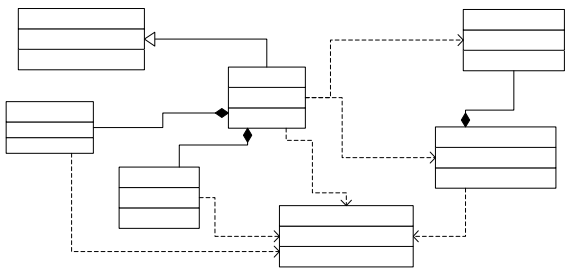
Pattern Languages are used to help describe idioms and/or philosophies (be they architectural, or design)

KSU J. Maletic 20

Design Pattern * 1 **Pattern Language**

Domain Model for X

Frameworks

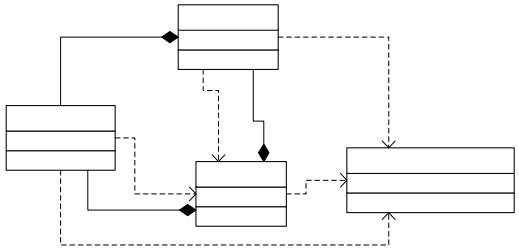


KSU J. Maletic 21

idioms +philosophy +design **Pattern Language**

* *

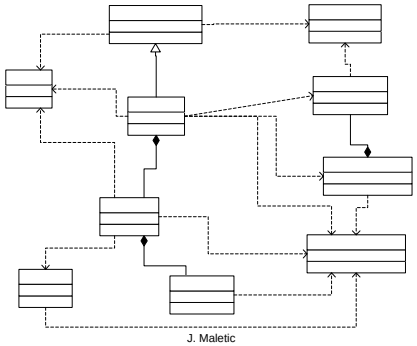
Components



KSU J. Maletic 22

1 **Component**

Unified Overview



KSU J. Maletic 23

*

1

Domain Model for X

Library/Kit

1

Final Notes

- One of the main success factors to the reuse of anything (software, ideas, or circuit designs) is standardization of terms and concepts.
- Given a standard vocabulary people can covey ideas to each other more quickly and reuse ideas and concepts over and over.
- Training is a very important part of reuse.

KSU J. Maletic 24

chitectural Style **Domain Model**

Idioms

Framework

Design Pattern

*

1

References

[Baumer97] Baumer, D., Gryczan, G., Knoll, R., Lilienthal, C., Riehle, D., Zullighoven, H., "Framework Development for Large Systems", *CACM* Vol. 40, No. 10, Oct. 1997, pp. 52-59.

[Coplien97] Coplien, J., "Idioms and Patterns as Architectural Literature", *IEEE Software*, Vol. 14, No. 1, Jan. 1997, pp. 36-42.

[Gamma95] Gamma, E., Helm, R., Johnson, R., Vlissides, J., *Design Patterns Elements of Reusable Object Oriented Software*, Addison Wesley, 1995.

[Johnson97] Johnson, R., "Frameworks = (Components + Patterns)", *CACM* Vol. 40, No. 10, Oct. 1997, pp. 39-42.

[Kerth97] Kerth, N., Cunningham, W., "Using Patterns to Improve Our Architectural Vision", *IEEE Software*, Vol. 14, No. 1, Jan. 1997, pp. 53-59.

[Monroe97] Monroe, R., Kompanek, A., Melton, R., Garlan, D., "Architectural Styles, Design Patterns, and Objects", *IEEE Software*, Vol. 14, No. 1, Jan. 1997, pp. 43-52.

[Schmid97] Schmid, H., "Systematic Framework Design by Generalization", *CACM* Vol. 40, No. 10, Oct. 1997, pp. 48-51.

[Shaw96] Shaw, M., Garlan, D. *Software Architecture perspectives on an emerging Discipline*, Prentice Hall, 1996.

[Tepfenhart97] Tepfenhart, W., Cusick, J., "A Unified Object Topology", *IEEE Software*, Vol. 14, No. 1, Jan. 1997, pp. 31-35.