Software Architecture in Practice

Chapter 6: Unit Operations and HCI Reference Architectures

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Sponsored by the U.S. Department of Defense

Lecture Objectives

This lecture will enable students to

- **describe the usefulness of unit operations and list six types of these operations**
- **discuss each type of unit operation in terms of their effects on achieving quality requirements**
- **see how unit operations may be used to create new architectural styles and reference architectures**
- **understand the evolution of the human computer interaction reference architectures in terms of unit operations**

Unit Operations

A unit operation is a codification of design operations that can be applied directly to an architecture. Examples include

- **abstraction**
- **compression**
- **part-whole decomposition**
- **is-a decomposition**
- **replication**
- **resource sharing**

Styles Vs. Unit Operations

Styles are pre-packaged off-the-shelf design solutions that have known quality properties.

Unit operations are the steps that one applies to derive styles, patterns, and reference architectures.

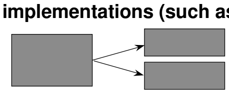
Unit operations affect the quality properties of the styles they lead to, but are not design solutions by themselves.

Abstraction as a Unit Operation

Abstraction creates a virtual machine (a component whose function is to hide its underlying implementation).

Abstraction is used for

- **simulated target platform (when that is distinct from development platform)**
- **layered systems**
- **common interface to heterogeneous set of underlying implementations (such as user interface toolkits)**

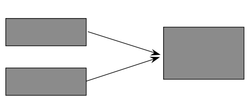


Compression as a Unit Operation

Compression combines two components into a single component.

Compression is used to

- **enhance performance**
  - **circumvent layering**
  - **eliminate overhead from calls**
- **speed up system development**



Decomposition as a Unit Operation

Decomposition breaks up a large component into smaller pieces. It can be done in two ways:

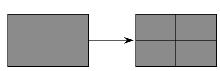
- **part-whole**
- **is-a**

Part-Whole Decomposition

Every component in the system can be built from a fixed small set of subcomponents. Example: model view controller, flight simulators

Used for supporting integrability, extensibility, and understandability

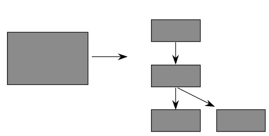
A small number of component types gives a small number of component type interfaces.



Is-A Decomposition

Each subcomponent represents a specialization of its parent's functionality. Examples: PAC, class hierarchies

Used for reuse by increments.

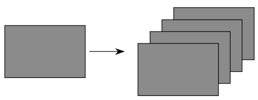


Carnegie Mellon University
Software Engineering Institute

Replication as a Unit Operation

Replication is the exact duplication of a component within an architecture.

Replication is used for enhancing
- **reliability (redundant operation)**
- **performance enhancement (data caching)**

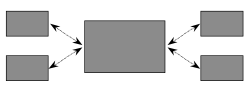


Carnegie Mellon University
Software Engineering Institute

Resource Sharing as a Unit Operation

Resource sharing encapsulates either data or services and shares them among multiple independent consumers.

Resource sharing is used for
- **integrability**
- **portability**
- **modifiability**



Examples
- **shared databases**
- **servers in client/servers**

Carnegie Mellon University
Software Engineering Institute

Unit Operations and Qualities

	Scalability	System modifiability	Integrability	Portability	Sequential performance	Concurrent performance	Fault tolerance	Ease of system creation	Component modifiability	Ease of component creation	Reusability
Abstraction		+	+	+	-		+	+	+	+	+
Compression		-	-	-	+	-		-	-		-
Part-whole decomposition		+	+			+					+
is-a decomposition		+	+		-	+		+		+	
Replication	+				-	+	+				
Resource sharing			+		-			+	-	+	

Reference Architecture

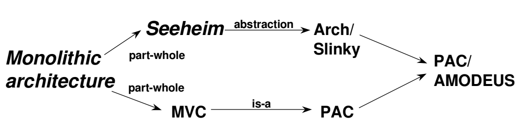
Reference architecture: a division of functionality, together with data flow between the pieces

Using Unit Operations to Build Reference Architectures

Strengths and weaknesses of unit operations can be demonstrated by exploring human-computer interface (HCI) reference architectures.

HCI reference architectures are important in their own right since user interface typically accounts for 50% of the cost of a system.

Unit Operations and HCI Reference Architectures



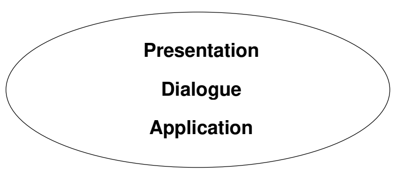
Monolithic Reference Architecture for Interactive Systems

An interactive system provides three functions.

- **presentation: controls interaction with user (example: windows interface)**
- **application: underlying purpose of system (example: database system)**
- **dialogue: decomposes/sequences user tasks (example: interaction)**

When functions are lumped together in a single structure, it is a *monolithic architecture*.

Monolithic Reference Architecture



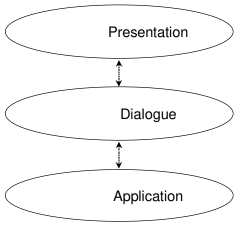
Achieving User Interface Quality Goals Through Unit Operations

The typical user interface goals are modifiability and portability.

Applying the unit operation of part-whole decomposition supports the modifiability goal and abstraction supports the portability goal.

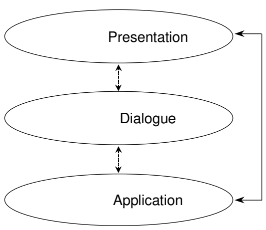
Seeheim Reference Architecture

The Seeheim reference architecture applies part - whole decomposition to the monolithic reference architecture.



Improved Seeheim

Uses compression to ameliorate performance problems



Strengths of Seeheim Reference Architecture

A separate presentation function supports portability and modifiability.

A separate application layer allows modification of function without affecting user interface.

A separate dialogue allows modifications to user interaction without rewriting the presentation.

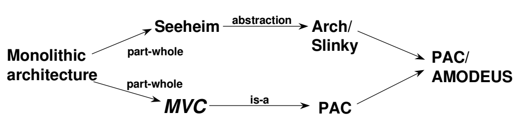
Weaknesses of Seeheim Reference Architecture

Many modifications affect all three functions.

There are performance problems with sophisticated semantic feedback (which result in the need for compression).

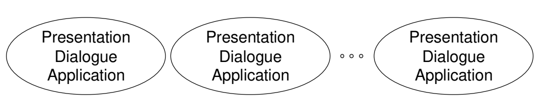
Use of the architecture leads to separate notations for dialogue, presentation, and application layer; this is cumbersome.

Unit Operations and HCI Reference Architectures



Model View Controller (MVC) -1

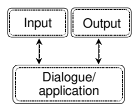
The MVC paradigm applies part-whole decomposition to portions of the monolithic reference architecture.



Model View Controller (MVC) -2

MVC: input = “controller”
output = “view”
dialogue application = “model”

Dialogue/application not divided



Strengths of MVC

MVC promotes
- **independent modification of objects and their presentation**
- **reuse at the object level**
- **language to describe whole application (e.g., Smalltalk)**

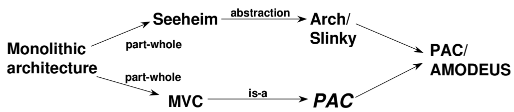
Weaknesses of MVC

It is cumbersome to reuse widgets across objects.

Portability is the responsibility of individual objects, rather than a system function. This is typically fixed by using layering.

Interactions involving presentations from multiple objects are difficult to coordinate (e.g., it is difficult to stretch a line that connects two boxes as one box is moved).

Unit Operations and HCI Reference Architectures

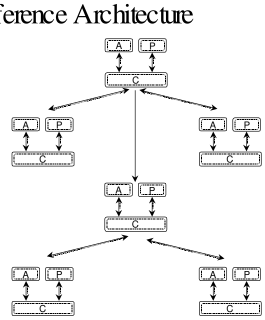


Presentation Application Control (PAC) Reference Architecture

PAC uses

- **part-whole decomposition within a PAC agent**
- **is-a decomposition to hierarchically decompose agents**

PAC Reference Architecture



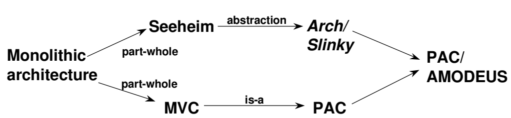
P = presentation A = application C = control

Strengths and Weakness of PAC

Strengths
- **aligns better with separations of concerns in user interface software**
- **allows for modifiability of dialogue through hierarchical decomposition**

Weakness: doesn't deal with reuse and consistency of user interface widgets

Unit Operations and HCI Reference Architectures

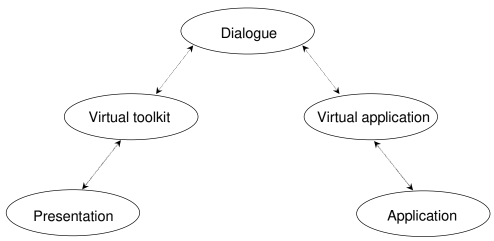


Arch/Slinky Reference Architecture -1

From Seeheim to Arch/Slinky: applied abstraction to presentation and application
- **interposed adapter between presentation and dialogue**
- **interposed adapter between application layers and dialogue**

Yields a five-component model

Arch/Slinky Reference Architecture -2

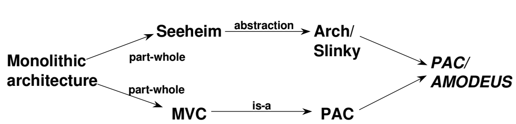


Strength and Weakness of Arch/Slinky

Strength: abstraction enhances portability of application and presentation

Weakness: abstraction introduces performance penalty

Unit Operations and HCI Reference Architecture



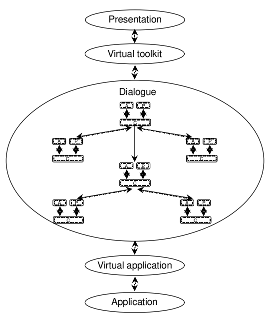
Unified Reference Architecture: PAC-AMODEUS*

PAC hierarchically applied part-whole decomposition to dialogue

Arch/Slinky applied abstraction to application and presentation

***AMODEUS = assimilating models of designers, users, and systems**

PAC-AMODEUS



Strength and Weakness of PAC-AMODEUS

Strength: supports modifiability and portability

Weakness: performance penalty for all subparts

Lecture Summary - 1

Quality goals can lead to different architectures.

Unit operations by themselves are insufficient to achieve quality goals; rather, they are used to understand architectural design options.

Unit operations represent primitive design decisions that lead to architectures.

Lecture Summary - 2

Unit operations are used in cases where existing styles are not appropriate.

Unit operations in isolation will not yield an architecture. Additional techniques are required for achieving quality goals.

Discussion Questions - 1

1. **Consider a large system with which you are familiar. Can you see how unit operations were applied to it? In what ways were they used and for what reasons. If they were not used, has it caused difficulties?**

Discussion Questions -2

2. **Unit operations are extremely coarse grained. Designs are always tempered by particular functional requirements and by other quality requirements (as Ralph Waldo Emerson - an early software engineer - noted, "Every good quality is noxious if unmixed"). Can you think of ways in which the application of unit operations is tempered by the environment in which they are applied?**

Discussion Questions -3

3. **Choose an architectural style and try to derive it from a monolithic model using unit operations. Do the qualities that the unit operations impart match those for which the style is known?**

4. **Choose a blank entry in the table on page 12 and try to decide when the corresponding quality is supported by or in conflict with the unit operation.**

