

CS3773 Software Engineering

Lecture 19 Software Architecture

Layered Style

- > Suitable for applications that involve distinct classes of services that can be organized hierarchically
- > Each layer provides service to the layer above it and serves as a client to the layer below it
- > Only carefully selected procedures from the inner layers are made available (exported) to their adjacent outer layer

2

UTSA CS3773

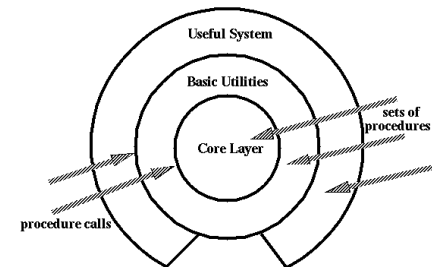
Layered Style (Cont'd)

- > Components: are typically collections of procedures
- > Connectors: are typically procedure calls under restricted visibility

3

UTSA CS3773

Layered Style (Cont'd)



4

UTSA CS3773

Layered Style Specializations

- Often exceptions are made to permit non-adjacent layers to communicate directly
 - This is usually done for efficiency reasons

5

UTSA CS3773

Layered Style Examples

- Layered Communication Protocols:
 - Each layer provides a substrate for communication at some level of abstraction
 - Lower levels define lower levels of interaction, the lowest level being hardware connections (physical layer)
- Operating Systems
 - Unix

6

UTSA CS3773

Layered Style Advantages

- Design: based on increasing levels of abstraction.
- Enhancement: since changes to the function of one layer affects at most two other layers
- Reuse: since different implementations (with identical interfaces) of the same layer can be used interchangeably

7

UTSA CS3773

Layered Style Disadvantages

- Not all systems are easily structured in a layered fashion
- Performance requirements may force the coupling of high-level functions to their lower-level implementations

8

UTSA CS3773

Repository Style

- Suitable for applications in which the central issue is establishing, augmenting, and maintaining a complex central body of information
- Typically the information must be manipulated in a variety of ways
- Often long-term persistence of information is required

9

UTSA CS3773

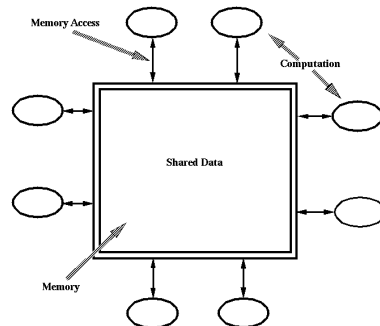
Repository Style (Cont'd)

- Components:
 - A central data structure representing the correct state of the system
 - A collection of independent components that operate on the central data structure
- Connectors:
 - Typically procedure calls or direct memory accesses

10

UTSA CS3773

Repository Style (Cont'd)



11

UTSA CS3773

Repository Style Specializations

- Changes to the data structure trigger computations
- Data structure in memory (persistent option)
- Data structure on disk
- Concurrent computations and data accesses

12

UTSA CS3773

Repository Style Examples

- Information Systems
- Programming Environments
- Graphical Editors
- AI Knowledge Bases
- Reverse Engineering Systems

13

UTSA CS3773

Repository Style Advantages

- Efficient way to store large amounts of data
- Sharing model is published as the repository schema
- Centralized management:
 - backup
 - security
 - concurrency control

14

UTSA CS3773

Repository Style Disadvantages

- Must agree on a data model a priori
- Difficult to distribute data
- Data evolution is expensive

15

UTSA CS3773

Interpreter Style

- Suitable for applications in which the most appropriate language or machine for executing the solution is not directly available

16

UTSA CS3773

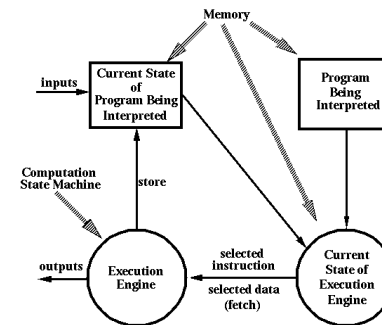
Interpreter Style (Cont'd)

- Components: include one state machine for the execution engine and three memories:
 - current state of the execution engine
 - program being interpreted
 - current state of the program being interpreted
- Connectors:
 - procedure calls
 - direct memory accesses

17

UTSA CS3773

Interpreter Style (Cont'd)



18

UTSA CS3773

Interpreter Style Examples

- Programming Language Compilers:
 - Java
 - Smalltalk
- Rule Based Systems:
 - Prolog
 - Coral
- Scripting Languages:
 - Awk
 - Perl

19

UTSA CS3773

Interpreter Style Advantages

- Simulation of non-implemented hardware
- Facilitates portability of application or languages across a variety of platforms

20

UTSA CS3773

Interpreter Style Disadvantages

- Extra level of indirection slows down execution
- Some interpreters have an option to compile code

21

UTSA CS3773

Process-Control Style

- Suitable for applications whose purpose is to maintain specified properties of the outputs of the process at (sufficiently near) given reference values
- Components:
 - Process Definition includes mechanisms for manipulating some process variables
 - Control Algorithm for deciding how to manipulate process variables

22

UTSA CS3773

Process-Control Style

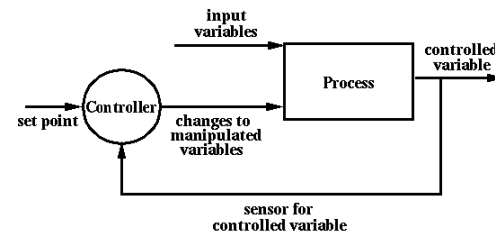
- Connectors: are the data flow relations for:
 - Process Variables:
 - Controlled variable whose value the system is intended to control
 - Input variable that measures an input to the process
 - Manipulated variable whose value can be changed by the controller
 - Set Point is the desired value for a controlled variable
 - Sensors to obtain values of process variables pertinent to control

23

UTSA CS3773

FeedBack Control System

- The controlled variable is measured and the result is used to manipulate one or more of the process variables

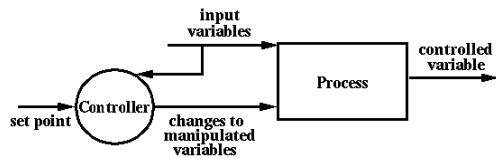


24

UTSA CS3773

Feedforward Control System

- Information about process variables is not used to adjust the system

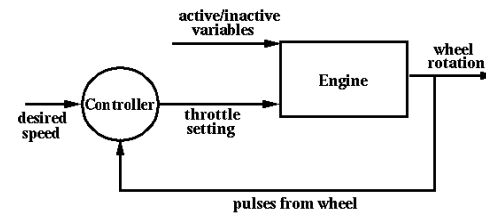


25

UTSA CS3773

Process Control Examples

- Real-Time System Software to Control:
 - Automobile Anti-Lock Brakes
 - Nuclear Power Plants
 - Automobile Cruise-Control



26

Reading Assignments

- Sommerville's Book
 - Chapter 12, "Distributed System Architecture"
 - Chapter 13, "Application Architectures"
- David Garlan and Mary Shaw, "An introduction to software architecture", URL :

http://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf

27

UTSA CS3773