Lecture 19: Software Architectures

Architectural Styles

- pipe and filter
- object oriented
- event based
- layered
- repositories
- process control

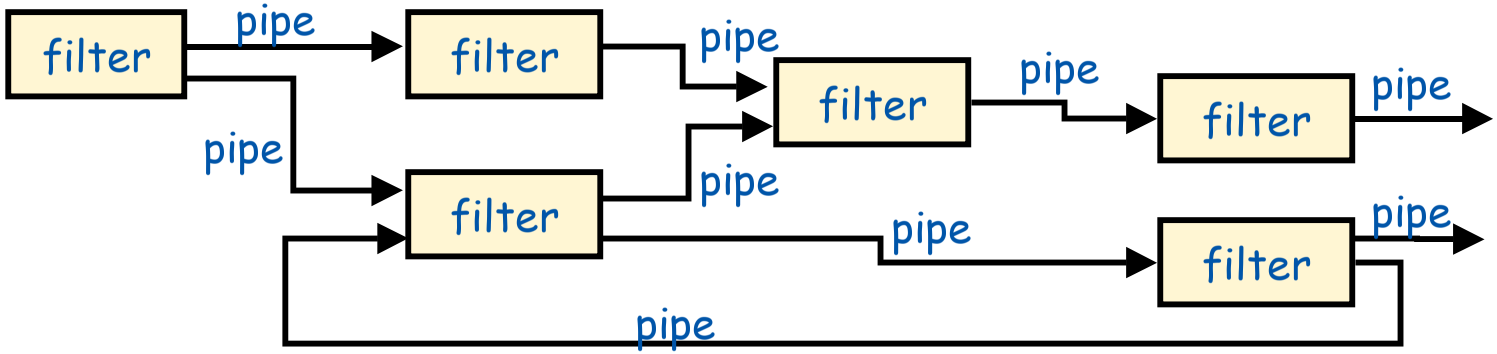
Why choice of style matters

- the KWIC example

Architectural Description Languages

Pipe-and-filter

Source: Adapted from Shaw & Garlan 1996, p21-2. See also van Vliet, 1999 Pp266-7 and p279



Examples:

UNIX shell commands

Compilers:

Lexical Analysis -> parsing -> semantic analysis -> code generation

Signal Processing

Interesting properties:

filters don't need to know anything about what they are connected to

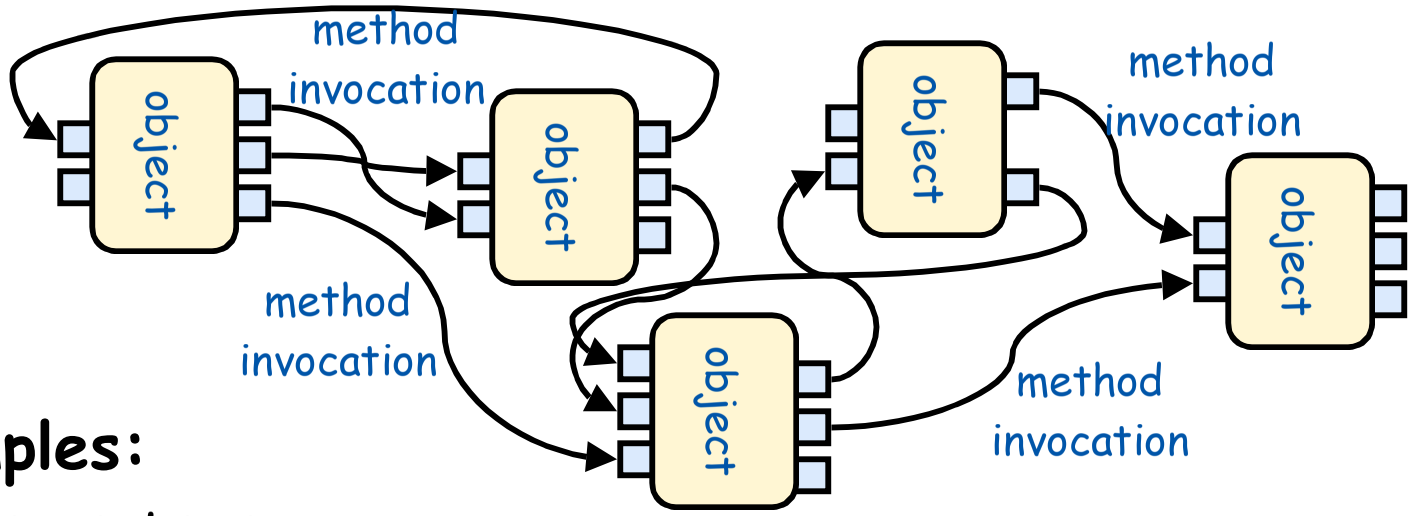
filters can be implemented in parallel

behaviour of the system is the composition of behaviour of the filters

specialized analysis such as throughput and deadlock analysis is possible

Object Oriented Architectures

Source: Adapted from Shaw & Garlan 1996, p22-3.



Examples:

abstract data types

object broker systems (e.g. CORBA)

Interesting properties

data hiding (internal data representations are not visible to clients)

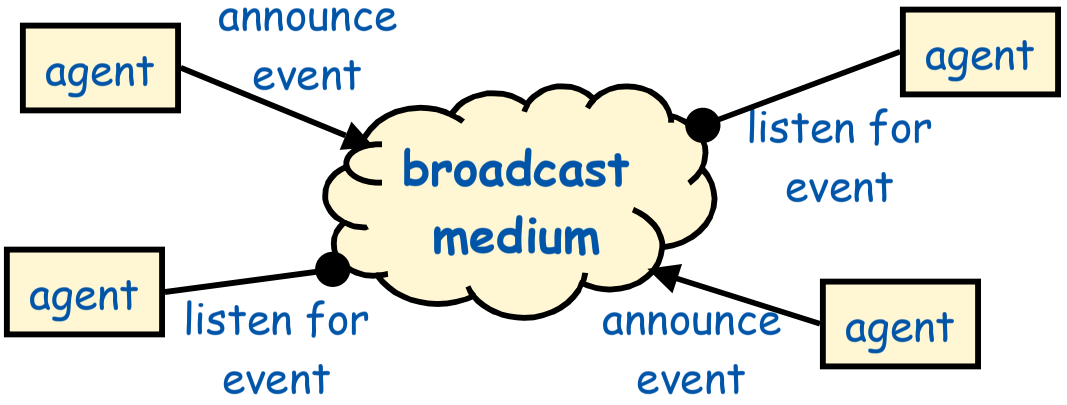
can decompose problems into sets of interacting agents

Disadvantages

objects must know the identity of objects they wish to interact with

Event based (implicit invocation)

Source: Adapted from Shaw & Garlan 1996, p23-4. See also van Vliet, 1999 Pp264-5 and p278



Examples

- debugging systems (listen for particular breakpoints)
- database management systems (for data integrity checking)
- graphical user interfaces

Interesting properties

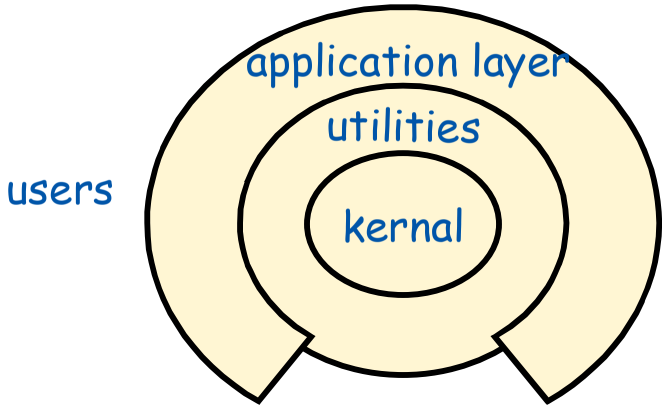
- announcers of events don't need to know who will handle the event
- Supports re-use, and evolution of systems (add new agents easily)

Disadvantages

- Components have no control over ordering of computations

Layered Systems

Source: *Adapted from Shaw & Garlan 1996, p25. See also van Vliet, 1999, p281.*



Examples

Operating Systems

communication protocols

Interesting properties

Support increasing levels of abstraction during design

Support enhancement (add functionality) and re-use

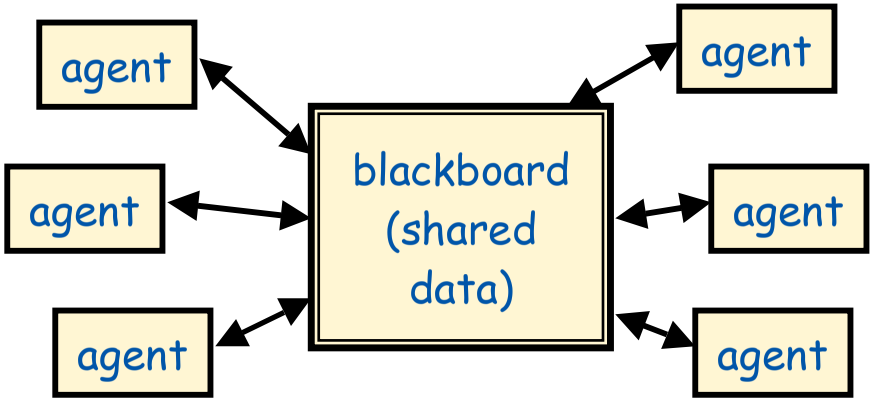
can define standard layer interfaces

Disadvantages

May not be able to identify (clean) layers

Repositories

Source: Adapted from Shaw & Garlan 1996, p26-7. See also van Vliet, 1999, p280



Examples

- databases
- blackboard expert systems
- programming environments

Interesting properties

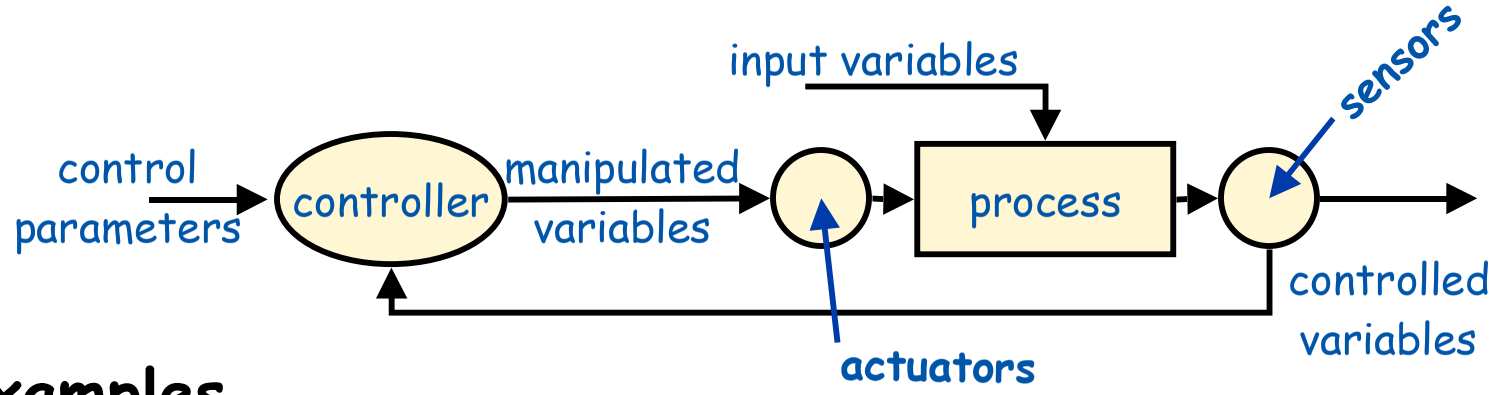
- can choose where the locus of control is (agents, blackboard, both)
- reduce the need to duplicate complex data

Disadvantages

- blackboard becomes a bottleneck

Process Control

Source: *Adapted from Shaw & Garlan 1996, p27-31.*



Examples

aircraft/spacecraft flight control systems

controllers for industrial production lines, power stations, etc.

chemical engineering

Interesting properties

separates control policy from the controlled process

handles real-time, reactive computations

Disadvantages

Difficult to specify the timing characteristics and response to disturbances

Parnas' KWIC example

Source: Adapted from Parnas 1972. See also van Vliet, 1999 Pp258-270

KWIC = KeyWord In Context

Task is to build a contextualized index for the text

Input is a set of lines of text

Output is the set of all circular shifts of all lines, in alphabetical order

Parnas identifies two different architectures:

1) shared data model
2) data abstraction model

} see next slide

Possible design changes:

change of input format

decision to store all text in memory

decision to index rather than copy

decision to alphabetize rather than search

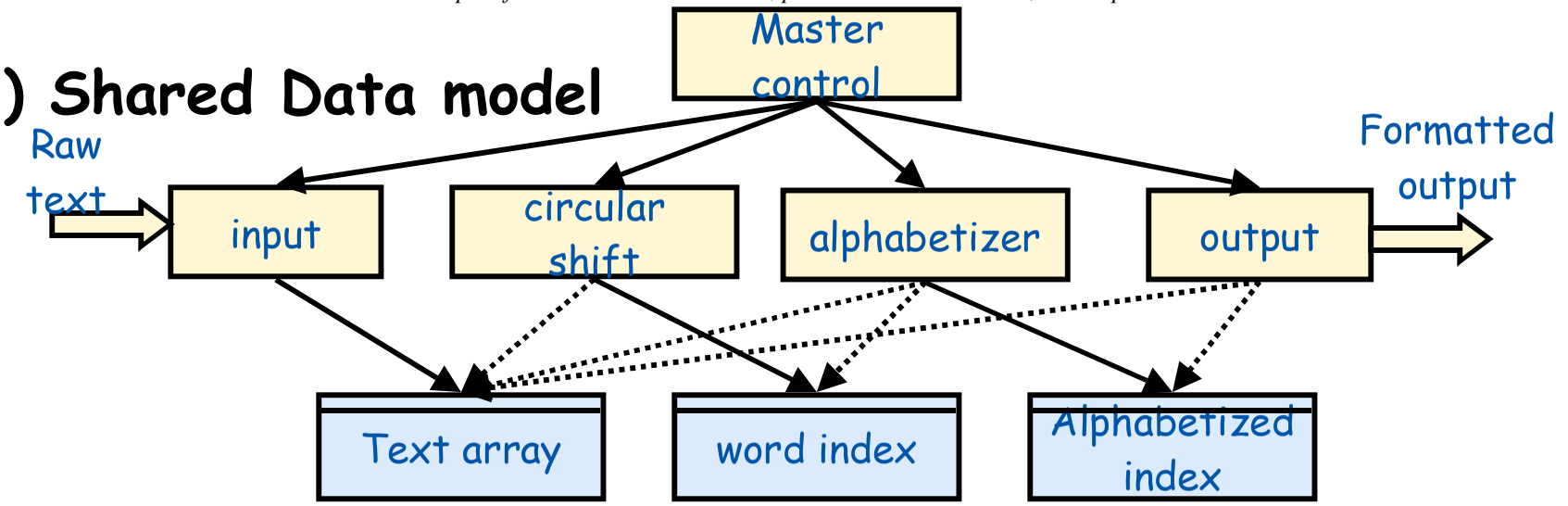
Different architectures support different changes:

1) good for adding functionality; poor for change in data rep, reusability
2) good for changing data rep, reusability; poor for change in functionality

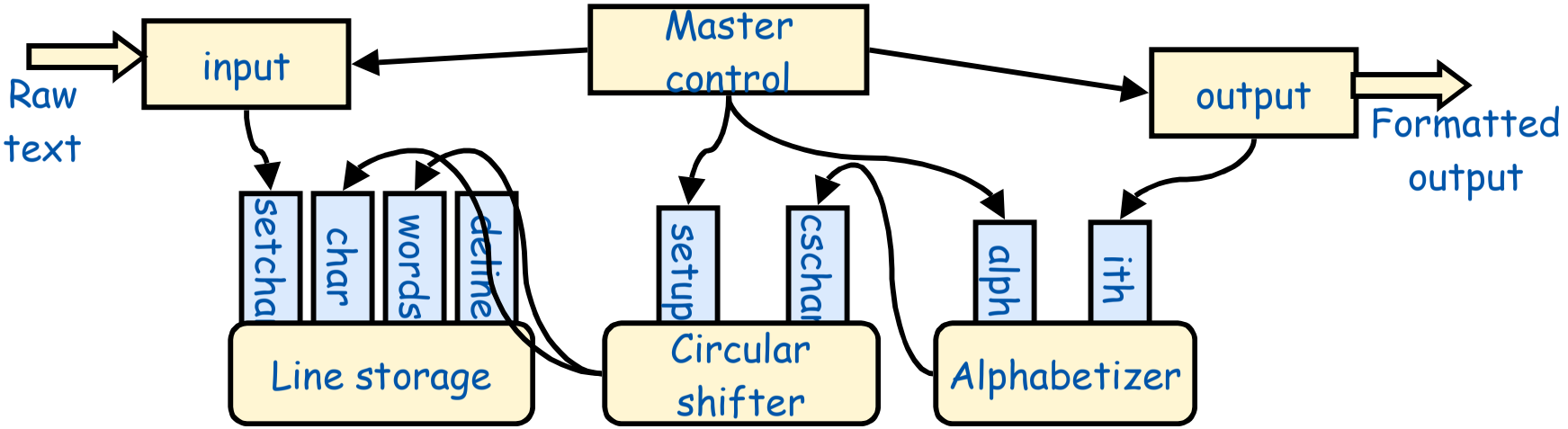
KWIC architecture solutions

Source: *Adapted from Shaw & Garlan 1996, p34-8. See also van Vliet, 1999 Pp258-270*

1) Shared Data model



2)Data abstraction model



Describing Architectures

Source: Adapted from Shaw & Garlan 1996, chapters 7 & 8. See also van Vliet, 1999, Pp270-281

Kinds of language

Module Interconnection Languages

- describe a system configuration separately from the components (programs)
- mainly concerned with name binding

Programming languages constructs

- e.g. UNIX pipes, Java Event handlers, Ada rendezvous
- Permit new forms of interaction beyond procedure call
- Do not permit creation of new abstractions, descriptions of architectural patterns

Architectural Description Languages (ADLs)

- Provide a language for describing components and connectors
- Connectors treated as first class objects
- Definition of roles & relationships (rather than algorithms & data structures)
- E.g. Unicon (Shaw); Darwin (Kramer)

Things to describe

Components

- computation; memory; object managers; process controllers; comms links

Connectors

- procedure call; dataflow; implicit invocation; message passing; shared data; instantiation

References

van Vliet, H. "Software Engineering: Principles and Practice (2nd Edition)" Wiley, 1999.

chapter 10 provides an excellent introduction to software architectures. van Vliet uses Parnas' KWIC example to motivate the entire chapter, and then covers the work of Shaw and Garlan quite thoroughly. Reading this chapter will save you having to refer to the originals, which are:

Shaw, M. and Garlan, D. "Software Architecture: Perspectives on an emerging discipline", 1996, Prentice Hall.

This book defined the field of software architecture. Most of this lecture is adapted from this book.

Parnas, D. L. "On the Criteria to be used in Decomposing Systems into Modules". 1972, Communications of the ACM, Vol 15, No 12

This paper, although dated, was the first to describe how the choice of software architecture affects modifiability. The KWIC example comes from this paper.