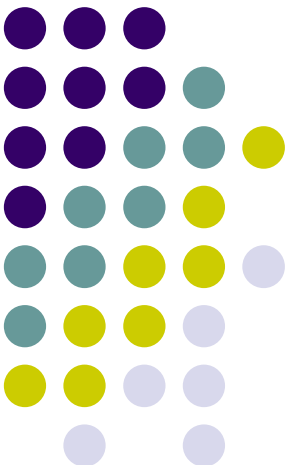
Introduction to Software Architecture

A Review of

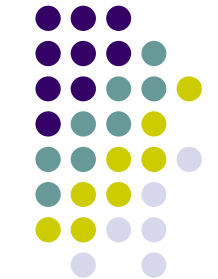
D. Garlan and M. Shaw, **An Introduction to Software Architecture**, Tech. Report CMU-CS-94-166, School of CS, Carnegie Mellon University, 1994.



Contents

- Introduction
- From Programming Languages to Software Architecture
- Common Architectural Styles
- Case Studies
- Past, Present, and Future

Architecture Level of Design



- Gross organization
- Global control structure
- Communication protocols
- Synchronization
- Data access
- Composition of design elements
- Scaling and performance

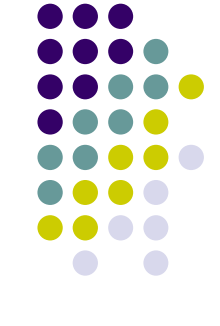
SW Arch and SE

- Ability to recognize common paradigms
  - Systems that are variations on old systems
- Getting the right architecture is crucial
- Principled choices among design alternatives
- Architectural system representation →
  - analysis of high-level properties

From PL to SA

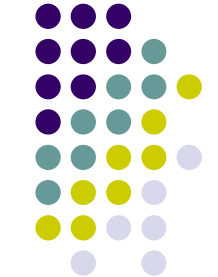
- Regular increase in abstraction level
- Historical develpment of abstraction techinques

High-level PLs



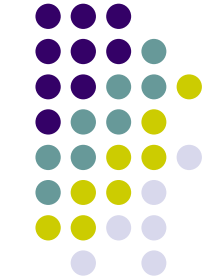
- 1950s
  - manual insertion of instructions
  - hand-checking updates
  - automated updates
  - symbolic names
  - macro processors
- Mathematical notations
  - expressions, invocations, loops, decisions, etc.

Abstract Data Types



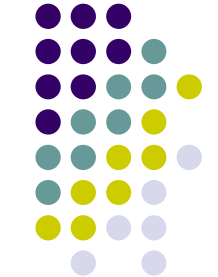
- If you get the data structures right
  - the rest of the development would be easier
- Understanding
  - the software structure
  - specifications
  - language issues
  - integrity
  - rules for combining types
  - information hiding

Software Architecture



- Useful system organizations
- ADTs are not the only way to organize a software system
  - C/S model
  - Abstraction layering
  - Distributed, OO approach
  - Pipelines

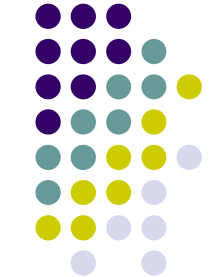
Common Architectural Styles



- A common framework to view Arch. Styles
  - Components
  - Connectors

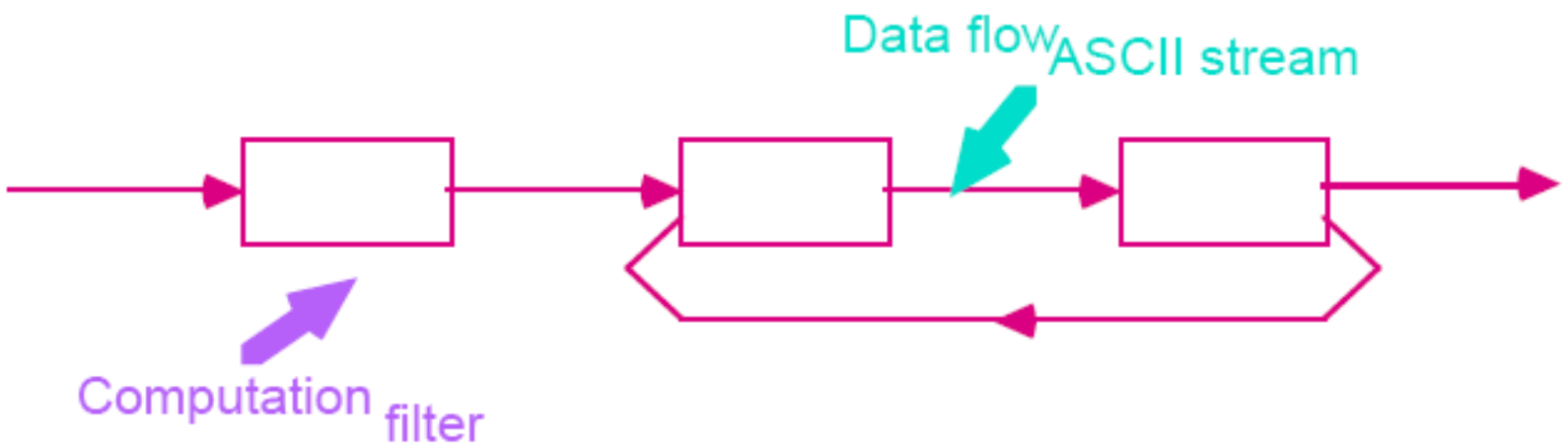
- Arch. Style
  - Family of systems in terms of a pattern of structural organization
  - Vocabulary of components and connectors
  - + Constraints

Questions on Arch. Styles

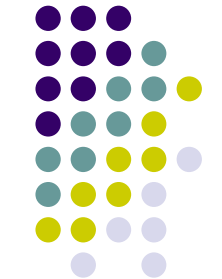


- What is the structural pattern?
  - Components, connectors, and constraints
- What is the underlying computation model?
- What are the essential invariants?
- What are some common examples?
- What are advantages and disadvantages?
- What are common specializations?

Pipes and Filters

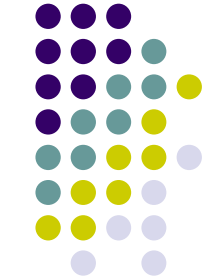


Pipe and Filter properties



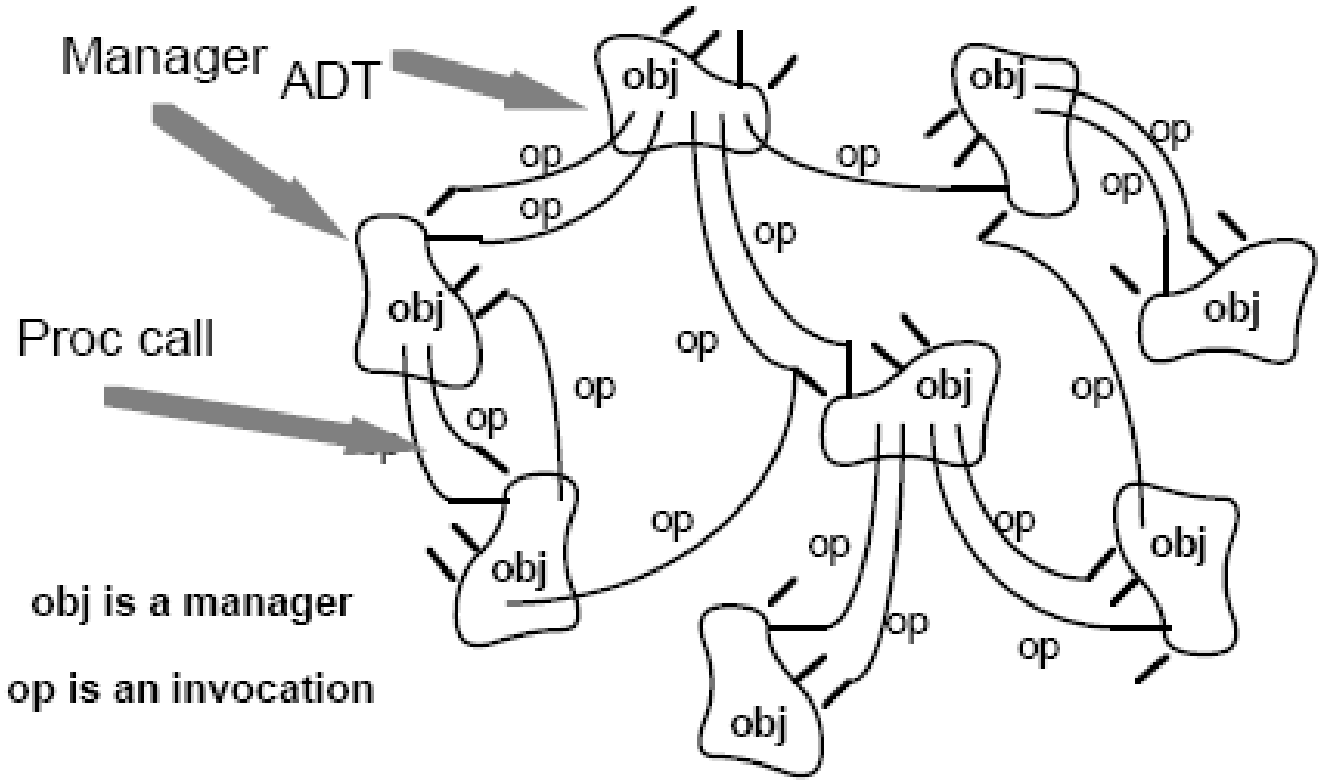
- Filters are independent
- Filters does not know the identity of the related filters
- Pros
  - A simple composition
  - Support for reuse
  - Easy maintenance
  - Specialized analysis

Pipe and Filter properties



- Cons
  - Batch organization of procesing
  - Incremental display updates
  - Added work for parse and unparse data

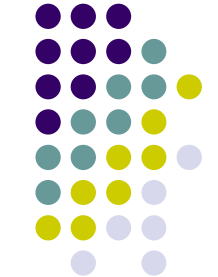
Data Abstraction



Pros and Cons

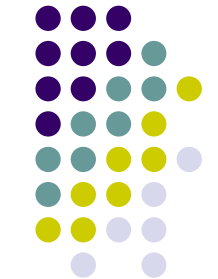
- Pros
  - Information hiding
  - Easy maintenance
- Cons
  - Having to know the identity of the communicating counterpart

Event-based, Implicit Invocation



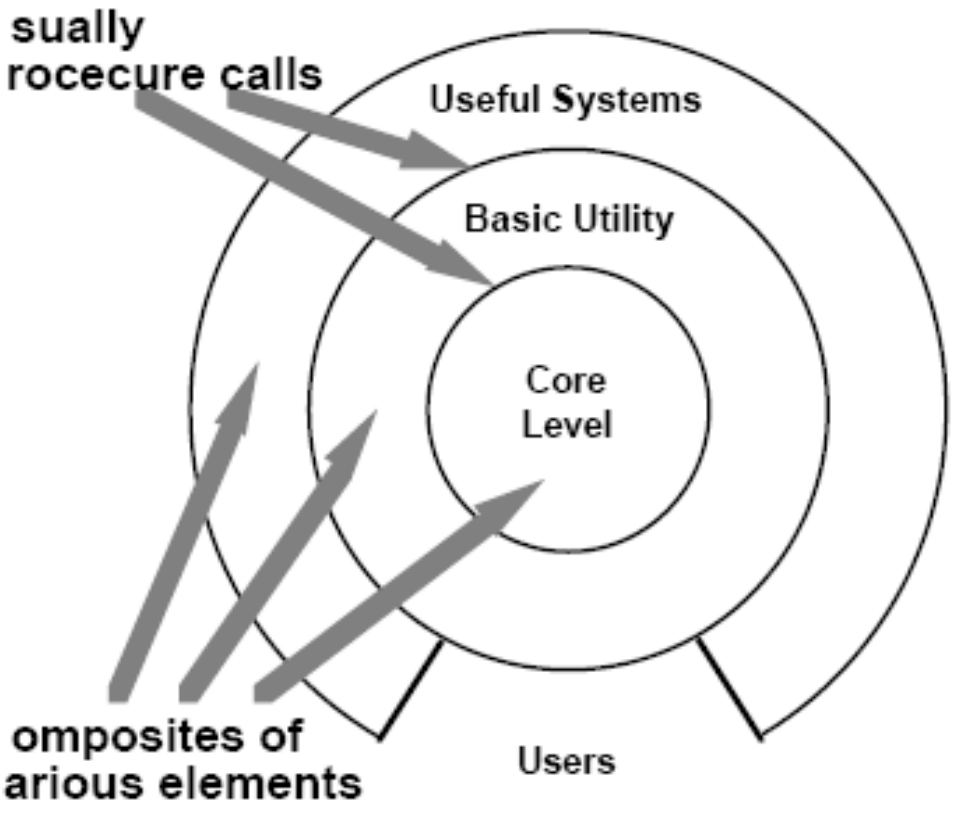
- Listeners, observers, etc.
- Announcers do not know which components will be affected
- Pros
  - Strong support for reuse
  - Easy system evolution

Event-based, Implicit Invocation



- Cons
  - Dependence on the identity of the announcer
  - The order of effects of an event is not known
  - Difficult to reason

Layered Systems



Pros and Cons

- Pros
  - Support for design based on increasing levels of abstraction
  - Support for enhancement
  - Support for reuse
- Cons
  - Not applicable to all systems
  - Performance considerations

Repositories

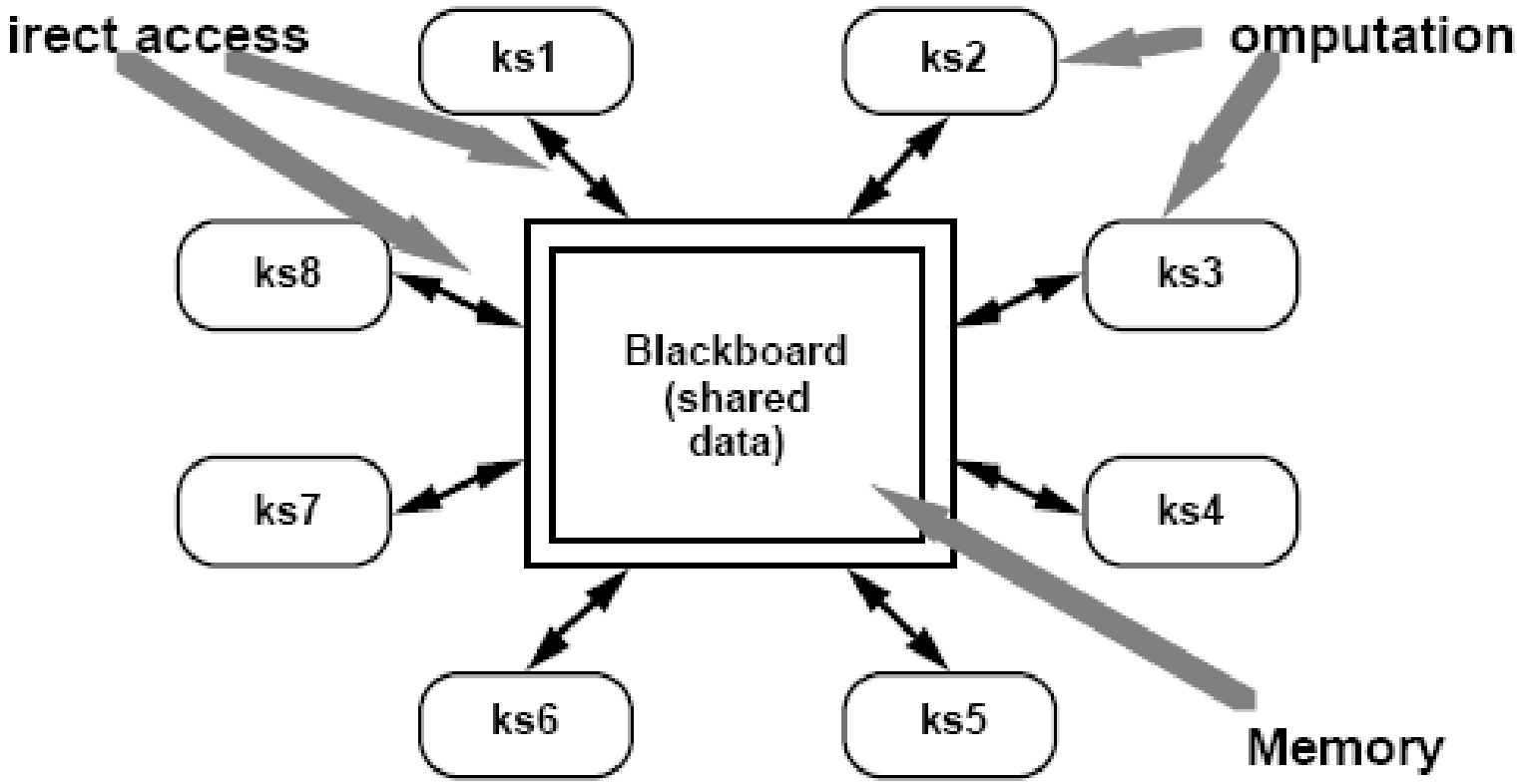
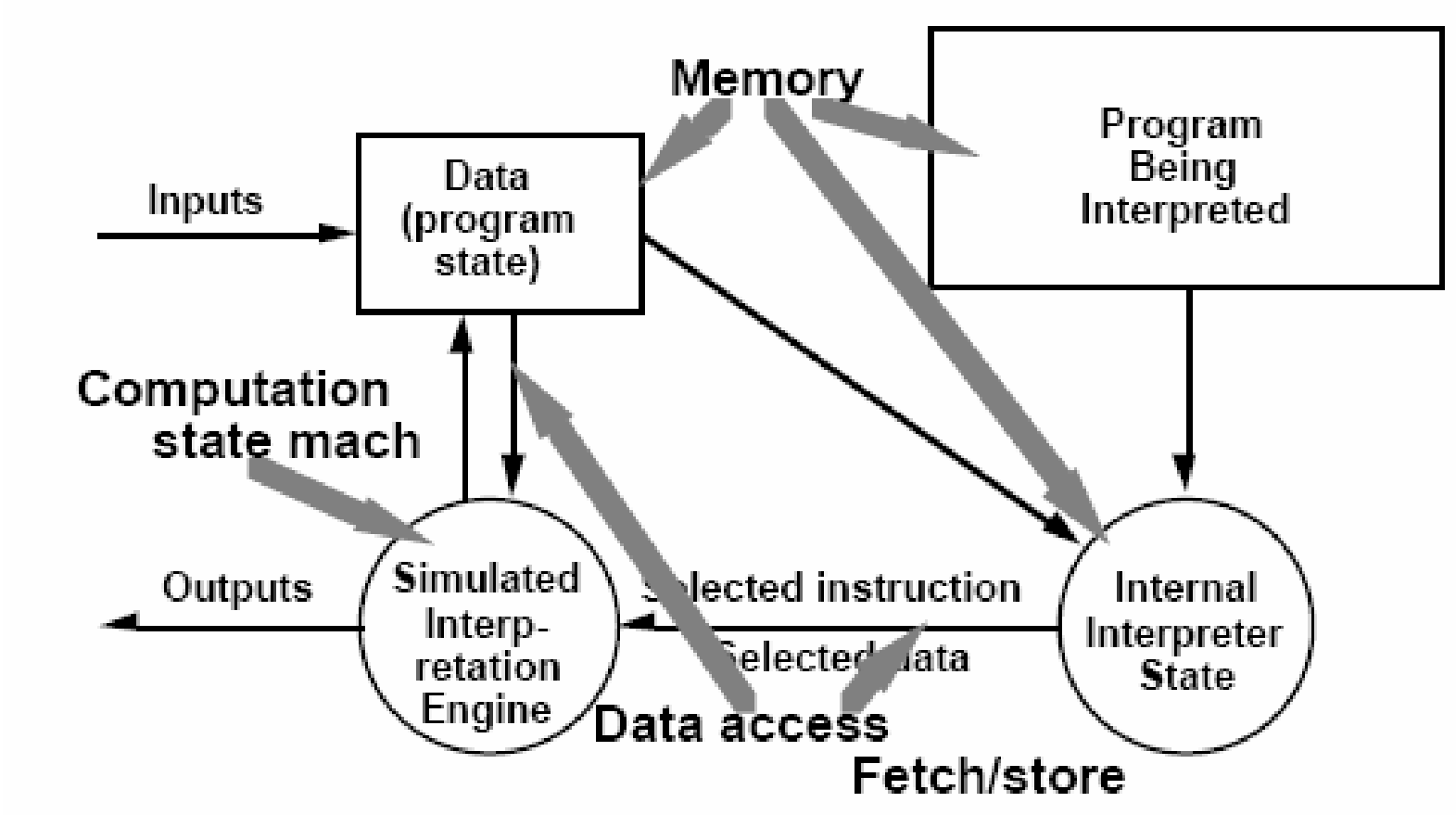
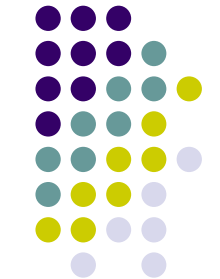


Table Driven Interpreters

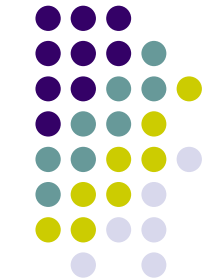


Other Families



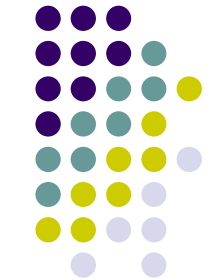
- Distributed processes
- Main program/subroutine organizations
- Domain-specific
- State transition systems
- Process control systems

Heterogenous Architectures



- Combination through hierarchy
- Single components use mixtures of arch. styles

Case Study 1: KWIC



- The KWIC [Key Word in Context] index system accepts an ordered set of lines, each line is an ordered set of words, and each word is an ordered set of characters. Any line may be ``circularly shifted'' by repeatedly removing the first word and appending it at the end of the line. The KWIC index system outputs a listing of all circular shifts of all lines in alphabetical order.

KWIC example

- Input: strings, each of which consists of several words.

  Clouds are white.

  Ottawa is beautiful.

- Output: a sorted list of all orderings of each input string.

  are white Clouds

  beautiful Ottawa is

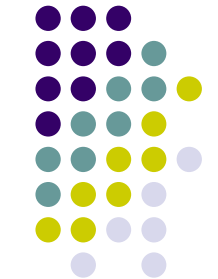
  Clouds are white

  is beautiful Ottawa

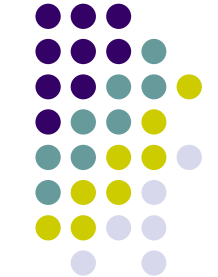
  Ottawa is beautiful

  white Clouds are

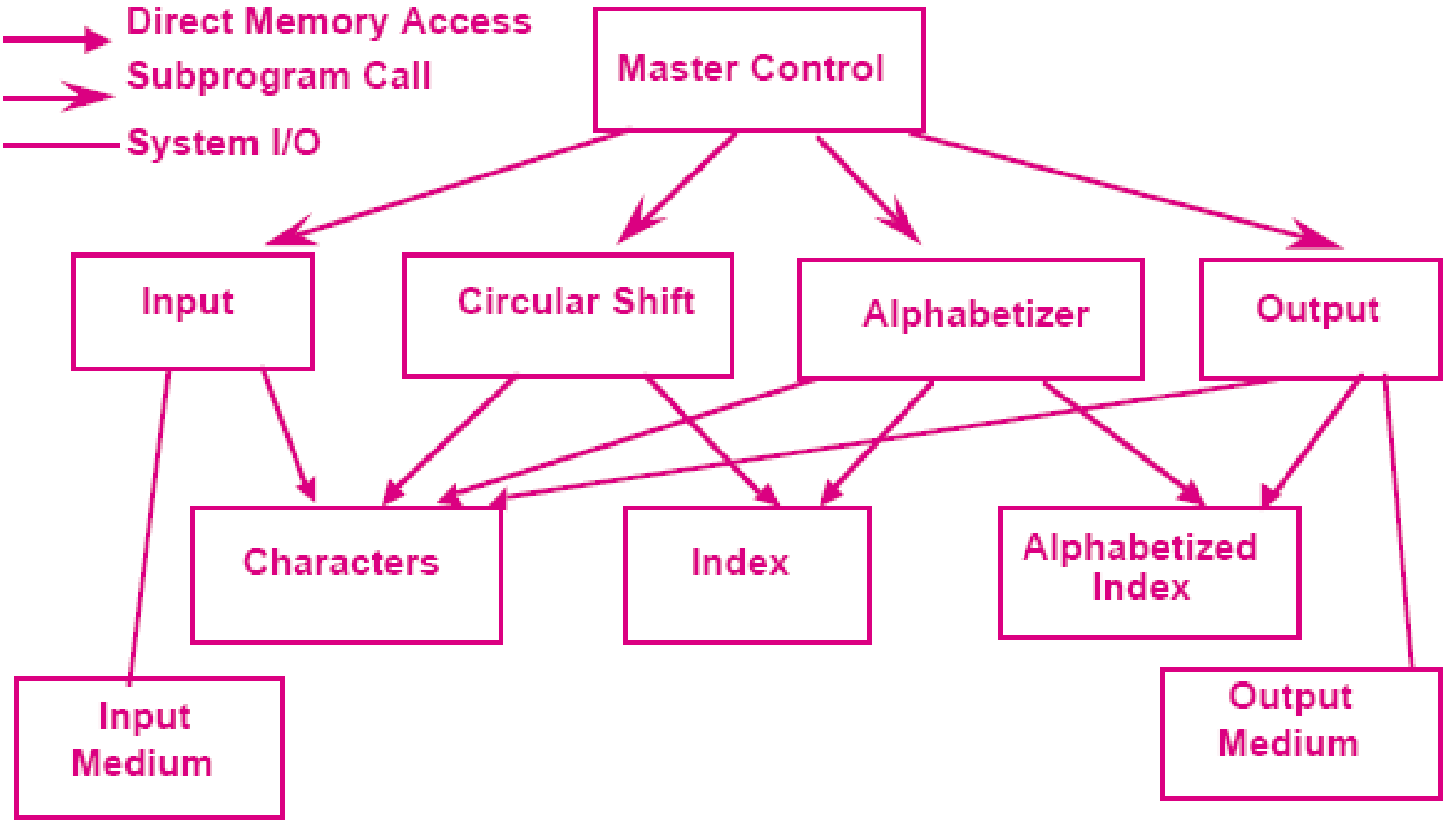
Possible Changes to the problem



- Changes in processing algorithm
- Changes in data representation
- Enhancements to system functions
- Performance
- Reuse

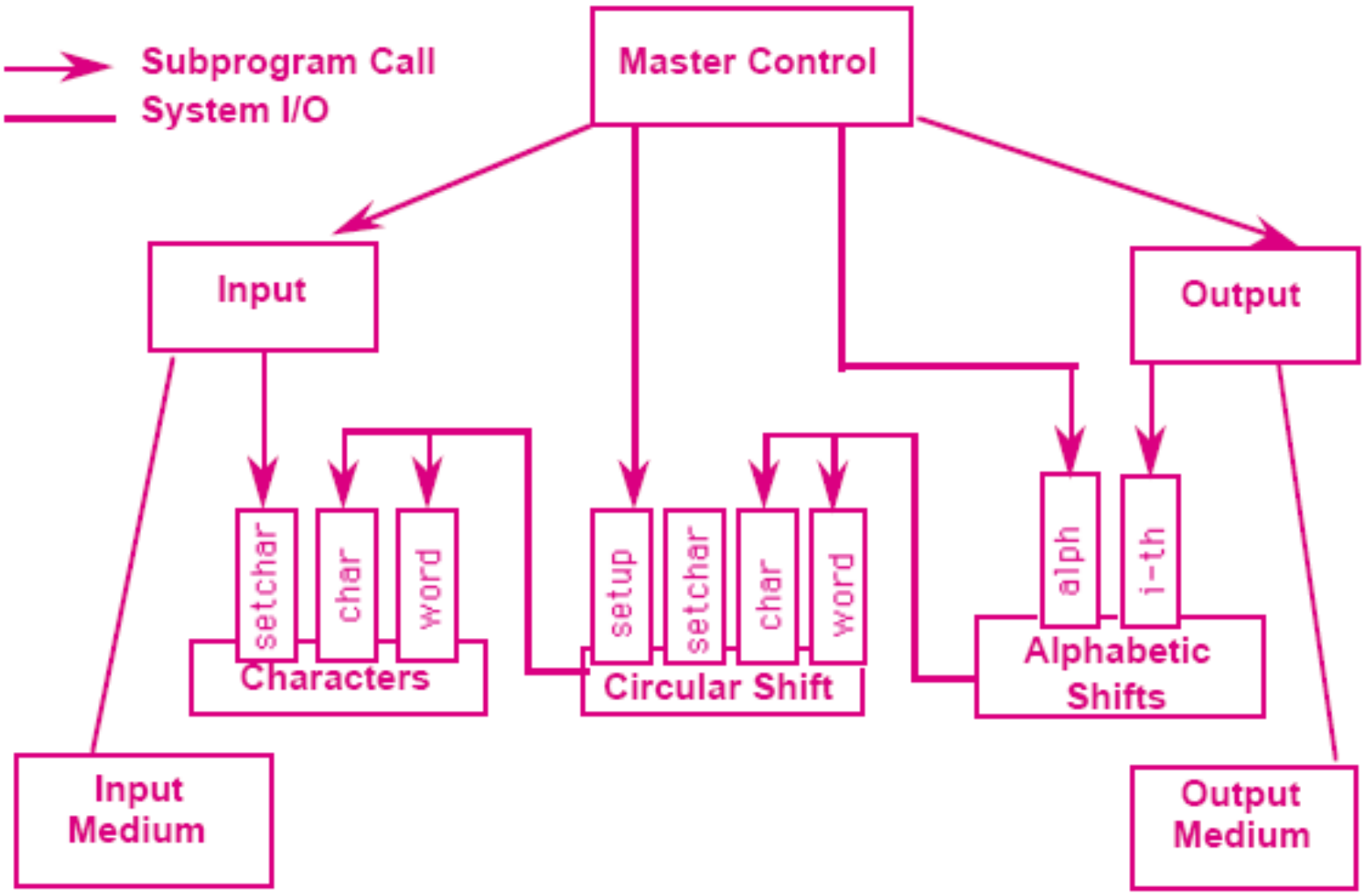


Solution 1 - Shared Data: Main Program & Subroutines
The main program sequences through sub-programs in turn. Shared Data Storage.
The coordinating program guarantees sequential access to the data.

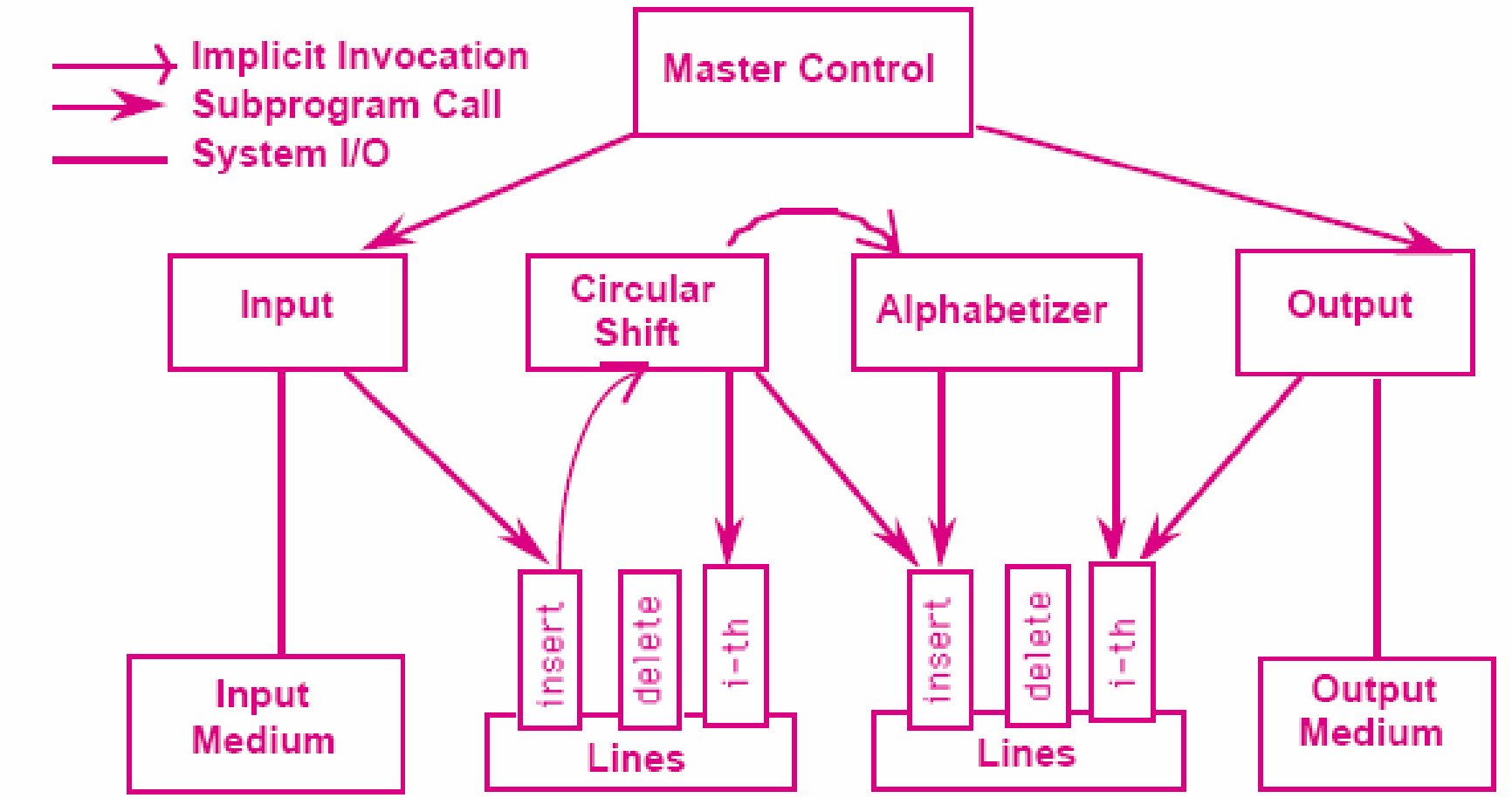


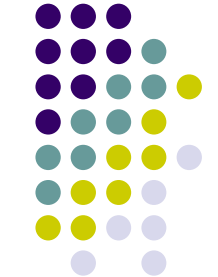
Solution 2 - ADT

Data is no longer directly shared by the computational components. Instead, each module provides an interface that permits other components to access data only by invoking procedures in that interface.

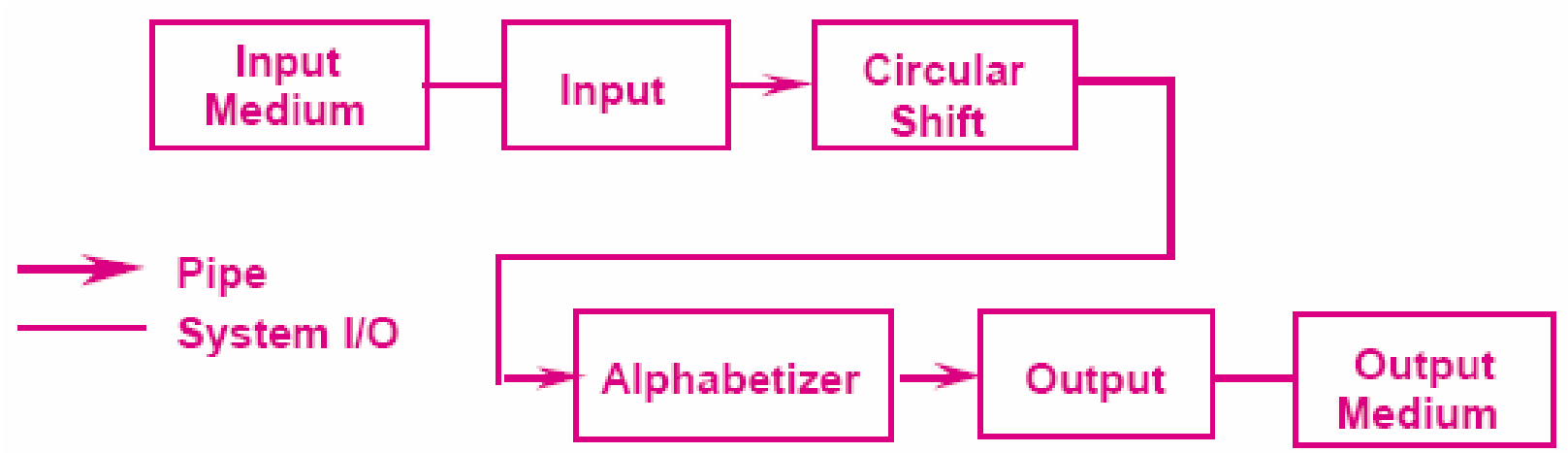


Solution 3 - Implicit Invocation:
data is accessed abstractly (for example, as a list or set)
computations are invoked implicitly as data is modified.
the act of adding a new line to the line storage causes an event to be sent to the Circular shift module. This in turn causes the alphabetizer to be implicitly invoked so that it can alphabetize the lines.

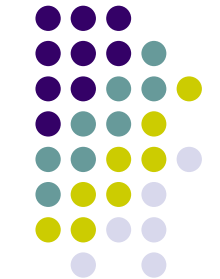




Solution 4 - Pipes and Filters
Each filter processes its data, sending it to the downstream filter. Control is distributed: each filter can run whenever it has data on which to compute.



Comparison

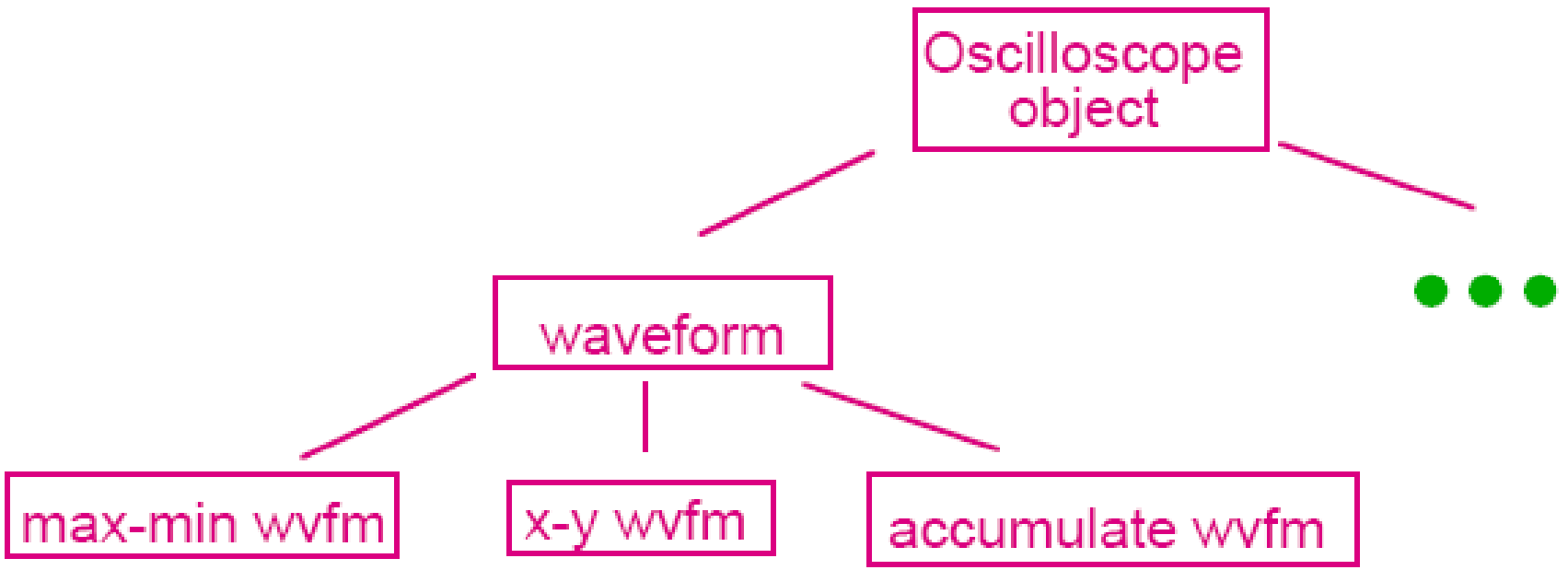


	Shared Memory	ADT	Events	Dataflow
Change in Algorithm	–	–	+	+
Change in Data Repn	–	+	–	–
Change in Function	+	–	+	+
Performance	+	+	–	–
Reuse	–	+	–	+

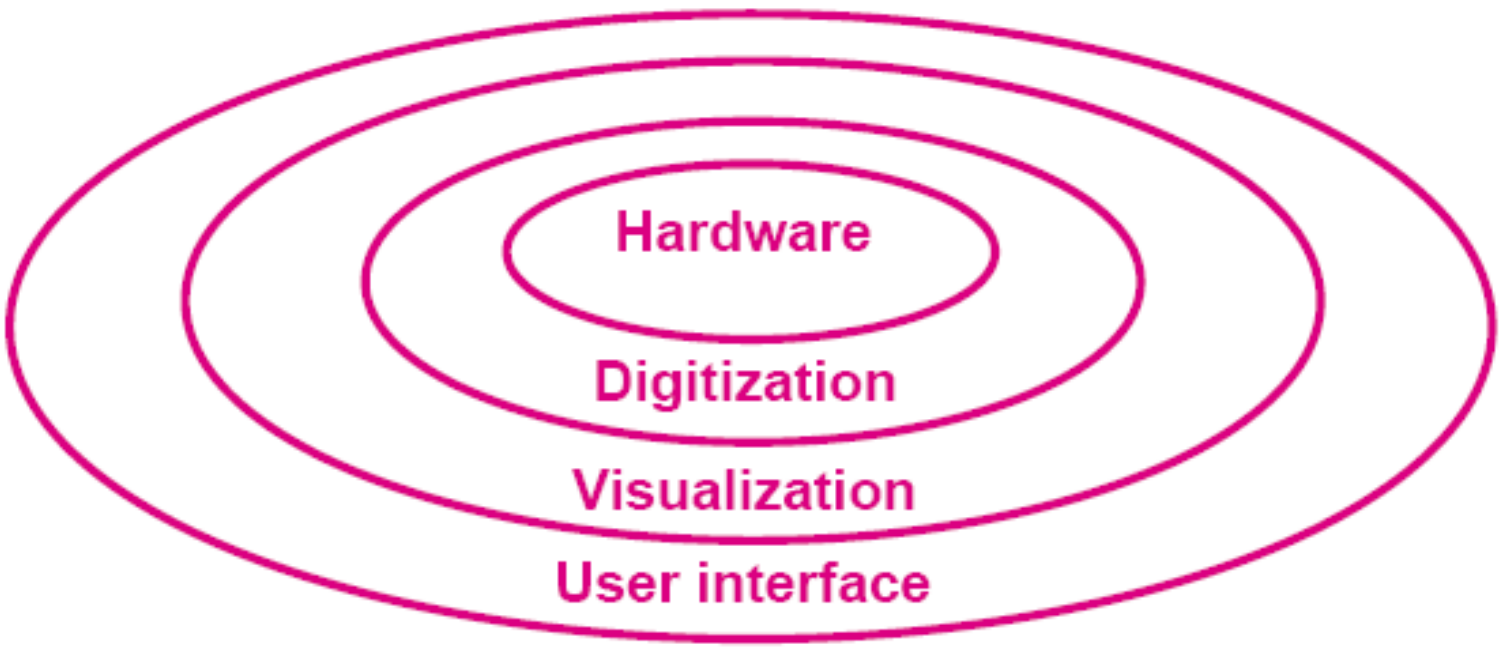
Case Study 2 - Instrumentation Software

- Developing architectural frameworks for oscilloscopes
  - Allowing rapid changes
  - High performance

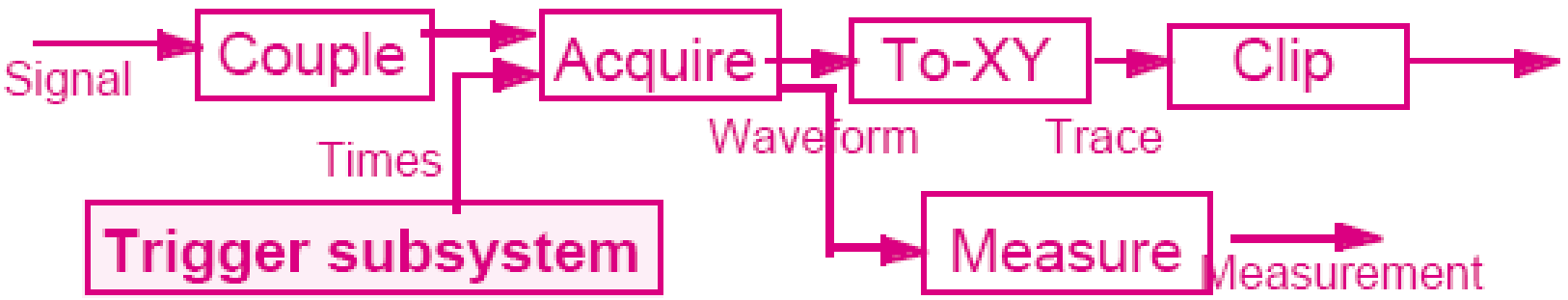
Object-oriented model



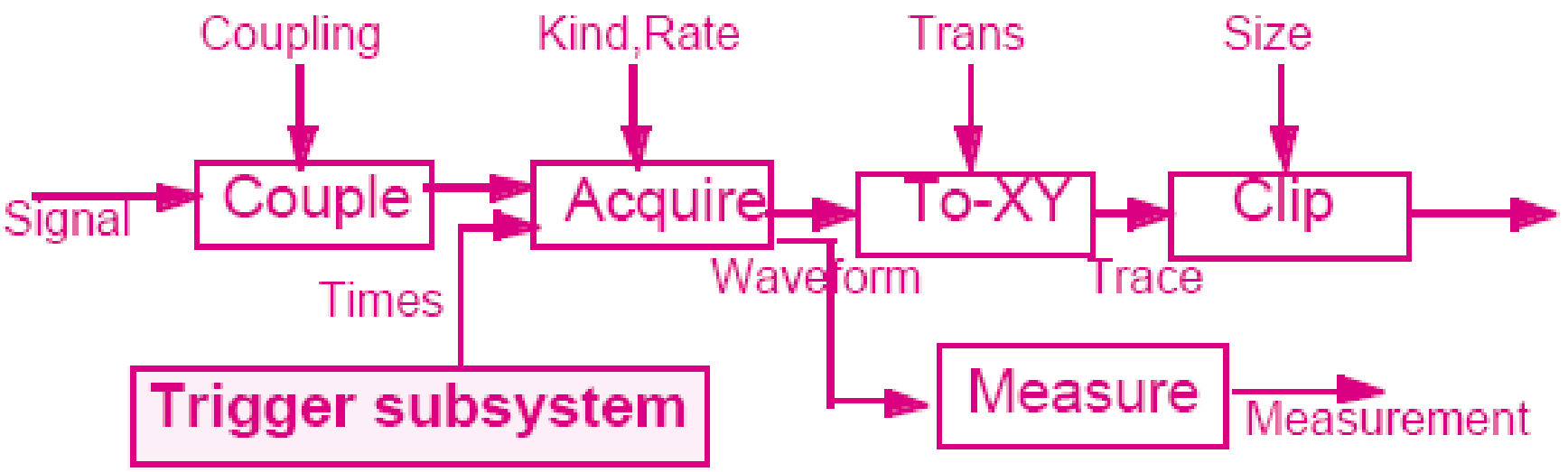
Layered Model



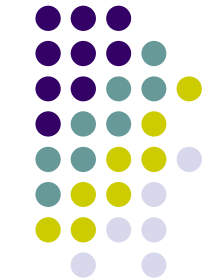
Pipe and Filter Model



Enhanced Pipe and Filter



Past, Present, and Future



- Architectural A&D: viable and worth doing
- Emerging vocabulary of architectural styles
- Future
  - Taxonomies
  - Formal methods
  - Primitive semantic entities
  - Tools and environments
  - Arch. extract tools
  - Role of arch. in development life-cycles

Research Suggestions

- Effectiveness of architectural research in certain domains
- Architectural references