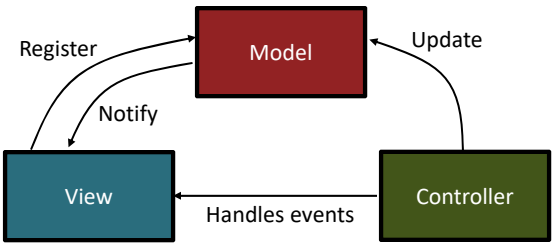
CSC40232: SOFTWARE ENGINEERING

Professor: Jane Cleland-Huang

Architecture 2 – A Birds Eye View of High-Level Design Decisions
Wednesday, April 24th
sarec.nd.edu/courses/SE2017

Department of Computer Science and Engineering

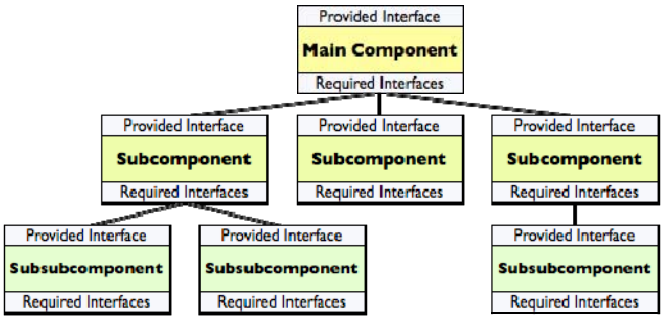
Model View Controller



- **Model**: data underlying the application, and changes in state require notifying the view. Model abstracts the data storage and/or maintains state.
- **View**: UI –A model may have many views
- **Controller**: receives input and initiates response by calling the model

http://www.thomasalspaugh.org/pub/fnd/architecture.html

Hierarchical Style:

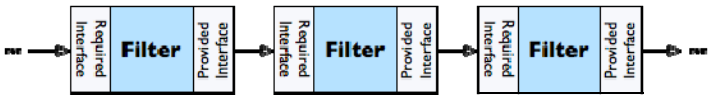


The interface of the system is provided by a single component (the Main Component above). Some components implement their interfaces with the help of subcomponents; no subcomponent contributes to more than a single parent component. The connectors are typically function or method calls.

http://www.thomasalspaugh.org/pub/fnd/architecture.html

3

Pipe and Filter



A pipe-and-filter architecture utilizes components that are filters: they take a sequence of inputs and produce a sequence of processed outputs from it. The outputs of each filter become the inputs of the next filter in the pipeline. The system may rearrange the filters in the pipeline at run time, substituting, adding, or deleting filters to achieve different results.

A typical application of this style is image, audio, or other signal processing.

http://www.thomasalspaugh.org/pub/fnd/architecture.html

4

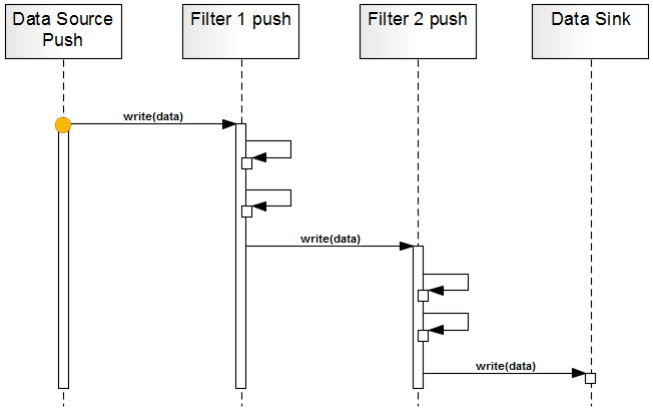
Filters

- **Passive filters**
  - Subsequent pipeline element **pulls** output data from filter.
  - Previous pipeline element **pushes** output data to filter.
- **Active filters**
  - **Starts processing on its own as a separate program or thread.**
  - **Most common:** filter is active in a loop, pulling its input from the pipeline and pushing it down the pipeline.

5

Scenario 1:

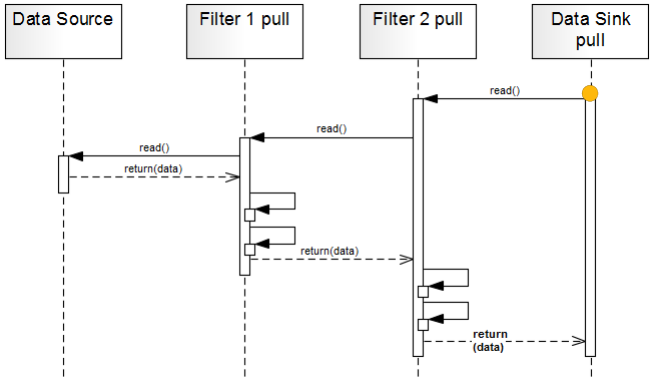
Push pipeline in which activity starts with the data source. Filter activity is triggered when data is written to passive filters.



6

Scenario 2:

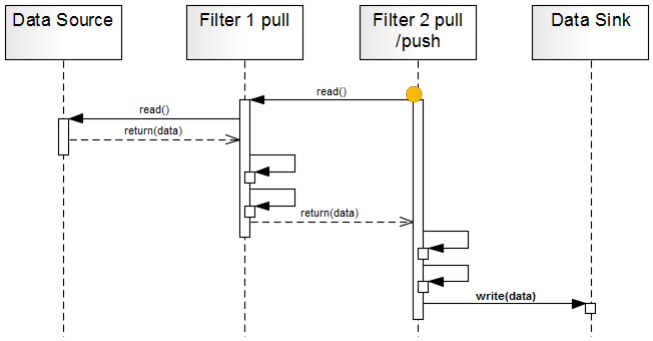
In the pull pipeline control flow is started by the datasink calling for data.



7

Scenario 3:

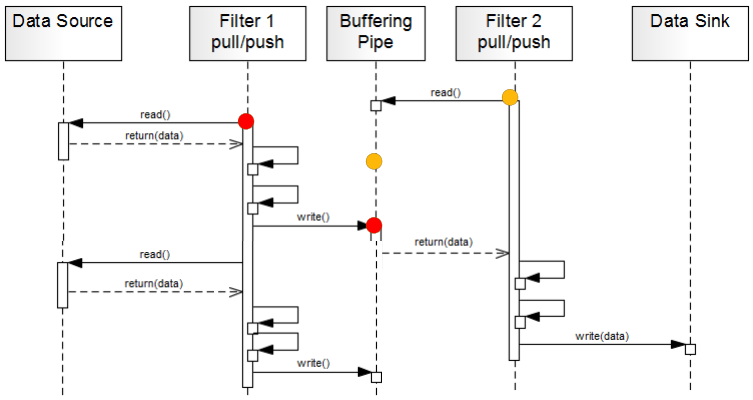
A mixed push-pull pipeline with passive data source and sink. The second filter plays the active role and starts the processing (in this example).



8

Scenario 4:

All filters actively pull, compute, and push data in a loop. Each runs its separate thread of control. Filters are synchronized by a buffering pipe between them. (In this example – the pipe buffers a single value only).



9

Think about it…

Joe

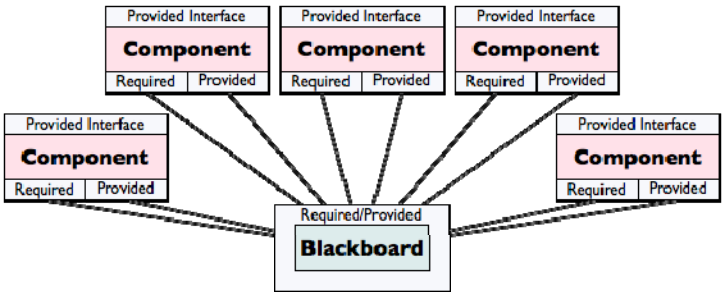
Joe has built an **active** pipeline with three filters. The amount of time a filter takes to process data is relative to the size of the filter in the diagram above (bigger = longer).

To what extent will Joe's pipeline improve performance (or not) over a non-pipelined style?

What changes would you recommend Joe makes?

10

Blackboard Architecture

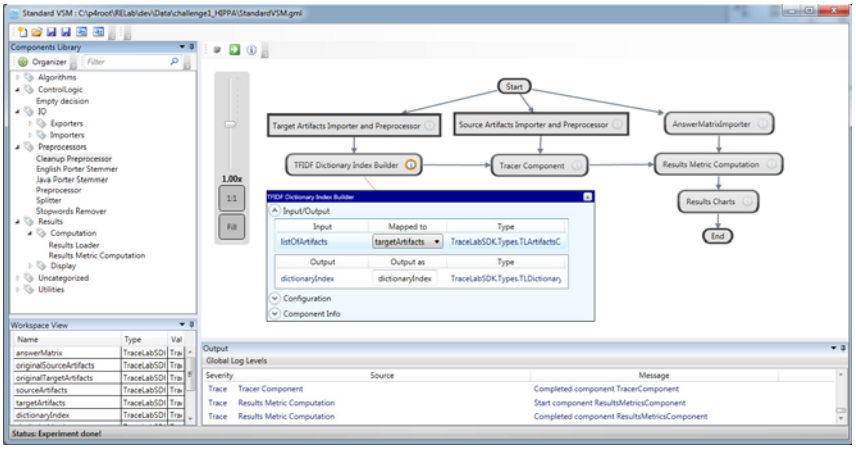


A blackboard architecture has a blackboard component, acting as a central data repository, and a number of components that act independently on the common data structure stored in the blackboard, responding to changes in it and in response making further changes. The components interact only through the blackboard.

http://www.thomasalspaugh.org/pub/fnd/architecture.html

11

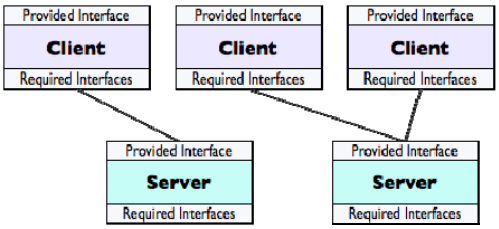
A Relaxed Blackboard Architecture



TraceLab – a tool for supporting traceability research.
(Similar systems MatLab, RapidMiner, etc etc.)

12

Client Server



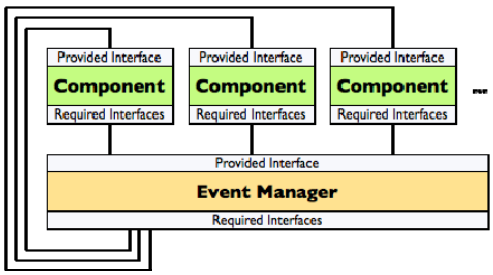
The *server* , provides an interface so that *clients*, connect as needed to request the service provided by the server.

The set of clients is often variable, & connections are commonly made only as needed. Several servers may provide the same services, in which case a client may connect to any, and may look up a server in some kind of directory in order to locate an appropriate one; clients may be directed to whichever server is least busy, in order to balance the load and provide fastest service.

http://www.thomasalspaugh.org/pub/fnd/architecture.html

13

Event Based Invocation



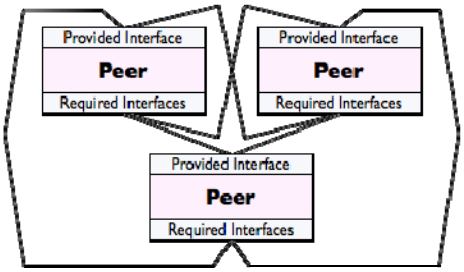
Components **register their interest** in specific events with an event manager (using its provided interface), making a *callback method* on its provided interface available to the event manager as one of its required interfaces.

When a **registered event is detected** by the event manager, it calls the callback method of every component registered for it and gives each one the event.

http://www.thomasalspaugh.org/pub/fnd/architecture.html

14

Peer to Peer



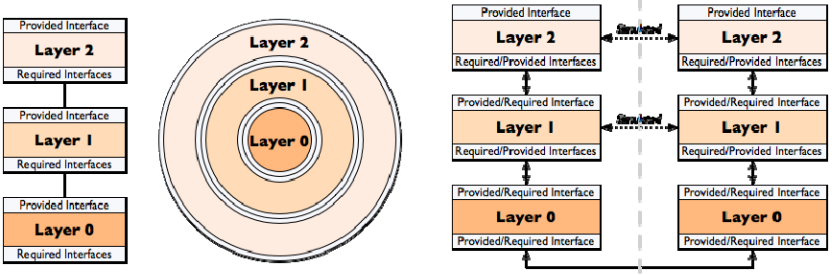
In a peer-to-peer architecture, all components are at the same "level" and each may require the interface of any or every other component. A peer-to-peer architecture provides little structure for a system.

http://www.thomasalspaugh.org/pub/fnd/architecture.html

15

OTHER STYLES

Layers and Protocol Stacks:



Two stacks simulate communication at each level, although in fact communication only occurs between the lowest levels, the Layer 0s. Each Layer 2 provides an interface for communicating with the other stack through its Layer 2 interface. It appears to the user of Layer 2 that the communication is occurring at that level. However, communication is really passed down to the next layer.

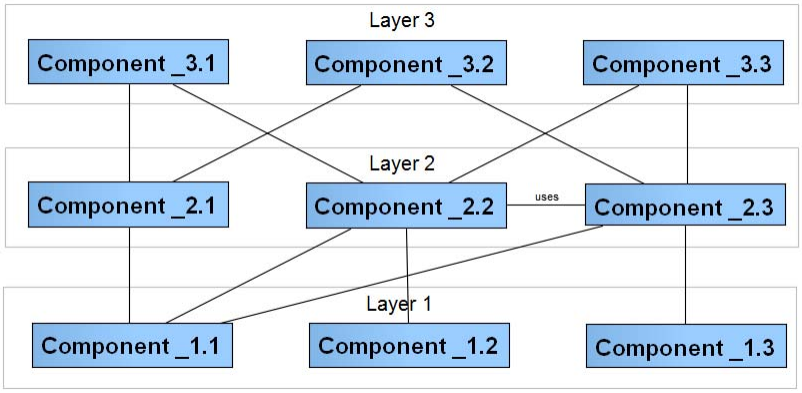
http://www.thomasalspaugh.org/pub/fnd/architecture.html

16

OTHER STYLES

LAYERS

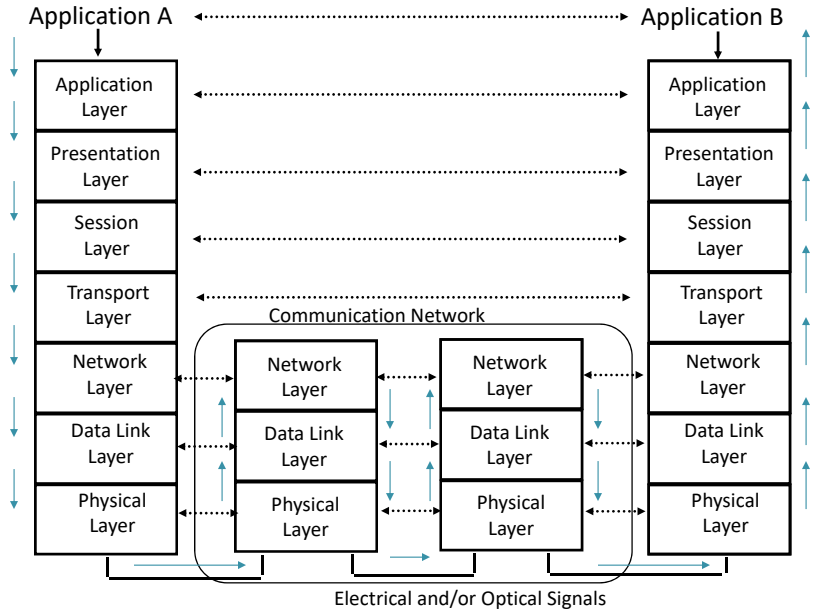
Layers



- Components in higher levels call lower level ones.
- Components use other components in the same level.
- Layers can be black, white, or gray box.

17

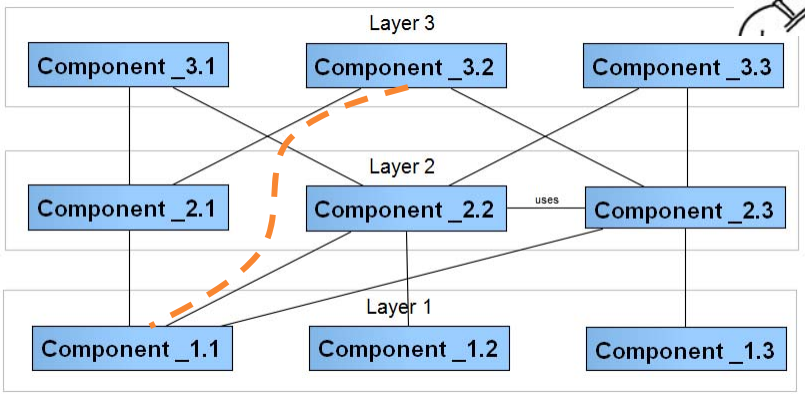
The ISO Reference Model



Borrowed from: *cs.iupui.edu/~xkzou/teaching/CS436/chapter2_l1.ppt*

18

Think about it…

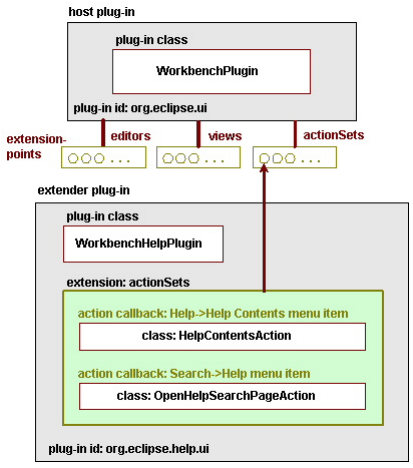


- Joe Architect wants to relax the layers architecture and allow Component 3.2 to call Component 1.1 directly.
- Explain the potential reasons for his decision and the possible consequences.

LAYERS

19

Plugin Architecture



The Plugin architectural pattern provides an extensible environment either by providing an API that allows *plugins to access your application* or an existing *application to access plugins*.

Example: Eclipse framework in which the base application is the plug-in framework.

LAYERS

20