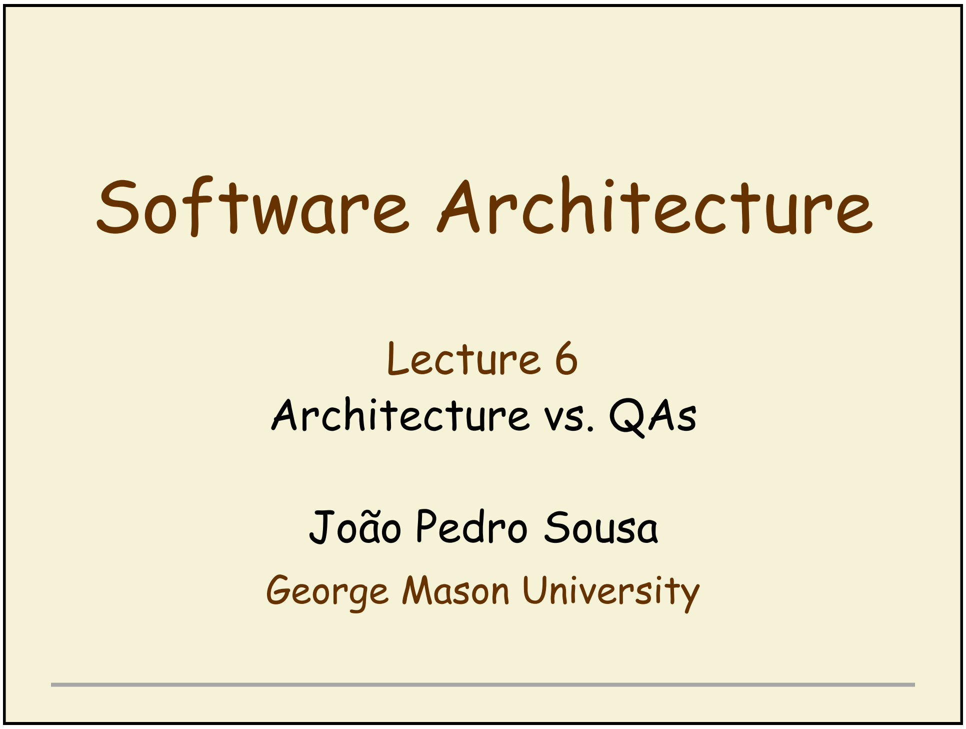


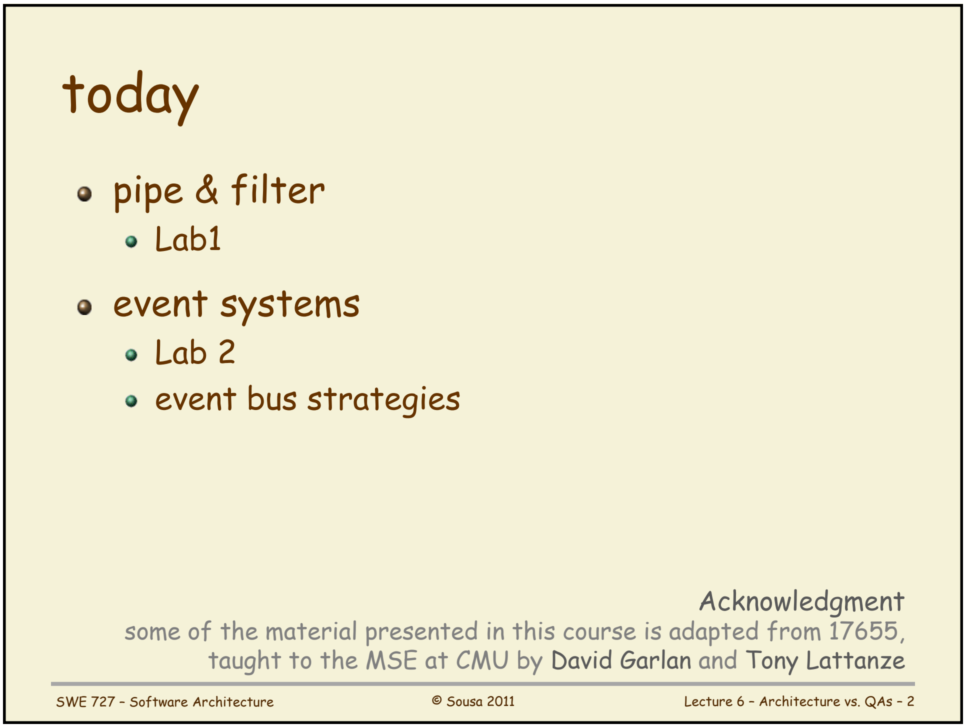
# Software Architecture

Lecture 6

Architecture vs. QAs

João Pedro Sousa

George Mason University



## today

- pipe & filter
  - Lab1
- event systems
  - Lab 2
  - event bus strategies

Acknowledgment
some of the material presented in this course is adapted from 17655,
taught to the MSE at CMU by David Garlan and Tony Lattanze

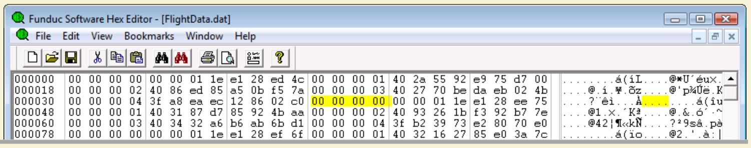
## lab 1: pipe & filter system
## build avionics instrumentation systems

- data comes in from airplane sensors

| ID | Data Descriptions and Units | Type | Number of Bytes |
|---|---|---|---|
| 00 | Time: number of milliseconds since the Epoch (00:00:00 GMT on January 1, 1970) | long int | 8 |
| 01 | Velocity: airspeed of the vehicle, measured in knots per hour | double | 8 |
| 02 | Altitude: vehicle's distance from the average surface of oceans, measured in feet | double | 8 |
| 03 | Pressure: atmospheric pressure external to the vehicle, measured in PSI | double | 8 |
| 04 | Temperature: temperature of the vehicle's hull, measured in degrees Fahrenheit | double | 8 |
| 05 | Pitch: angle of the nose of the vehicle, if positive, the vehicle is climbing | double | 8 |

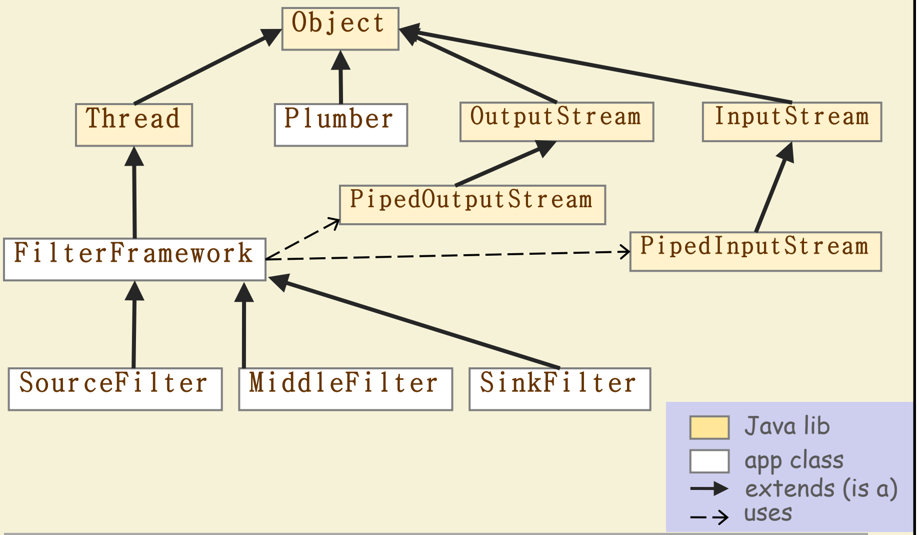
- framed as

| | | | | | | |
|---|---|---|---|---|---|---|
| 0000 | Time | 0001 | Velocity | … | *n* | *data* |
| 0000 | Time | 0001 | Velocity | … | *n* | *data* |
| | | | … | | | |


Funduc Software Hex Editor - [FlightData.dat]
File Edit View Bookmarks Window Help

```
000000  00 00 00 00 00 00 01 1e e1 28 ed 4c 00 00 00 01 40 2a 55 92 e9 75 d7 00  ........á(íL....@*U´éux.
000018  00 00 00 02 40 86 ed 85 a5 0b f5 7a 00 00 00 03 40 27 70 be da eb 02 4b  ....@.í.¥.õz....@'p¾Úë.K
000030  00 00 00 04 3f a8 ea ec 12 86 02 c0 00 00 00 00 00 00 01 1e e1 28 ee 75  ....?¨êì...À........á(îu
000048  00 00 00 01 40 31 87 d7 85 92 4b aa 00 00 00 02 40 93 26 1b f3 92 b7 7e  ....@1.×.´Kª....@.&.ó´·~
000060  00 00 00 03 40 34 32 a6 b6 ab 6b d1 00 00 00 04 3f b2 39 73 e2 80 70 e0  ....@42¦¶«kÑ....?²9sâ.pà
000078  00 00 00 00 00 00 01 1e e1 28 ef 6f 00 00 00 01 40 32 16 27 85 e0 3a 7c  ........á(ïo....@2.'.à:|
```

## lab 1
## existing system: module view


Object

Thread

Plumber

OutputStream

InputStream

PipedOutputStream

PipedInputStream

FilterFramework

SourceFilter

MiddleFilter

SinkFilter

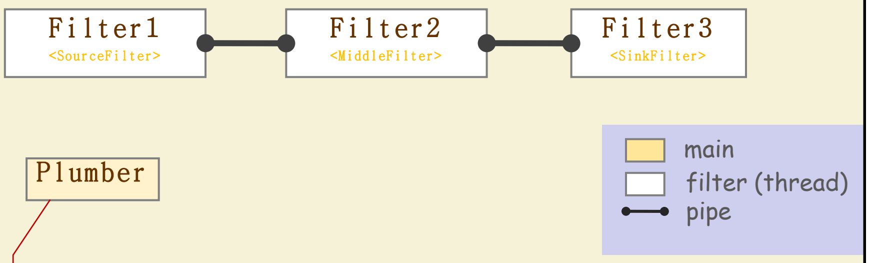
🟨 Java lib

□ app class

→ extends (is a)

⇢ uses

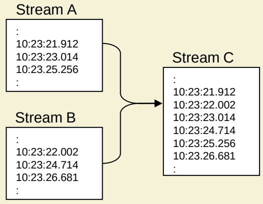
## lab 1
## existing system: C&C view



- creates and connects the filters
- doesn't intervene during system operation
  - therefore not normally represented

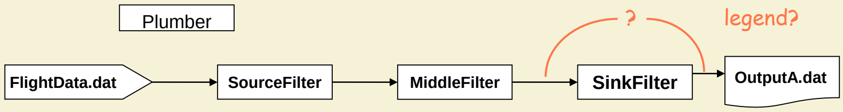
## lab 1: pipe & filter system
## build avionics instrumentation systems

- system A: reads flight data and
  - converts Temp to Celsius
  - converts altitude to meters
  - removes other fields
- system B: same plus
  - includes pressure data
  - removes pressure outliers > 80psi or <50 psi
    and replaces them by interpolated values (avg of previous and next)
- system C: merges streams
  from two sets of sensors
  - output frames are sorted by time
  - filter pressure & pitch



# sample 1: system A C&C view

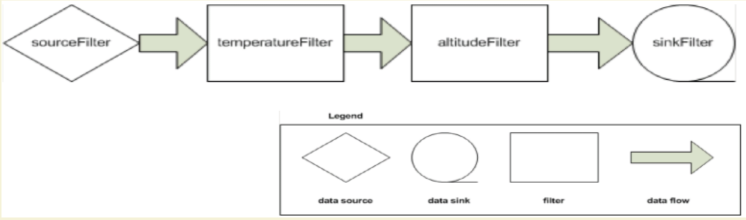
conv. temperature and altitude, remove others



```
public class MiddleFilter extends FilterFramework {
public void run() {
  <...>
  while(true) {
    id = readID();
    if ( id == 2 ) {
      data = readMeasurement();
      WriteFilterOutputPort(convert.intToByteArray(id));
      WriteFilterOutputPort(convert.longToByteArray(
                  convert.feetToMeters(Double.longBitsToDouble(data))));
    } else if ( id == 4 ) {
      data = readMeasurement();
      WriteFilterOutputPort(convert.intToByteArray(id));
      WriteFilterOutputPort(convert.longToByteArray(
                  convert.fahrenheitToCelsious(Double.longBitsToDouble(data))));
    } else {
      readMeasurement();
  }}
}}
```

# sample 2: system A C&C view

conv. temperature and altitude, remove others

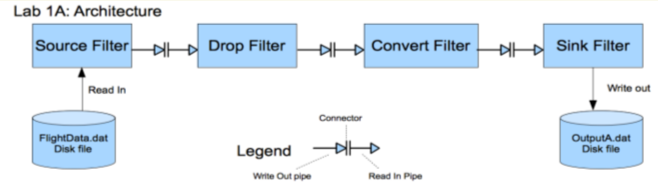


```
public class temperatureFilter extends FilterFramework {
public void run() {
  <...>
  while(true) {
    id = readID();
    data = readMeasurement();
    WriteFilterOutputPort(convert.intToByteArray(id));
    if ( id == 2 ) {
      Double altitude = Double.longBitsToDouble(data);
      Double meters = altitude * 0.3048;
      data = Double.doubleToLongBits(meters);
    }
    WriteFilterOutputPort(convert.longToByteArray(data);
  }
}}
```

```
public class altitudeFilter <...> {
public void run() {
 <...>
 while(true) {
   id = readID();
   data = readMeasurement();
   WriteFilterOutputPort(<..>id));
   if ( id == 4 ) {
     Double t = <…>data);
     Double c=(5.0/9.0)*(t-32);
     data=Double.doubleToLongBits(c);
   }
   WriteFilterOutputPort(<…>data);
 }
}}
```

## sample 3: system A C&C view

conv. temperature and altitude, remove others



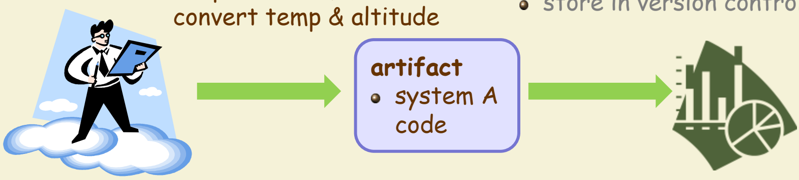
```
public class dropFilter extends FilterFramework {
public void run() {
  <...>
  while(true) {
    id = readID();
    data = readMeasurement();
    if ( id==0 || id==2 || id==4 ) {
      WriteFilterOutputPort(<..>id));
      WriteFilterOutputPort(<…>data);
    }
}}
```

```
public class ConvertFilter <...> {
public void run() {
 <...>
 while(true) {
   id = readID();
   data = readMeasurement();
   if ( id == 2 ) {
     Double a = <…>data);
     Double m= a*0.3048;
     data=Double.doubleToLongBits(m);
   } else if ( id == 4 ) {
     Double t = <…>data);
     Double c=(5.0/9.0)*(t-32);
     data=Double.doubleToLongBits(c);
   }
   WriteFilterOutputPort(<..>id));
   WriteFilterOutputPort(<…>data);
}}}
```

## which system is better for QA scenario cost of change aka modifiability

**stimuli**
- generate 2 new products for diff customers:
  1. keep only temp & altitude, convert only temp
  2. keep *all* fields, convert temp & altitude

**responses**
- make all changes
- store in version control



**artifact**
- system A code

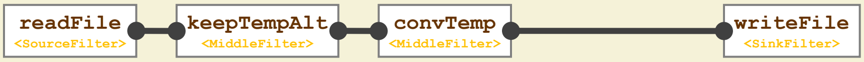
**source**
- developer

**environment**
- off line, design time

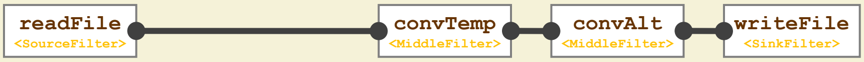
**response measures**
- 10 minutes
- no new bugs/ side effects

## revised system for
## cost of change scenario

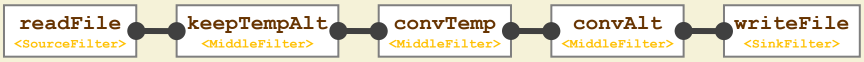
1. keep only temp & altitude,
   convert only temp



2. keep *all* fields,
   convert temp & altitude



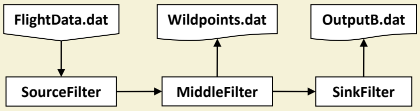
3. system A:
   keep only temp & altitude,
   convert temp & altitude

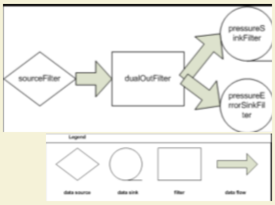


trivial changes to Plumber

## samples 1, 2: system B C&C view

same as A plus remove pressure wild points



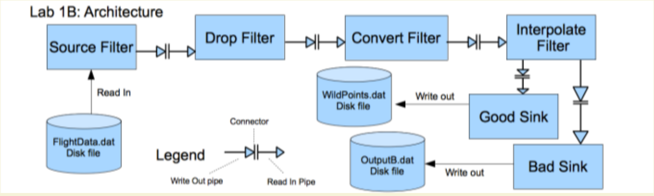


```
public class MiddleFilter extends FilterFramework {
public void run() {
   <...>
   while(true) { <...>
       processDataframe(frame);
   }
}
public void processDataframe(Dataframe f) { <...>
   f.setTemperature((f.getTemperature() - 32) * 5/9);
   f.setAltitude((f.getAltitude() * .3048);
   double pressure = f.getPressure();
   if (pressure > 80 || pressure < 50) { /* wild datapoint */
      wildPoints.println(time + pressure);
   }
   WriteFilterOutputPort(frame.toByteArray(),0,frame.size());
}}
```

```
all the action is...
   in the PressureSinkFilter
```

## sample 3: system B C&C view

same as A plus remove pressure wild points



```
public class InterpolateFilter extends FilterFramework {
public void run() {
  <...>
  while(true) {
     frame = readDataframe();
     if (frame.pressureOk()) {
       badPressureQ.procAll((frame.getPressure()+lastPressure)/2);
       lastPressure = frame.getPressure();
       WriteFilterOutputPort(frame.toByteArray(),0,frame.size());
     } else { /* wild datapoint */
       WriteFilterOutputAlternatePort(frame.toByteArray(),0,frame.size());
       badPressureQ.add(frame);
     }
  }
}}
```

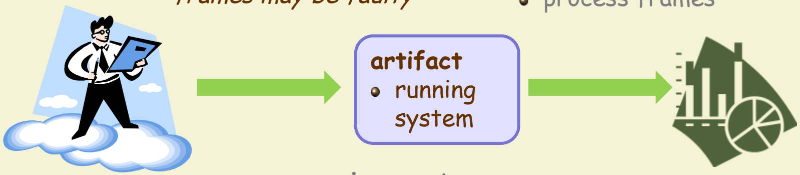
## does sample 3 satisfy response time scenario

**stimuli**

- send data frames
- *up to 20 consecutive frames may be faulty*

**responses**

- process frames



**artifact**

- running system

**source**

- sensor instrumentation

**environment**

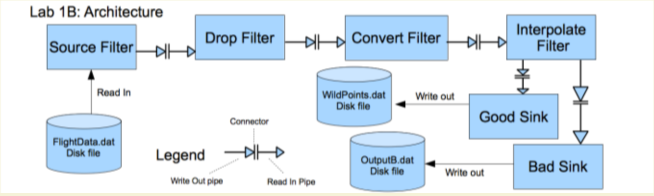
- normal operation
- input rate ~ 20 fps

**response measures**

- jitter < 0.1s

# revise sample 3 for QA scenario

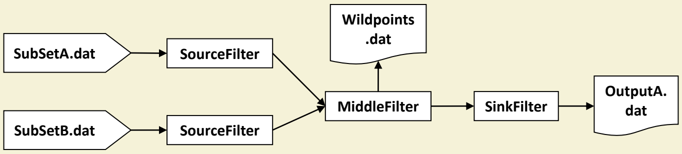
## change architecture or code?



```
public class InterpolateFilter extends FilterFramework {
public void run() {
   <...>
   while(true) {
      frame = readDataframe();
      if (frame.pressureOk()) {
         badPressureQ.procAll((frame.getPressure()+lastPressure)/2);
         lastPressure = frame.getPressure();
         WriteFilterOutputPort(frame.toByteArray(),0,frame.size());
      } else { /* wild datapoint */
         WriteFilterOutputAlternatePort(frame.toByteArray(),0,frame.size());
         badPressureQ.add(frame);
      }
   }
}}
```

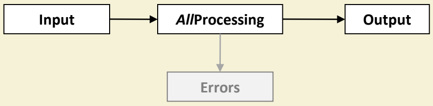
# sample 1: system C C&C view

## merge streams, remove wild points on pressure and pitch



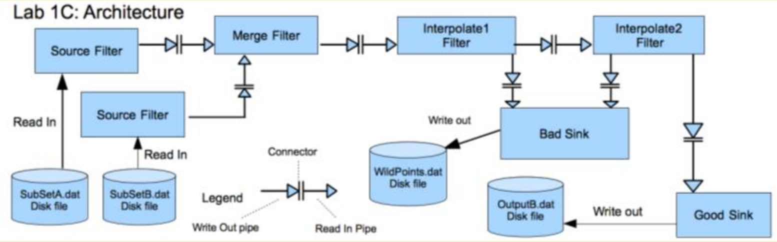
```
all the action is...
   in the MiddleFilter
```

- what is the real architectural style at play here?
  - does it respond to modifiability architectural drivers?

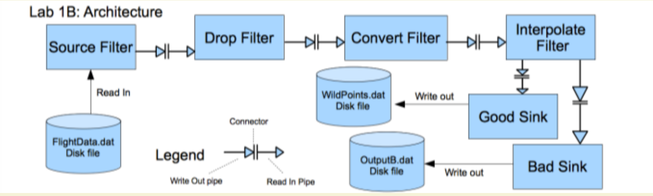


## sample 3: system C C&C view

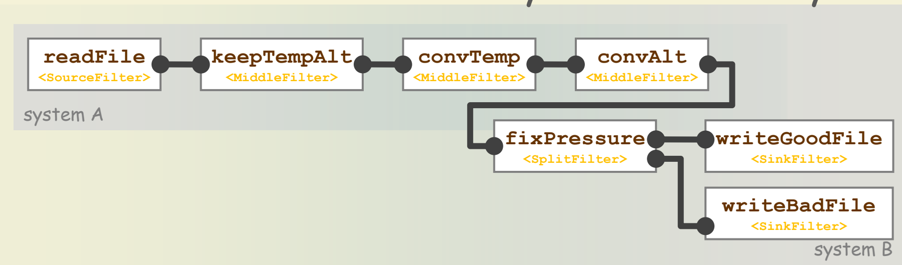
merge streams, remove wild points on pressure and pitch



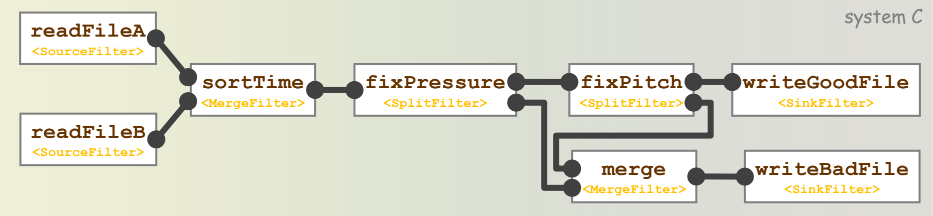
- identify reuse of parts from system B
  - could it be made more obvious/effective?



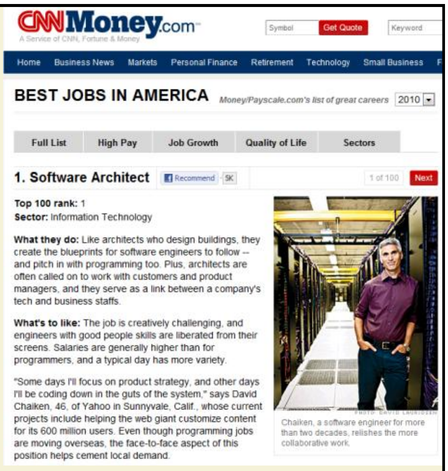
## revised systems
## maximize filter reuse & sys modifiability



- identify all instances of reuse



## architects learn by doing



**BEST JOBS IN AMERICA** *Money/Payscale.com's list of great careers*

**1. Software Architect**

**Top 100 rank:** 1
**Sector:** Information Technology

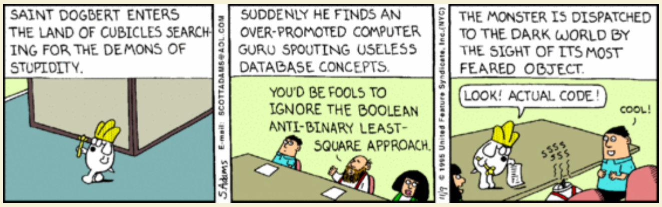
**What they do:** Like architects who design buildings, they create the blueprints for software engineers to follow -- and pitch in with programming too. Plus, architects are often called on to work with customers and product managers, and they serve as a link between a company's tech and business staffs.

**What's to like:** The job is creatively challenging, and engineers with good people skills are liberated from their screens. Salaries are generally higher than for programmers, and a typical day has more variety.

"Some days I'll focus on product strategy, and other days I'll be coding down in the guts of the system," says David Chaiken, 46, of Yahoo in Sunnyvale, Calif., whose current projects include helping the web giant customize content for its 600 million users. Even though programming jobs are moving overseas, the face-to-face aspect of this position helps cement local demand.

Chaiken, a software engineer for more than two decades, relishes the more collaborative work.

≠


SAINT DOGBERT ENTERS THE LAND OF CUBICLES SEARCHING FOR THE DEMONS OF STUPIDITY.
S.Adams E-mail: SCOTTADAMS@AOL.COM
SUDDENLY HE FINDS AN OVER-PROMOTED COMPUTER GURU SPOUTING USELESS DATABASE CONCEPTS.
YOU'D BE FOOLS TO IGNORE THE BOOLEAN ANTI-BINARY LEAST-SQUARE APPROACH.
11/7 © 1995 United Feature Syndicate, Inc.(NYC)
THE MONSTER IS DISPATCHED TO THE DARK WORLD BY THE SIGHT OF ITS MOST FEARED OBJECT.
LOOK! ACTUAL CODE!
COOL!

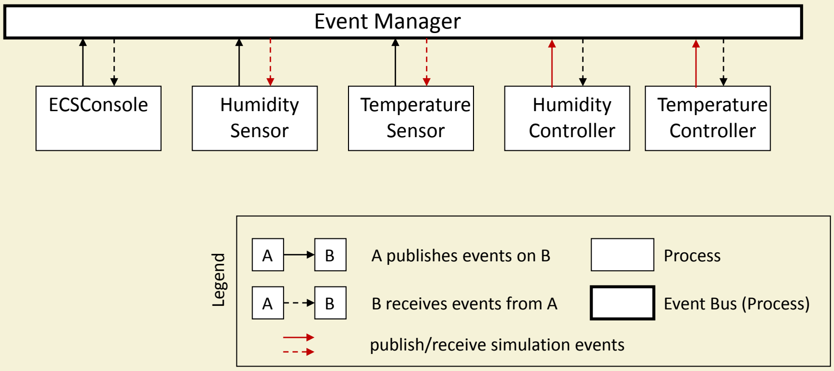
## take 5

# today

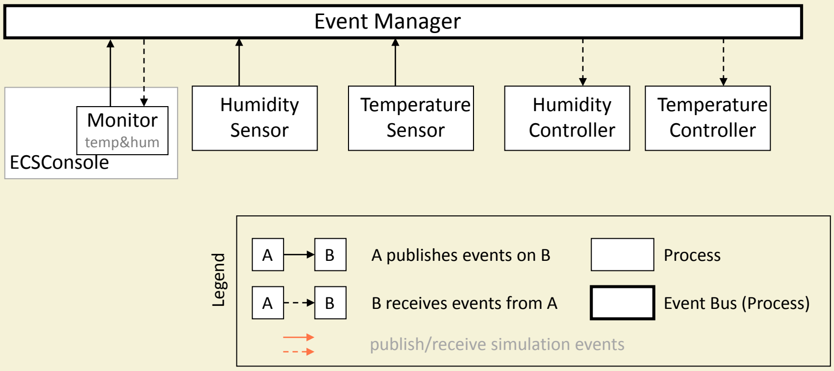
- pipe & filter
  - Lab1
- event systems
  - Lab 2
  - event bus strategies

Acknowledgment
some of the material presented in this course is adapted from 17655,
taught to the MSE at CMU by David Garlan and Tony Lattanze

# lab2
## C&C view

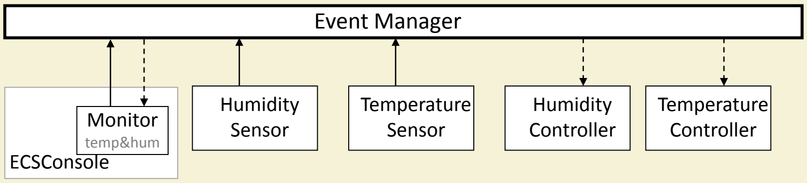


## lab2
## C&C view highlights

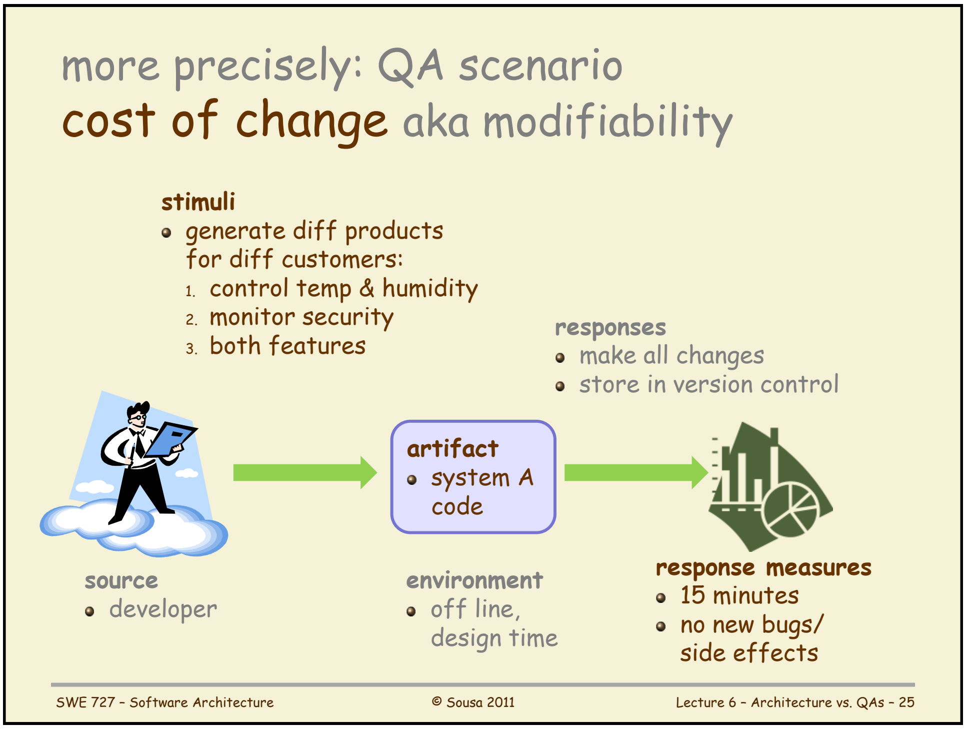


## lab2 system A
## add intrusion alarm

- new features
  - sensors: windows, doors…
  - intrusion alerts
- where to add the new features?



- *"a key requirement for this organization is to have a highly extensible system where sensors, equipment controllers, and consoles can be easily added to the system at runtime."*



## more precisely: QA scenario
## cost of change aka modifiability

**stimuli**
- generate diff products for diff customers:
  1. control temp & humidity
  2. monitor security
  3. both features

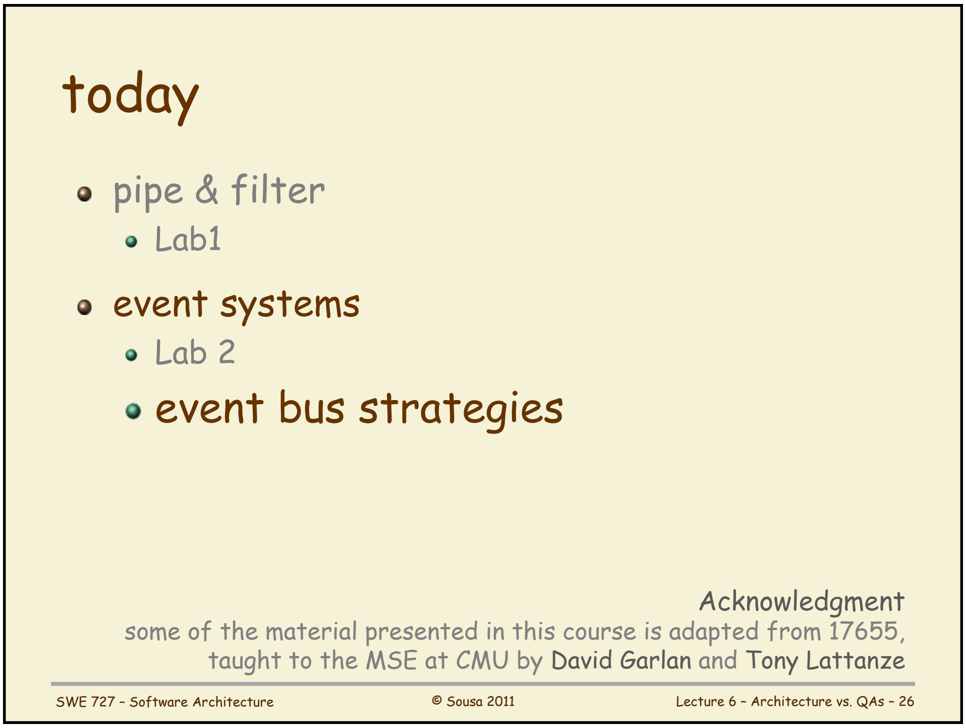
**responses**
- make all changes
- store in version control

**artifact**
- system A code

**source**
- developer

**environment**
- off line, design time

**response measures**
- 15 minutes
- no new bugs/ side effects



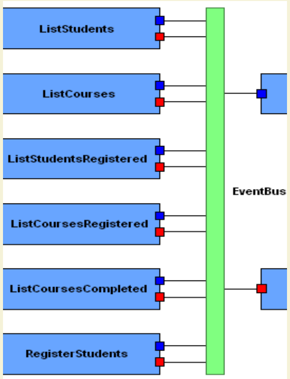
## today

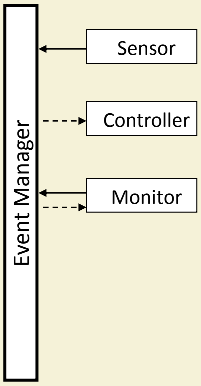
- pipe & filter
  - Lab1
- event systems
  - Lab 2
  - event bus strategies

Acknowledgment
some of the material presented in this course is adapted from 17655,
taught to the MSE at CMU by David Garlan and Tony Lattanze

# different bus strategies

## we saw two



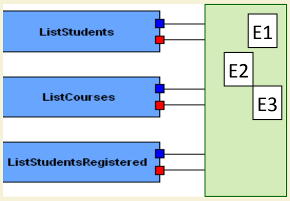


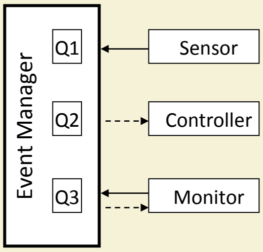
- interactive student registration
- single process
  - Java Observer/Observable
  - method call/callback

- lab 2
- distributed processes
  - Java RMI

# discuss effects on QA scenarios

## scalability, availability, latency…





- one Q per type of event (Observable)
- Observers register for specific type
  - Java manages Q
- sendEvent(e)
  - triggers notifyObservers()
  - Observers receive update(e) callback

- one Q per reg component
- sendEvent(e)
  - appends e to all Qs
- getEventQ(id)
  - each component pulls and processes its Q
  - ignores irrelevant events