Software Architecture

Lecture 6
Architecture vs. QAs

João Pedro Sousa
George Mason University

today

- relate architectural decisions to QA
  - Lab1

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

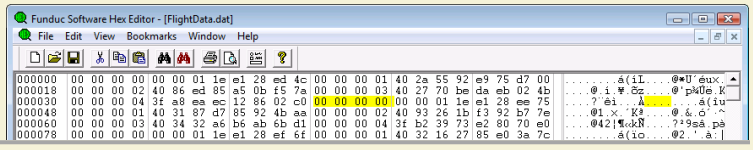
lab 1: pipe & filter system
build avionics instrumentation systems

- data comes in from airplane sensors

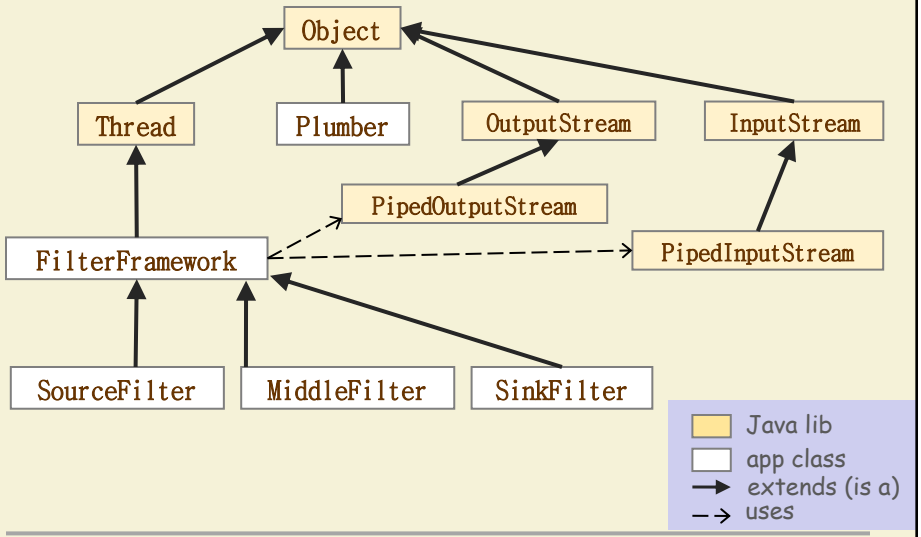
ID	Data Descriptions and Units	Type	Number of Bytes
00	Time: number of milliseconds since the Epoch (00:00:00 GMT on January 1, 1970)	long int	8
01	Velocity: airspeed of the vehicle, measured in knots per hour	double	8
02	Altitude: vehicle's distance from the average surface of oceans, measured in feet	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

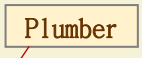
0000	Time	0001	Velocity	…	*n*	*data*
0000	Time	0001	Velocity	…	*n*	*data*
…						

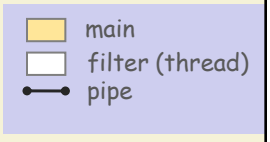


lab 1
existing system: module view



lab 1
existing system: C&C view

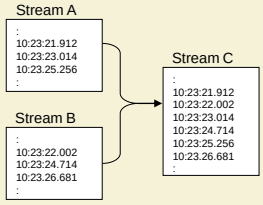




- creates and connects the filters
- doesn't intervene during system operation
  - therefore not normally represented

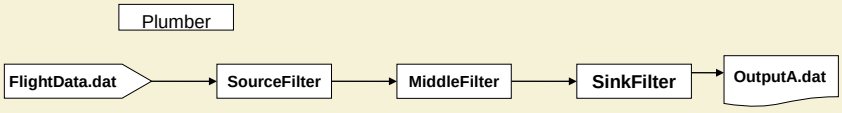
lab 1: pipe & filter system
build avionics instrumentation systems

- system A: reads flight data and
  - converts Temp to Celsius
  - converts altitude to meters
  - removes other fields
- system B: same conversions plus
  - keeps all fields
  - removes altitude wild jumps > 100m adjacent frames 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

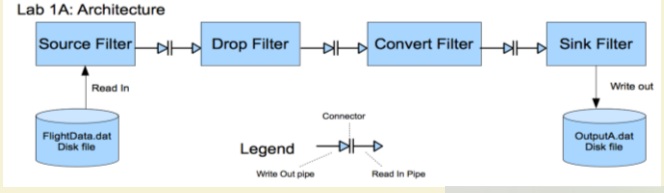
conv. temperature and altitude, remove others




```
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

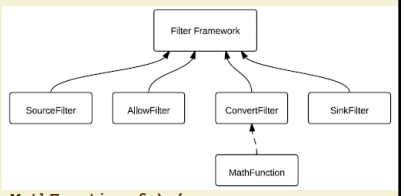


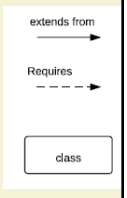

```
public class dropFilter extends FilterFramework {
public void run() {
   <...>
   while(true) {
      id = readID();
      data = readMeasurement();
      if ( id==0 || id==2 || id==4 ) {
         WriteFilterOutputPort(<..>id));
         WriteFilterOutputPort(<…>data);
      }
}}
```

```
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);
}}}
```

sample 3: system A code view

conv. temperature and altitude, remove others



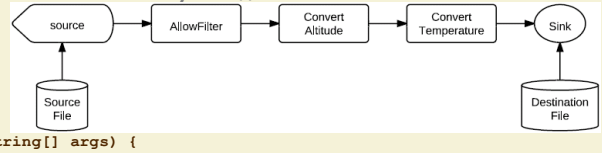



```
public interface MathFunction {
    public double calculate(Double d);
}

public class ConvertFilter extends FilterFramework {
  ...
    public ConvertFilter(int convertId, String name, MathFunction fx) {
        super();
        this.fx = fx;
        this.convertId = convertId;
        this.name = name;
    } ...
    public void run() { ...
        while (true) {
            try { ...
                if (id == convertId) {
                    measurement = super.getMeasurement();
                    double d = fx.calculate(measurement);
                    super.pipeOut(id, d);
                } else {
                    super.pipeOut(id);
                }
            }...
}
```


sample 3: system A C&C view

conv. temperature and altitude, remove others




```
public class A extends Thread{
    public static void main(String[] args) {
       A a = new A();
       a.start();
    }
    public void run() {
        SourceFilter source = new SourceFilter("FlightData.dat");
        AllowFilter allow = new AllowFilter(Arrays.asList(0,2,4));

        ConvertFilter convertTemp = new ConvertFilter(4, "Convert Temp Filter",
                                            new MathFunction() {
                                                  public double calculate(Double d) {
                                                     return (5.0 / 9.0) * (d - 32); }
                                            });
        ConvertFilter convertAlt  = new ConvertFilter(2, "Convert Alt Filter",
                                            new MathFunction() {
                                                  public double calculate(Double d) {
                                                     return d * 0.3048;             }
                                            });
        SinkFilter sink = new SinkFilter("OutputA.txt");

        sink.Connect(convertTemp); ...
}
```

sample 4: system A C&C view

conv. temperature and altitude, remove others


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

```
public class SystemA {
  public static void main() {
    SourceFilter readFile = new SourceFilter ("...");
    AllowFilter  clip = new AllowFilter(...));
    ConvTemp     temp = new ConvTemp();
    ConvAlt      alt  = new ConvAlt();
    SinkFilter  writeFile = new SinkFilter("...");

    sink.Connect(convertTemp); ...
}
```


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

source
- developer

artifact
- system A code

environment
- off line

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

systems for
cost of change scenario

1. keep only temp & altitude,
   convert only temp

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

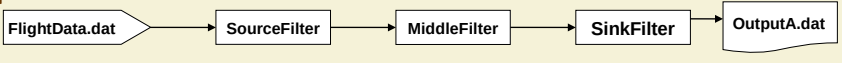
3. system *A*:
   keep only temp & altitude,
   convert temp & altitude

trivial changes to Plumber

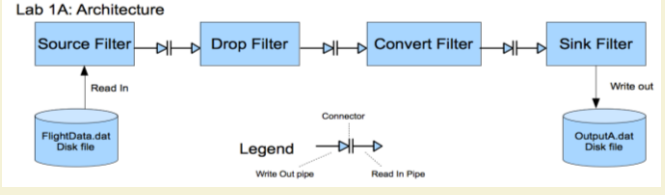
which system takes better advantage
deployment on multiple cores

key: thread/proc

sample 1



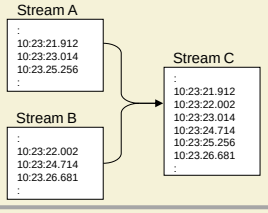
sample 2



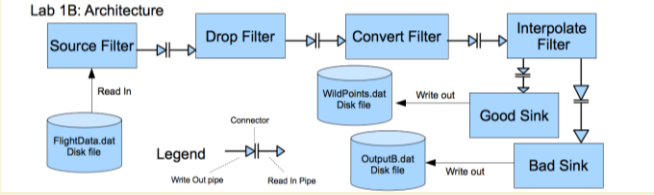
samples 3, 4

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 conversions plus
  - keeps all fields
  - removes altitude wild jumps > 100m adjacent frames 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 5: system B C&C view
same as A plus remove altitude wild jumps




```
public class InterpolateFilter extends FilterFramework {
public void run() {
   <...>
   while(true) {
      frame = readDataframe();
      if (frame.altitudeOk()) {
         badAltitudeQ.procAll((frame.getAltitude()+lastAltitude)/2);
         lastAltitude = frame.getAltitude();
         WriteFilterOutputPort(frame.toByteArray(),0,frame.size());
      } else { /* wild jump */
         WriteFilterOutputAlternatePort(frame.toByteArray(),0,frame.size());
         badAltitudeQ.add(frame);
      }
   }
}}
```

does sample 3 satisfy
response time scenario

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

responses
- process frames

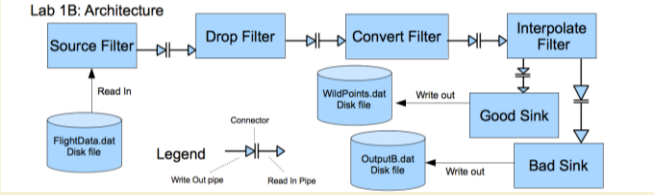
artifact
- running system

source
- sensor instrumentation

environment
- normal operation
- input rate ~ 20 fps

response measures
- jitter < 0.1s

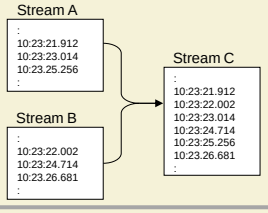
revise sample 5 for QA scenario
change architecture or code?



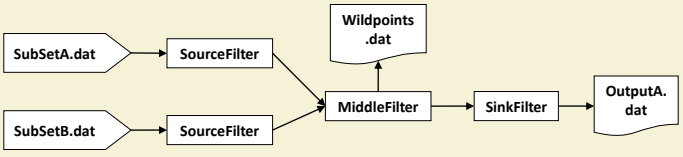

```
public class InterpolateFilter extends FilterFramework {
public void run() {
   <...>
   while(true) {
      frame = readDataframe();
      if (frame.altitudeOk()) {
         badAltitudeQ.procAll((frame.getAltitude()+lastAltitude)/2);
         lastAltitude = frame.getAltitude();
         WriteFilterOutputPort(frame.toByteArray(),0,frame.size());
      } else { /* wild jump */
         WriteFilterOutputAlternatePort(frame.toByteArray(),0,frame.size());
         badAltitudeQ.add(frame);
      }
   }
}}
```

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 conversions plus
  - keeps all fields
  - removes altitude wild jumps > 100m adjacent frames 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

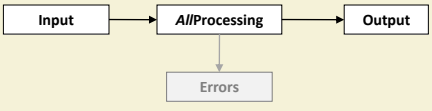


sample 6: system C C&C view
merge streams, remove wild points on pressure and pitch



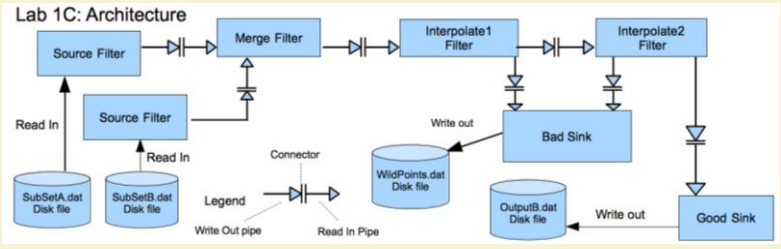

```
all the action is...
   in the MiddleFilter
```

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

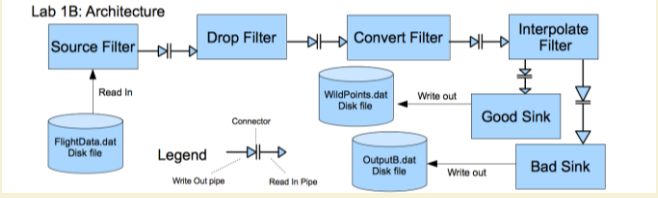


sample 7: system C C&C view

merge streams, remove wild points on pressure and pitch

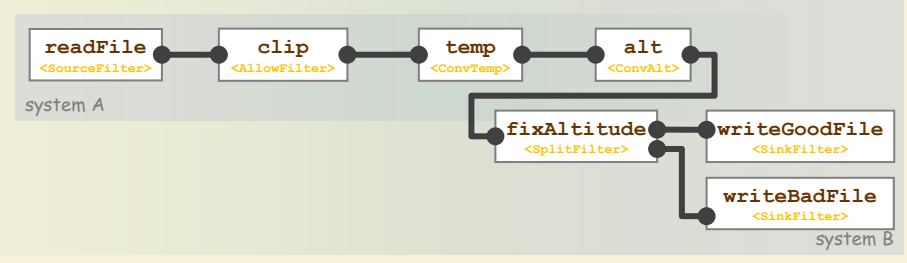


- identify reuse of parts from system B
  - could it be made more obvious/effective?

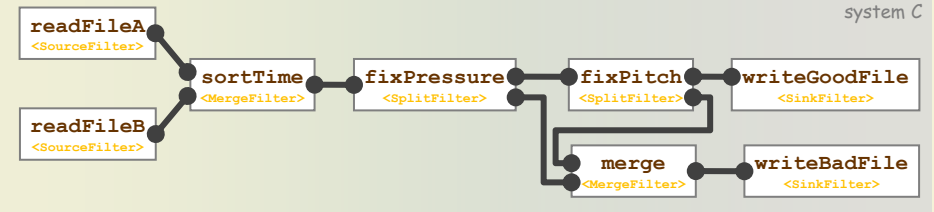


revised systems

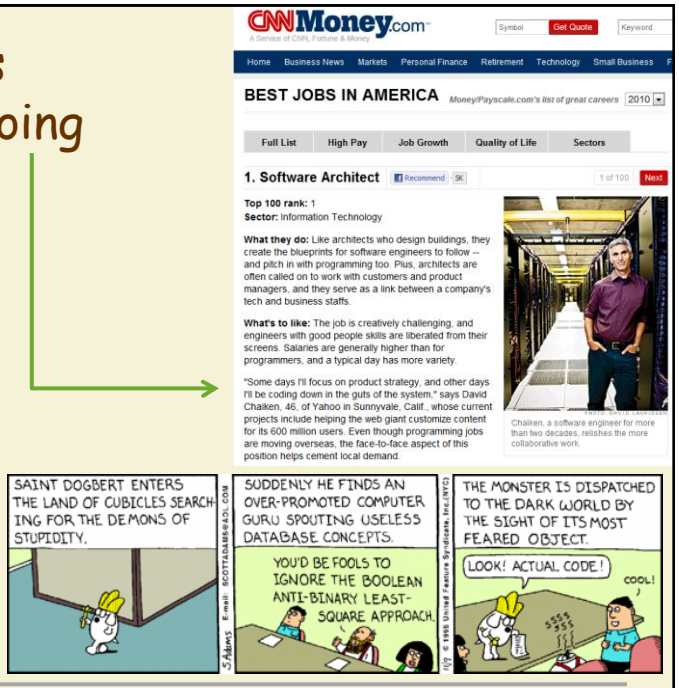
maximize filter reuse & sys modifiability



- identify all instances of reuse



architects learn by doing



≠