

# **Leverage Commonality at Model Level with AspectJ Assistance**

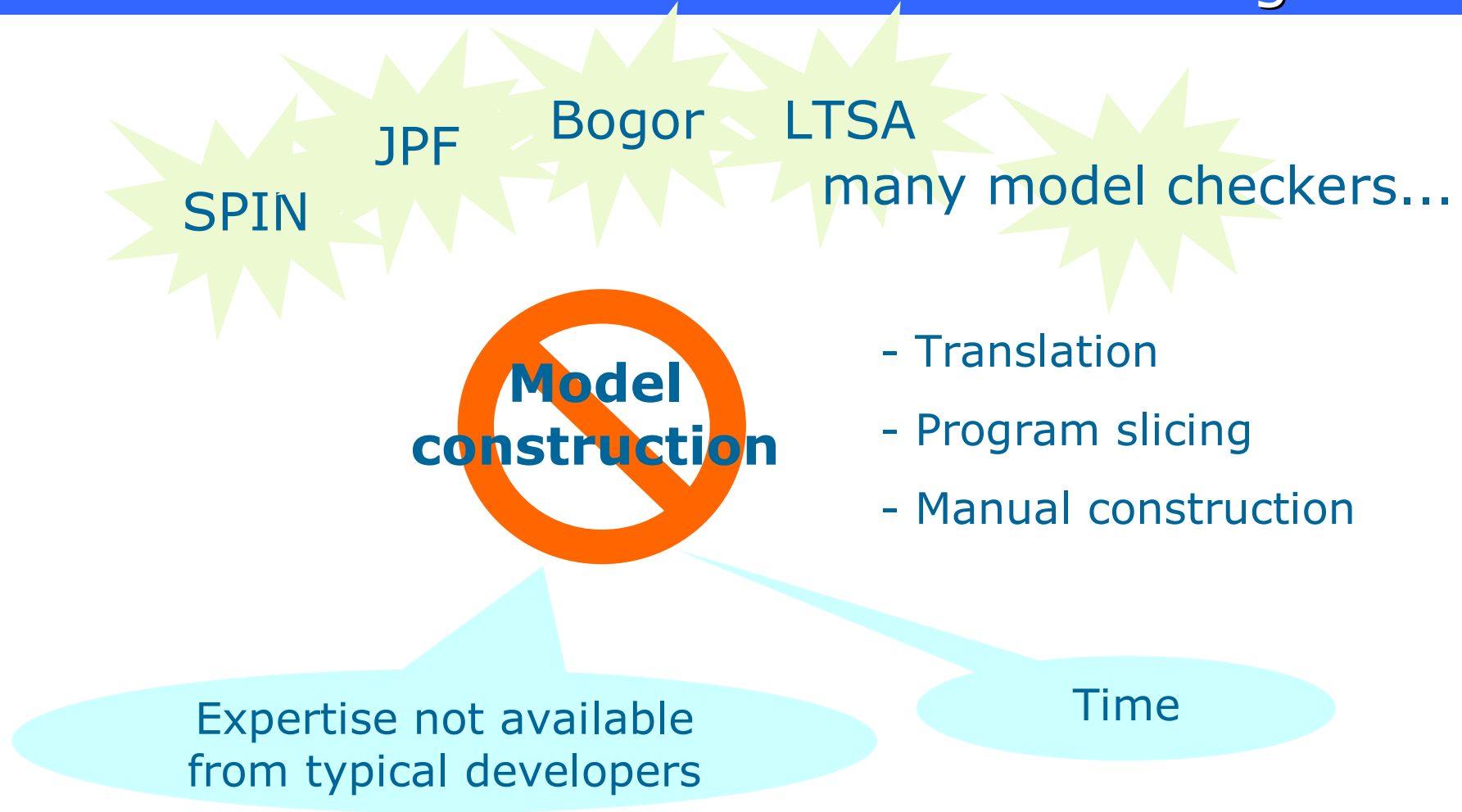
**Zebin Chen**

**Computer and Information Science  
University of Oregon**

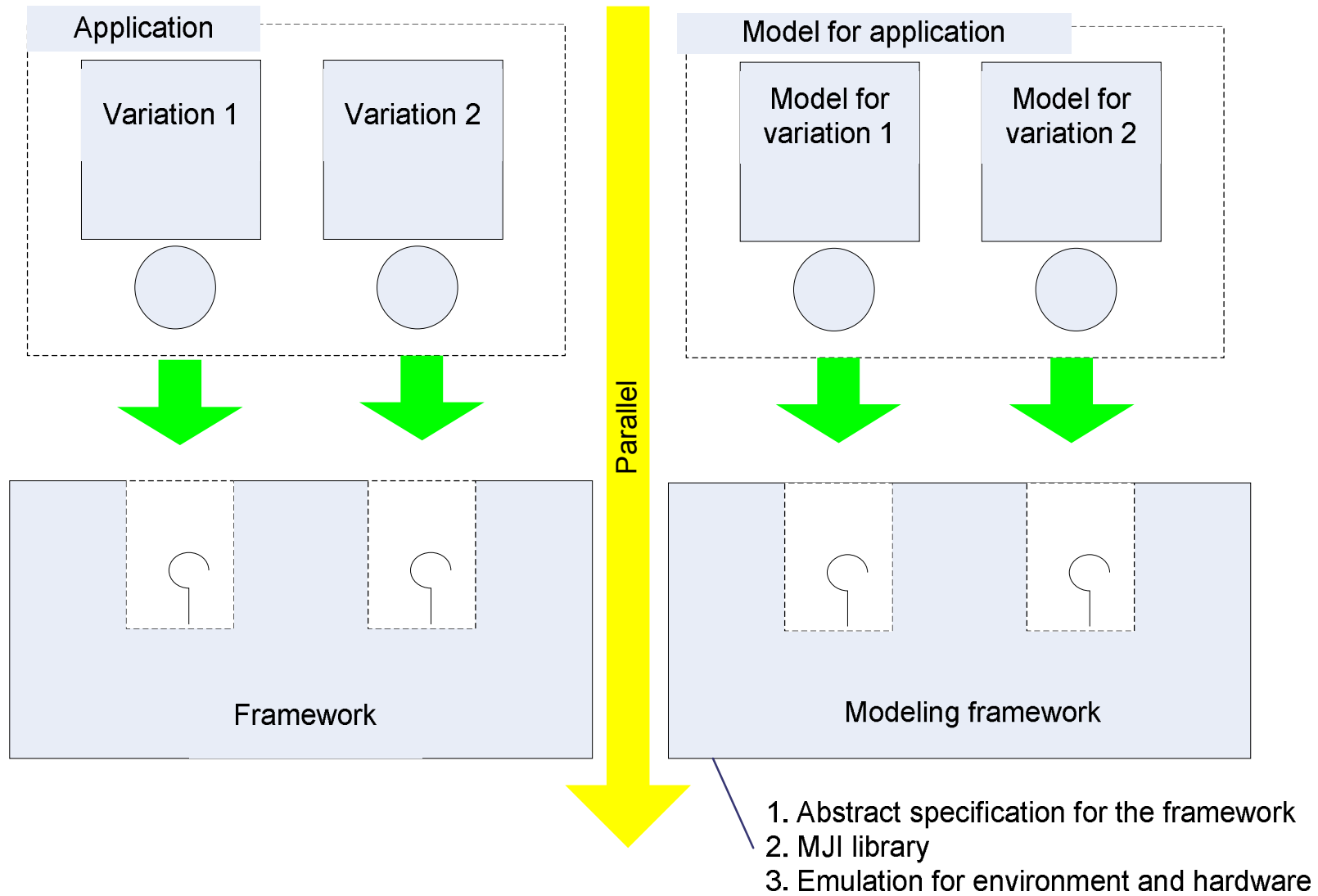
# Motivation

- We build Smart Home applications for cognitive impaired users
- We base our applications on OSGi
  - Ease integration and maintenance
  - Have implemented many commonalities
- We are concerned about correctness.

# Model construction hinders model checking



# An intuitive but vague idea



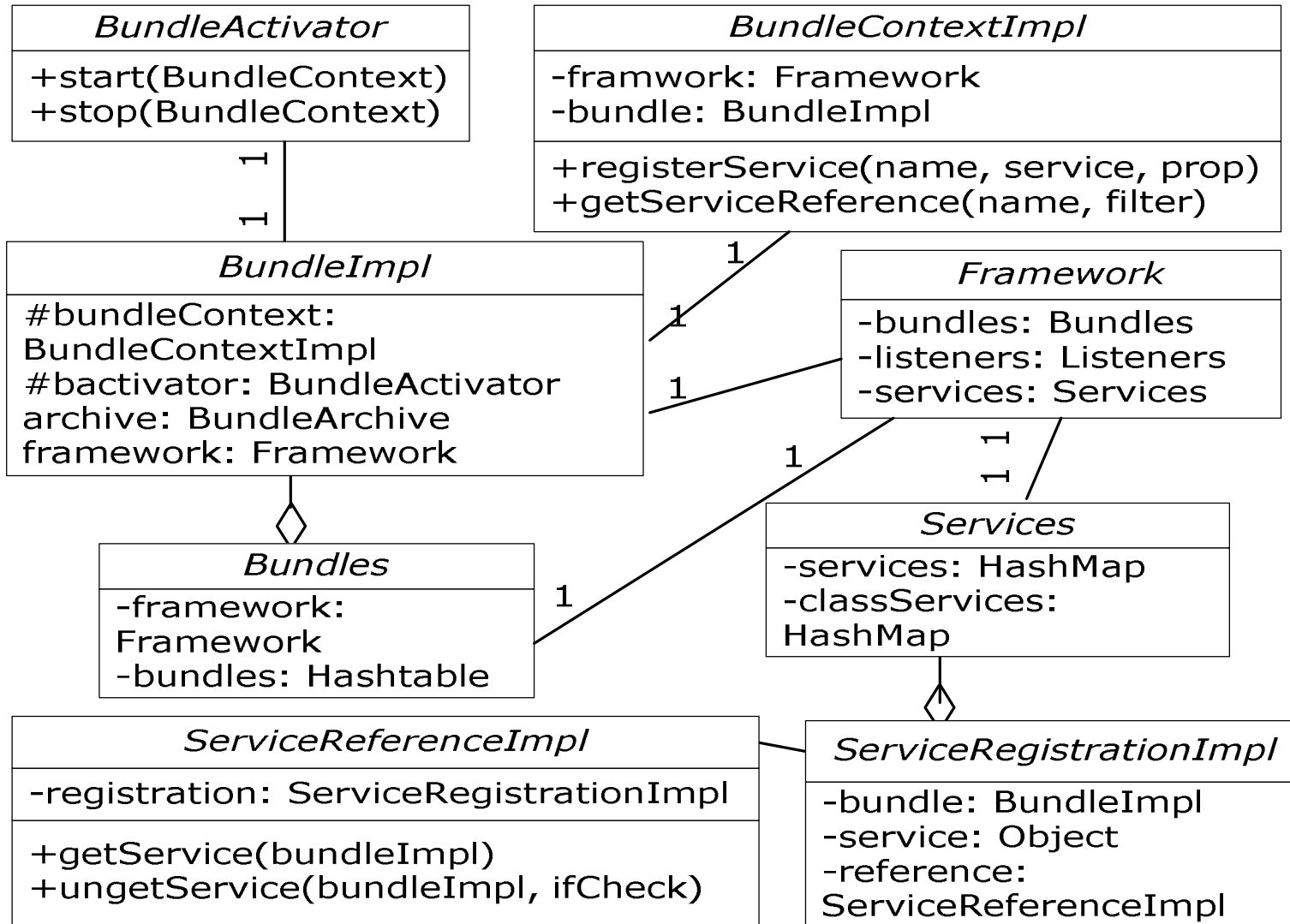
## Research questions:

- Can we build a modeling framework to ease model construction in a specific domain?
  - Has an Object-Oriented language provided enough variation points?
  - Concern: state space explosion
  - Concern: more dimensions for specialization
    - » Adding specifications
    - » Functionality and feature
    - » Vary granularity of atomicity
  
- Little literature about this

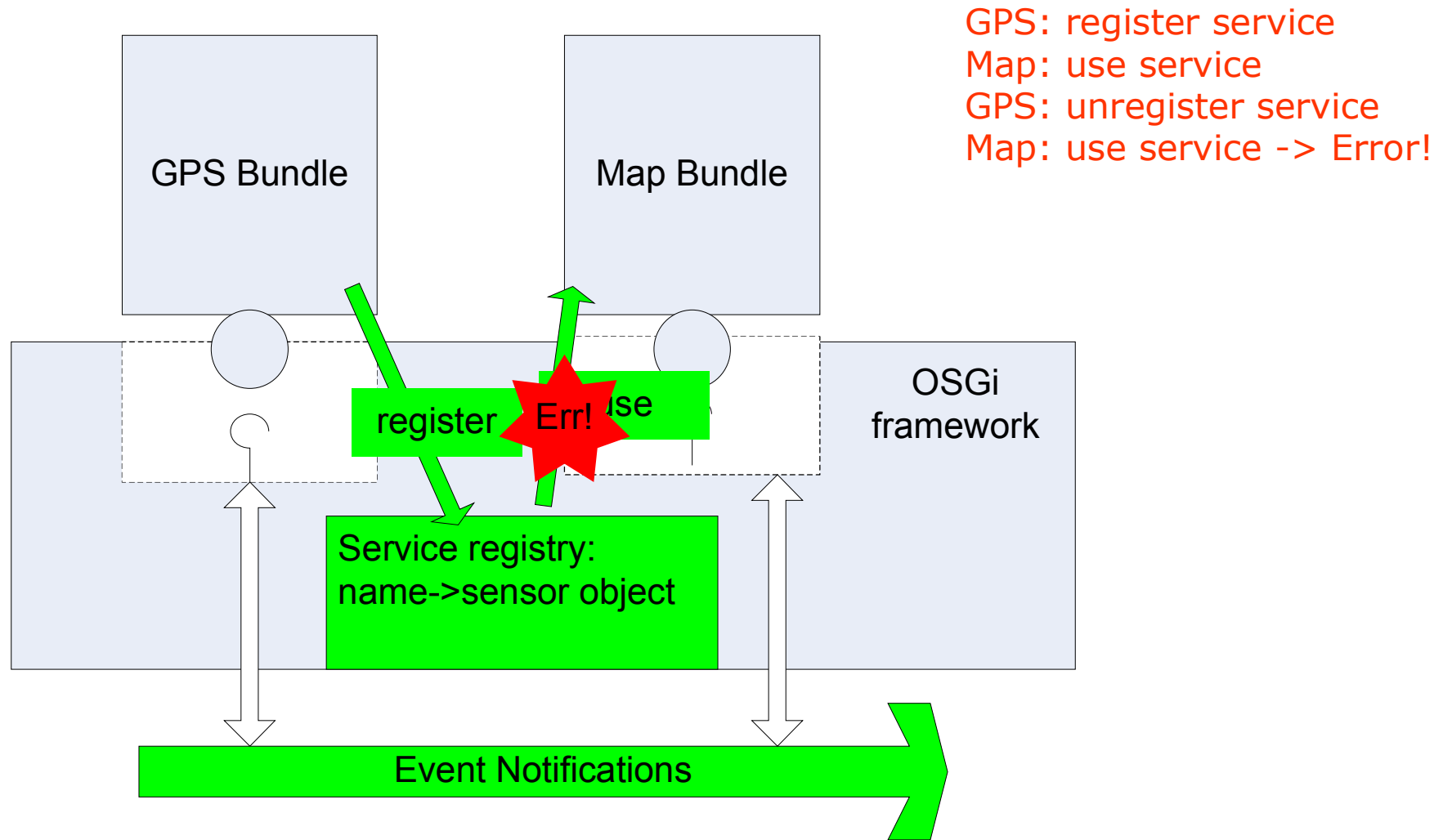
# Model checker: Java Pathfinder

- Directly check Java bytecode
- Restrictions:
  - No more than 10K lines of code
  - Native methods

# A simplified UML diagram of the OSGi framework



# The stale reference problem



# Use the modeling framework: add specification code

```
class GPSImpl implements GPSService {  
    public String getGPS() {  
        ...// do the actual work  
    }  
}
```

assert(valid);

```
public class BasicService {  
    public volatile boolean valid = false;  
    public ReentrantLock service_lock=  
        new ReentrantLock();  
}  
class GPSImpl extends BasicService...
```

```
public class BundleContextImpl {  
    public ServiceRegistration registerService(String[]  
        clazzes, Object service, Dictionary properties) {  
        ServiceRegistration sr=...// do the actual work  
        return sr;  
    }  
    public ServiceRegistration registerService(String  
        clazz, Object service, Dictionary properties) {  
        ServiceRegistration sr=...// do the actual work  
        return sr;  
    }  
}
```

```
if (sr!=null) {  
    if (service instanceof BasicService) {  
        BasicService ds = (BasicService)service;  
        ds.valid=true;  
    }  
}
```

```
public class ServiceRegistrationImpl {  
    Object service;  
    private void unregister_removeService() {  
        ...//remove the service  
    }  
}
```

```
if (service!=null && service instanceof BasicService) {  
    BasicService ds = (BasicService)service;  
    ds.valid=false;  
}
```

# Use the modeling framework: add specification code

```
class GPSImpl implements GPSService {
    public String getGPS() {
        ...// do the actual work
    }
}

public class BundleContextImpl {
    public ServiceRegistration registerService(String[]
        clazzes, Object service, Dictionary properties) {
        ServiceRegistration sr=...// do the actual work
        return sr;
    }
    public ServiceRegistration registerService(String
        clazz, Object service, Dictionary properties) {
        ServiceRegistration sr=...// do the actual work
        return sr;
    }
}

public class ServiceRegistrationImpl {
    Object service;
    private void unregister_removeService() {
        ...//remove the service
    }
}
```

Problem: We have entangled code spread in various places of multiple files!

# Use the modeling framework: feature variation

```
public class Framework {
    PermissionOps perm;
    public Framework(Object m) throws
        Exception {
        ...// Do other initialization
        ...// instantiate perm with proper
            permission policy
    }
}
public class Listeners implements {
    PermissionOps secure;
    Listeners(PermissionOps perm) {
        secure = perm;
    }
    void addBundleListener(Bundle bundle,
        BundleListener listener) {
        if (listener instanceof
            SynchronousBundleListener) {
            secure.checkListenerAdminPerm(bundle);
            ...// do the actual work to add listeners
        } else
            ... // do the actual work to add listeners
    }
}
```



# Potential problems (2): feature variation

```
public class Framework {
    PermissionOps perm;
    public Framework(Object m) throws
        Exception {
        ...// Do other initialization
        ...// instantiate perm with proper
            permission policy
    }
}
public class Listeners implements {
    PermissionOps secure;
    Listeners(PermissionOps perm) {
        secure = perm;
    }
    void addBundleListener(Bundle bundle,
        BundleListener listener) {
        if (listener instanceof
            SynchronousBundleListener) {
            secure.checkListenerAdminPerm(bundle);
            ...// do the actual work to add listeners
        } else
            ... // do the actual work to add listeners
    }
}
```



Problem: No modular way to turn on/off a feature

# Problem: Object-Oriented is not enough?

- Solution: add another modularization unit
- Hypothesis: use AOP techniques
- AspectJ: Aspect-Oriented Programming in Java
  - Pointcut: match method calls
  - Advice: change program logic
  - Intertype data: add additional data
  - “Compatible” with Java!

# Apply AOP techniques at model level (1)

```
class GPSImpl implements GPSService {
    public String getGPS() {
        ...// do the actual work
    }
}

public class BundleContextImpl {
    public ServiceRegistration registerService(String[]
        clazzes, Object service, Dictionary properties) {
        ServiceRegistration sr=...// do the actual work
        return sr;
    }
    public ServiceRegistration registerService(String
        clazz, Object service, Dictionary properties) {
        ServiceRegistration sr=...// do the actual work
        return sr;
    }
}

public class ServiceRegistrationImpl {
    Object service;
    private void unregister_removeService() {
        ...//remove the service
    }
}
```

```
declare parents : ActivatorAJ.GPSImpl extends
BasicService;

pointcut ServiceFunction(Object service) :
    execution(* getGPS(..) && this(service)
    && this(GPSService));
before(Object service): ServiceFunction(service) {
    if (service instanceof BasicService) {
        BasicService bs = (BasicService) service;
        assert (bs != null && bs.valid);
    }
}

pointcut ServiceUnregister(ServiceRegistrationImpl sr):
    execution(* unregister_service(..) &&
    this(ServiceRegistrationImpl) && this(sr);
after(ServiceRegistrationImpl sr) :
    ServiceUnregister(sr) {
    if (sr.service != null && sr.service instanceof
    BasicService)
        ((BasicService) sr.service).valid = false;
}
```

# AspectJ brings good and bad...

```
public class Framework {
    public static void main(String[] args) {
        Framework fw = new Framework();
    }
}
aspect FrameworkConstructor {
    pointcut FrameworkConstructor(Framework f) :
        execution(Framework.new(..))
        && !cflow(adviceexecution()) && this(f);
    after(Framework f): FrameworkConstructor(f) {
        Signature sig = thisJoinPoint.getSignature();
    }
}
```

```
error #1: gov.nasa.jpfi.jvm.NoUncaughtExceptionsProperty
java.lang.NullPointerException: calling
'getProperty(Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;' on null object
at java.lang.System.getProperty(System.java:663)
at org.aspectj.runtime.internal.CFlowCounter.getSystemProperty-
WithoutSecurityException(CFlowCounter.java:78)
at org.aspectj.runtime.internal.CFlowCounter.selectFactoryForVMVersion(CFlowCounter.java:55)
at org.aspectj.runtime.internal.CFlowCounter.<clinit>(CFlowCounter.java:29)
at edu.uoregon.osgi.mc.FrameworkConstructor.ajc$preClinit(Framework.aj:1)
at edu.uoregon.osgi.mc.FrameworkConstructor.<clinit>(Framework.aj:native)
at edu.uoregon.osgi.mc.Framework.<init>(Framework.aj:3)
at edu.uoregon.osgi.mc.Framework.main(Framework.aj:5)
```

# Handle native code: build an abstraction library

## ➤ AspectJ runtime library: ~180K

```
1. public static int getProperty__Ljava_lang_String_2Ljava
   _lang_String_2__Ljava_lang_String_2 (MJNIEnv env, int clsObjRef, int keyRef, int
   defRef) {
2.   int r = MJNIEnv.NULL;
3.   if (keyRef != MJNIEnv.NULL) {
4.     String k = env.getStringObject(keyRef);
5.     String defaultString = env.getStringObject(defRef);
6.     if (k==null)
7.       return MJNIEnv.NULL;
8.     String v = System.getProperty(k);
9.     if (v != null)
10.      r = env.newString(v);
11.     else if (defaultString!=null) {
12.      r = env.newString(defaultString);
13.    }
14.    return r;
15.}
```

# Summary

- Research question: Can we build a modeling framework to verify applications in a specific domain?
  - Crosscutting concerns: OO is not enough
  - Introduce AOP to model construction
  - Build an abstraction library
- Expected contributions
  - Ease model construction with a modeling framework
  - Enable model checking a new class of applications
  - Build an OSGi modeling framework that can be directly reused

# The End

This is a short version of the talk – contact [zbchen@cs.uoregon.edu](mailto:zbchen@cs.uoregon.edu) if you are interested in the full details.