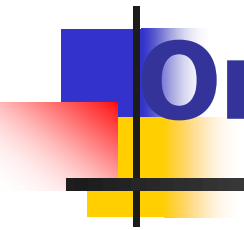
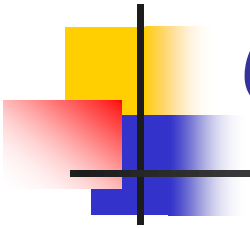


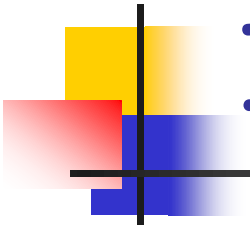
Pointcut Matching based on Ontology for Aspect of Assembly

Presentation of *Projet de Fin d'Etude*
Student: VU Trong Tuan



Outline

- Introduction.
- Weaving process of Aspect of Assembly
- Syntactic Pointcut Matching
- Proposed solution: a semantic extension
- Experiments
- Conclusion

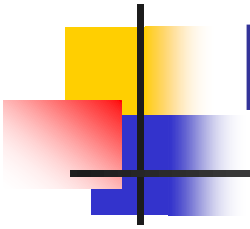


Introduction

- In ubiquitous computing, the disappearance/appearance of devices lead the changes of the context (physical or software infrastructure).
- Dynamic adaptation of the application based on the context plays an important role.

Software Adaptation

- There are three main approaches [1]:
  - Laissez faire: adaptation is entirely supported by the applications.
    - Put too much burden on the application.
    - Make the development of application become more difficult,
  - Application transparent: adaptation is entirely supported by the system
    - Adaptation may be not adequate or counter-productive
  - Middleware approach: collaboration between application and the system to achieve the best adaptation.
- Most of the works converges to manage the adaptation by Middleware layer

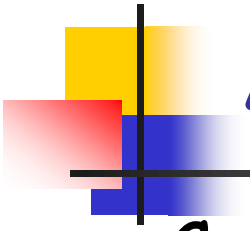


Middleware approach

- Middleware has two roles:
  - Simplify the development of distributed applications.
  - Support dynamic adaptation.
- Three approaches to perform adaptation in literature [2]
  - Reflection
  - Policy-based adaptation.
  - Aspect-oriented programming.

Adaptation in Middleware

- Reflection: the capability of a system to reason and possibly alter its own behavior.
- Policy-based adaptation:
  - Application states policy rules for adaptation.
  - For each particular condition, the matching rule is applied to change the middleware behavior.
  - Example: ECA (Event-Condition-Action)
    - Event specifies the context changes.
    - Condition test if the context change is satisfied
    - If yes, Adaptation (action) will be carried out.



Aspect oriented approach [5]

- Complex programs are composed of different intervened crosscutting concerns.
  - Cross cutting concerns are properties or area of interest: security, persistence. logging, …
- Crosscutting concerns are implemented at separate modules called aspect, and can be adapted at Runtime
- There are 3 basic elements:
  - JointPoint: an event in executing program where the advice may be executed.
  - Pointcut: set of jointpoint where the associated advice should be executed.
  - Advice: concern implementation.

Aspect oriented approach

```
public class HelloWorld {
   public static void main (String []args){
      new HelloWorld().sayHello();
   }
   public void sayHello(){
      System.out.println("Hello World");
   }
}
```

Weaver

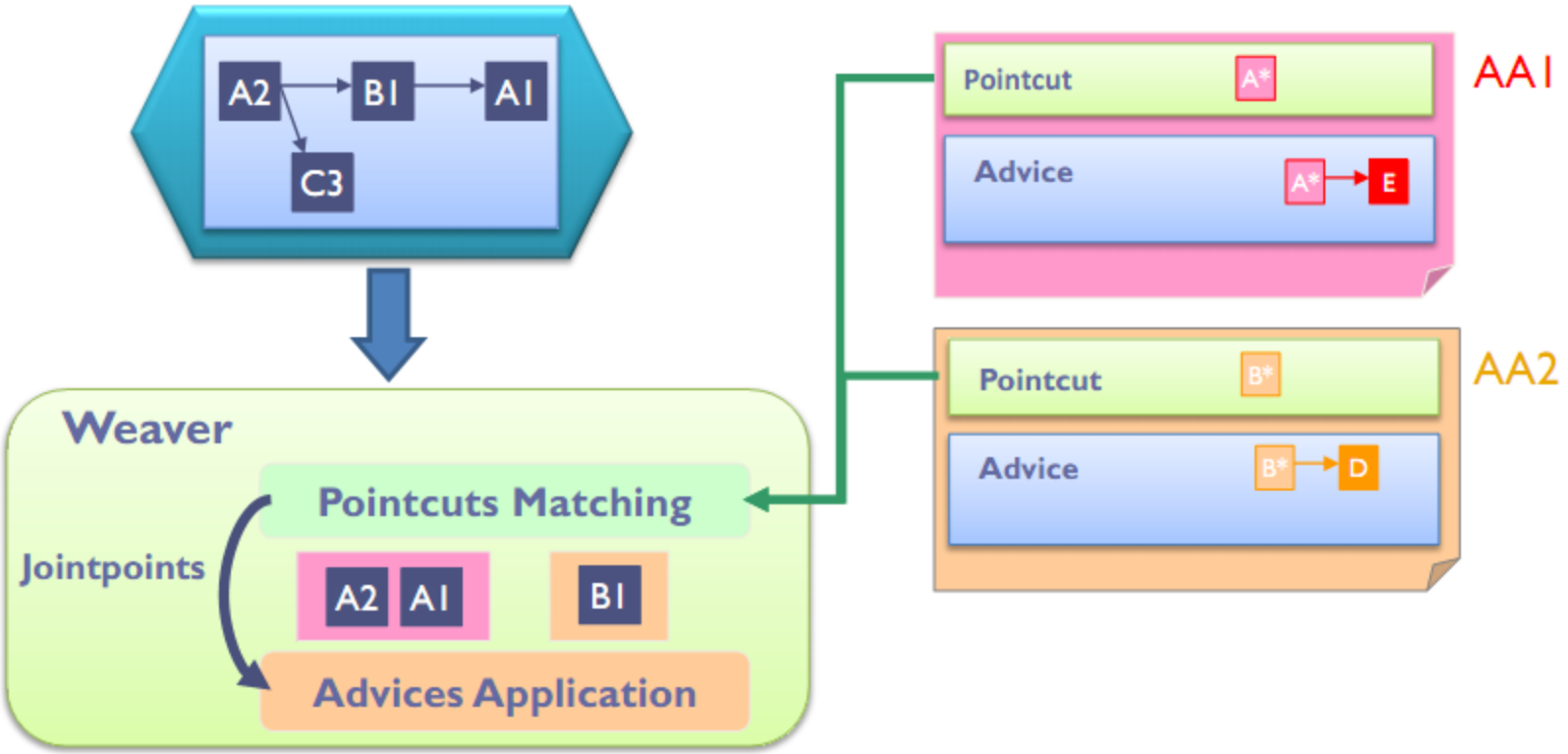
Pointcut Matching
Advice Application

```
pointcut One():
   execution("HelloWorld.sayHello")
before():One():{
   System.out.println("Hello One");
}
```

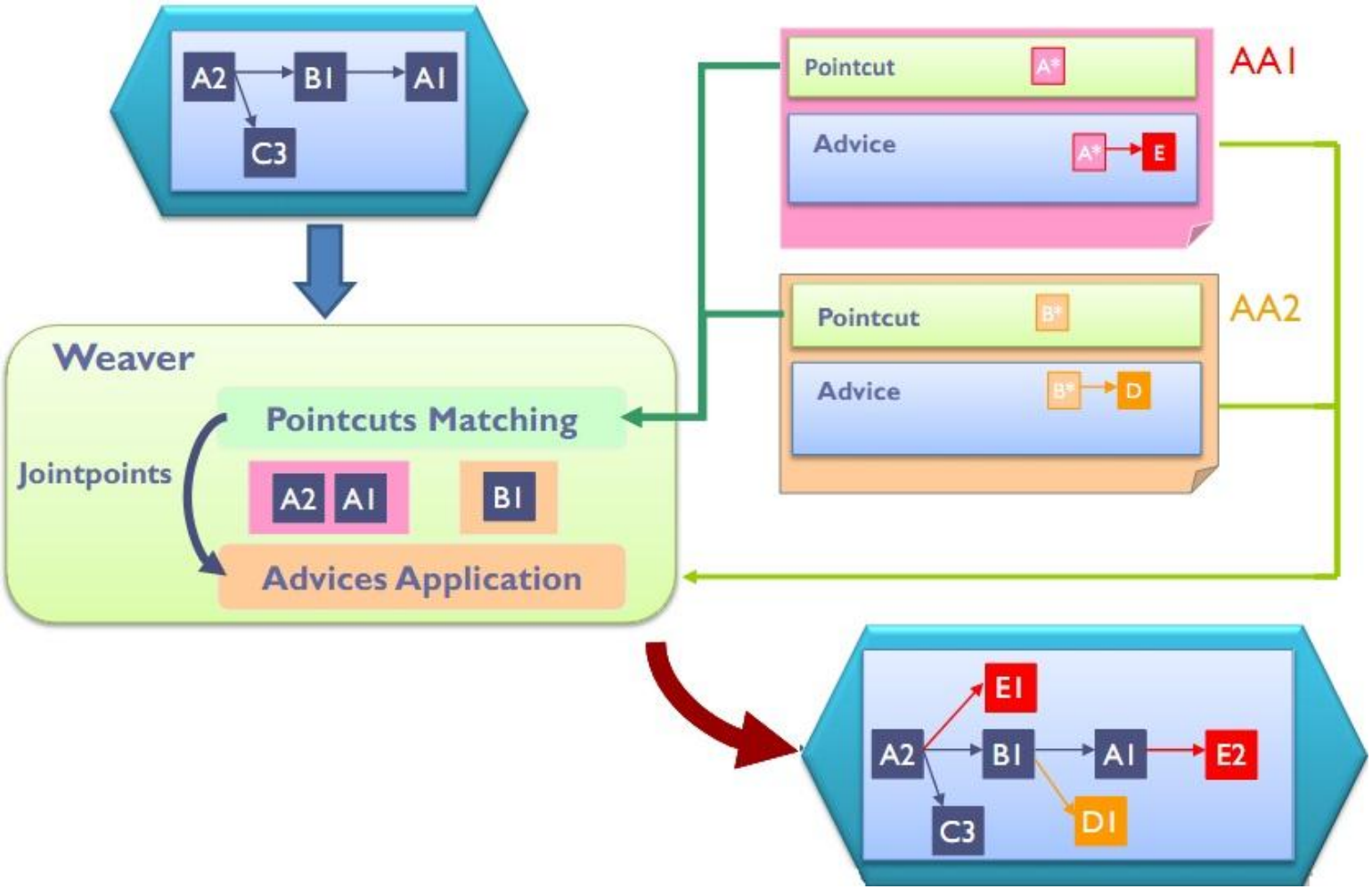
```
pointcut Two():
   execution("HelloWorld.sayHello")
after(): Two():{
   System.out.println("Hello Two");
}
```

```
public class HelloWorld {
   public static void main (String []args){
      System.out.println("Hello One");
      new HelloWorld().sayHello();
      System.out.println("Hello Two");
   }
   public void sayHello(){
      System.out.println("Hello World");
   }
}
```

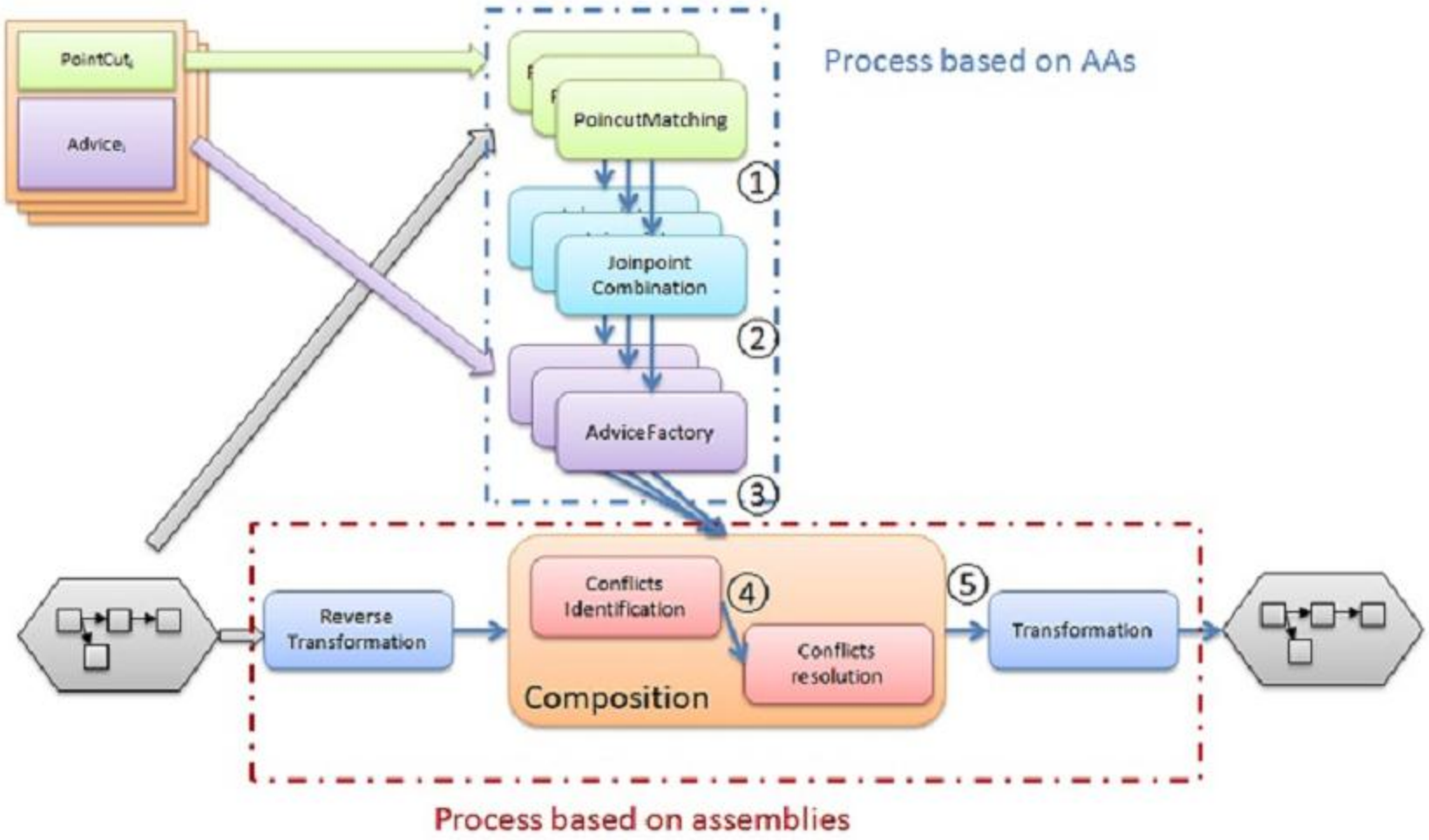
Principle of Aspect of Assembly [3]



Principle of Aspect of Assembly

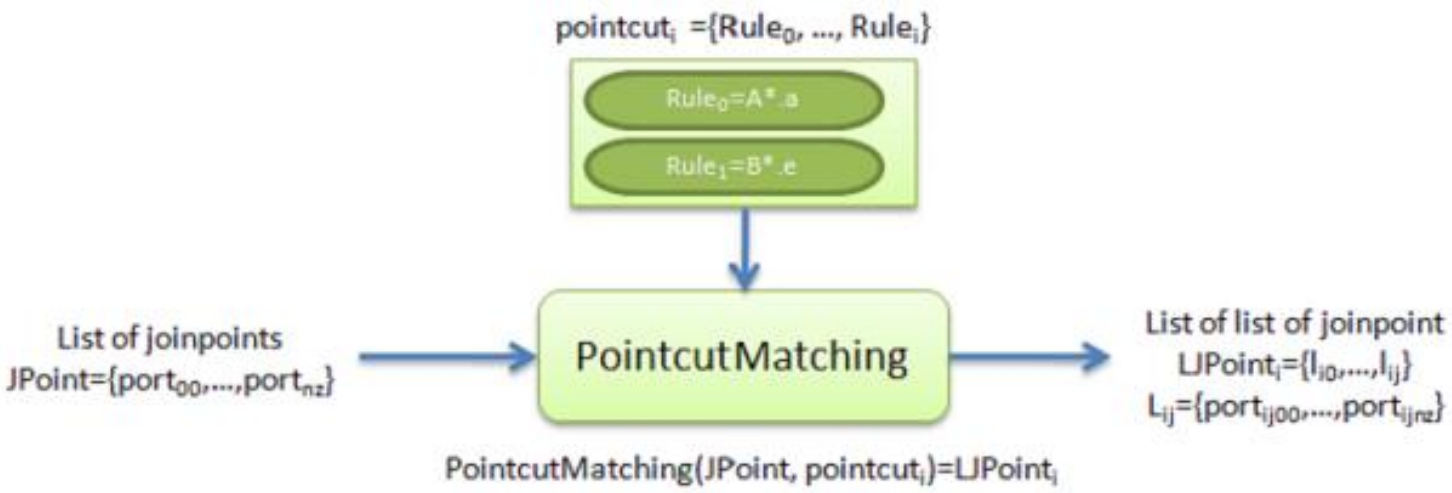


Weaving process of Aspect of Assembly

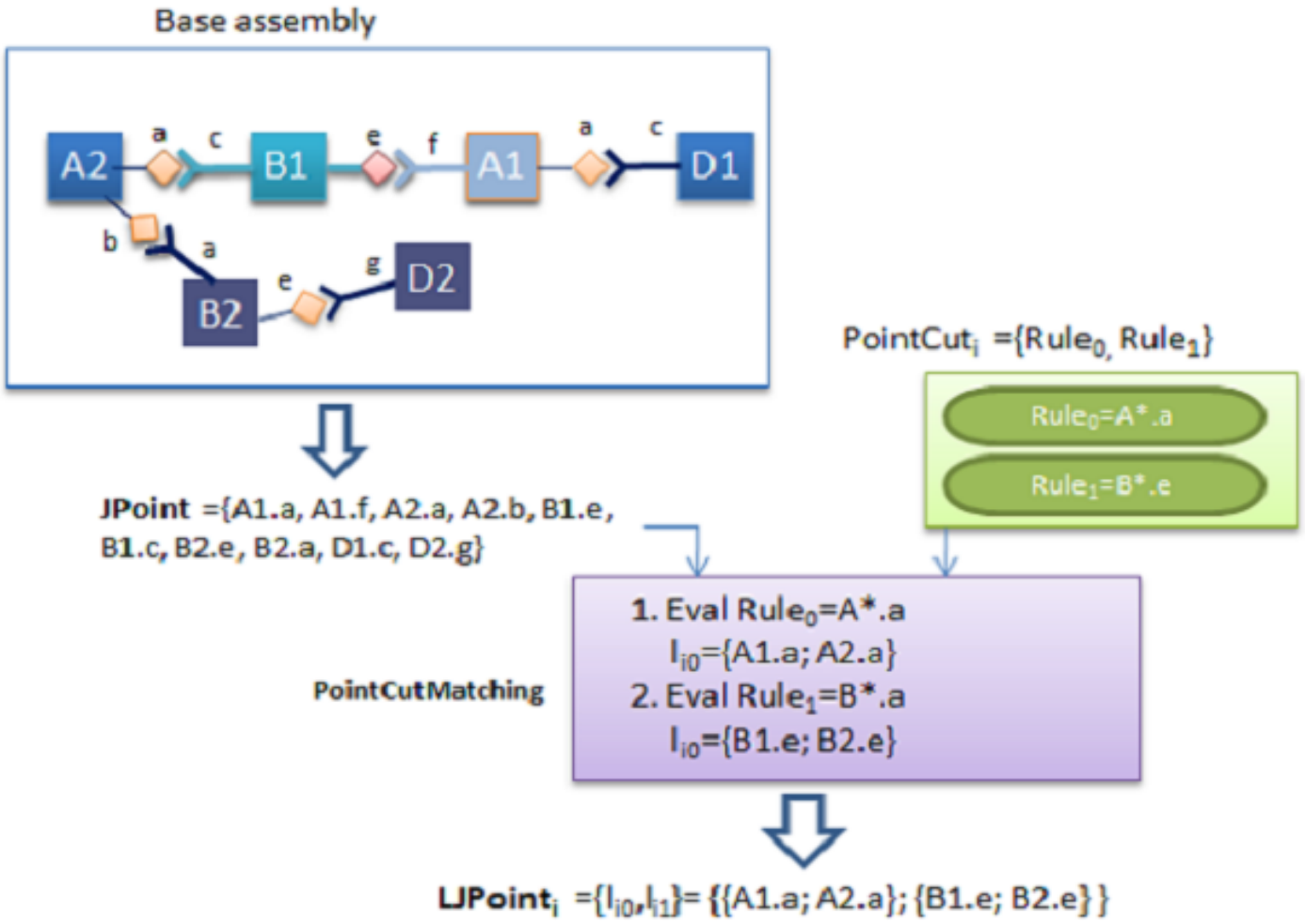


Syntactic Pointcut Matching

- Pointcut matching is used to determined in the based assembly all places where changes described in an AA can be applied.



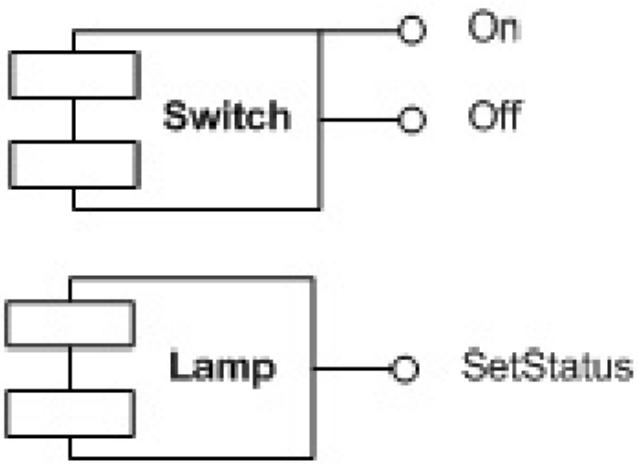
Pointcut Matching Example



Syntactic Pointcut Example

- Example: when a lamp comes to my room, I will connect it to switch1

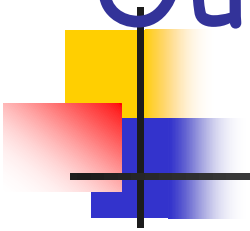
```
//Description of needed components
l := /lamp1/
sw := /switch1/

//AA description
//pointcut->advice
sw.on -> l.setStatus(true)
sw.off -> I. setStatus(false)
```



Syntactic Pointcut Fragility

- Current Pointcut is expressed by syntactic regular expression, i.e sw.on, sw.off.
- Syntactic pointcuts are fragile
  - Their semantic may change "silently" when changes are made to base program
  - Need to revise the pointcut when there is any change in base program.
- Solution: adding some meta-data to the component and use them to construct the pointcut.
- Example:
  - Pointcut: list all switches and lamps which are in the same location.
  - Advice: Connect them together.

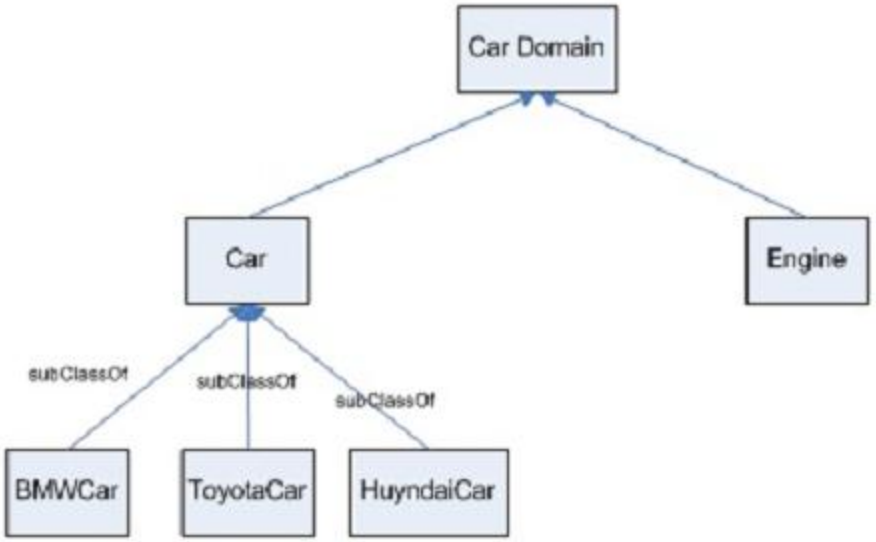
Outline



- Introduction.
- Weaving process of Aspect of Assembly
- Syntactic Poincut Matching and Semantic Interoperability
- Proposed solution: a semantic extension
  - Ontology
  - Resource Description Framework
  - Semantic Pointcut based on Ontology.
- Experiments
- Conclusion

Ontology

- Ontology is a formal representation of knowledge as a set of concepts within a domain, and the relationships between those concepts.
- There are three main part for a ontology:
  - Class
  - Individual
  - Property
- Example: Car domain



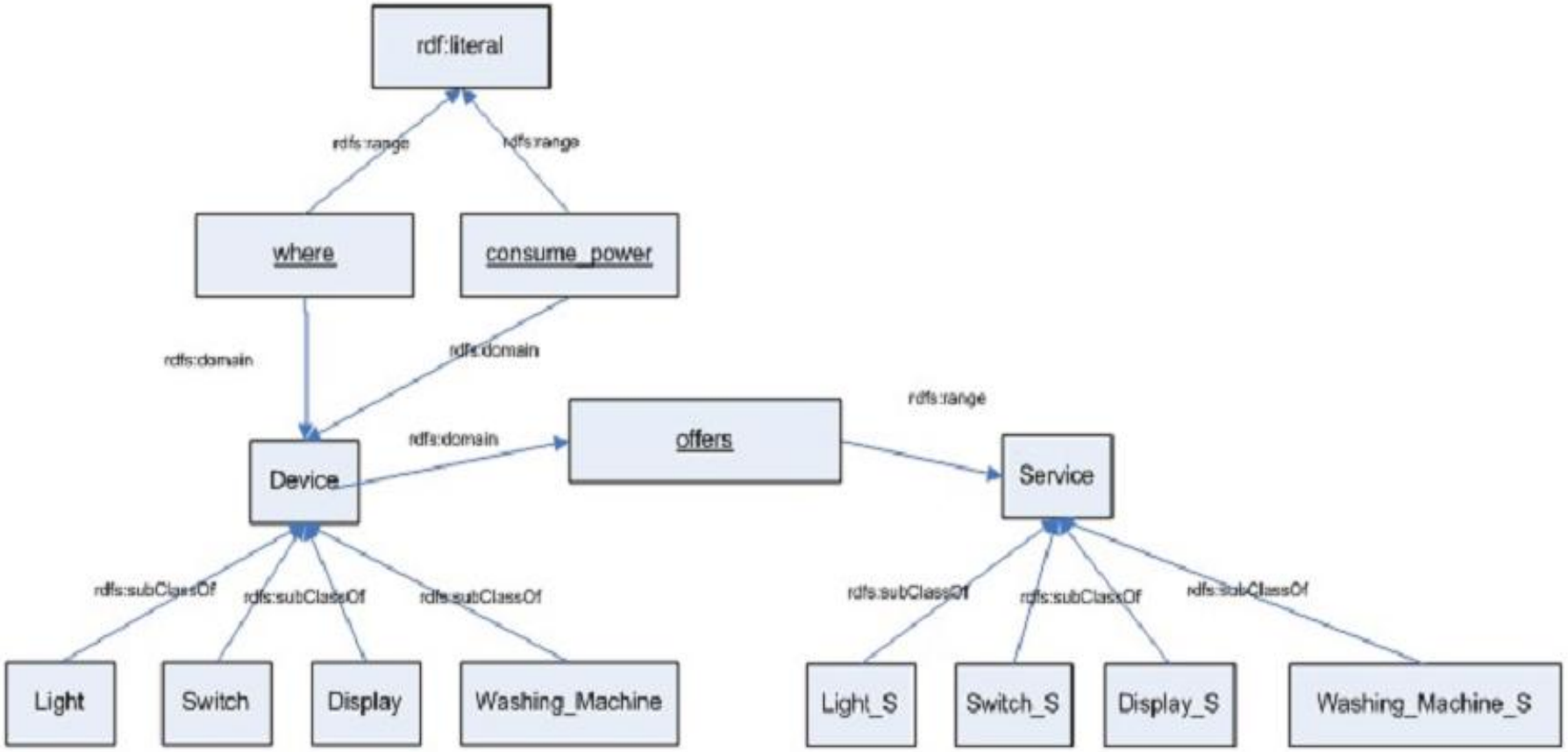
Resource Description Framework (RDF)

- The Resource Description Framework is an XML-based language to represent information on the Web.
- It is based upon the idea of making statements about resources (in particular Web resources) in the form of *subject-predicate-object* expressions called *triples*.
- *Example:*

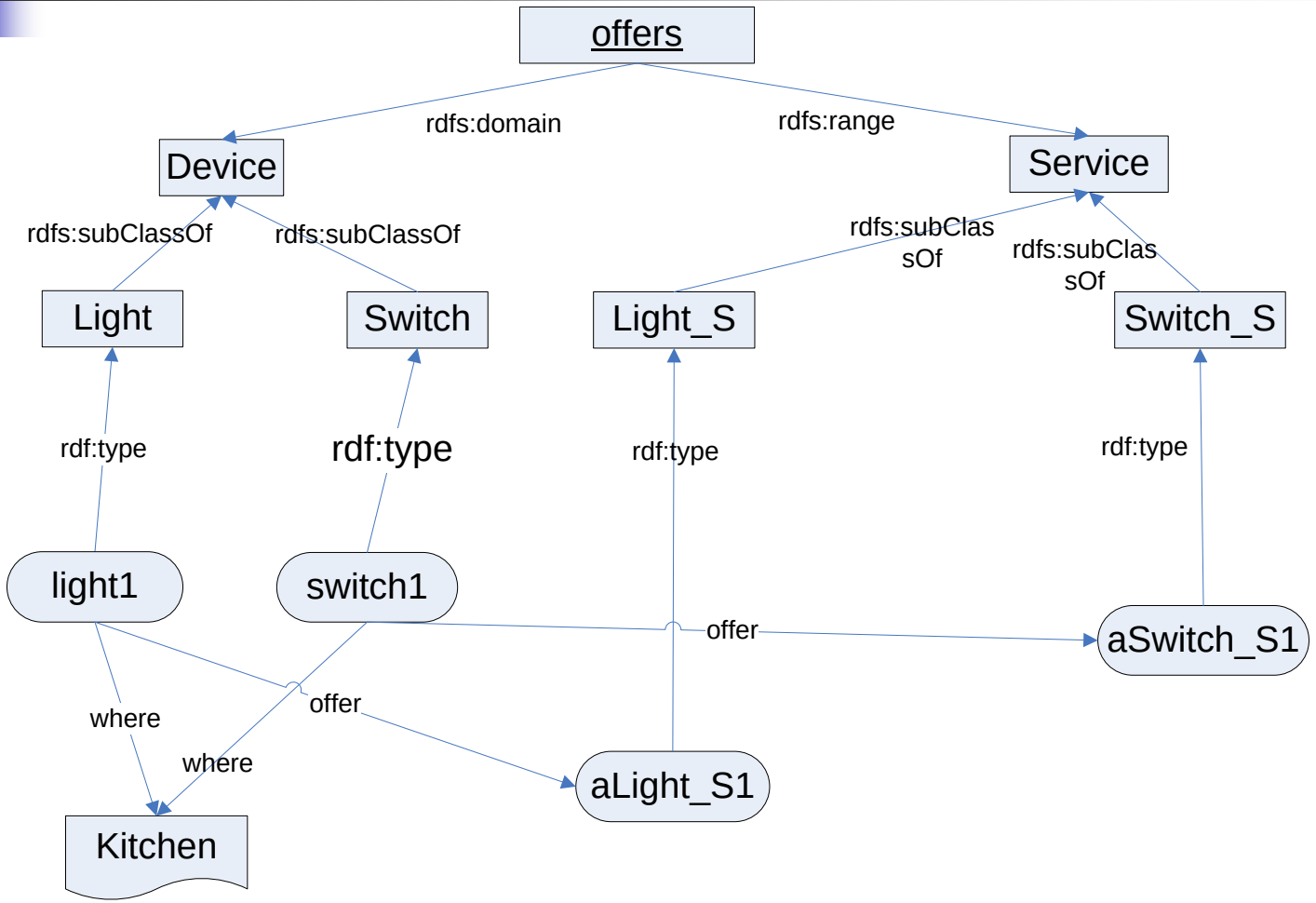
Semantic Pointcut based on Ontology

- The initial model of context with two basic classes *Device* and *Services.*
  - *Device: instance of this class are the physical processing devices that are currently available in a given context.*
    - *Offers: which service does the device offers*
    - *Where: the current location of the device.*
    - *Consume_power: the current power consumption of device.*
  - *Service: this class represent all types of services offered by the devices.*

RDFS based Context Model



Context Information

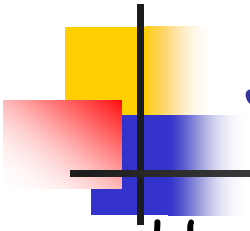


Context Information

- Query example on this context:
  - Device(D) && where (D, Kitchen) && (exist S such that Light_S (S) && offers (D, S)
  - SPARQL:

```
SELECT ?d WHERE {
  ?d type Device .
  ?d where Kitchen .
  ?d offers ?s .
  ?s type Light_S .
};
```

- The query mentions about the abstraction information (Classes: Device, Light_S), but the result returns their instances
  - The same query can be applicable and yield the correct results if there is a new device (of type VerySmartLight) appears in the environment.



Service Annotation

- How to add the meta-data to the component?
  - Used Annotation to add meta-data in the source code.
- Annotation for Device

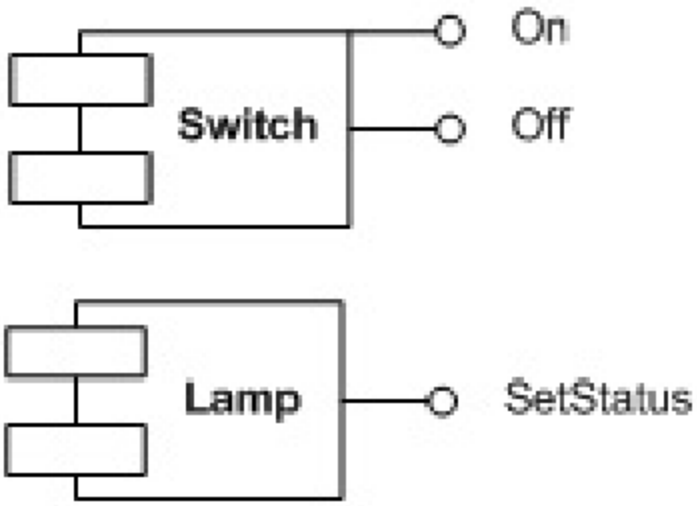
```
public @interface Device {
    public String name() default "{unassigned}";
    public String deviceType() default "{unassigned}";
    public String consume_power() default "{unassigned}";
    public String location() default "{unassigned}";
}
```

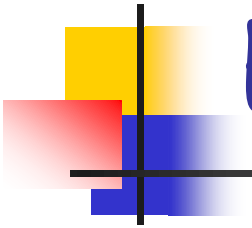
- Annotation for the Service

```
public @interface Service {
    public String name() default "{unassigned}";
    public String serviceType() default "{unassigned}";
}
```

Example Revisited

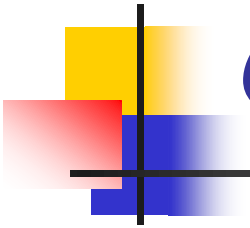
```
//Pointcut
 A = @Device (deviceType=''Switch'', location=''Kitchen'') &&  @Service
(serviceType=''Switch_S'')

 B = @Device (deviceType=''Light'', location=''Kitchen'') && @Service
(serviceType=''Light_S'')

 //Advice
 A->B
```





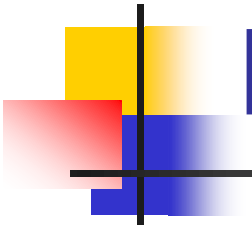
Experiments

- Conquer is tool for Ontology development, developed by the collaboration with HADAS-LIG
- The context data presented is implemented using Conquer [4] with the following functionalities:
  - Edit the context data, i.e adding or deleting the class, an instance of a class or a property of an instance.
  - Query the context data and give the corresponding answer.



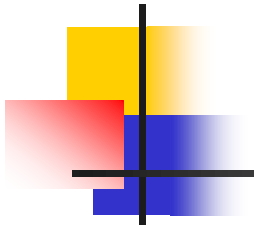
Conclusion

- Study about the mechanisms, approaches and new challenges to solve the problems of adapting application to their context in the ubiquitous environment
- Study detail about the Aspect of Assembly Mechanism and its current syntactic pointcut problem
- Propose one mechanism to express the pointcut in high abstraction level to solve the syntactic fragile pointcut problem.



References

- [1] M. Satyanarayanan. Fundamental challenges in mobile computing. In Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing (PODC'96), pages 1-7, New York, NY, USA, 1996. ACM.
- [2] Paul Grace. Dynamic adaptation. In H. Miranda B. Garbinato and L. Rodrigues, editors, Middleware for Network Eccentric and Mobile Applications, pages 285{304. Springer, 2009.
- [3] Jean-Yves Tigli, Stéphane Lavirotte, Gaetan Rey, Nicolas Ferry, Sana Fathallah Ben Abdenneji, and Michel Riveill. Aspect of assembly: from theory to performance.
- [4] Anis Benyelloul, Fabrice Jouanot, and Marie-Christine Rousset. Conquer tool. http://conquer.liglab.fr/home.php.
- [5] Gregor Kiczales and Erik Hilsdale. Aspect-oriented programming. SIGSOFT Softw. Eng. Notes, 26-313 September 2001



QUESTIONS?