Towards Model Driven Design of Service Based Context Aware Applications

Vincenzo Grassi Andrea Sindico

Key Concepts

- Pervasive applications should be context aware
- Context-aware adaption as a crosscutting concern
- Aspect Oriented Software Development

Agenda

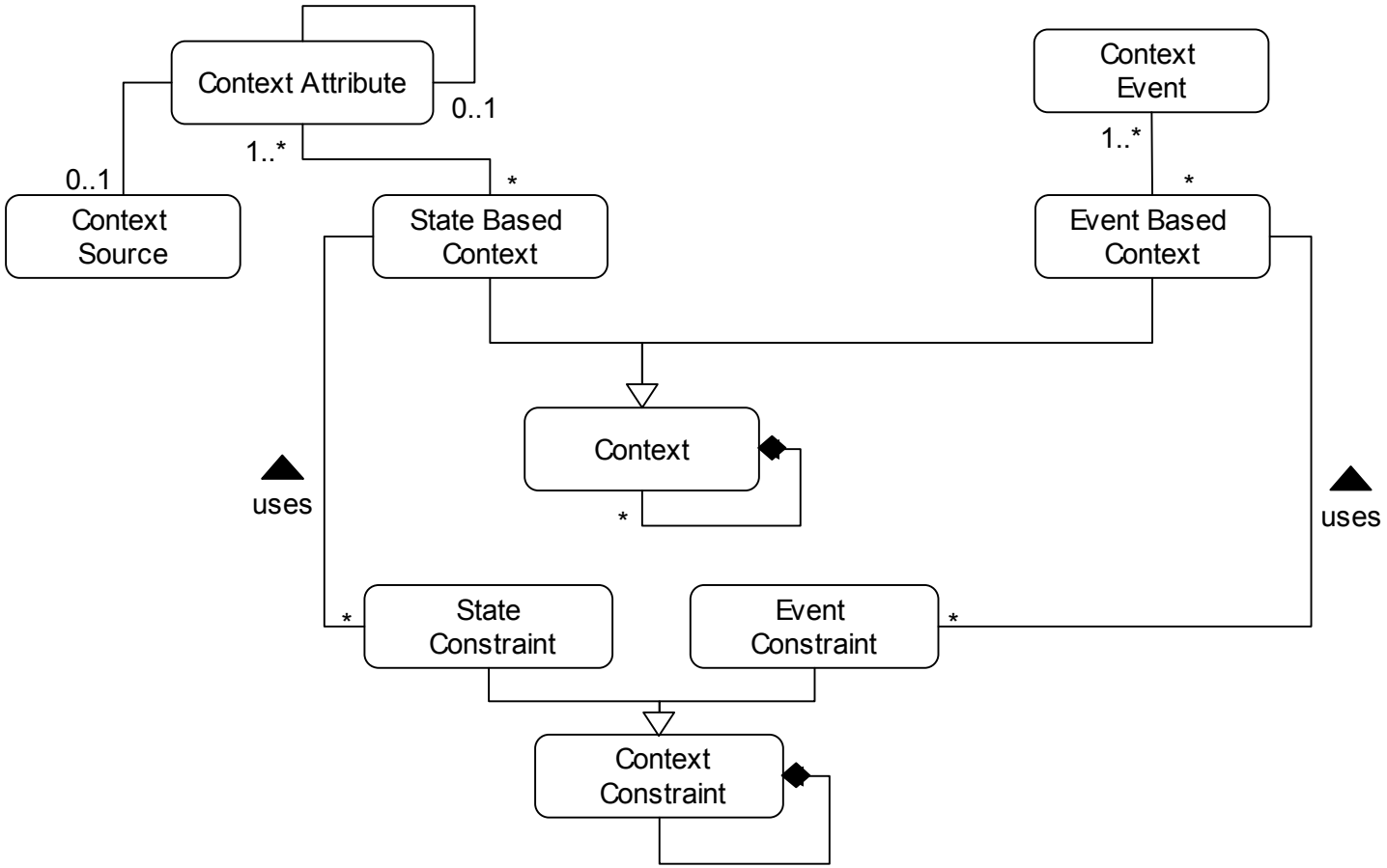
1. Conceptual Model

2. UML Model

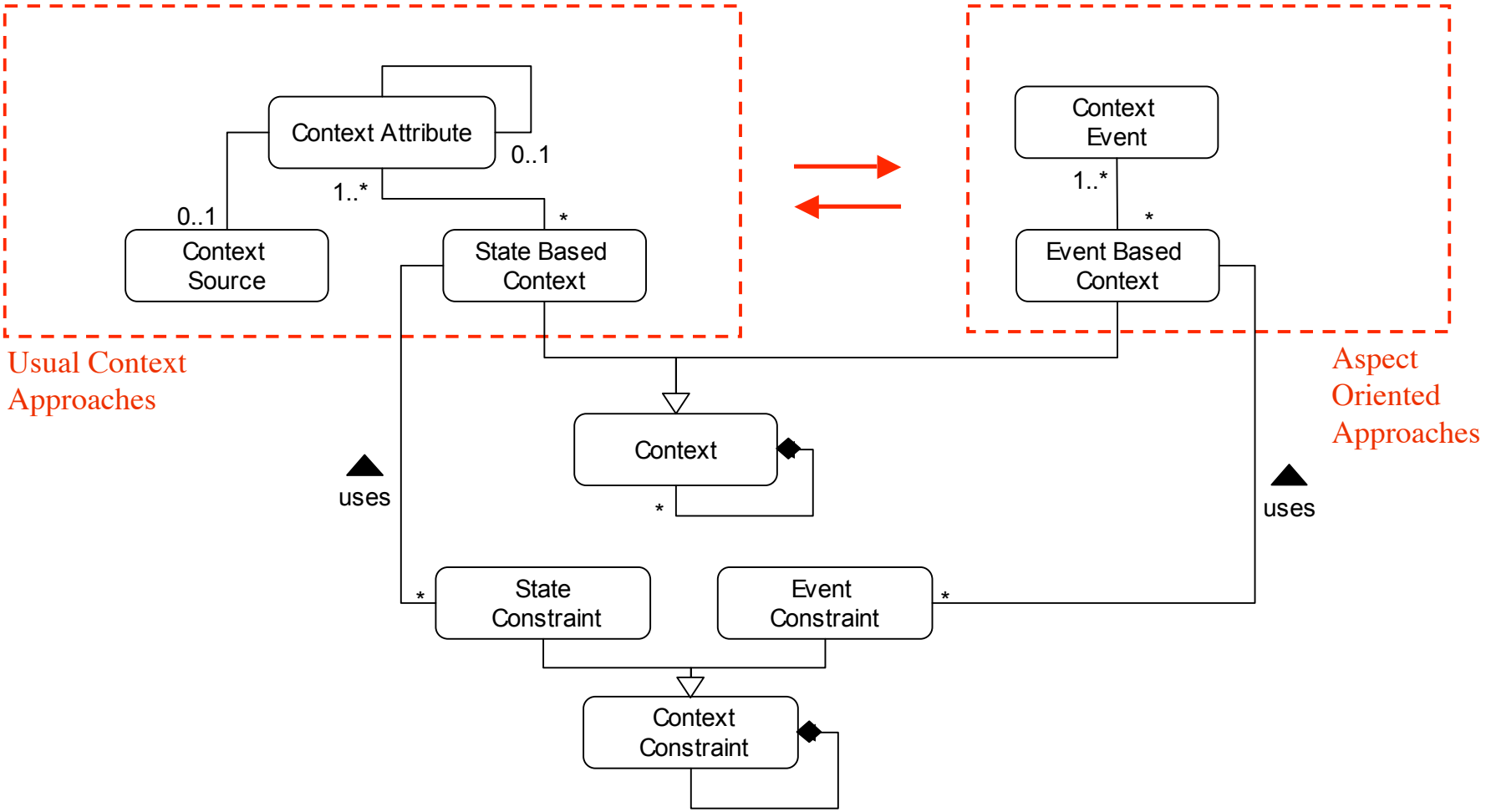
3. Code

Conceptual Model

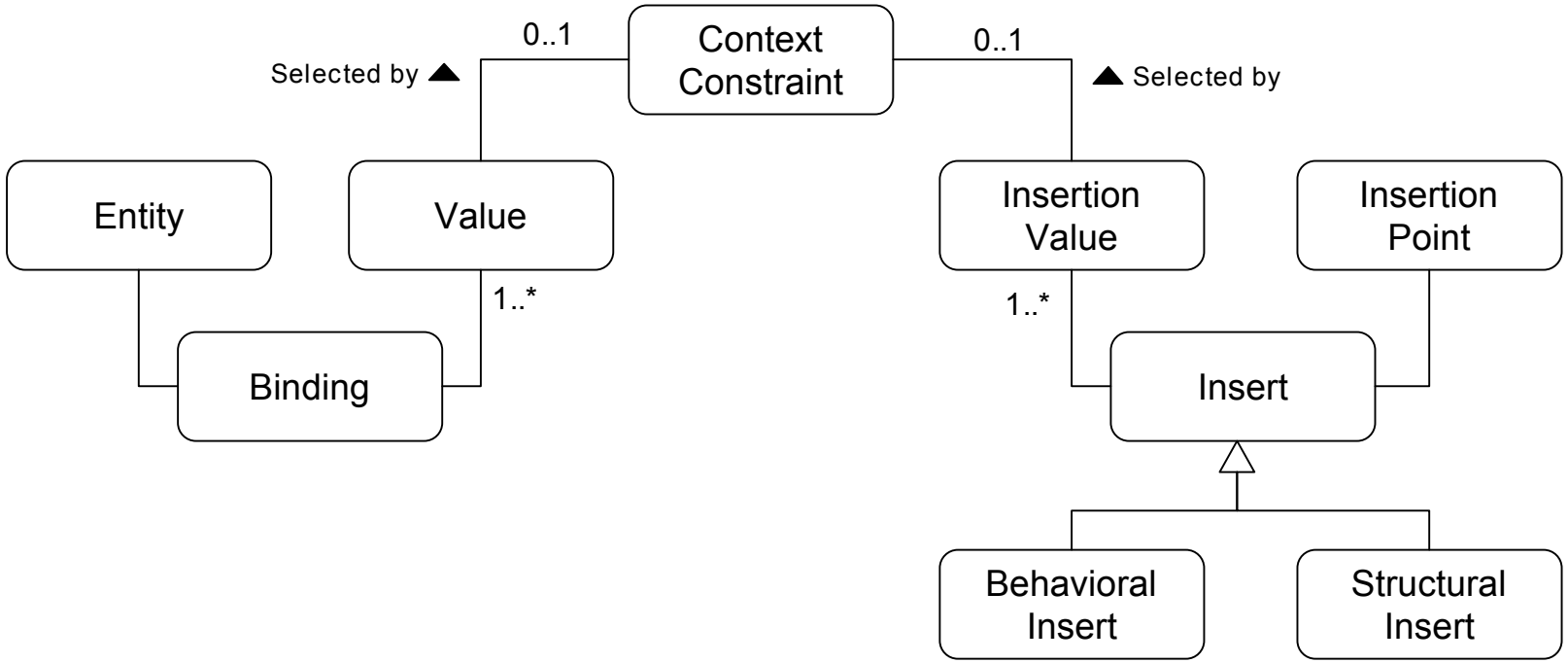
Conceptual model of context



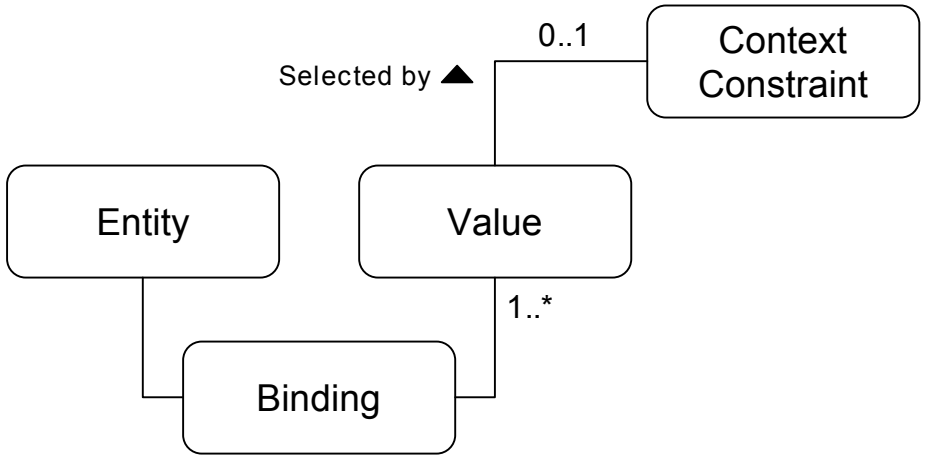
Conceptual model of context



Conceptual model of context aware adaption



Conceptual model of context aware adaption



Binding

can be used to achieve different types of adaption

For example:

- to bind a service interface to different implementations
- to bind a service ivocation to different services
- to bind a parameter in a service invocation to different values

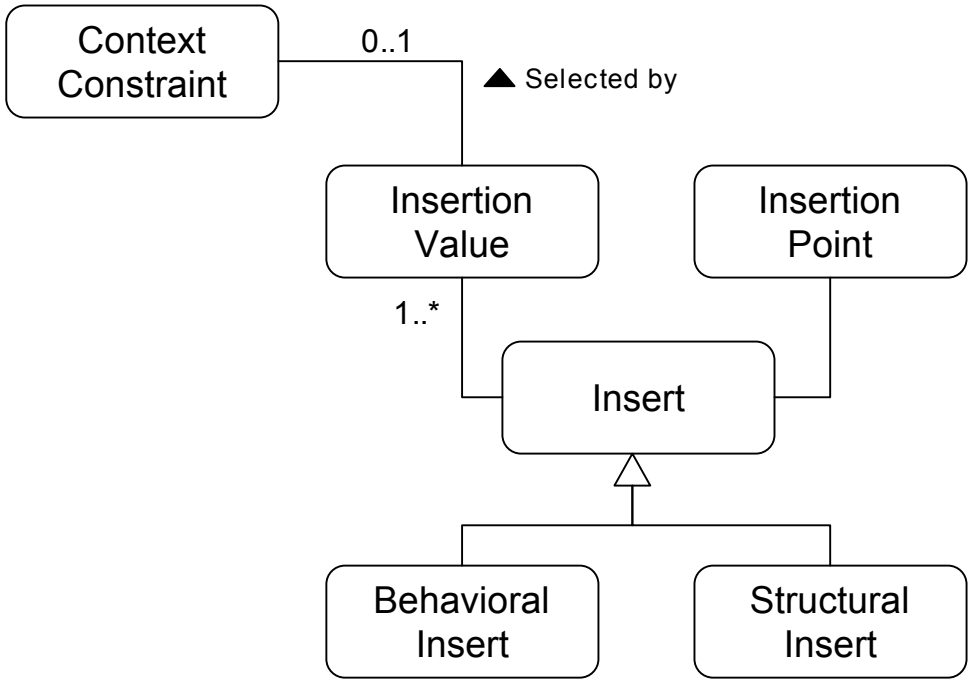
Conceptual model of context aware adaption

Insert

The concept of context-aware insert is derived from AOSD

It can be structural or behavioral

It consists of a specification of the value that must be inserted and of the point where it must be inserted



UML Model

New UML Elements

Context Monitor

- is a Container for State based Context, Event based Context, State Constraints and Event Constraint
- captures events in the execution of a system and produce signals on context changing

Context Adaptor

- is a Container for adaption mechanisms (binding, insert)
- reacts to signals generated by Context Monitors

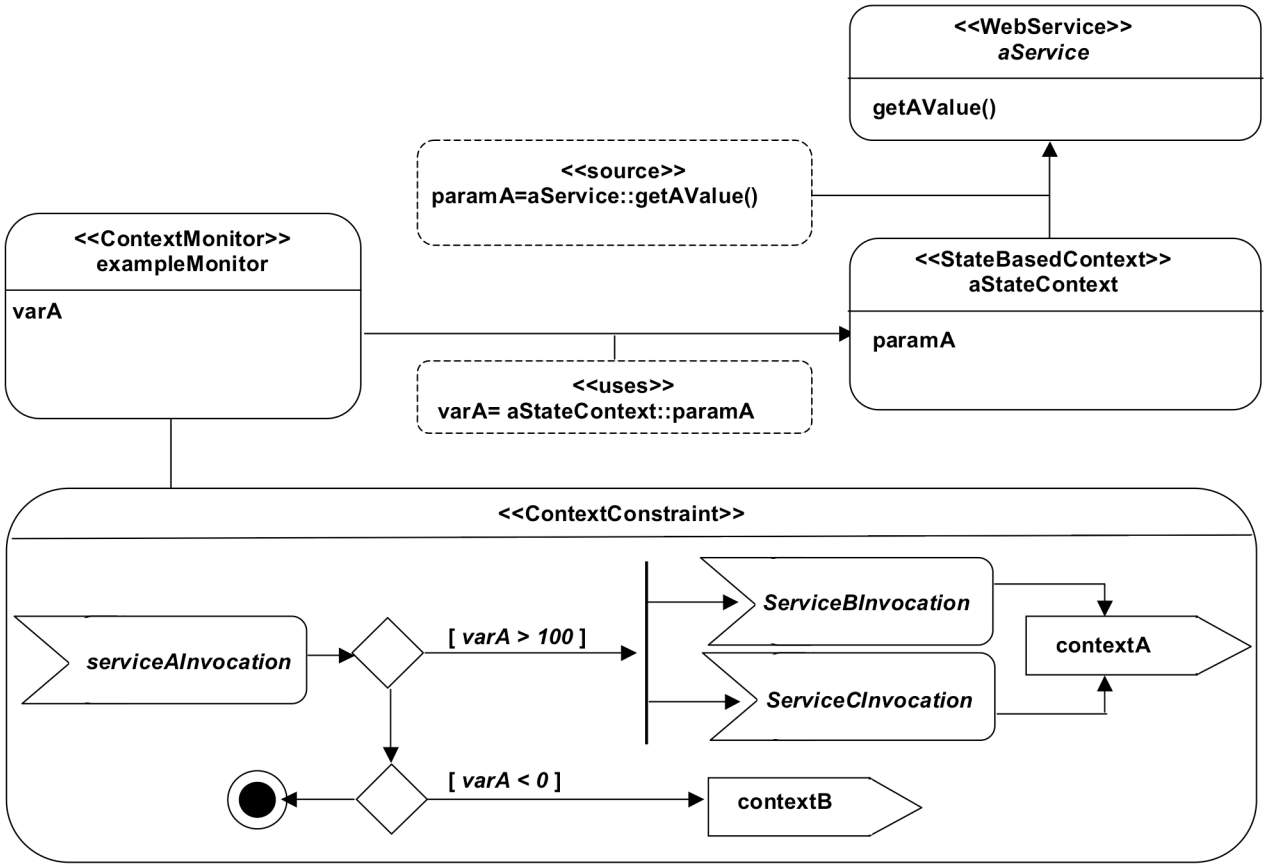
New UML Elements

A Context Monitor may provide context informations to several Context Adaptors

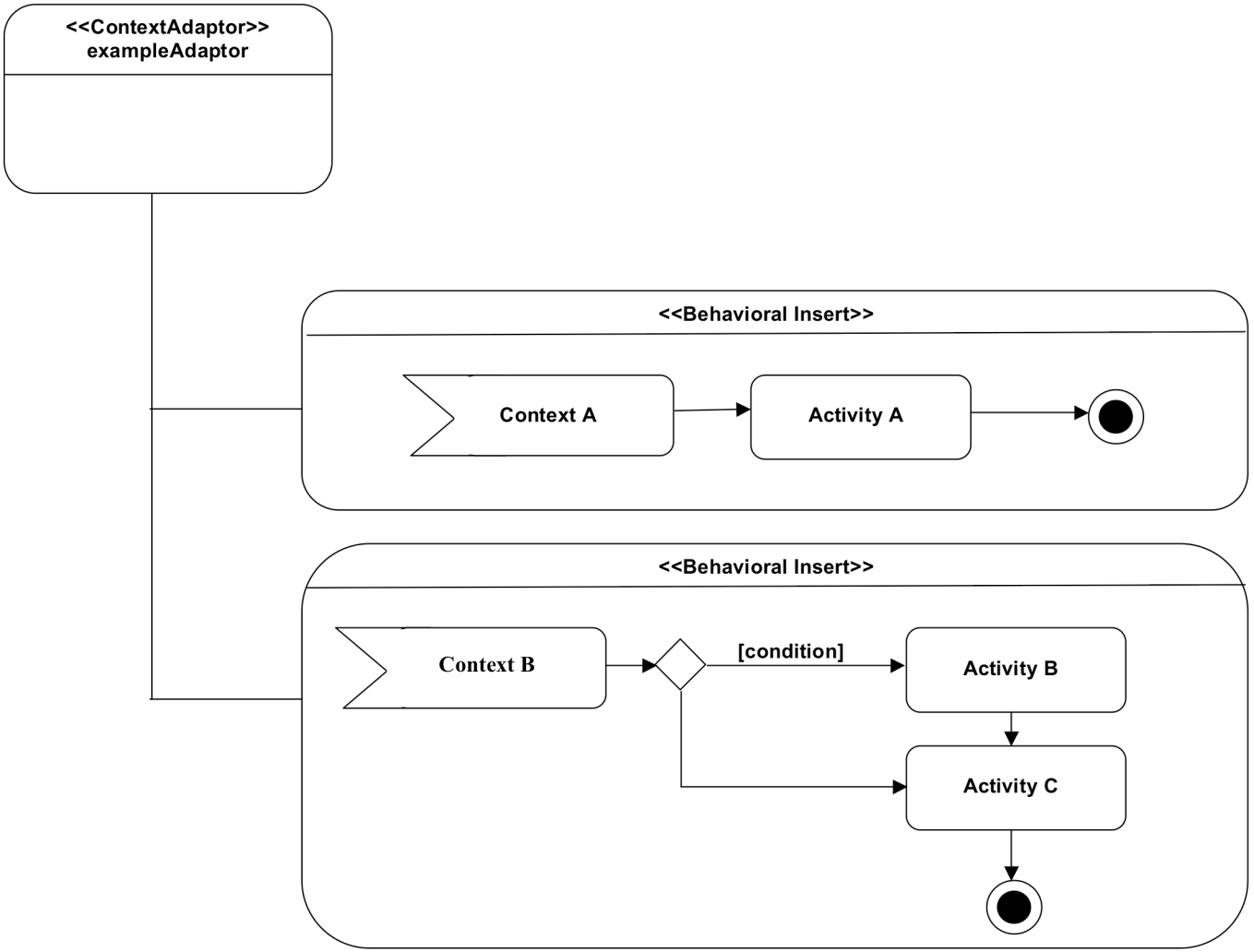
A Context Adaptor may be driven by several Context Monitors

These new elements realize a separation of concerns among “Context Monitoring” and “Context Adaption”

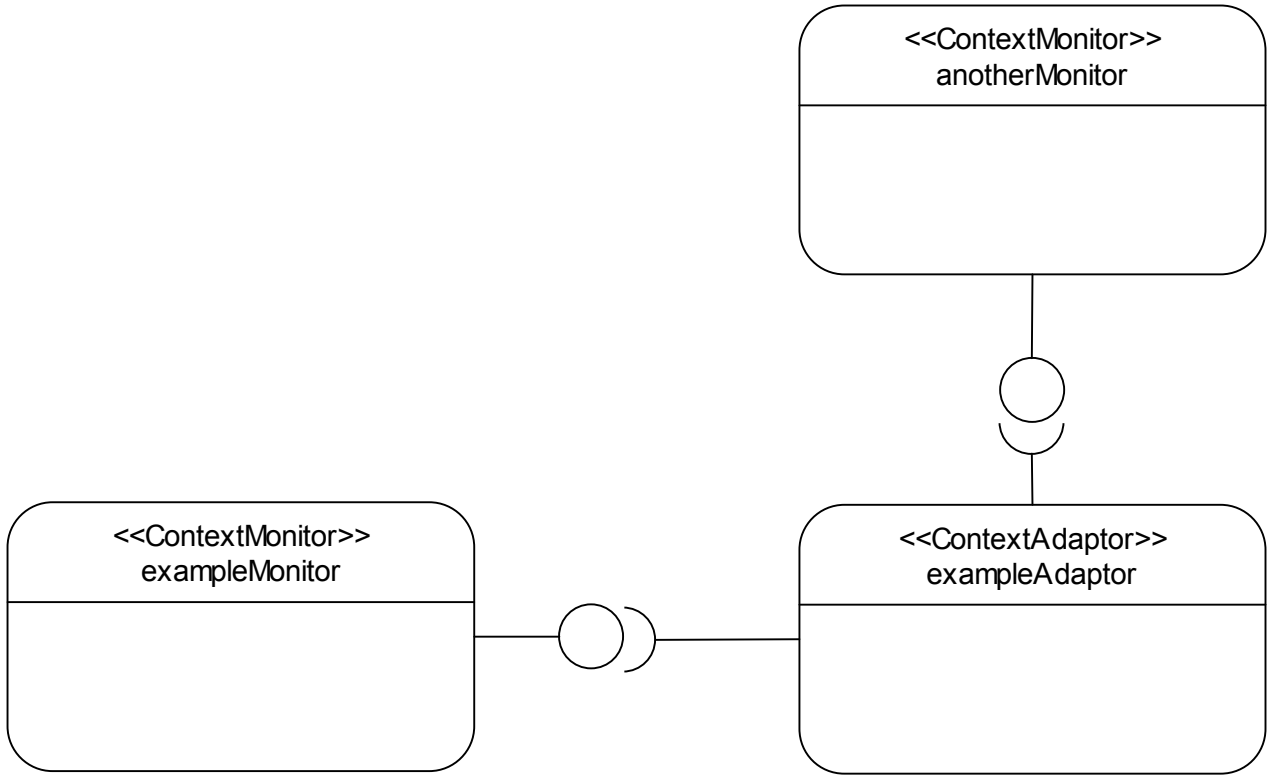
Context Monitor

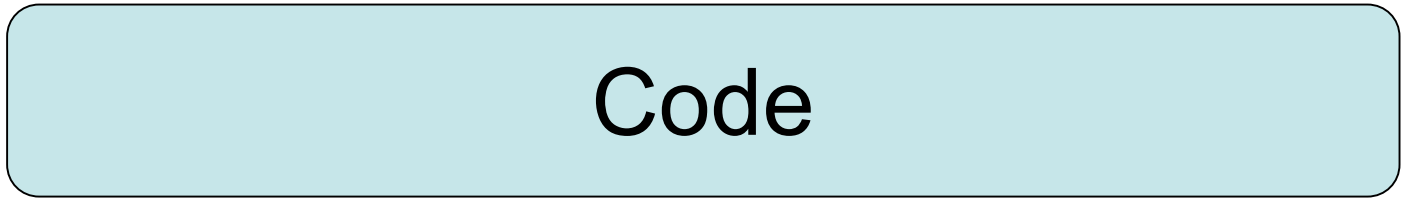


Context Adaptor



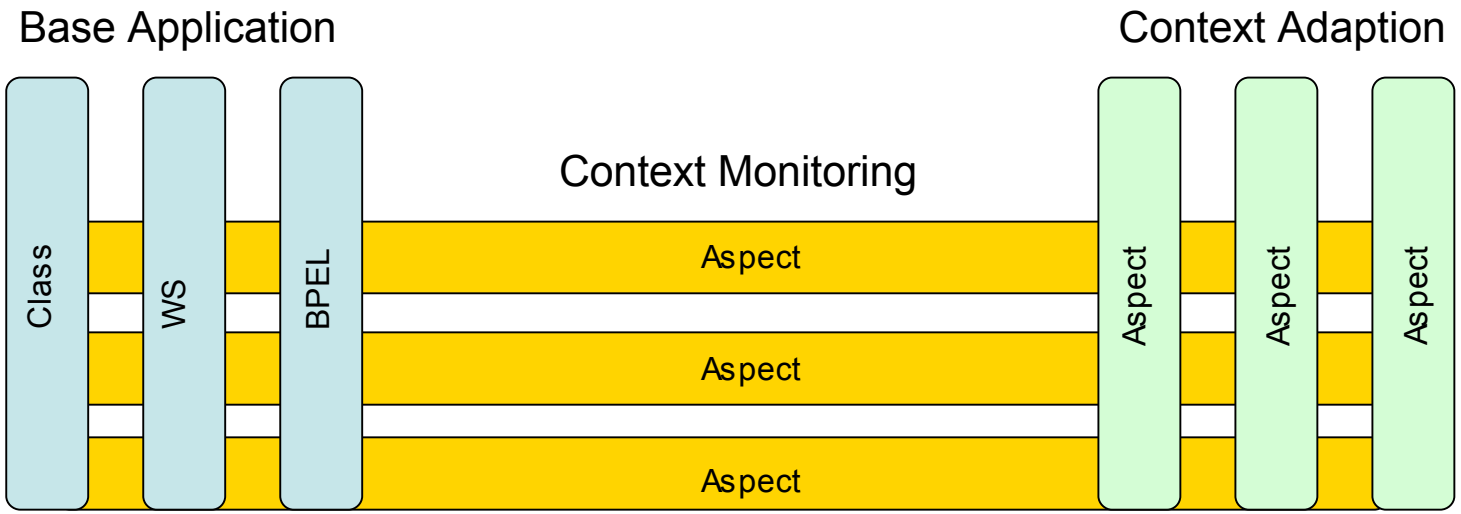
Connecting Monitors to Adaptors




Code

AO Approach

Monitor and Adaptors can be considered as two distinct aspects wich crosscut each other



Implementation

```
aspect ExampleMonitor {

            //events to be sent to the adaptors
            public void contextA(){}
            public void contextB(){}

            pointcut serviceBorC(): call(* WSB.sericeB(..)) ||
                                    call(* WSA.serviceA(..));
            pointcut serviceA() : call(* WSA.servicA(..));

            void around(): serviceBorC(){

                        if(serviceAInvocated && conditionA){
                                    proceed();
                                    contextA();
                        }

            }

            after(): serviceA(){
                        serviceAInvocated=true;
                        if(conditionB){
                                    contextB();
                        }
            }

}
```

Inner events among Monitor and Adaptors

Base Application Crosscuts

Context Monitoring Logic

Implementation

```
aspect ExampleAdaptor {

    pointcut contextA(): call(* ExampleMonitor.contextA());
    pointcut contextB(): call(* ExampleMonitor.contextB());

     after (): contextA(){
            //ActivityA

      }

     after (): contextB(){

            If(condition){
              //ActivityB
            }
            //ActivityC
      }

}
```

Context changes captured by Context Monitors

Context Adaption Activities

Future Works

A COntext Oriented Language to bring at language level concepts as State, Context, ContextMonitor and ContextAdaptor

New techniques to deal with unforeseen adaptation requirements in context-aware applications

Automatic generation of AO4BPEL code