Danmarks Tekniske Universitet

DTU

Towards Separation of Concerns in Flow-Based Programming

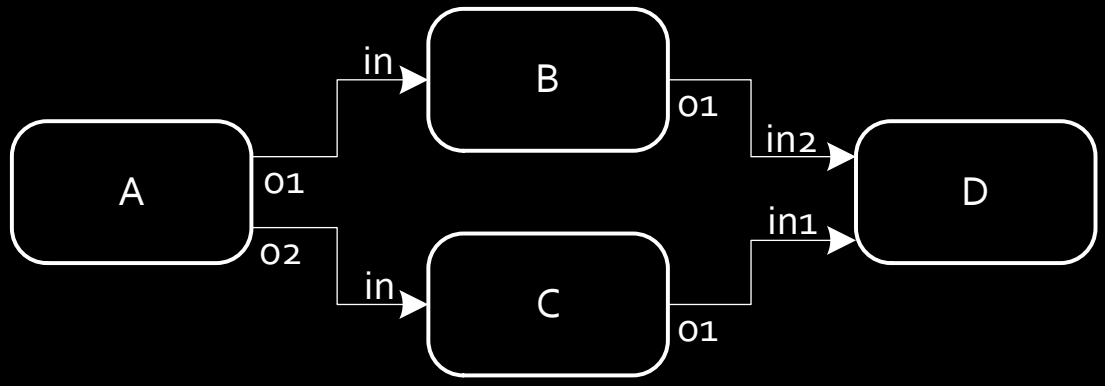
Bahram Zarrin

Hubert Baumeister

FOAL 2015

Flow Based Programming

- Invented by J. Paul Morrison in the early 1970s
- Models software systems as a directed graph of predefined processes which run asynchronously and exchange data through input and output ports.

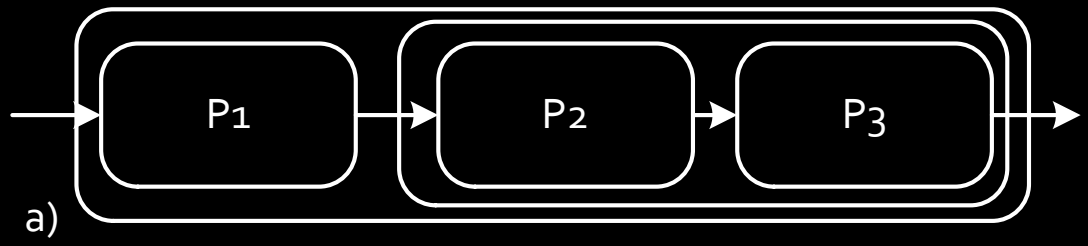


- Atomic or composite process, ports, connections, scheduler
- Implementation (C#FBP, CppFBP, JavaFBP)

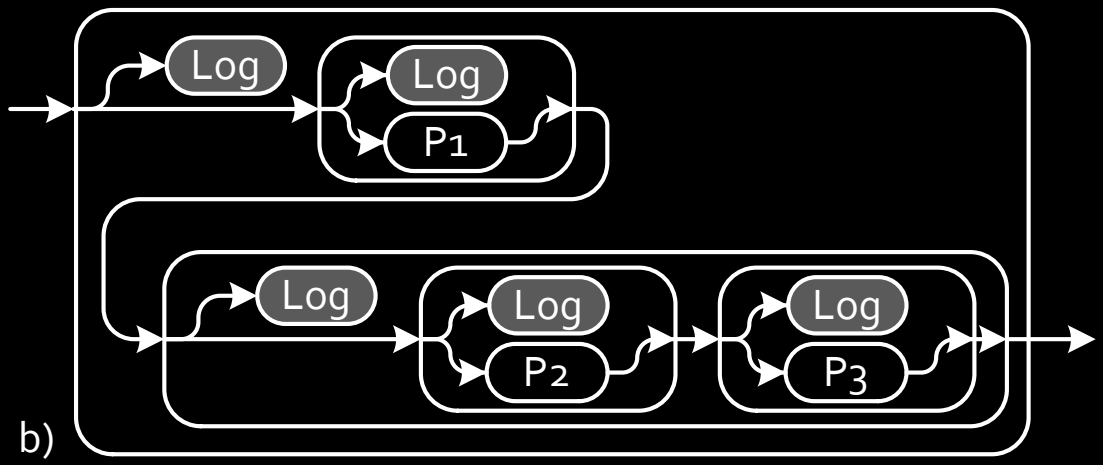
Motivation

- There are concerns in software systems which cannot be modularized well in FBP.
- Examples
  - **Logging**
  - **Life Cycle Assessment**
  - **Costing**

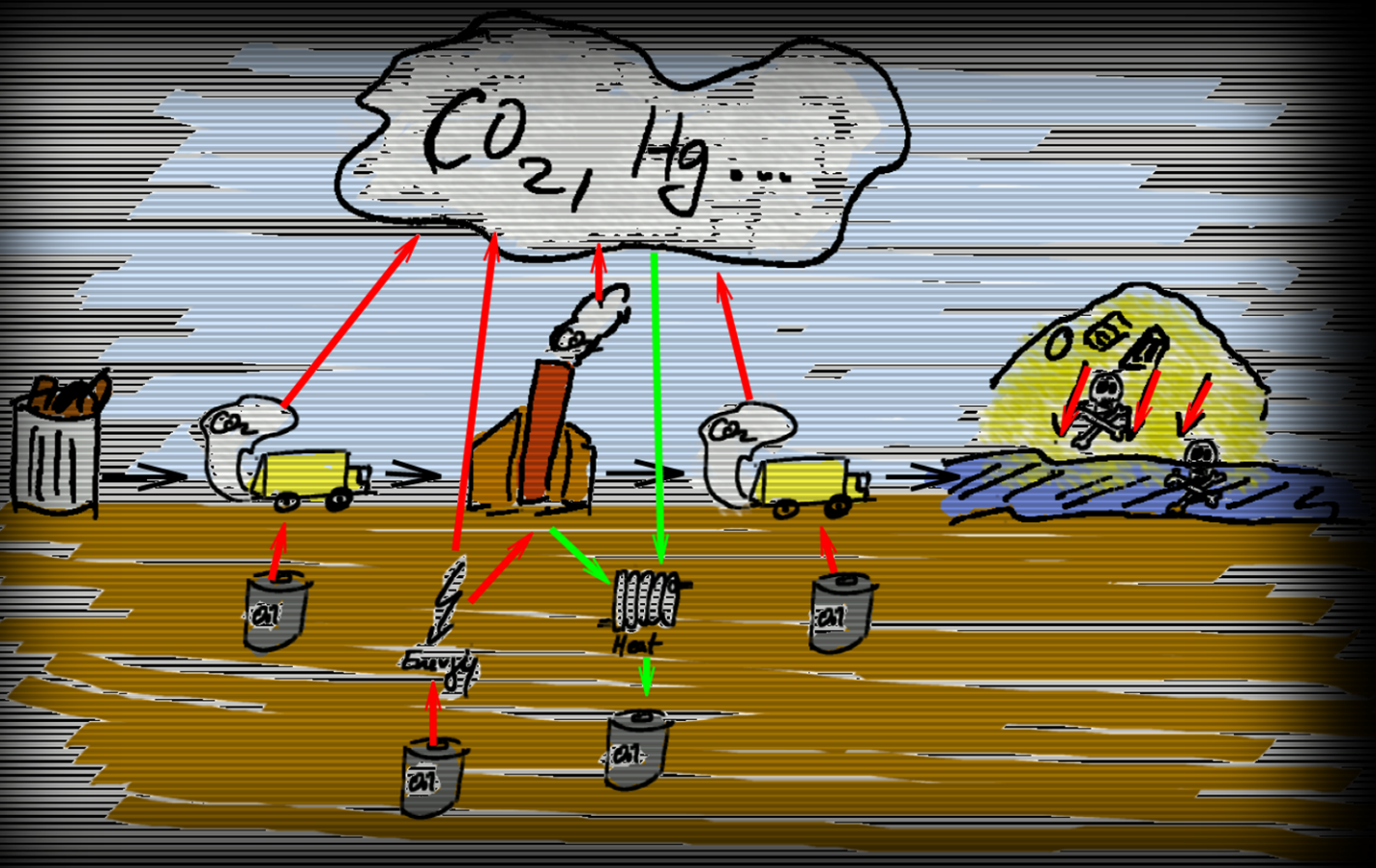
Logging



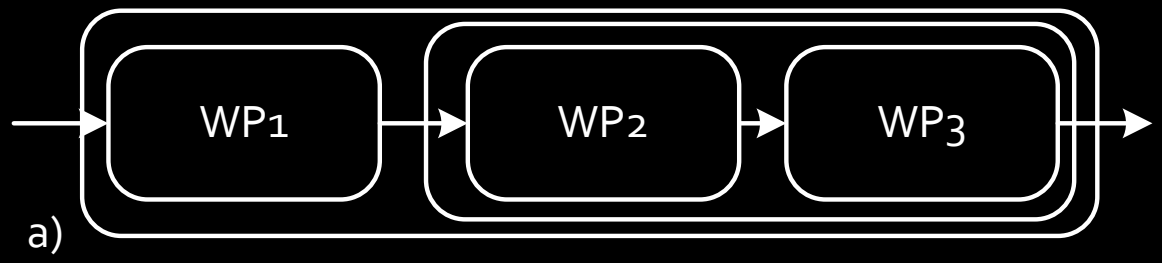
Logging



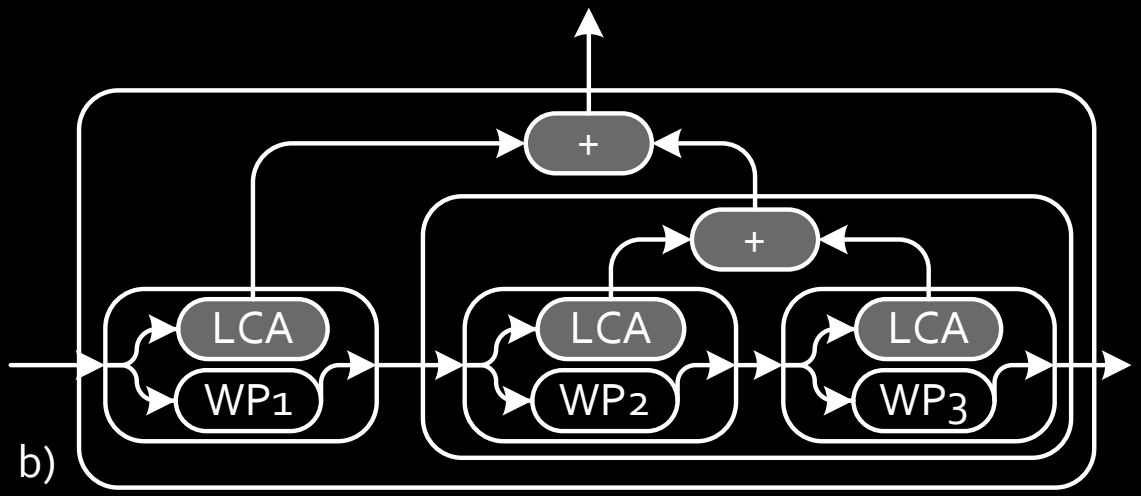
Waste Management Modeling



Waste Management Modeling

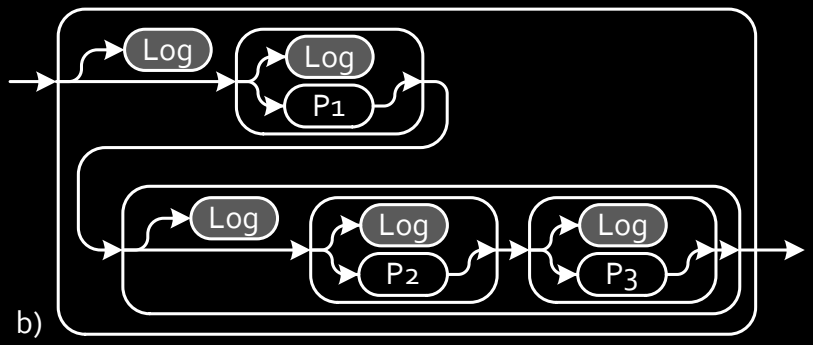


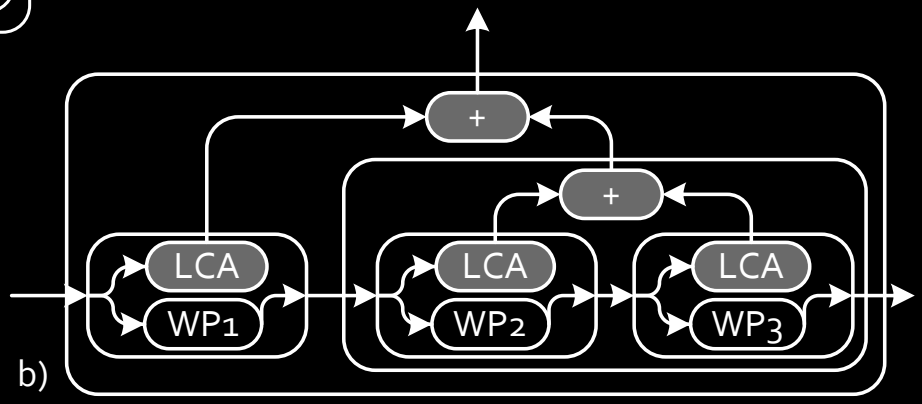
Waste Management Modeling



Cross-Cutting Problem in FBP

- Tangled and scattered process definitions.
- On the one hand, one process addresses several concerns. On the other hand, the implementation of a single concern is scattered through many places in the other process definitions.



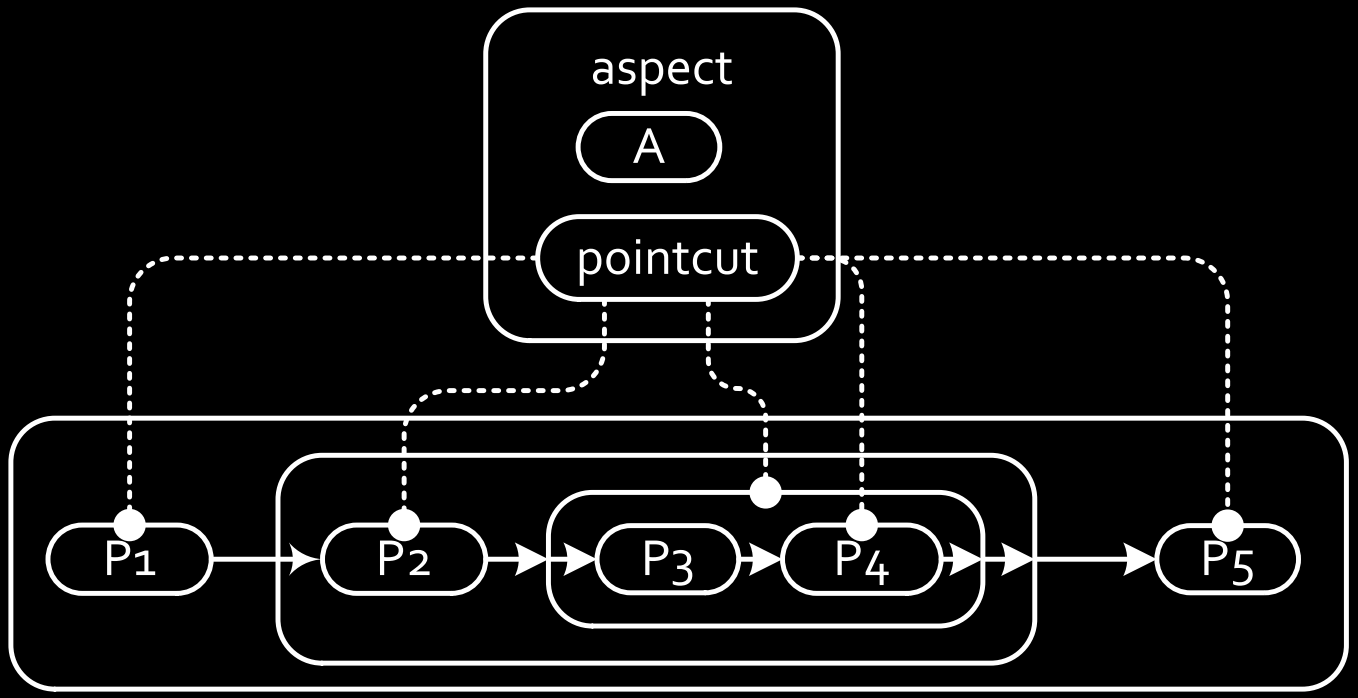


Extending FBP with Aspect-Oriented Concepts

- Aspect-Oriented Flow-Based Programming (AOFBP)
  - Join Point Model and Pointcut Language
  - Advice
  - Weaving

- Tool Support
  - AOC#FBP

AOFBP Join Point Model

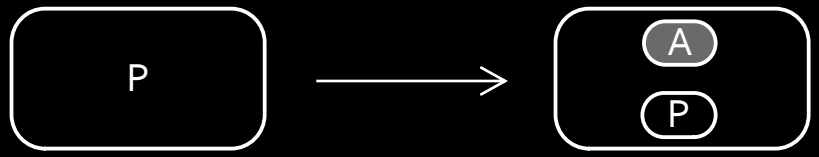


AOFBP Pointcut Language

```
<PortDesignator> ::= inPort (<String>,<String>,<String>)
|outPort (<String> , <String> , <String>)
|port (<String> , <String> , <String>)
<LevelDesignator> ::= level (<String>)
<ContextDesignator> ::= child (<PointcutExp> , <String>)
|parent (<PointcutExp> , <String>)
<ConDesignator> ::= inCon (<PointcutExp> , <String>)
|outCon (<PointcutExp> , <String>)
<Designator> ::= procType (<String>)
|<PortDesignator>|<LevelDesignator>
|<ContextDesignator>|<ConDesignator>
<ParExpr> ::= (<PointcutExp>)
<UnNot> ::= ^<PointcutExp>
<BinAnd> ::= <PointcutExp> & <PointcutExp>
<BinOr> ::= <PointcutExp> '|' <PointcutExp>
<BinExpr> ::= <BinAnd>|<BinOr>
<PointcutExp> ::= <Designator>
|<Identifier>|<ParExpr>|<UnNot>|<BinExpr>
```

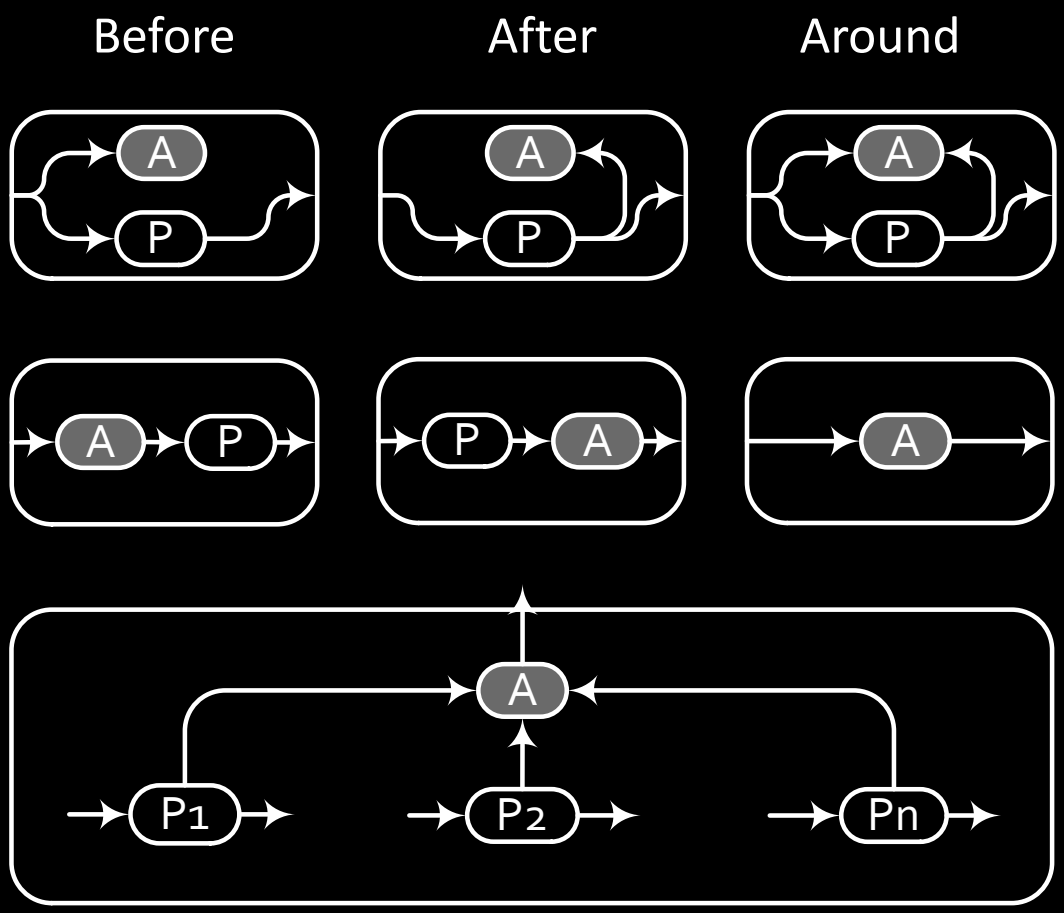
```
procType("*foo") & inPort("*","int","2") &
        inCon(procType("*foo"),"2..4")
```

AOFBP Advice



AOFBP Advice

- Observer
- Adaptor
- Collector



AOFBP Network Definition

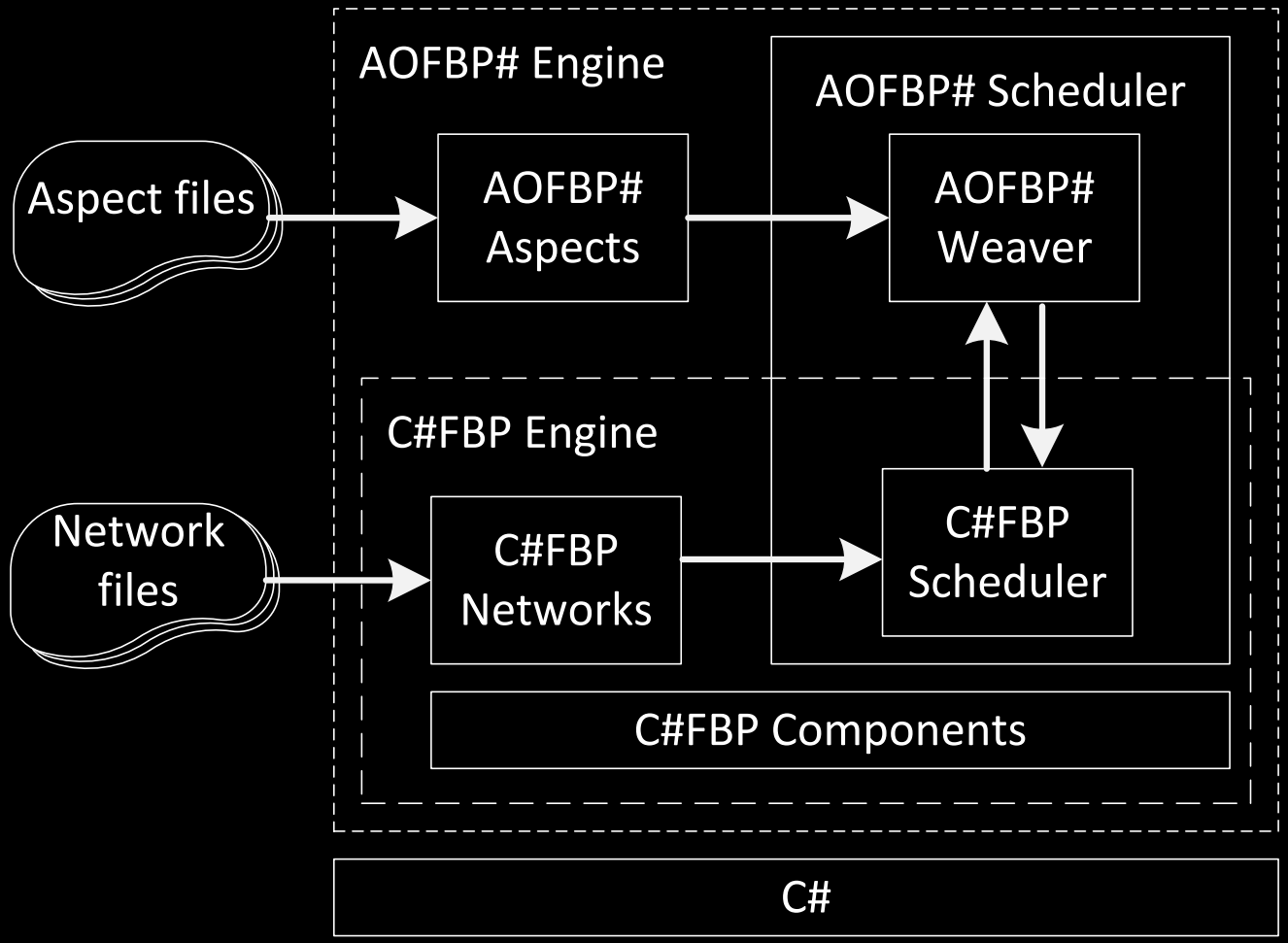
```
<Attribute> ::= name |type |parent
<PortFilter> ::= in (<String> , <String>)
|out (<String> , <String>)
<PortCtor> ::= <Identifier> (<Type>)
<ProcRef> ::= <Identifier>()
<Param> ::= <Identifier> = <Value>
<ParamList> ::= <ParamList> , <Param> | <Param>
<ProcCtor> ::= <Identifier> (<ComponentID>)
|<Identifier> (<ComponentID> : <ParamList>)
<ProcExp> ::= <ProcRef> |<ProcCtor> |<Connection> |this
<Value> ::= <ProcExp> [<Attribute>] |<Number> |<String>
|<Object>
<InExp> ::= <Identifier> <ProcExp> |<PortCtor>
<OutExp> ::= <ProcExp> <Identifier>
|<ProcExp> <PortFilter> |<PortCtor> |<Value>
<Connection> ::= <OutExp> -> <InExp>
<Network> ::= <Network> ; <Connection> | <Connection>
<NetworkDef> ::= network <ComponentID> <Network> end
```

```
network sample
    "some data" -> X P1(Componenet1);
    P1() Y -> Z P2(Component2) K -> R P3(Component3);
end
```

AOFBP Aspect

```
<NamedPortFilter> ::= <PortFilter> as <Identifier>
<PortFilterList> ::= <PortFilterList> ,
<NamedPortFilter> |<NamedPortFilter>
<AdviceType> ::= before |after |around
<Collector> ::= collector <Identifier>
(<PortFilterList>) : <PointcutExp> <Network> end
<Observer> ::= observer <Identifier> <AdviceType> :
<PointcutExp> <Network> end
<Adapter> ::= adapter <Identifier> <AdviceType> :
<PointcutExp> <Network> end
<AdviceDef> ::= <Observer>|<Adapter>|<Collector>
<PonitCutDef> ::= pointcut <Identifier> : <PointcutExp>
<Statement> ::= <PonitcutDef>|<AdviceDef>
<StatementList> ::= <StatementList> ; <Statement>
|<Statement>
<Aspect> ::= aspect <Identifier> <StatementList> end
```

AOFBP Architecture



Examples

```
aspect logging
   pointcut all_processes: procType("*");
   observer logger before: all_processes
        this in("*","*") -> arguments L(Logger : name=
        this [name],       type= this [type])
   end
end
```

Examples

```
aspect LCA
   pointcut p: inPort("*","waste","1..*");
   observer process_LCA () before : p & ^isComposite
        this in("*","waste") -> WASTE_IN lca_process(
        LCAComponent: p_name= this [name], p_type =
        this [type]);
        lca_process() LCA -> LCA (LCA)
   end;
   collector composite_LCA(out("LCA","LCA") as
   inventory):
        p & isComposite
        inventory -> values AP(aggregation);
        AP() result -> LCA (LCA)
   end
end
```

Related Work

- At the moment none of the FBP implementations have addressed the cross-cutting-concerns and provided mechanisms to implement them. PyF, DSPatch, Pypes , and NoFlo.
- AO4BPEL (A. Charfi and M. Mezini. 2007) improves the modularity and increases the flexibility of Web Service composition.
- Composition Filters (L. Bergmans and M. Aksit. 2001) provides separation of concerns for object-based systems.

Conclusion

- Address the cross-cutting concerns in FBP.
- Propose an aspect-oriented approach to FBP called AOFBP to support aspect-oriented concepts in FBP.

- Provide means to specify sub graphs of the processes in a network as join points and to add mechanisms for advice to substitute the subgraph with alternatives as future works.