

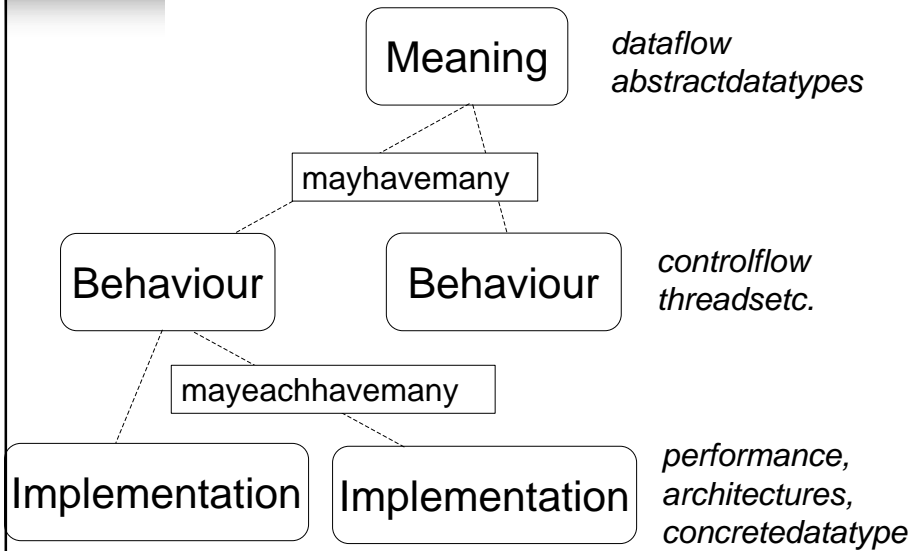
Component Programming Model

Anthony Mayer
London *e*-Science Centre
Reality Grid - 15.05.02

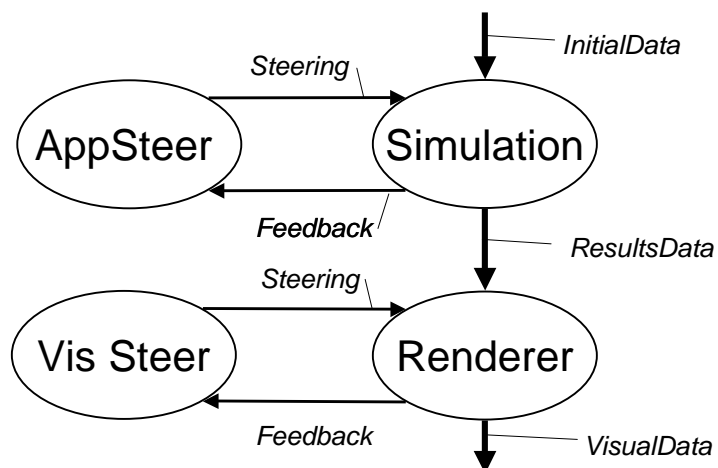
Separation of Concerns

1. Abstraction
 - Annotations preserve information
 - Better programming
2. Encapsulation
 - Software engineering benefits
3. Automation
 - Selection according to run-time information
 - Grid deployment

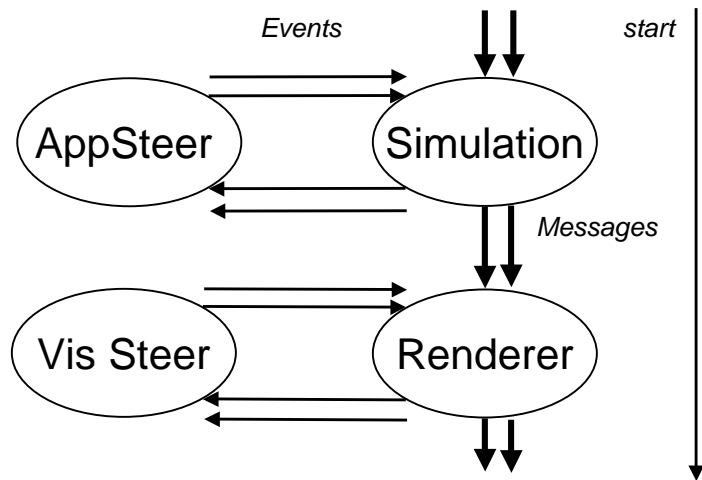
Layered Abstraction



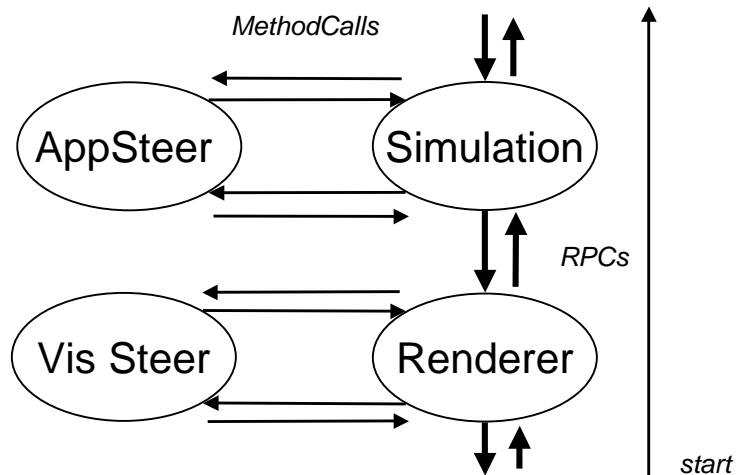
Strawman: Dataflow



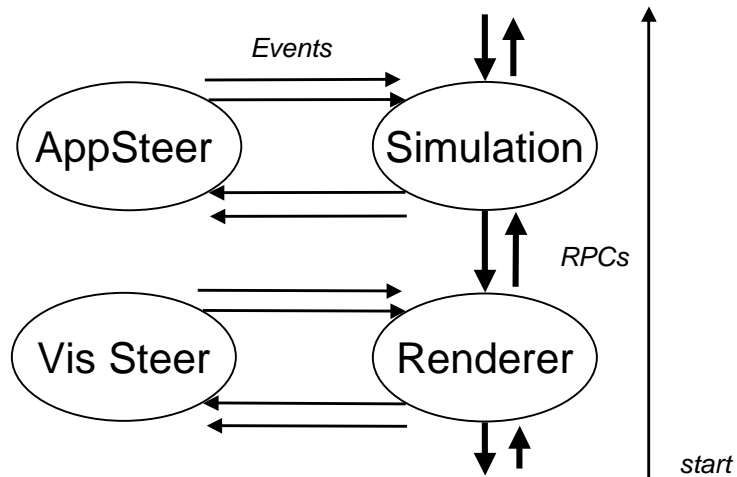
Strawman: ControlFlow Push



Strawman: ControlFlow Pull



Strawman: ControlFlow Mix



Endpoints

	Control Flow In	Control Flow Out
Dataflow In	Message Receive	Call Method
Dataflow Out	Offer Method	Message Send

Added Value: Abstraction

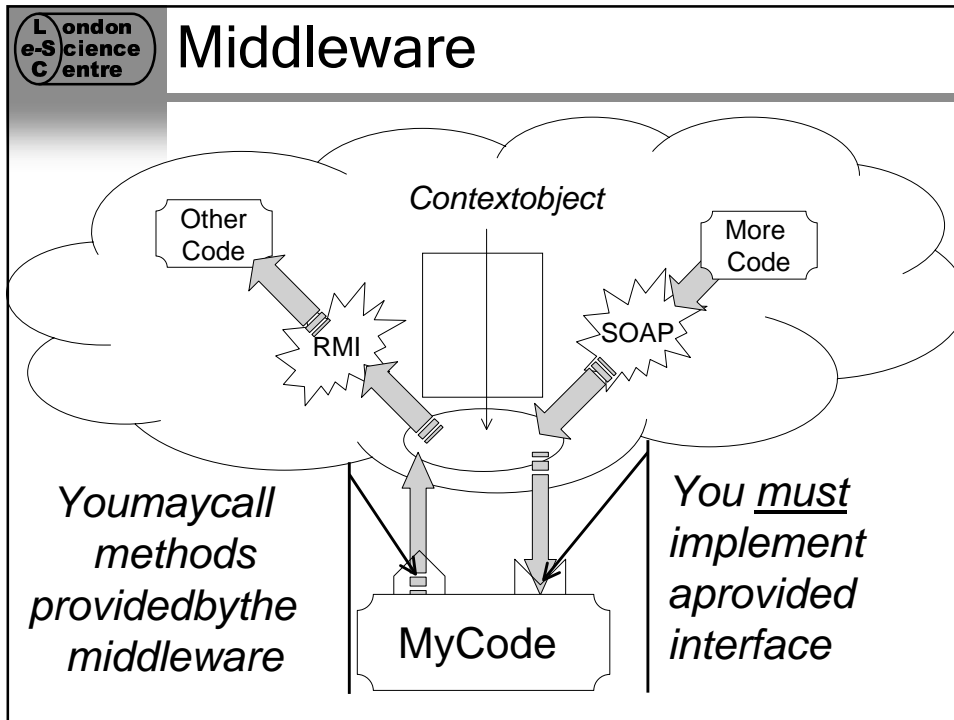
- Composition through Meaning:
 - Flow Based Programming
 - Concurrently existing components
- Multiple patterns supported
- Connections easy
 - Dataflow converters (Glue Code)
 - Controlflow converters
 - Thread Transparency

Added Value: Automation

- Control Flow (user provided)
- Implementation (user provided)
- Communication (machine provided)
- Platform (scheduling decisions)

Utilisation of XML meta-data

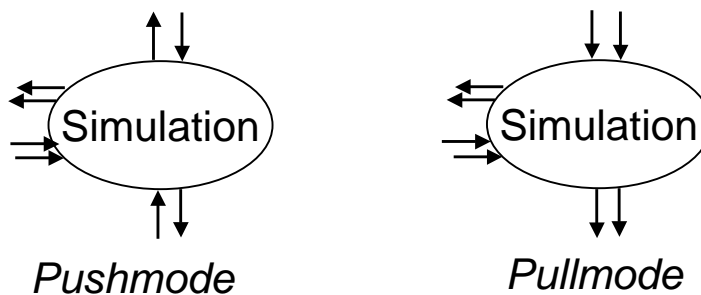
Middleware



Single Implementation

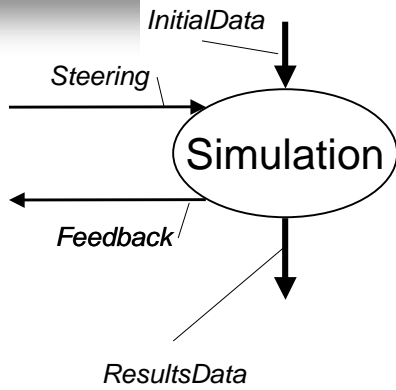
Multiple Behaviour

- A piece of code could have multiple modes



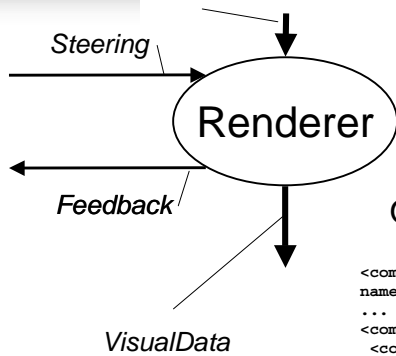
- Obviously must satisfy multiple interfaces
- Allows *run-time* selection of mode, without changing the implementation code!

XML: Meaning



```
<componentDefinitionDocument ...
name="SimulationDefinitions">
<componentTypeDefinition>
<componentTypeName>simulation</componentTypeName>
<port>
<name>DataChannelIn</name>
<portType>init</portType>
<dataflow>in</dataflow>
</port>
<port>
<name>DataChannelOut</name>
<portType>results</portType>
<dataflow>out</dataflow>
</port>
<port>
<name>ControlChannelIn</name>
<portType>steering</portType>
<dataflow>in</dataflow>
</port>
<port>
<name>ControlChannelOut</name>
<portType>feedback</portType>
<dataflow>out</dataflow>
</port>
</componentTypeDefinition>
</componentDefinitionDocument>
```

XML: Meaning 2

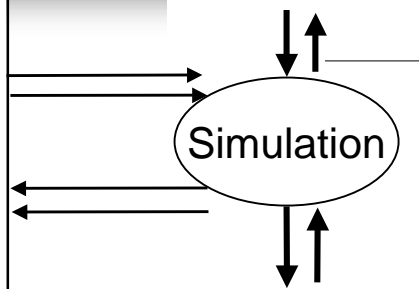


```
<componentDefinitionDocument ...
name="SimulationDefinitions">
...
<portTypeDefinition name="results">
<ddl:adt>streamingResults</ddl:adt>
</portTypeDefinition>
...
```

Or inline them in the definition:

```
<componentDefinitionDocument ...
name="SimulationDefinitions">
...
<componentTypeDefinition>
<componentTypeName>renderer</componentTypeName>
<port>
<name>DataChannelOut</name>
<portTypeDefinition>
<ddl:adt>visualData</ddl:adt>
</portDefinition>
</port>
</componentTypeDefinition>
</componentDefinitionDocument>
```

XML: Behaviour



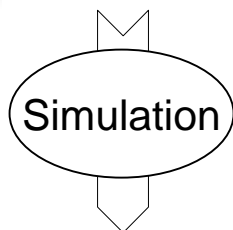
Blocking Communication

Dependencies on other Component's methods

Overlay onto the dataflow

```
<behaviourDescriptionDocument ...
  name="Simulation behaviour definitions">
<behaviour ComponentDescriptionDocument=
"/homes/aem3/xml/cdlexample2.xml">
<componentName>simulation</componentName>
<behaviourName>FullDataPushControl
</behaviourName>
<portBehaviour>
<name>DataChannelIn</name>
< <controlFlowOut>
<threads> ... </threads>
</controlFlowOut>
</portBehaviour>
<portBehaviour>
<name>DataChannelOut</name>
<ddl:dataStruct>FormattedStreamingResults
</ddl:dataStruct>
<controlFlowIn>
<dependencies> ... </dependencies>
</controlFlowIn>
</portBehaviour>
...
</behaviour>
...
</behaviourDescriptionDocument>
```

XML: Implementation



Concrete Data Structure

Performance Data

Overlay onto the dataflow

```
<implementationDescriptionDocument ...
  name="Simulation implementation definitions">
<implementation ComponentDescriptionDocument=
"/homes/aem3/xml/cdlexample2.xml">
<componentName>simulation</componentName>
<implementationName>DefaultImplementation
</implementationName>
<software> ... </software>
<portImplementation>
<name>DataChannelIn</name>
<ddl:dataStruct>FormattedInitialData
</ddl:dataStruct>
<performanceCharacteristics>
...
</performanceCharacteristics >
</portImplementation>
<portImplementation>
<name>DataChannelOut</name>
<ddl:dataStruct>FormattedStreamingResults
</ddl:dataStruct>
<performanceCharacteristics>
...
</performanceCharacteristics >
</portImplementation>
...
</implementation>
...
</implementationDescriptionDocument>
```

Current Status

- Internal Testing:
 - XML dialects
 - Interface generation tools:
 - Java
 - Web Services
- Development:
 - Collective Communication
 - Context objects
 - Converters (Controlflow; Dataflow to follow)

Further Reading

- Component Based Programming:
 - Szyperski
- Flow Based Programming:
 - J P Morrison
- Separation of Control & Data Flow:
 - Infopipes (Oregon)
- Collective Communication & Coordination
 - Skeletons: Murray Cole, John Darlington et al
 - Message Passing: MPI, WSFL standards