2/13/2013

Platform Project
Interaction in MITK

GERMAN
CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

Problems

Jurisdiction:

- no clear rules about who gets an event
- several interactors may respond to one event
- each interactor can give an estimate:

State machine description

- two versions
- it’s all ids and numbers
- monolithic (~ 4000 LOC)
- in the core

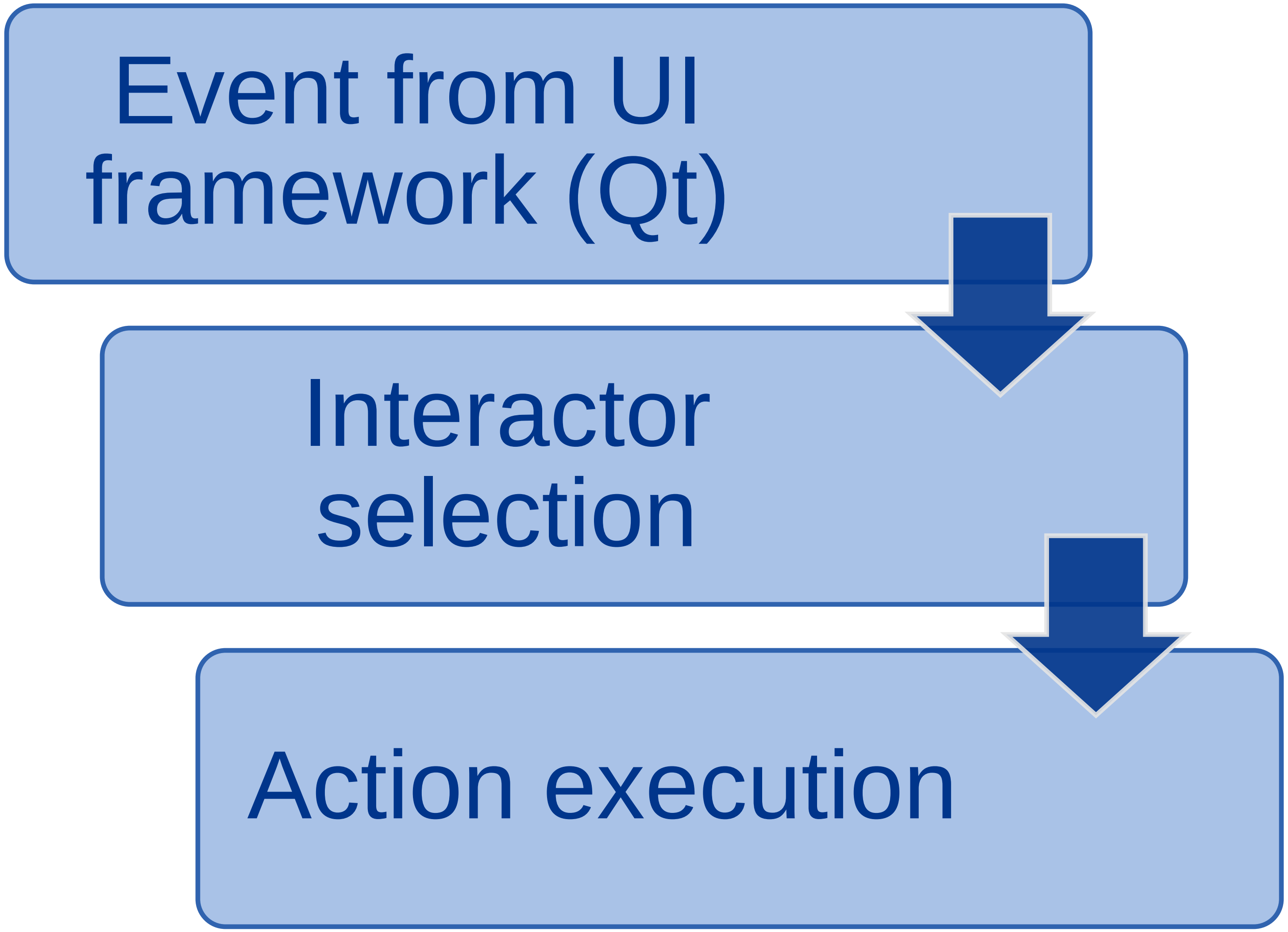
New Concept

Topics

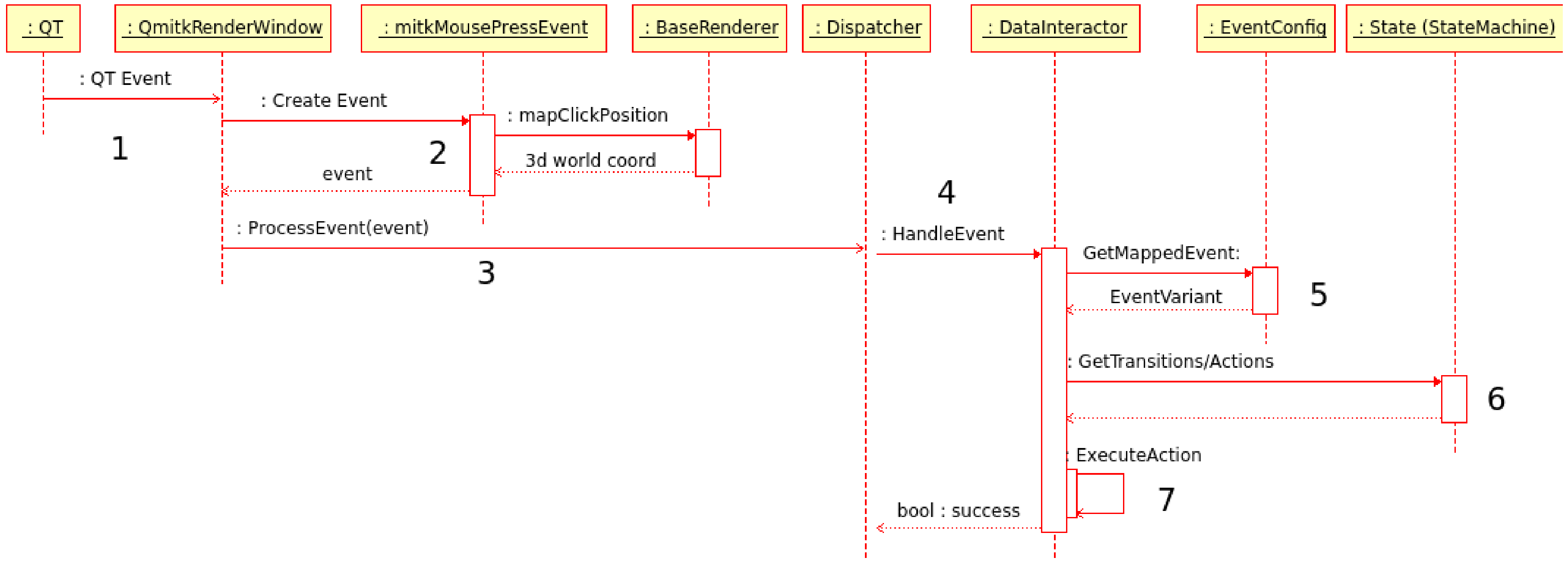
- Event creation & event types
- Event distribution
- State machine patterns & configuration
- Action execution

- How to implement new interactors
- How to use new interactors

Event Handling – The general Idea

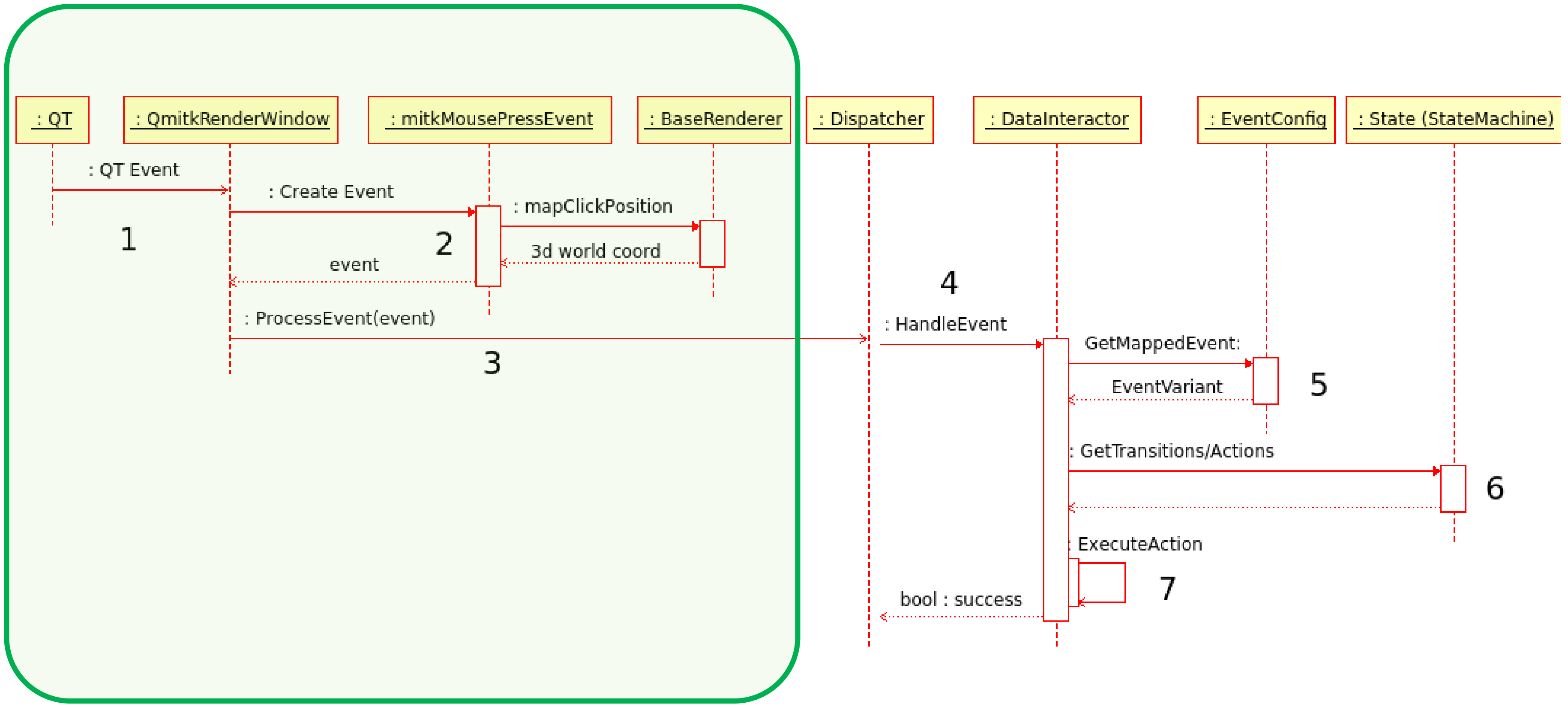


Event Handling – Sequence Diagram



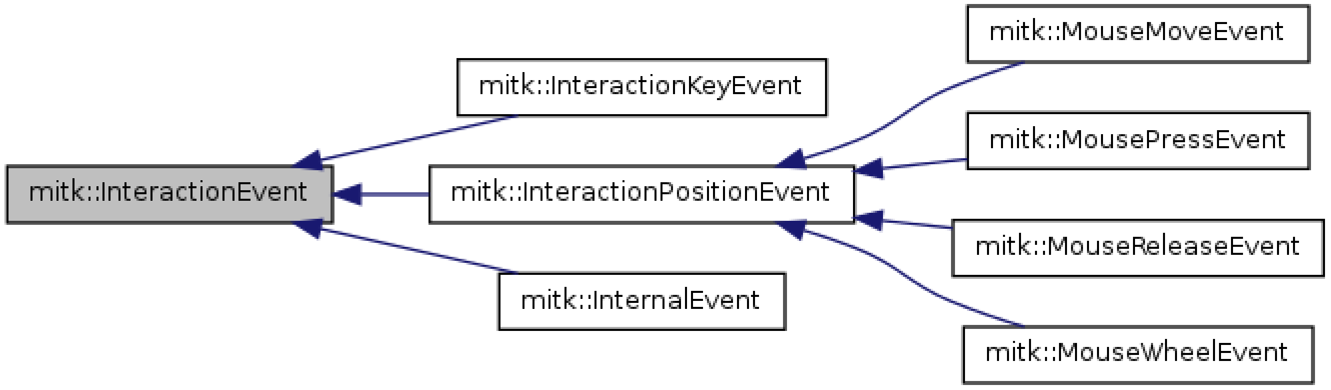
http://docs.mitk.org/nightly-qt4/DataInteractionPage.html

Event Handling – Event Creation



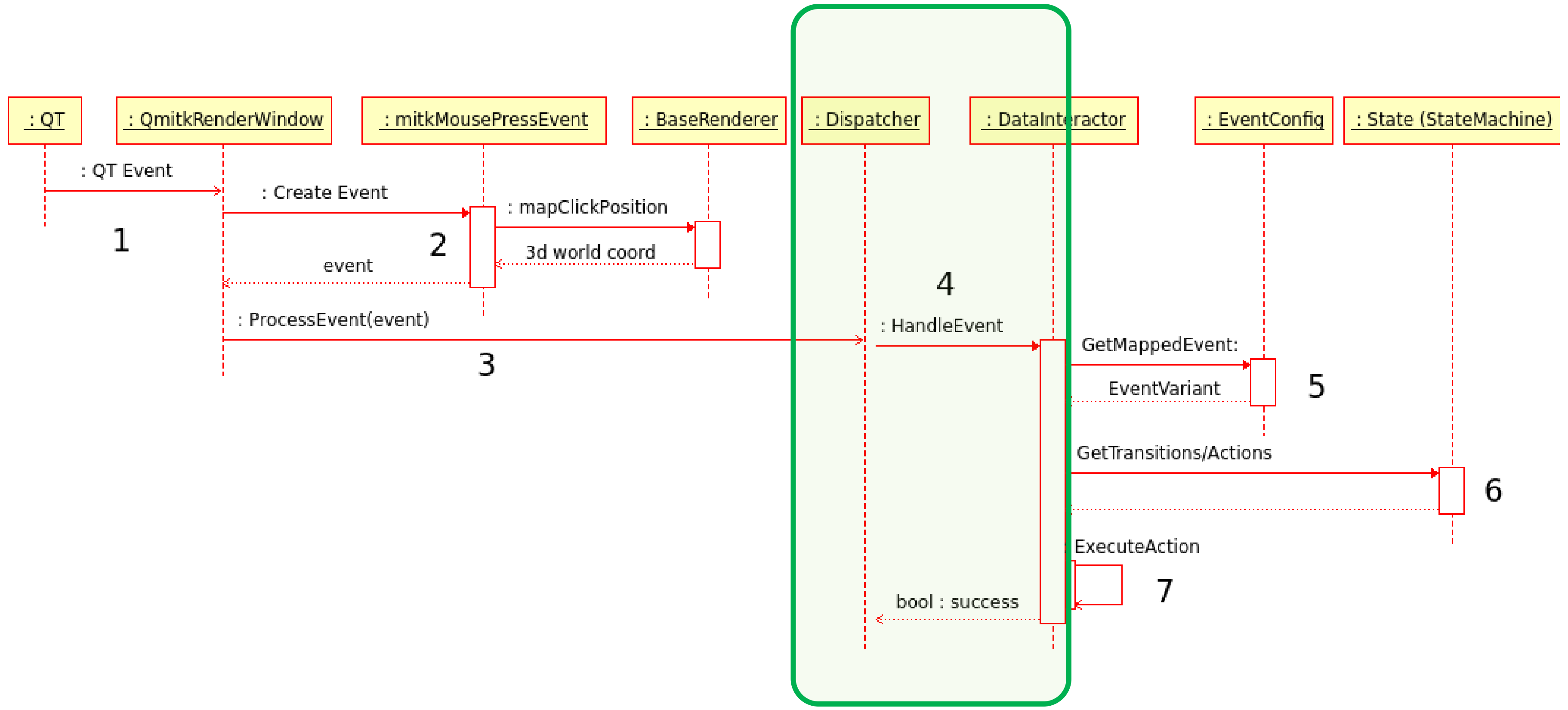
http://docs.mitk.org/nightly-qt4/DataInteractionPage.html

Events

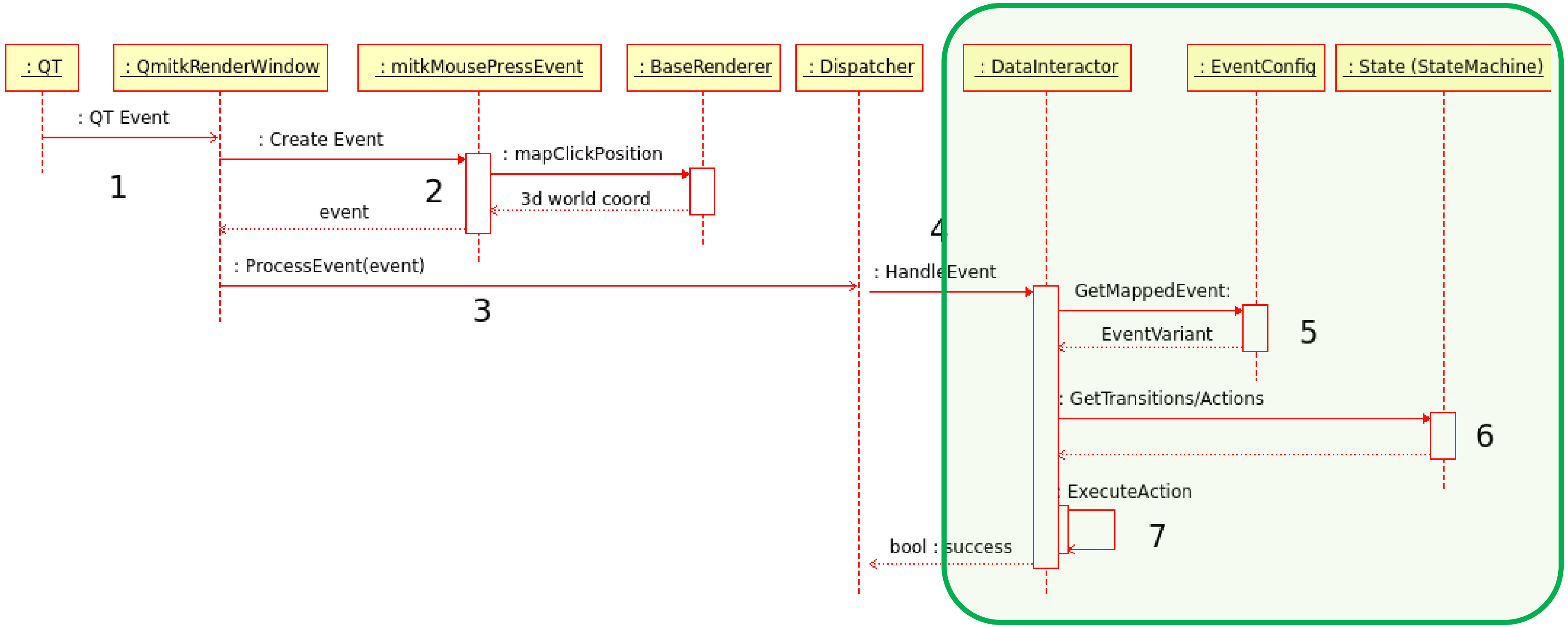


- Each event type is represented by a class
- State machines support polymorphism of events
- Easy to extend

Event Handling – Distribution



http://docs.mitk.org/nightly-qt4/DataInteractionPage.html



http://docs.mitk.org/nightly-qt4/DataInteractionPage.html

State Machines - Description & Configuration

State machine description:

```
<state name="CollectPoints">
   <transition event_class="PositionEvent" event_variant="ConfigurableEvent"
               target="CollectPoints">
     <action name="AddPoint"/>
   </transition>
</state>
```

Configuration description:

Example 1 :

```
<config>
 <input event_class="MousePressEvent" event_variant="ConfigurableEvent">
   <attribute name="EventButton" value="LeftMouseButton"/>
 </input>
</config>
```

Example 2 :

```
<config>
 <input event_class="MouseReleaseEvent" event_variant="ConfigurableEvent">
   <attribute name="EventButton" value="RightMouseButton"/>
 </input>
</config>
```

See Documentation for detailed description:
http://docs.mitk.org/nightly-qt4/Step10Page.html

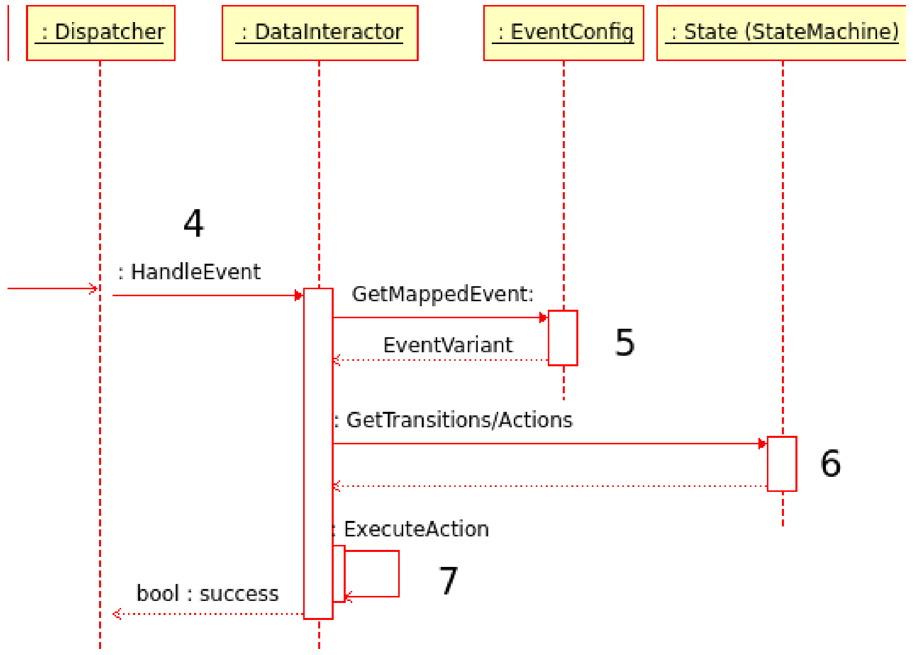
Configuration Objects

ConfigurableEvent

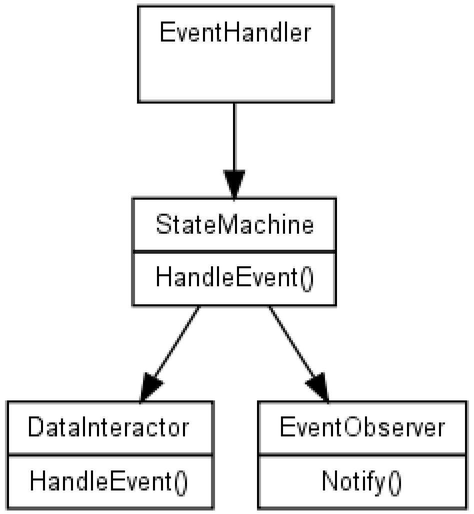
- Event class = MousePressEvent
- Event button = LeftMouseButton
- Modifiers = Ctrl

Event variant – tag of event object

Event object is built according to description in configuration file.



Implementation of DataInteractors



Inherit from mitk::DataInteractor

Implement your functionality with the following interface:

```
bool SomeFunctionality(StateMachineAction* , InteractionEvent*);
```

Connected with actions by implementing the function ConnectActionsAndFunctions() and using the CONNECT_FUNCTION macro.

```
void mitk::ExampleInteractor::ConnectActionsAndFunctions()
{
   CONNECT_FUNCTION("actionNameFromPattern", SomeFunctionality);
   CONNECT_FUNCTION("actionName2FromPattern ", AnotherFunctionality);
}
```

How to use new DataInteractor

First we need a DataNode that is added to the DataStorage.

```
mitk::DataNode::Pointer dataNode = mitk::DataNode::New();
GetDataStorage()->Add(dataNode.GetPointer());
```

Then we create an instance of to PointSetDataInteractor and load a statemachine pattern as well as a configuration for it:

```
m_CurrentInteractor = mitk::PointSetDataInteractor::New();
m_CurrentInteractor->LoadStateMachine("PointSet.xml");
m_CurrentInteractor->LoadEventConfig("PointSetConfig.xml");
```

Lastly the DataNode is added to the DataInteractor

```
m_CurrentInteractor->SetDataNode(dataNode);
```

Thank you for your attention!

Question ???