Progettazione avanzata di software di controllo industriale

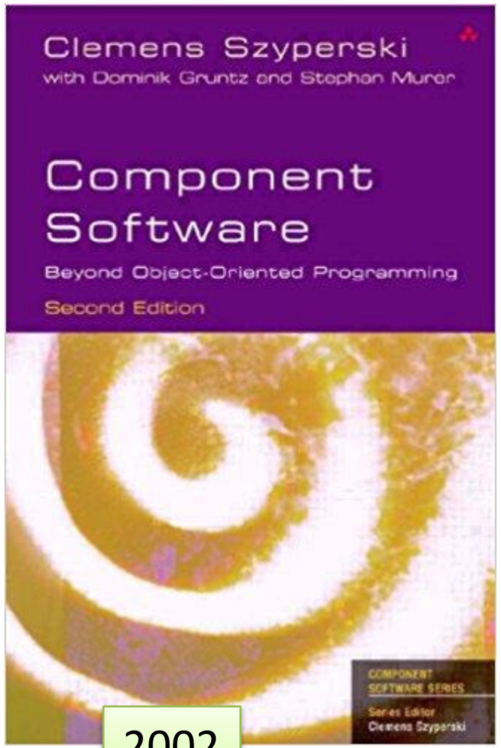
Elettric80

https://github.com/anatali/lss0

Sviluppo di sistemi software

PROGETTAZIONE, COMPONENTI, RIUSO, TESTING

Software components

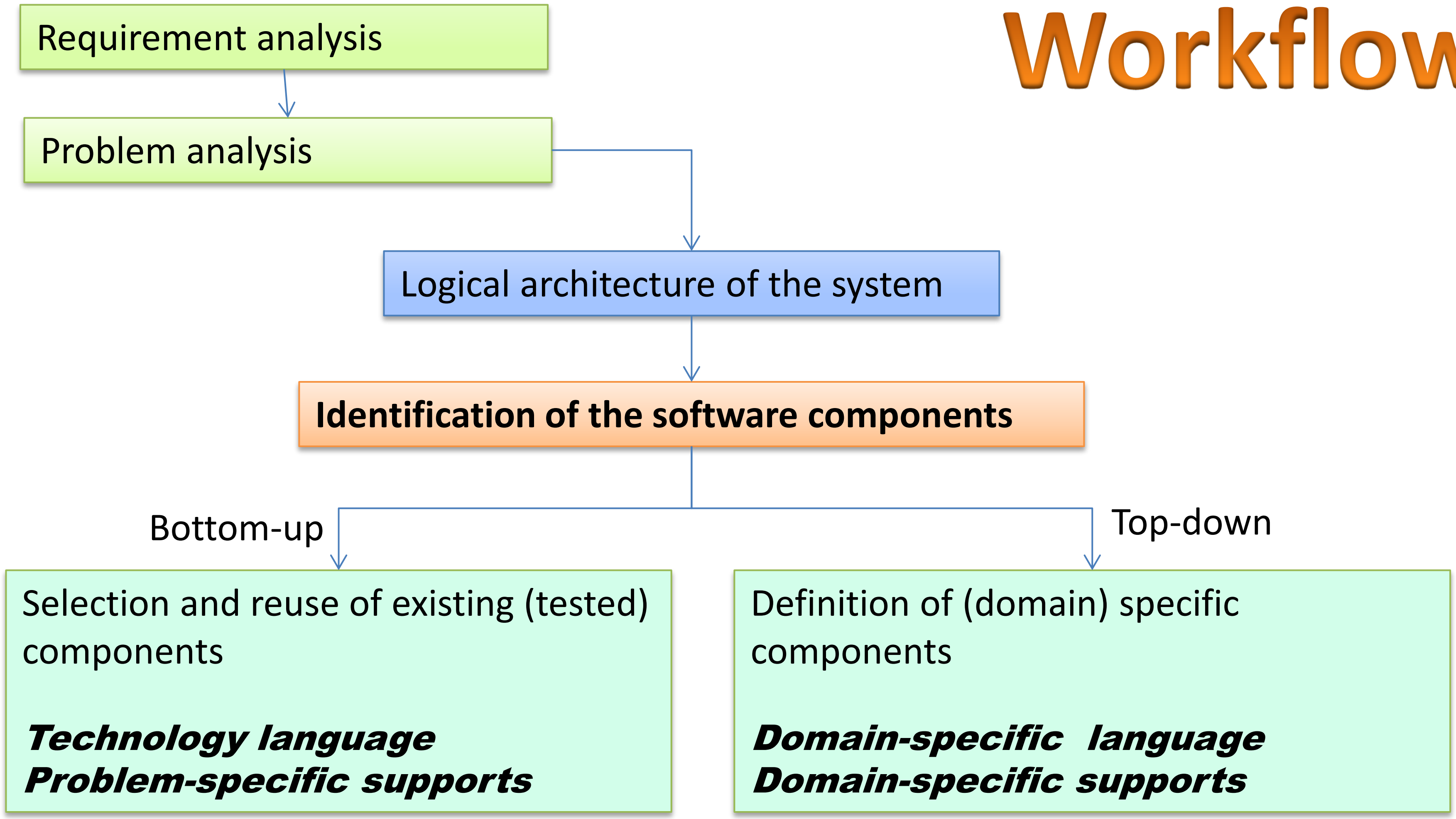

Clemens Szyperski
with Dominik Gruntz and Stephan Murer
Component Software
Beyond Object-Oriented Programming
Second Edition
COMPONENT SOFTWARE SERIES
Series Editor Clemens Szyperski

2002

The book gives us an objective survey of the component landscape, blended with unique insights into the market forces that influence deployment and in-depth coverage of real problems and their solutions.
Highlights of the Second Edition include:

- A comprehensive update of market-leading technologies including COM+, CORBA, EJB and J2EE
- New sections evaluating the strengths and weaknesses of emerging technologies like .NET, the CORBA Component Model, XML Web Services, showing how they work together with components and XML-related standards
- New examples in C# in addition to Java and Component Pascal

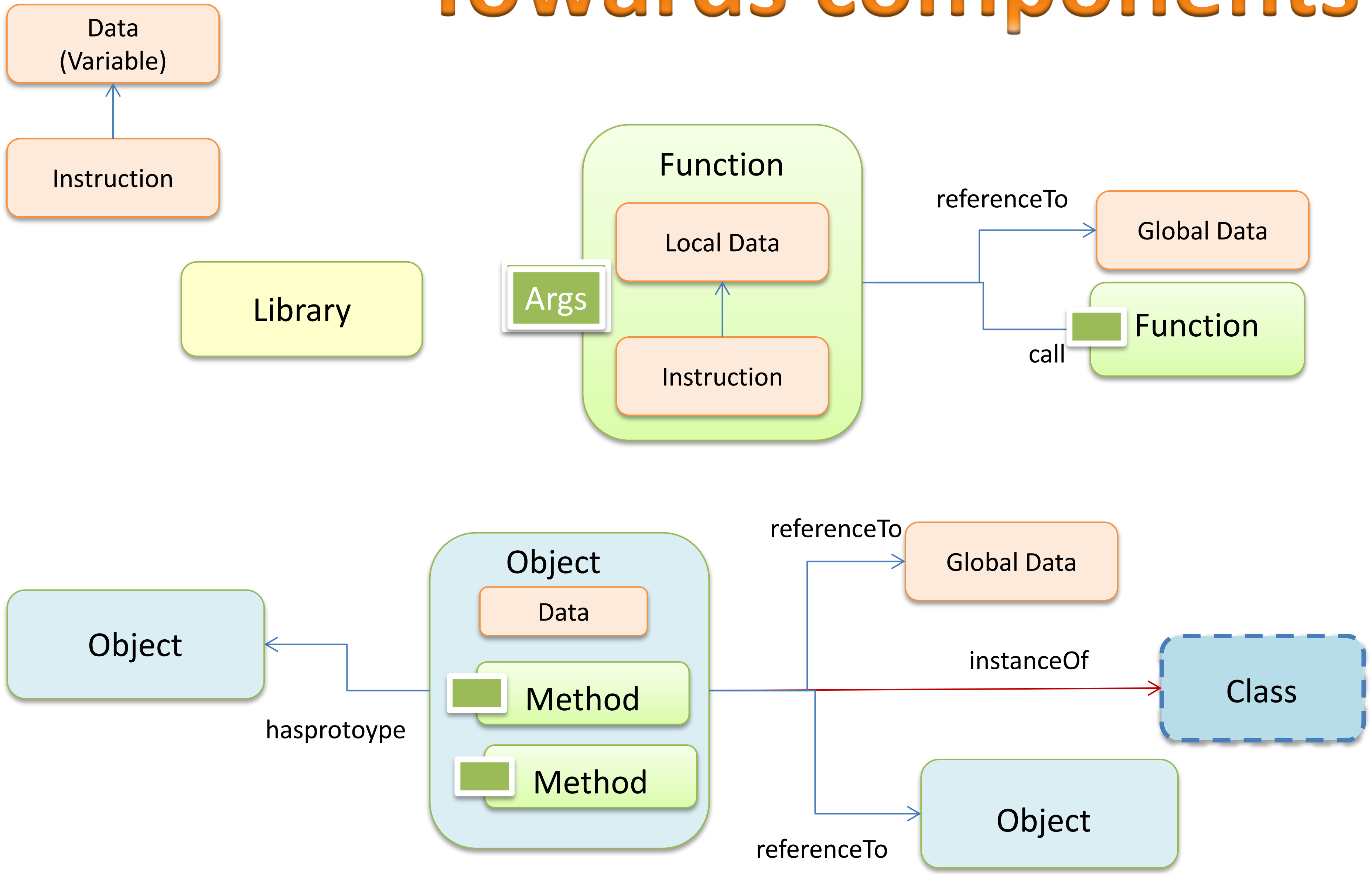
Workflow


Requirement analysis
Problem analysis
Logical architecture of the system
Identification of the software components
Bottom-up
Top-down
Selection and reuse of existing (tested) components
Technology language
Problem-specific supports
Definition of (domain) specific components
Domain-specific language
Domain-specific supports

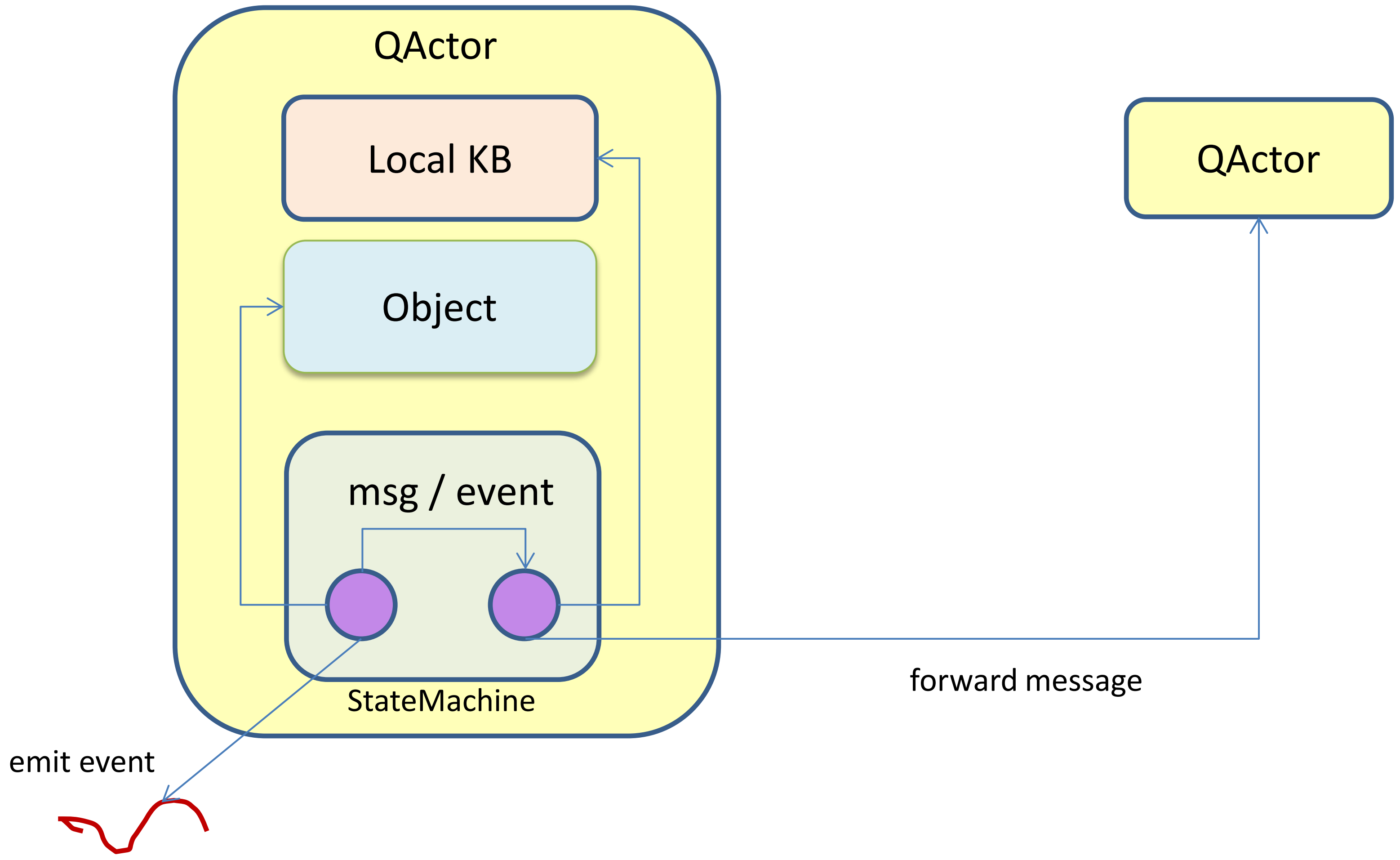
In any case:

What is a software component?
How components interact?
Which components embed the ***business logic***?

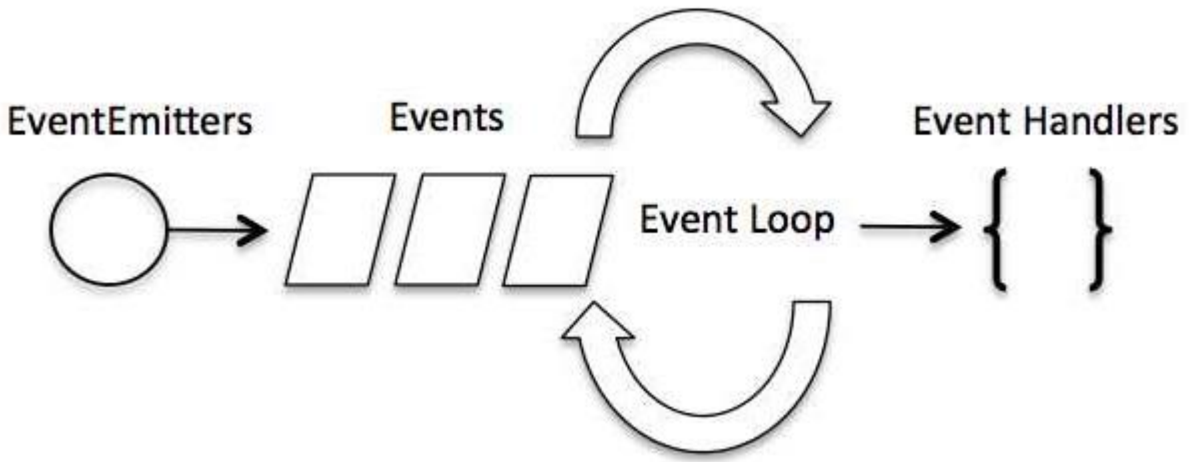
Towards components



Beyond procedure calls



Event loop



```
setTimeout(  function(){ console.log(“1000a1”); console.log(“1000a2”);  } , 1000 );

setTimeout(  function(){ console.log(“1000b1”); console.log(“1000b2”);  } , 1000 );

setTimeout(  function(){ console.log(“500”);  } , 500);
```

Fact asynch

```
factAsynch = function( n,  callback ){ factIterAsynch(n,n,1,callback); }
factIterAsynch = function( n, n0, v, callback ){
var res = n*v;          //ACCUMULATOR
        console.log( "factIterAsynch n0=" + n0 + " n=" + n, " v=" + v + " res=" + res);
        if( n == 1 ) callback( "factIterAsynch(" + n0 + ") RESULT="+res );
        else setTimeout( function(){  factIterAsynch( n-1, n0, res, callback ) ; } , 0 );
 }
console.log("START");
console.log("CALL= ", factAsynch(4, console.log) );
factAsynch(6,console.log);
console.log("END");
```

```
START
factIterAsynch n0=4 n=4  v=1 res=4
CALL=  undefined
factIterAsynch n0=6 n=6  v=1 res=6
END
factIterAsynch n0=4 n=3  v=4 res=12
factIterAsynch n0=6 n=5  v=6 res=30
factIterAsynch n0=4 n=2  v=12 res=24
factIterAsynch n0=6 n=4  v=30 res=120
factIterAsynch n0=4 n=1  v=24 res=24
factIterAsynch(4) RESULT=24
factIterAsynch n0=6 n=3  v=120 res=360
factIterAsynch n0=6 n=2  v=360 res=720
factIterAsynch n0=6 n=1  v=720 res=720
factIterAsynch(6) RESULT=720
```

Fibonacci asynch

```
fibonacciAsync  = function( n, callback ){
if( n==1  || n == 2 || n == 3 ) {   callback( n  );  }
else{
console.log( "fibonacciAsync for " + n   );
  process.nextTick(function() {
            fibonacciAsync( n -1 , function(val1){
            process.nextTick(function() {
                    fibonacciAsync( n -2, function(val2){
                                   callback(  val1 + val2  );
                    });
                });
            });
        });
    }}
console.log("fibAsynch  STARTS ");
fibonacciAsync(10, console.log);
console.log("fibAsynch  ENDS ");
```

Actions (Asyhnchronous)

Components

Responsibilities

Business logic

System (Microservice)

Architecture

ButtonLed system

Project it.unibo.qa.nodeserver

actions	actions types
blsHlCustom:	a ‘onion’ system on PC /Rasp
blshlBlink	a system that executes reactive actions
blsHlNode	a system that works with Node
helloMqtt	a system that does publish/subscribe

blsHlCustom

A button-led system working on a PC

1. it.unibo.buttonLed.components. DevLed
2. it.unibo.buttonLed.components.DeviceLedImpl
3. it.unibo.custom.led. LedFactory
4. it.unibo.custom.button. ButtonFactory
5. blsHLCustom.qa
6. --
7. srcMore/it.unibo.ctxBlsHlCustom/QActorWebUI.html
8. Context ctxBlsHlCustom ip [host="localhost" port=8029] -httpserver
9. --
10. Events and event conversion

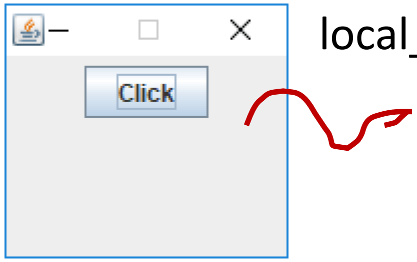
blsHlNode

A button-led system working in Node on a PC and on Raspberry

1. it.unibo.qa.nodeserver\node\blsOop\Led.js
2. it.unibo.qa.nodeserver\node\blsOop\LedImplPcjs
3. it.unibo.qa.nodeserver\node\blsOop\LedHlPc.js
4. blsHLNode.qa (a qactor that interacts with a Led implemented in Node)
5. --
6. it.unibo.qa.nodeserver\cmd.txt
7. --
8. it.unibo.qa.nodeserver\node\blsOop\LedHlRasp.js
9. it.unibo.qa.nodeserver\node\blsOop\LedImplGpiojs

Project it.unibo.qa.nodeserver

CustomBlsGui.*createCustomButtonGui(*QActor qa)



local_click : click(N)

AD HOC

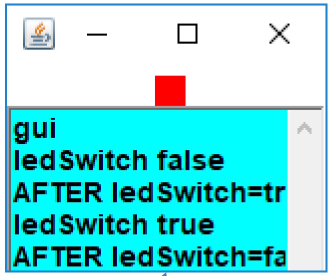
createLedObjecGui

Logical

Concrete

DevLed → DeviceLedGui

DeviceLedRasp

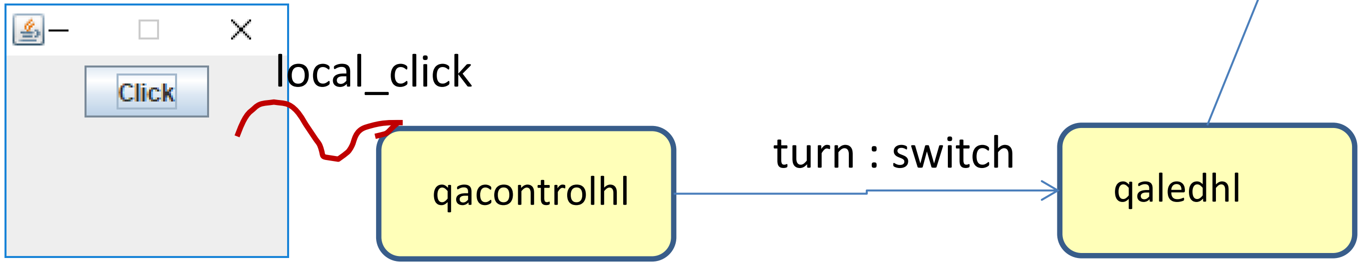


```
gui
ledSwitch false
AFTER ledSwitch=tr
ledSwitch true
AFTER ledSwitch=fa
```

DEIVICE ONTOLOGY

showOntology

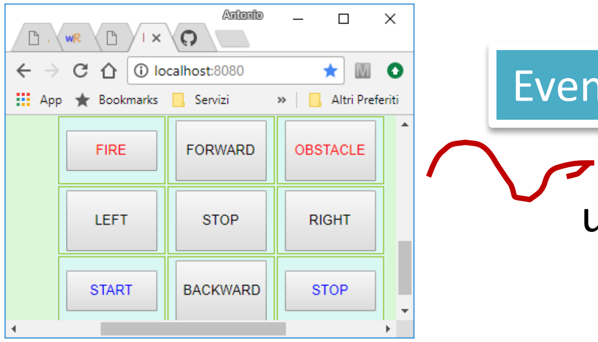
blsHl.qa



INTEGRATION

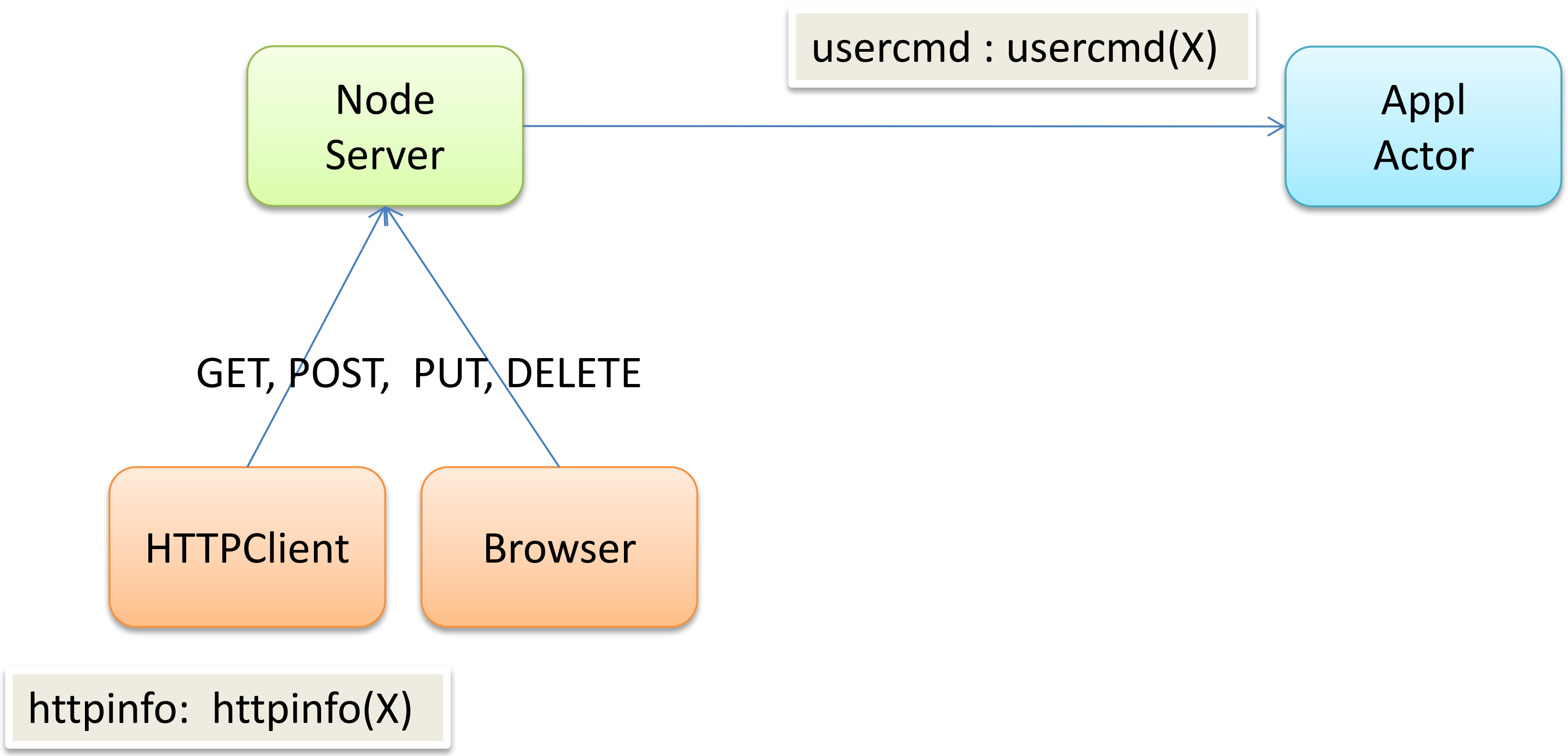
DISTRIBUTION

HETEROGENEITY

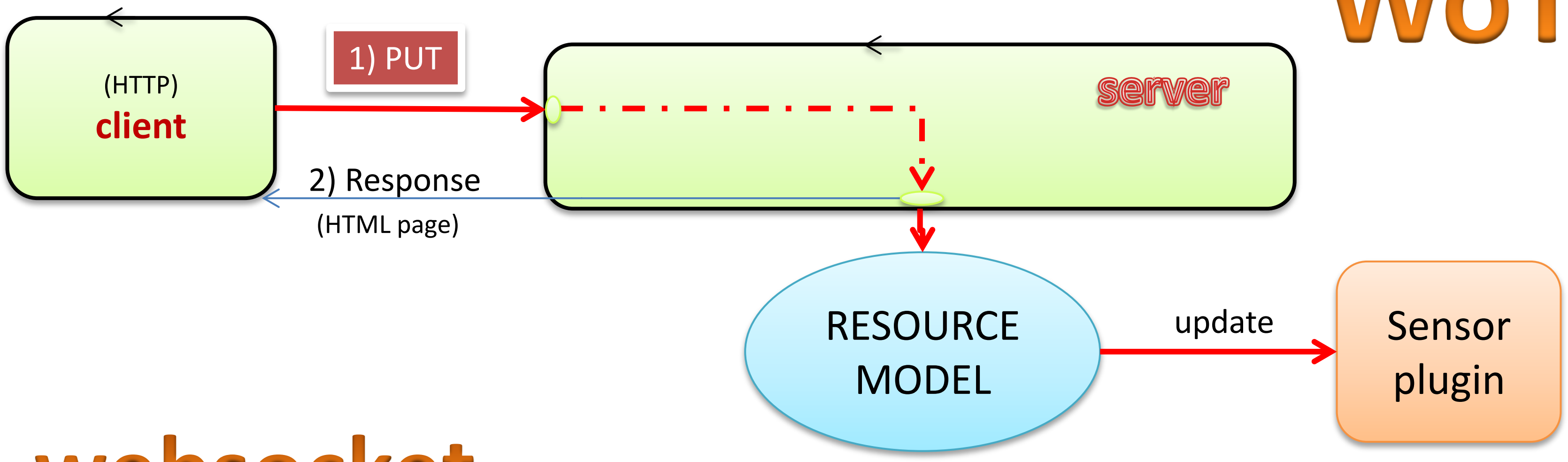


EventHandler as event converter

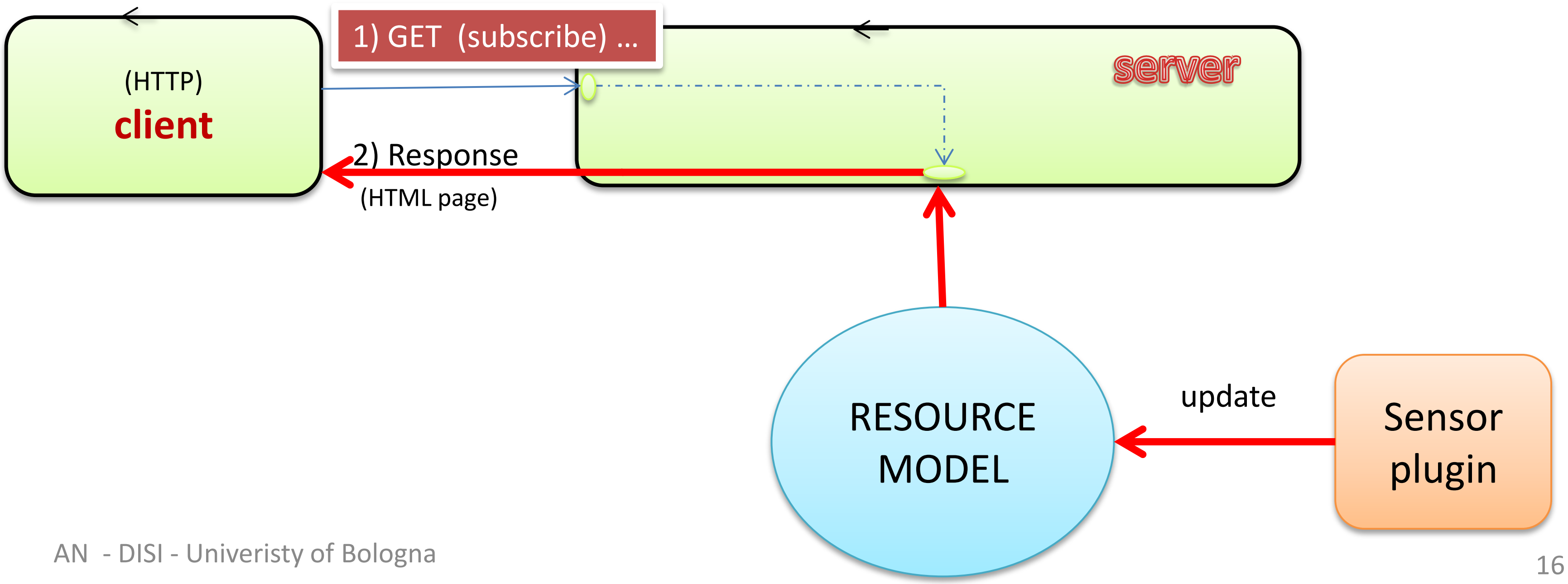
usercmd : usercmd(N)


usercmd : usercmd(X)
Node Server
Appl Actor
GET, POST, PUT, DELETE
HTTPClient
Browser
httpinfo: httpinfo(X)

WoT



websocket


(HTTP)
client
1) GET (subscribe) …
server
2) Response
(HTML page)
RESOURCE MODEL
update
Sensor plugin