

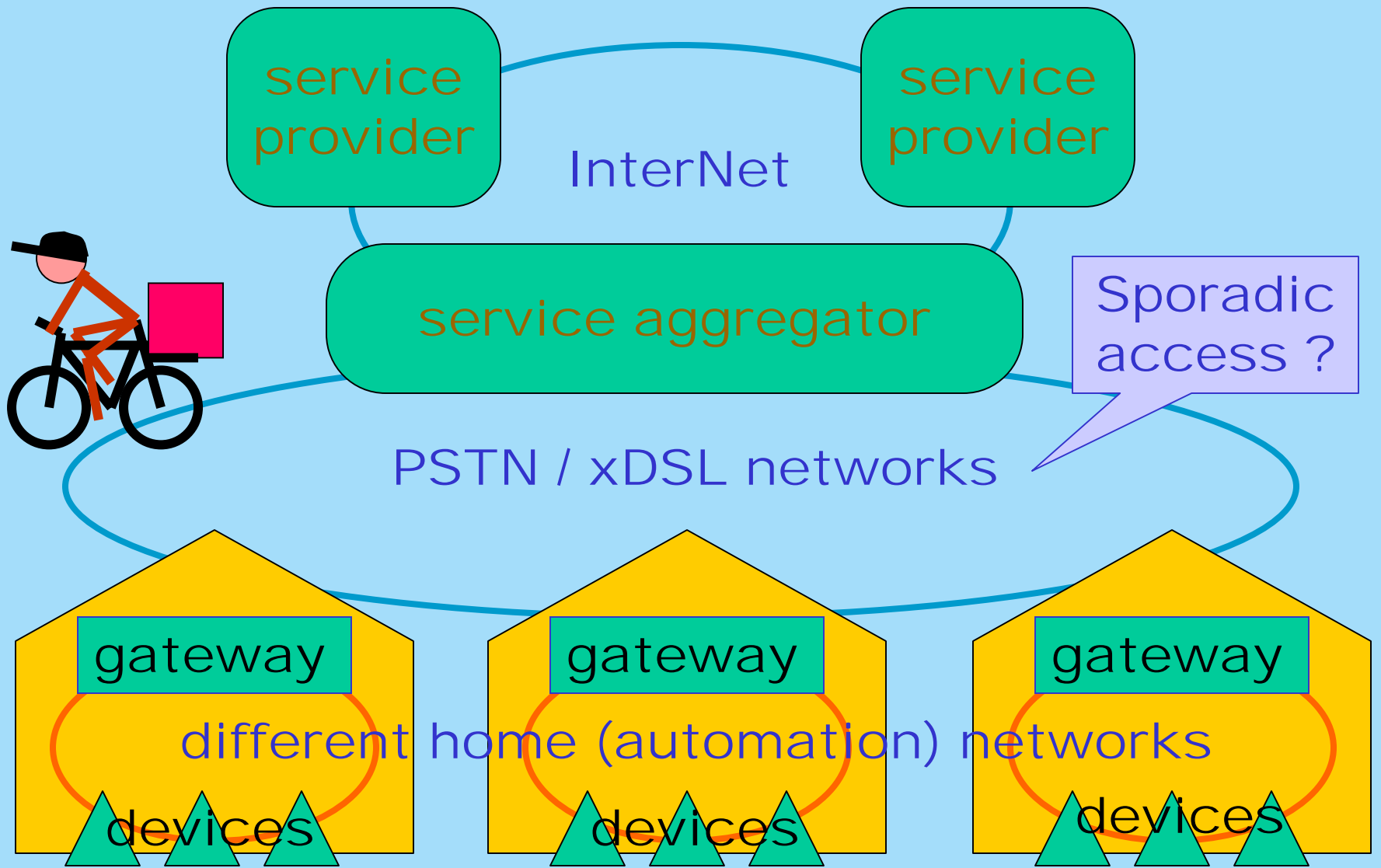


# Smart Metering in Smart Homes

Dr Paul KOPFF

Electricité de France

# General architecture

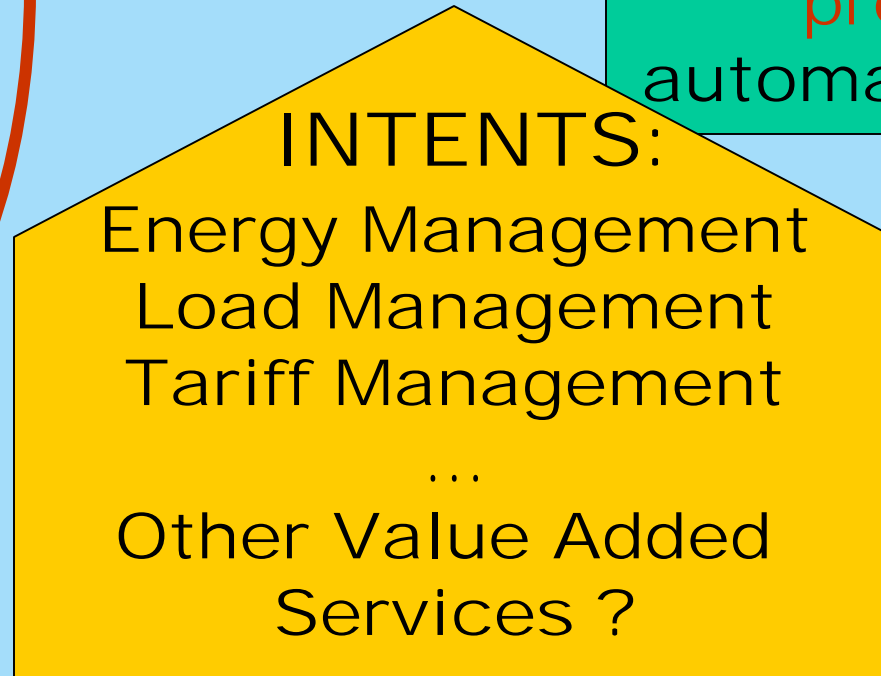
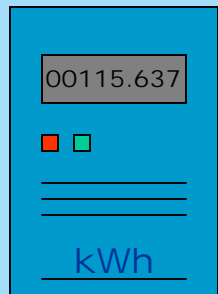


Our initial perspective

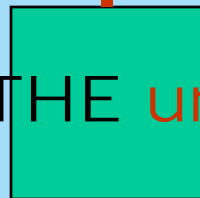
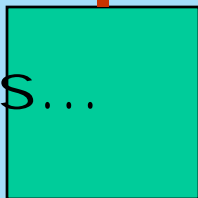
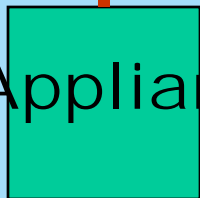
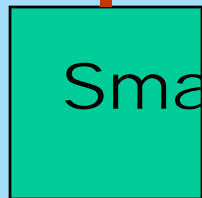
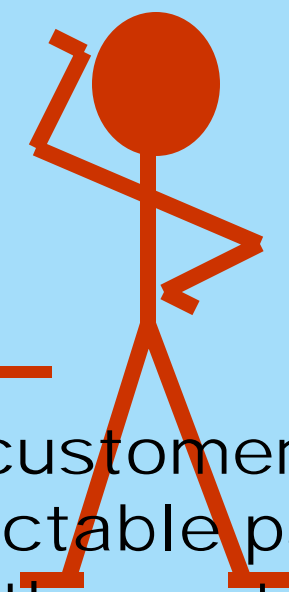


predefined data from utilities

Smart Power Meter



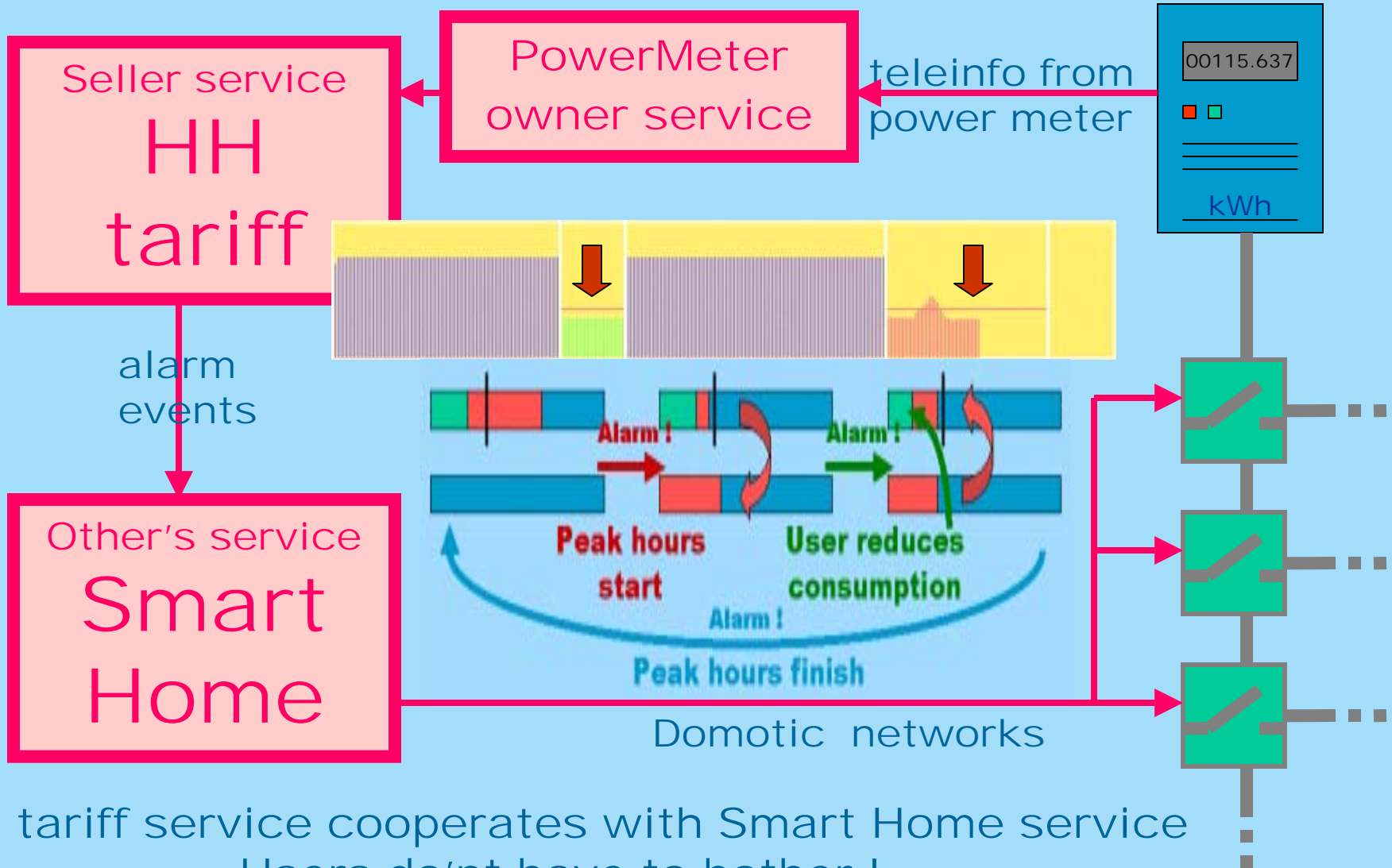
I will disturb any preprogrammed automation scheme



Smart Appliances...

THE unpredictable part of the system

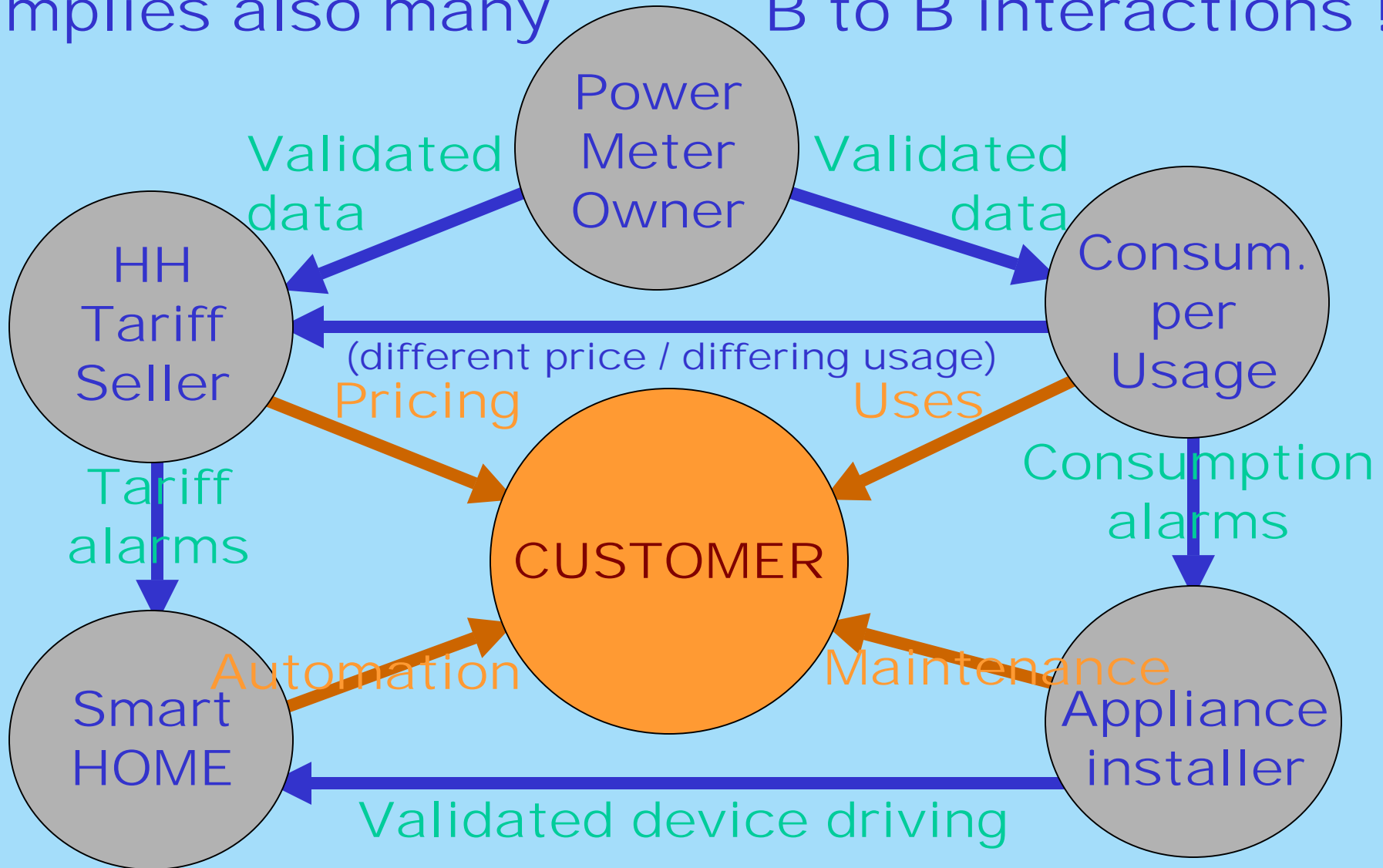
# A simple example



HH tariff service cooperates with Smart Home service  
Users do'nt have to bother !

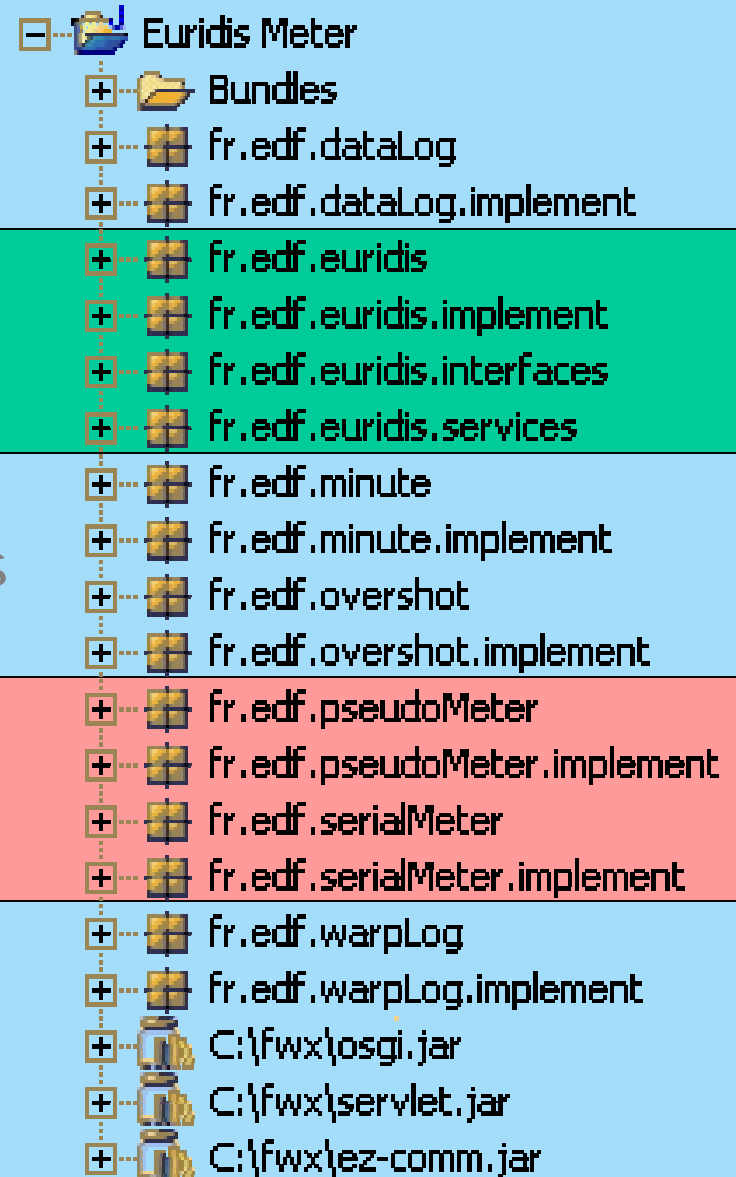


implies also many B to B interactions !





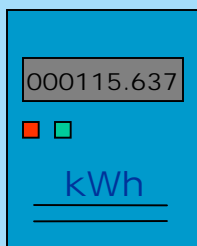
The power meter owner will provide the basic power metering services !



EURIDIS Bundle  
for the basic services  
and interfaces definition

Others : basic test bundles

2 implementation bundles  
- Serial Meter (RS232 port)  
- PseudoMeter (simulation)



EURIDIS TELEINFO  
Serial port : 1200 bauds  
1 data frame per sec.



## Services

- fr.edf.euridis.services
  - DataService.java
  - GivenLoad.java
  - GivenUse.java
  - GivenWarp.java
  - IndexWatcher.java
  - LoadService.java
  - OvershotWatcher.java
  - TariffWatcher.java
  - UseService.java
  - WarpService.java

## Interfaces

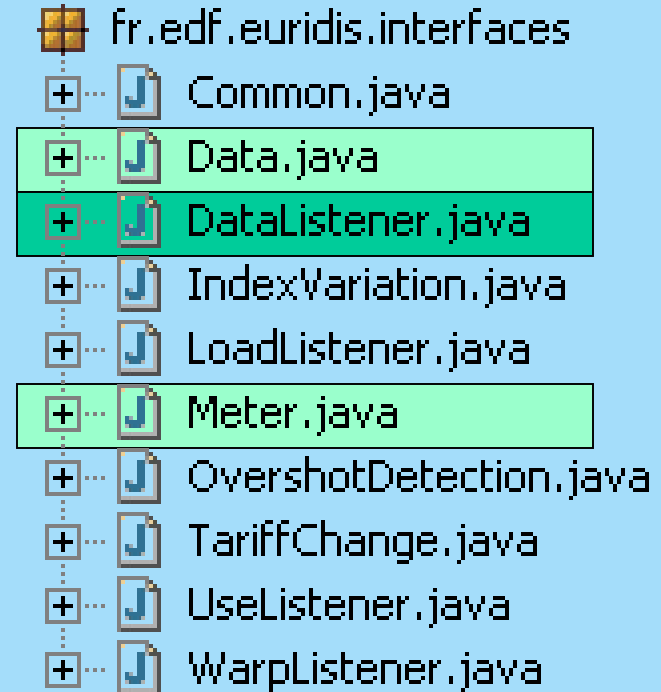
- fr.edf.euridis.interfaces
  - Common.java
  - Data.java
  - DataListener.java
  - IndexVariation.java
  - LoadListener.java
  - Meter.java
  - OvershotDetection.java
  - TariffChange.java
  - UseListener.java
  - WarpListener.java



## Services



## Interfaces



Subscribing to DataService  
the client gets **all the data**

from a DataListener through the Data interface  
the Meter interface is for the implementations  
of various power-meters and simulations



```
package fr.edf.euridis.services ;
```

```
public interface DataService {  
    public void addDataListener(DataListener dl) ;  
    public void removeDataListener(DataListener dl) ;  
}
```

```
package fr.edf.euridis.interfaces ;
```

```
public interface DataListener {  
    public void available(Data d) ;  
}
```

New data frame arrived



available(Data d)



## Services

- fr.edf.euridis.services
  - DataService.java
  - GivenLoad.java
  - GivenUse.java
  - GivenWarp.java**
  - IndexWatcher.java
  - LoadService.java
  - OvershotWatcher.java
  - TariffWatcher.java
  - UseService.java
  - WarpService.java**

## Interfaces

- fr.edf.euridis.interfaces
  - Common.java
  - Data.java
  - DataListener.java
  - IndexVariation.java
  - LoadListener.java
  - Meter.java
  - OvershotDetection.java
  - TariffChange.java
  - UseListener.java
  - WarpListener.java**

Subscribing to WarpService

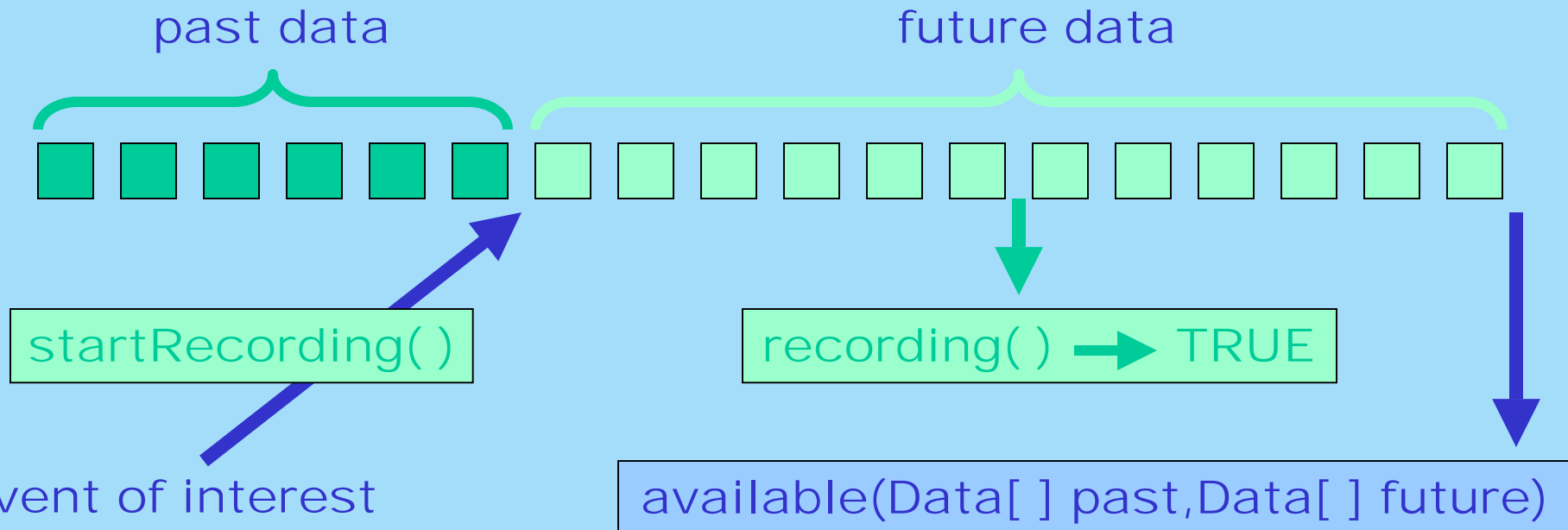
the client gets data around an event of interest  
from a WarpListener for a GivenWarp

this service is generically implemented in the Warp, Weft  
and Loom classes of fr.edf.euridis.implement



```
package fr.edf.euridis.services ;
```

```
public interface WarpService {  
    public GivenWarp add(int past,int future) ;  
    public void remove(GivenWarp gw) ;  
}
```





```
package fr.edf.euridis.services ;
```

```
public interface GivenWarp {  
    public void addWarpListener(WarpListener wl) ;  
    public void removeWarpListener(WarpListener wl) ;  
    public void startRecording( ) ;  
    public boolean recording( ) ;  
}
```

```
package fr.edf.euridis.interfaces ;
```

```
public interface WarpListener {  
    public void available(Data[ ] past,Data[ ] future) ;  
}
```



## Services

- fr.edf.euridis.services
  - DataService.java
  - GivenLoad.java
  - GivenUse.java
  - GivenWarp.java
  - IndexWatcher.java**
  - LoadService.java
  - OvershotWatcher.java**
  - TariffWatcher.java**
  - UseService.java
  - WarpService.java

## Interfaces

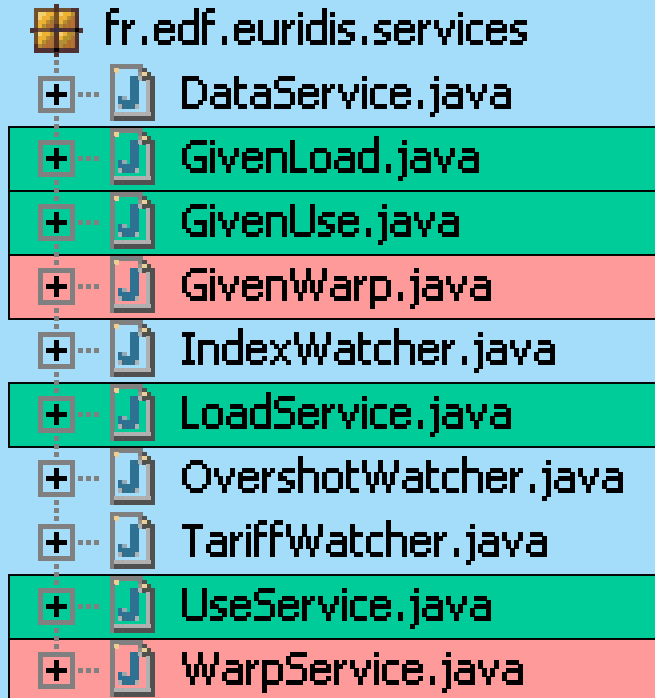
- fr.edf.euridis.interfaces
  - Common.java
  - Data.java
  - DataListener.java
  - IndexVariation.java**
  - LoadListener.java
  - Meter.java
  - OvershotDetection.java**
  - TariffChange.java**
  - UseListener.java
  - WarpListener.java

The events of interest could be simple metering events, like tariff change, overshoot detection, etc.

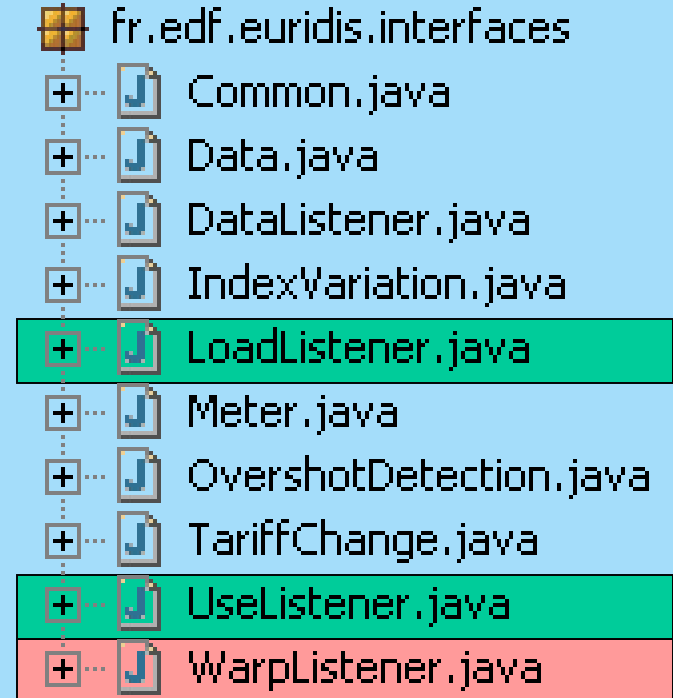
With every Watcher service goes a listener type interface for the client. It may start a given warp...



## Services



## Interfaces



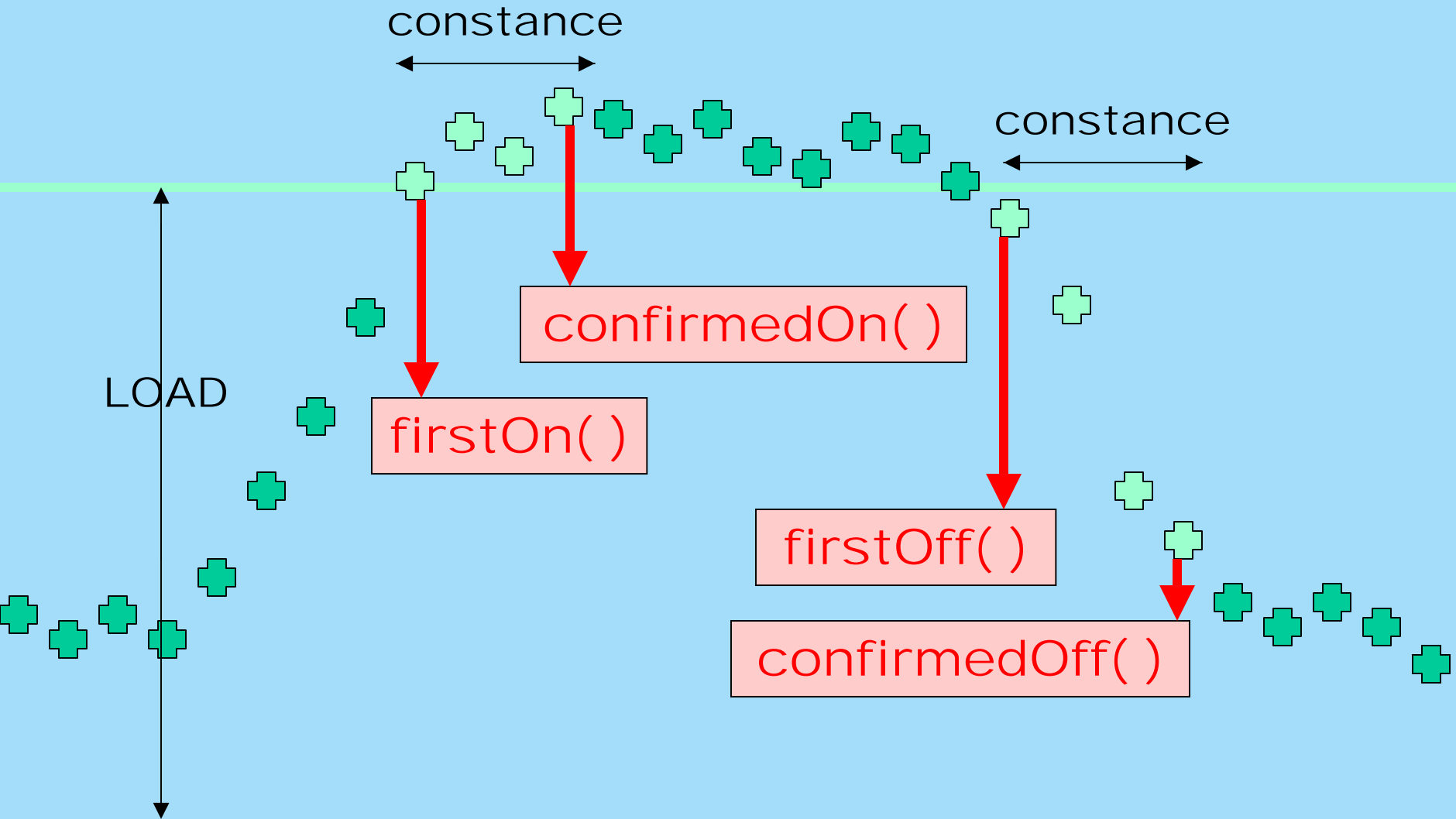
two client configurable events are defined for

- load management client applications,
- per use consumption evaluations etc.

Their configuration follows the same model as the Warp service, with GivenLoad and GivenUse specs.



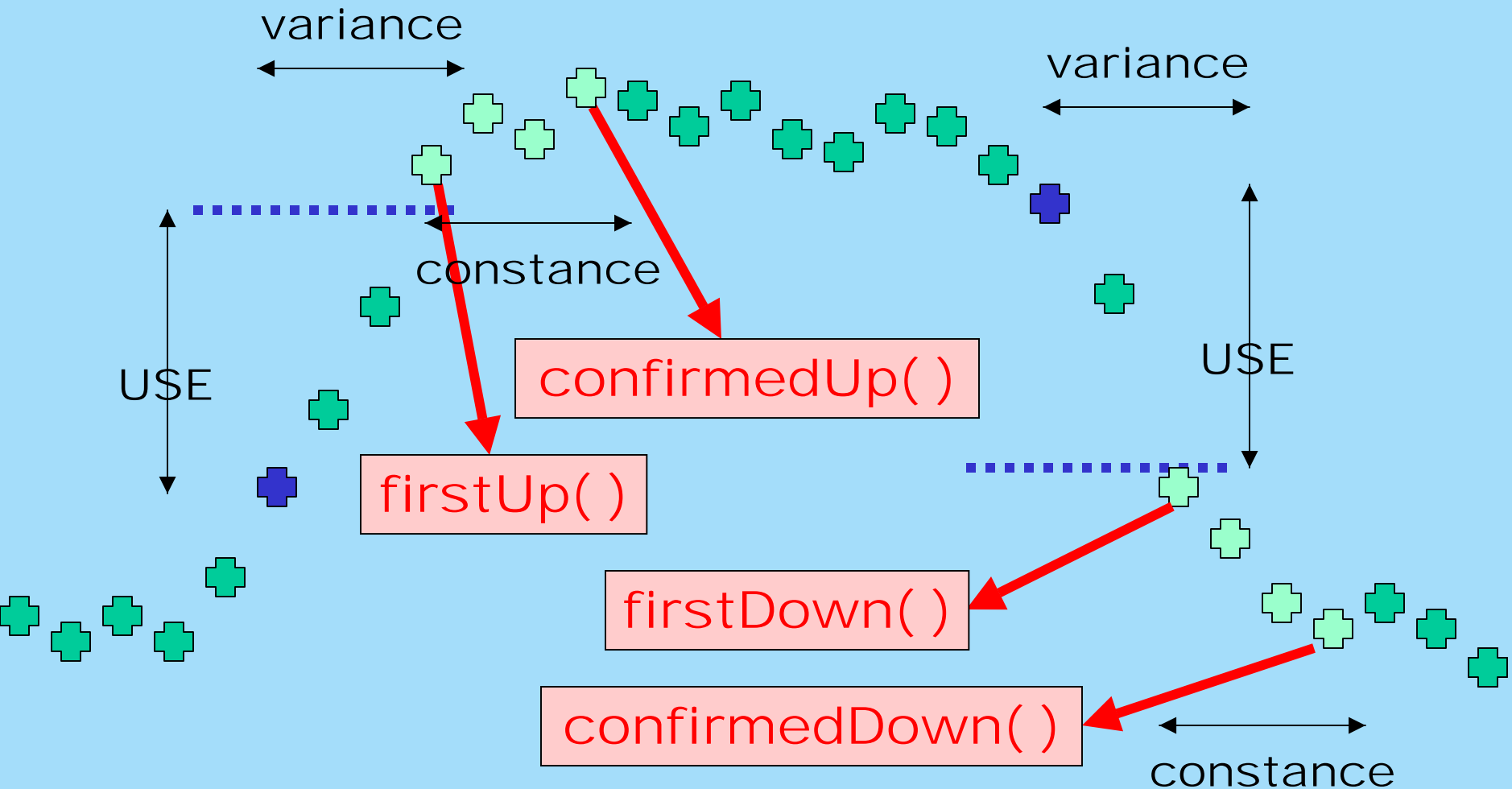
The LoadListeners « hear » four different types of events :



use  
events



The UseListeners « hear » four different types of events :





```
package fr.edf.overshotClient.implement ;
```

```
public class RecordOvershot implements OvershotDetection {  
    private GivenWarp w ;
```

```
    public RecordOvershot( GivenWarp gw ) { w = gw ; }
```

```
    public void triggered(int overAmp) {  
        System.out.println("Overshot of "  
                            + String.valueOf(overAmp)  
                            + " A detected");
```

```
        if (! w.recording( ))  
            w.startRecording( ) ;
```

```
    }
```

```
}
```



```
package fr.edf.overshotClient.implement ;
```

```
public class LogOvershot implements WarpListener {
```

```
    public void available(Data[ ] before, Data[ ] after) {
```

```
        if (before != null) {
```

```
            System.out.println(String.valueOf(before.length)  
+ " samples before the overshoot");
```

```
        }
```

```
        if (after != null) {
```

```
            System.out.println(String.valueOf(after.length)  
+ " samples after the overshoot");
```

```
        }
```

```
    }
```

```
}
```



## Successful services will be REAL TIME

They will be immediately responsive to all kinds of signals and also to the perturbative customer (who will not be bothered)

## The service providers community will dynamically manage rich local resources

In order to optimize the use of servers and access resources thus promoting cheaper offers to more potential customers

## The services running in the gateways will also cooperate (B to B interactions)

To achieve this, they will have to support a common standard tailored as a model of services : current best fit, OSGi !



Questions?