# LSD and the ODA framework

## Empirical Modelling

empirical = based on observation and experiment

empirical = given in experience

*modelling* because it is intended to support an activity that relies upon establishing a correlation between the experience offered by the computer and some external experience moment by moment ... and thus is (as if) carried out in a situation in which there is a referent

## Fundamental concepts 1

an observable

some feature of a situation to which a value or status can be attributed. Empirical procedures and conventions are involved in identifying a particular observable and assigning its value. Not all the observables associated with a situation need be present in a particular state.

## Fundamental concepts 2

an agent

a family of observables whose presence and absence in a situation is correlated in time, that is typically deemed to be responsible for particular changes to observables. All changes to the values of observables in a situation are typically construed as due to actions on the part of agents.

## Fundamental concepts 3

a dependency

a relationship between observables that pertains in the view of a particular agent. It expresses the observation that when the value of a particular observable x is changed, other observables (the dependants of x) are of necessity changed in a predictable manner as if in one and the same action. The changes to the values of x and its dependants are indivisible in the view of the agent. That is: no action or observation on the part of the agent can take place in a context in which x has changed, but the dependants of x have yet to be changed.

## Fundamental concepts 4

The identification of observables, dependencies and agents and all matters concerning their integrity and status is an informal empirical activity (“What EM is”)

It is arguably an activity that is implicit in all system construction, whatever development method or programming paradigm is used

# LSD

An LSD account for an agent classifies observables:

**oracle** - an observable to which it responds
**state** - an observable that it owns
**handle** - an observable conditionally under its control
**derivate** - an observable determined by a dependency

+ **protocol** = list of privileges of the form

*enabling condition -> sequence of actions*
*where an action is a redefinition,*
*an agent invocation or a deletion*

# Room user LSD account

```
agent room_user

state          loc, armslength
oracle         door_open
handle         door_open
derivate
   nearDoor = dist(loc, door_hinge) < armslength
protocol
   nearDoor -> door_open = true
   nearDoor -> door_open = false
```

Can optionally give types to observables: bool / real etc

# Architect LSD account

```
agent architect

state
oracle door_hinge, door_width, door_type, door_cost
handle door_width, door_open, door_hinge
derivate
  poss_door is (door_unit_cost * door_width < door_cost)
                         AND door_width > reg_door_size
protocol
   TRUE -> door_width++
   TRUE -> door_width--
```

# Automatic door LSD account

```
agent auto_door

state          hinge, lock, width, aperture
oracle         door_open
handle         aperture
derivate
   door_open is aperture > 0
protocol
   aperture > 0 -> aperture--
```

# Central heating LSD account

```
agent boiler

state          boilerOn, currentBoilerTemperature
oracle         desiredBoilerTemperature
handle         currentBoilerTemperature, flameNeeded
derivate
   needsToHeat = currentBoilerTemperature <
   (desiredBoilerTemperature - tolerance)
protocol
   needsToHeat -> flameNeeded = true
```