PHILIPS

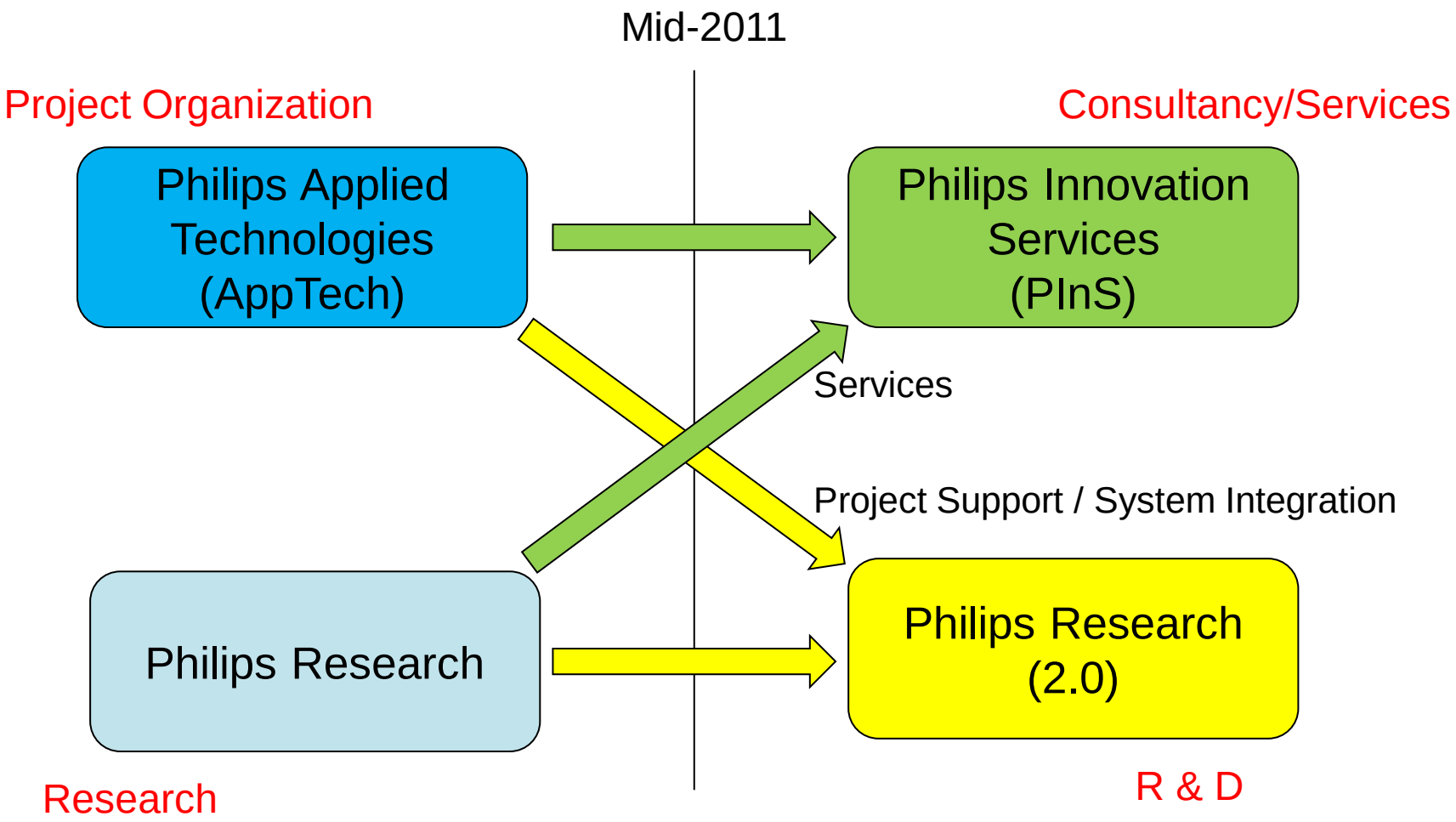
sense and simplicity

Proton Therapy System

Patrick Bonné, Philips Research
SASG June 2012

PHILIPS

Restructuring of Philips Corporate Technologies



Project Context

- External customer
- Proton Therapy System (Medical Device, Class C – IEC:62304)
- (Single) software component not stable
  - Working for several years with multiple teams
  - No accreditation → facility may not be used to treat patients

PHILIPS

sense and simplicity

Simulator based development & testing of a sub-system

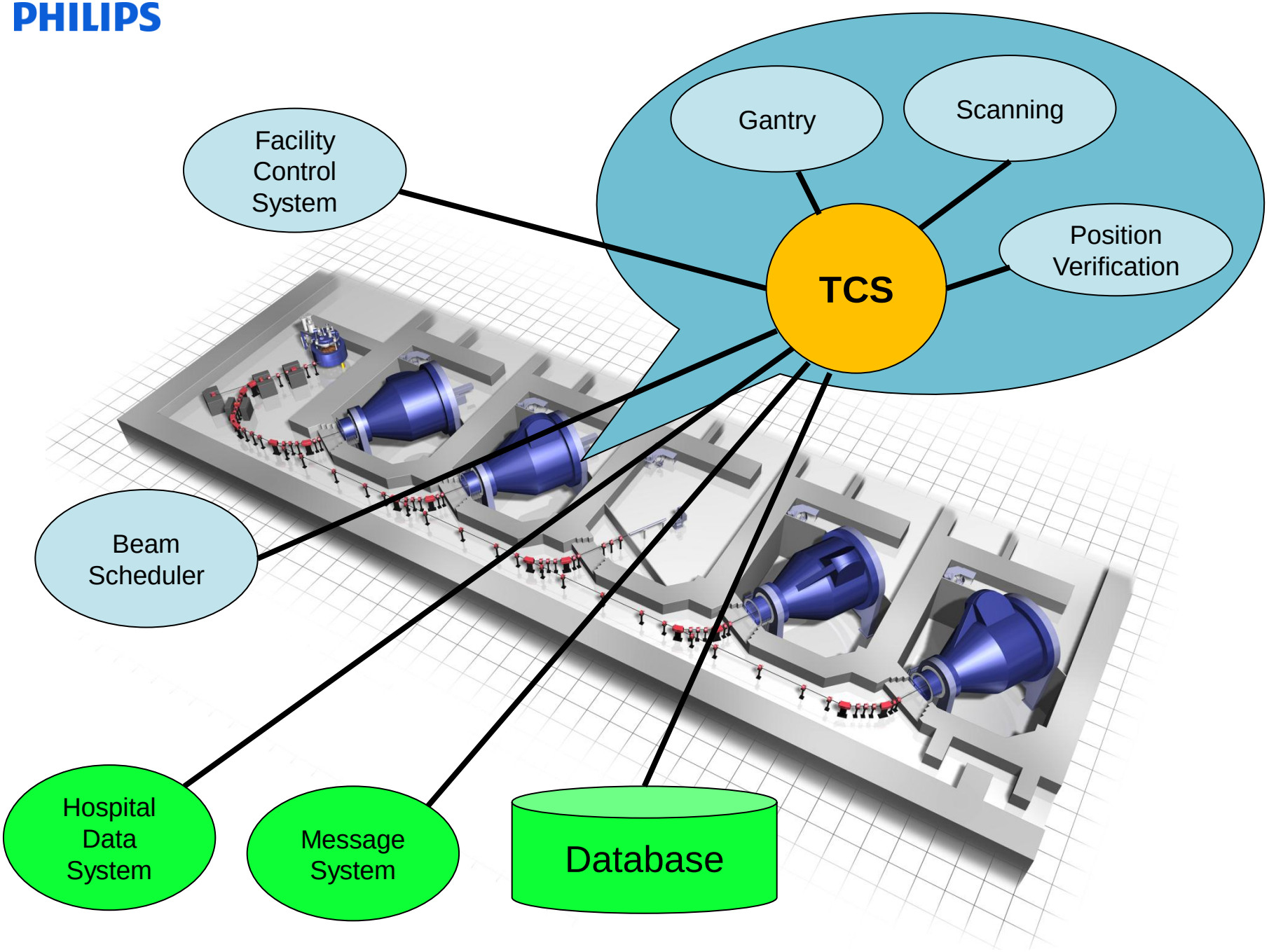
Patrick Bonné, Philips Research
SASG June 2012

Goal:

- Share some backgrounds on the simulation based development/test environment setup that has become a “best practice” in Philips Research.

PHILIPS

PHILIPS


Facility Control System
Gantry
Scanning
Position Verification
TCS
Beam Scheduler
Hospital Data System
Message System
Database

PHILIPS

Why simulators

- Real product not available
  - intelligent stubbing (development / integration / testing)

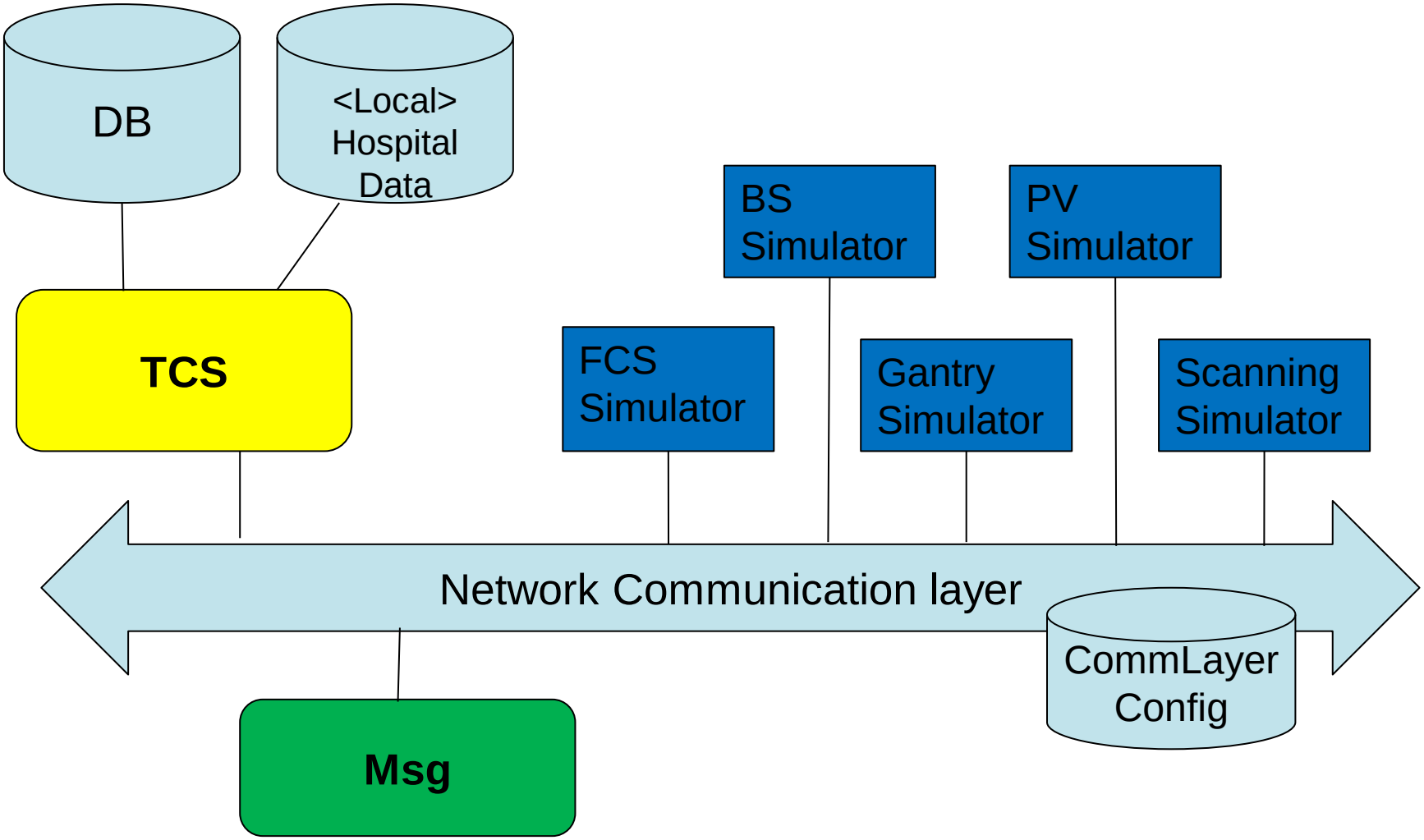
- Interface specification incomplete (actual status exists in code)
  - Qualitative good interface documentation
  - Used as acceptance criteria (→ gap management)

- Unbalanced requirements of TCS
  - User observable behavior

- Need to exercise special situations
  - You can’t just stop a cyclotron
  - Playing around with protons is dangerous

PHILIPS

Development Environment

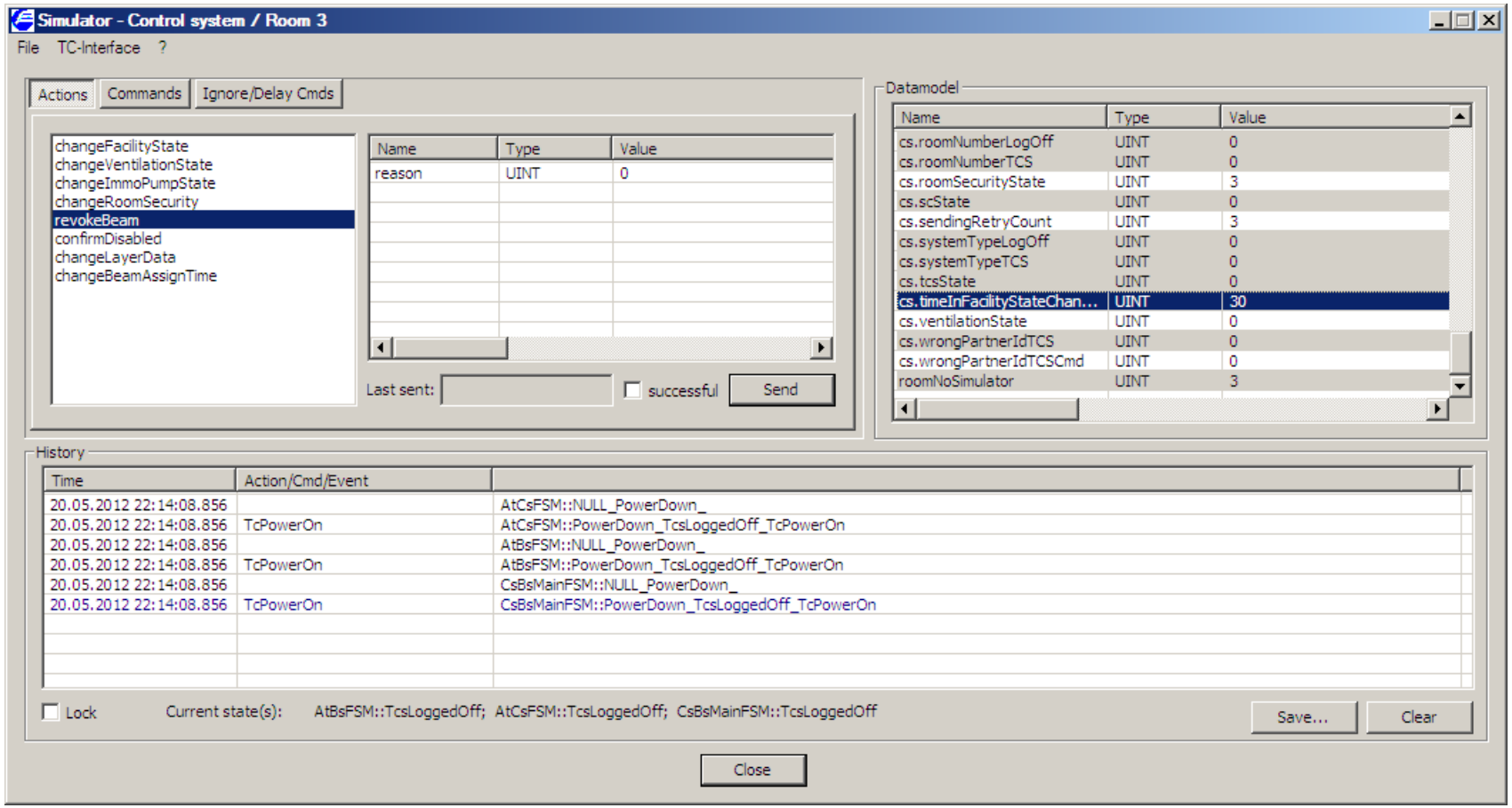


Simulator requirements

- Must display user observable state & behavior
- Option to enable automatic scenario support
- Possibility to alter timing and “simulator generated data”
- Controls to trigger exceptional situations
- Controls to trigger “out-of-specification” behavior
- Logging
- API: execute action, set/get properties, wait for states

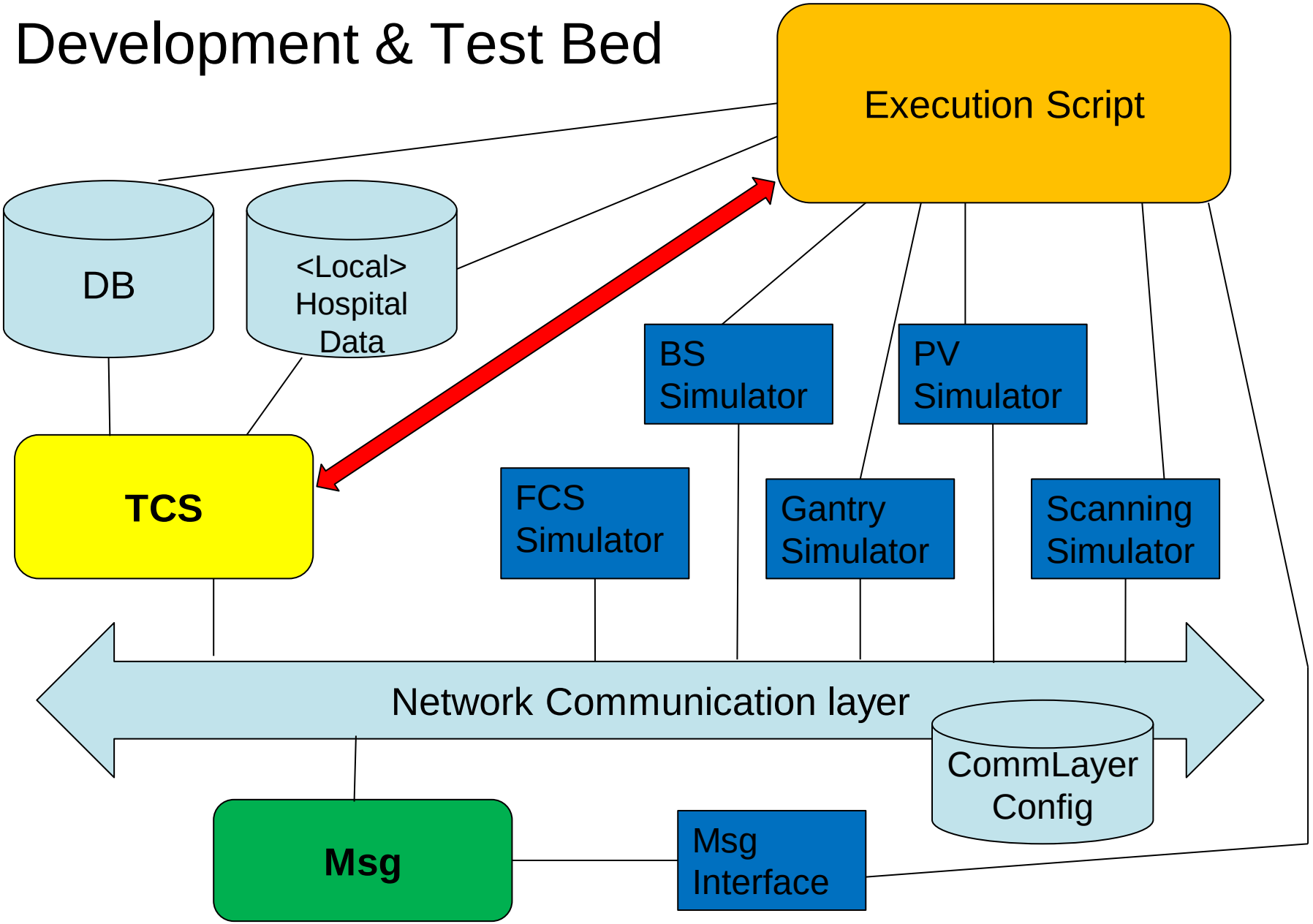
PHILIPS

Simulator: appearance

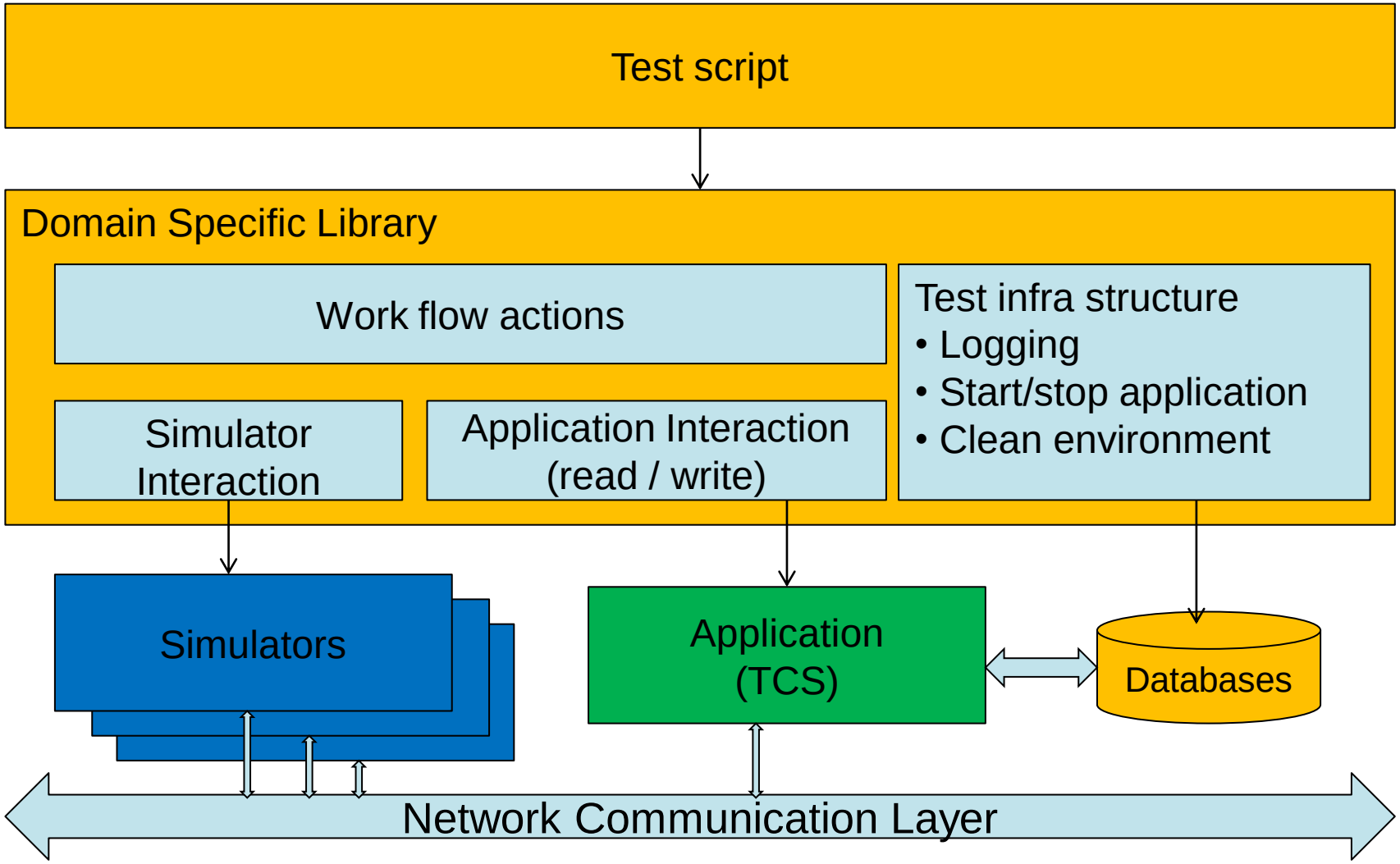


PHILIPS

Development & Test Bed



Automatic test architecture

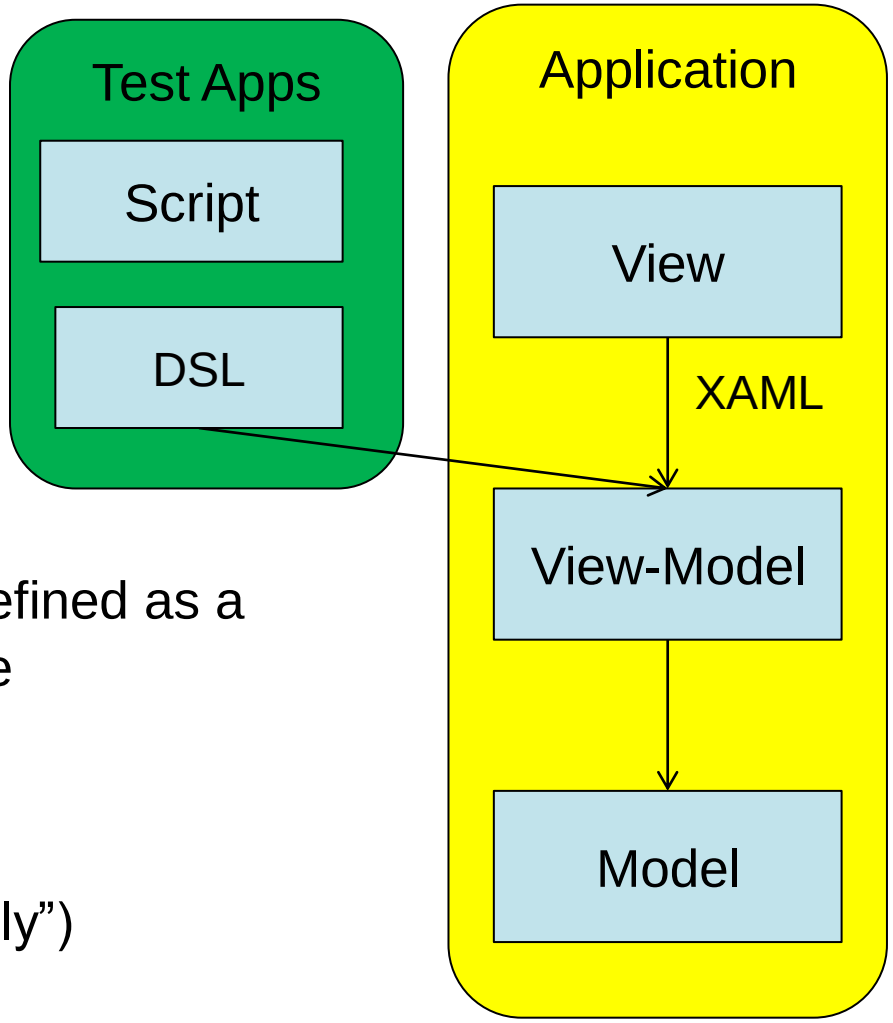


Application Test Interface

- *GUI Record/Playback tool (Rational Robot)*
  - Difficulties in recognition of screen elements
  - Script execution very sensible for external (window) events
  - Script language and development environment not state of the art
  - Tests can only use visible information on the screen
    - ➢ Difficult to check the absence of elements

Application Test Interface (2)

- *Test Interface*
  - Use the MVVM paradigm (based on MS-WPF)
  - Views do not have any additional intelligence
  - The View-Model interface is defined as a remoting interface, used by the Domain Specific Library
  - Used for functional tests (appearance is tested “manually”)



Summary

- Discussed some aspects used to set-up a generic (automated) development/test environment based on simulators
  - Balanced requirements
  - Simulate 2nd order and “out-of-specification” behavior
  - Automatic basic workflow
  - Logging
  - Provide an API to all elements of the execution environment
    - Application under development
    - Simulators
  - “Domain Specific Library” to control the environment

Concluding

- We have applied this set-up successfully in multiple projects (we think it's worth the investment)

- The particular Proton Therapy Centre is currently operational and patients are being treated with the system


PHILIPS