SEN9110 Simulation Masterclass
Lecture 09: Real-time Simulation and Emulation

Alexander Verbraeck, a.verbraeck@tudelft.nl

Parts of the course slides are based on
research of Yvo Saanen and Corne Versteegt, TU Delft

Agenda

- Extended use of Simulation

  Discussed in PhD thesis of Yvo A. Saanen, An Approach for Designing Robotised Container Terminals. TU Delft, 2004. (see TU Delft repository if you are interested)

- Real-time simulation (with eM-Plant)

  See background paper: C. Versteegt, A. Verbraeck. “Evaluating the design of fully automated logistic systems using a combination of simulation, emulation, and prototyping”. In: E. Yücesan, C.-H. Chen, J.L. Snowdon and J.M. Charnes (Eds.). Proceedings of the 2002 Winter Simulation Conference, San Diego, 8-11 December 2002. pp. 1659-1666. (plus video on Brightspace)

- Emulation (with DSOL and eM-Plant)

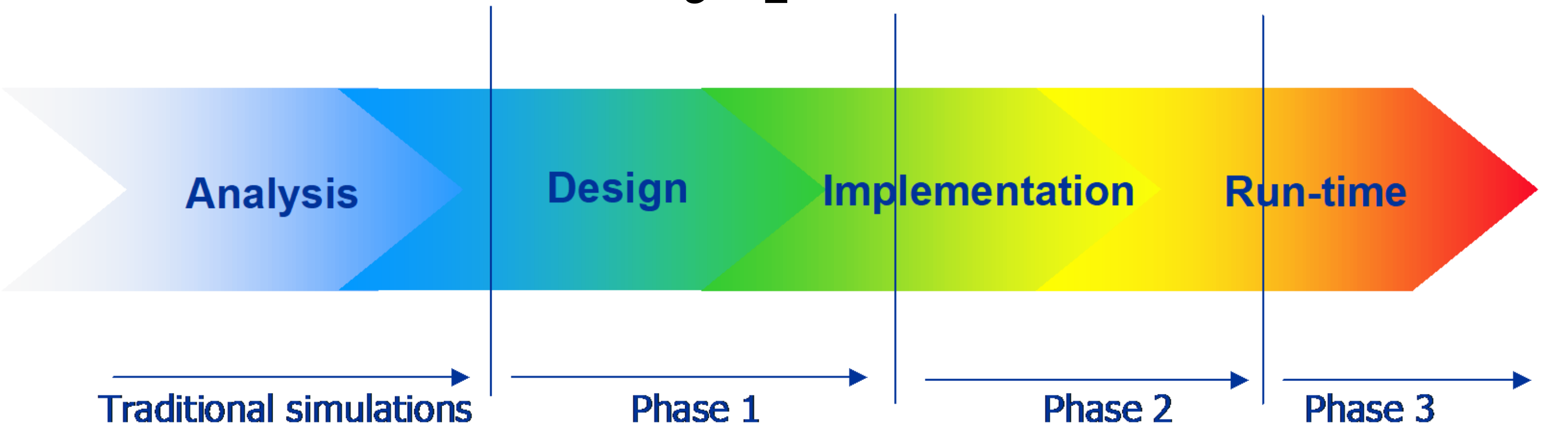
  Paper: Peter H.M. Jacobs, Alexander Verbraeck, William Rengelink. Emulation with DSOL. In: M.E. Kuhl, N.M. Steiger, F.B. Armstrong, and J.A. Joines, (Eds.). Proceedings of the 2005 Winter Simulation Conference. IEEE, 2005. pp. 1453-1462.

1.

Extended Use of Simulation

Discussed in PhD thesis of Yvo A. Saanen, An Approach for Designing Robotised Container Terminals. TU Delft, 2004. (see TU Delft repository if you are interested)

The evolutionary path of simulation



- Using a single simulation library during the whole engineering process:
- Analysis: 'what if' questions and dimensioning of the system
- Design: testing detailed design concepts
- Implementation: testing implemented parts of the control system
- Run time: real-time decision support and ex-post analysis

Benefits of extended use of simulation

Why?

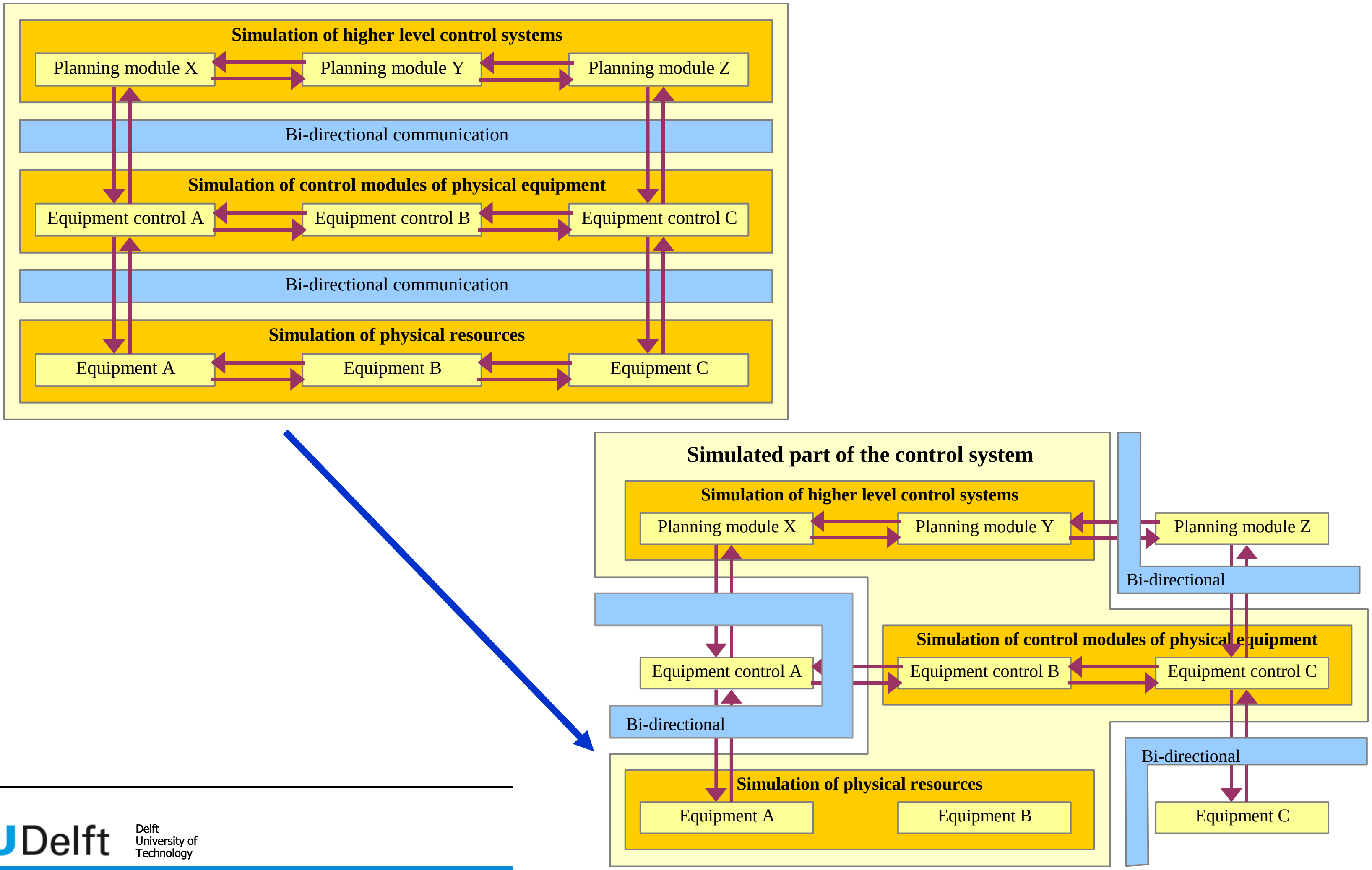
- Faster development of complex systems due to immediate feedback
- Ability to test modules in complete environment under a wide range of circumstances
- Easier testing and evaluation of technical alternatives
- Less cumbersome implementation
- Extra functionality in runtime phase
- Reduction of double effort in total engineering process by re-use software components

Possible threats

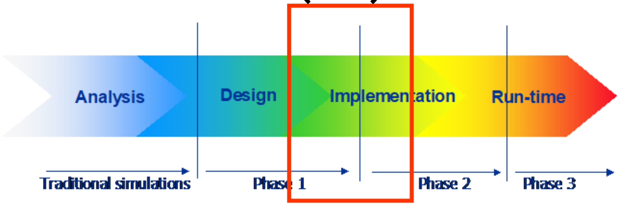
Why not?

- Simulation versus real application
- Different approach in terms of reduction, conceptualisation, et cetera
- Different objectives and therefore not always done with the same perspective or by the same organisation
- More effort in building the simulation model

Model structure:
design -> implementation (1)



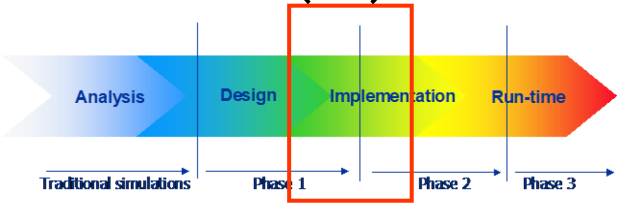
From design to implementation (2)



Applications:

- Testing of production software as part of the whole system, which is simulated:
  - under a wide range of (exceptional) circumstances
  - in collaboration with other implemented parts
- Evaluating performance of production software in terms of speed, robustness and effectiveness
- Verification of production software by advanced debugging and animation
- Pre-structuring applications and clearly defining functionality of components

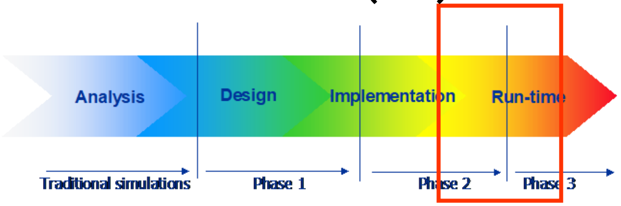
From design to implementation (3)



Requirements:

- Open software (access to databases, other programs, other computers over a network)
- Clearly defined interfaces
- Modular model structure (no interlaced components)
- Robust external information exchange (sequence, accessibility, re-entrancy watch, speed)
- Synchronization of distributed components

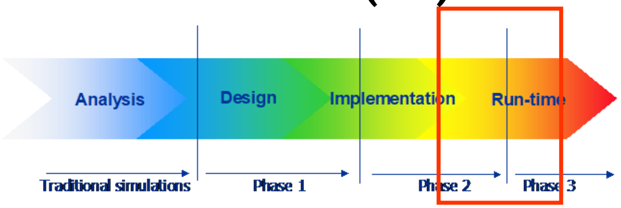
From implementation to runtime (1)



Applications:

- Replay of errors occurred without disturbance of ongoing operations and analysis of causes
- Anticipation on problems by forward play of simulation (e.g. deadlocks, local peaks)
- Real-time support of choice of alternative control strategies (priority setting, loading/unloading strategies)
- Evaluation of fine-tuning changes (lay-out, speed of equipment, routing algorithms, order assignment algorithms, recovery strategies, traffic control)

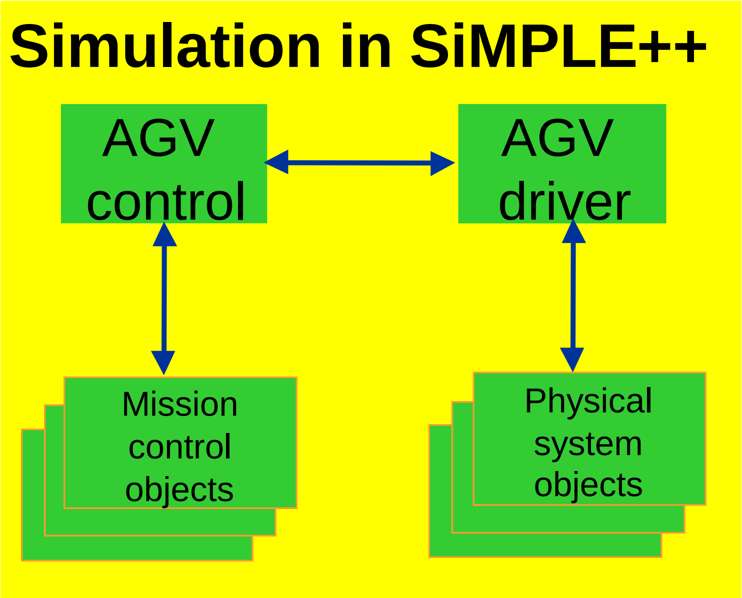
From implementation to runtime (2)



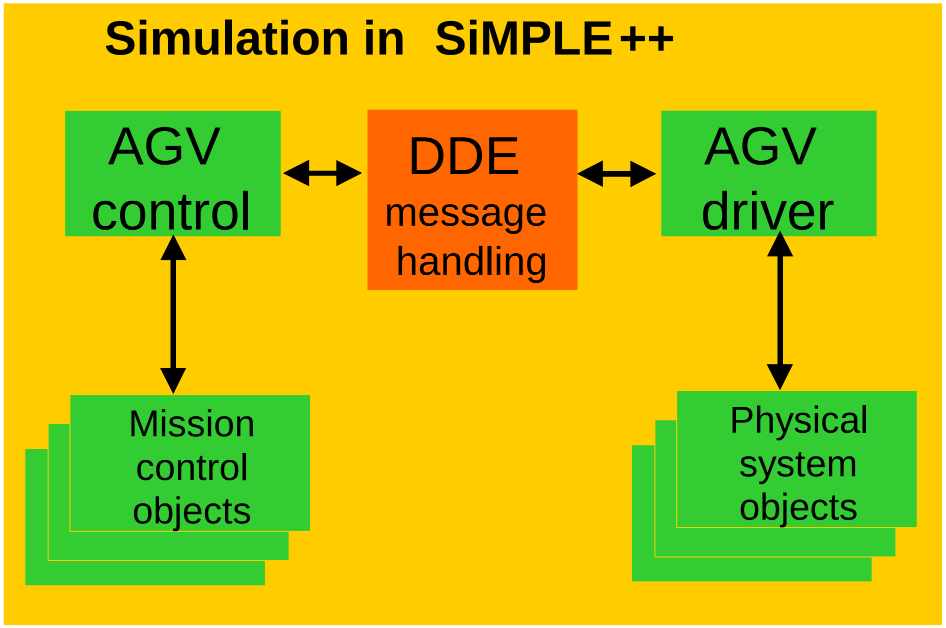
Requirements:

- Starting with simulation as preparation for real planning and control system (PCS)
- Simulation with identical structure and interfaces as real PCS, including explicit communication between modules
- Real time information exchange between PCS and simulation regarding actual events (deviating from planning)
- Logging of events (so that simulation can use logs to replay situations)
- Ability to run simulation real-time as well as faster than real-time (>>10X real-time)

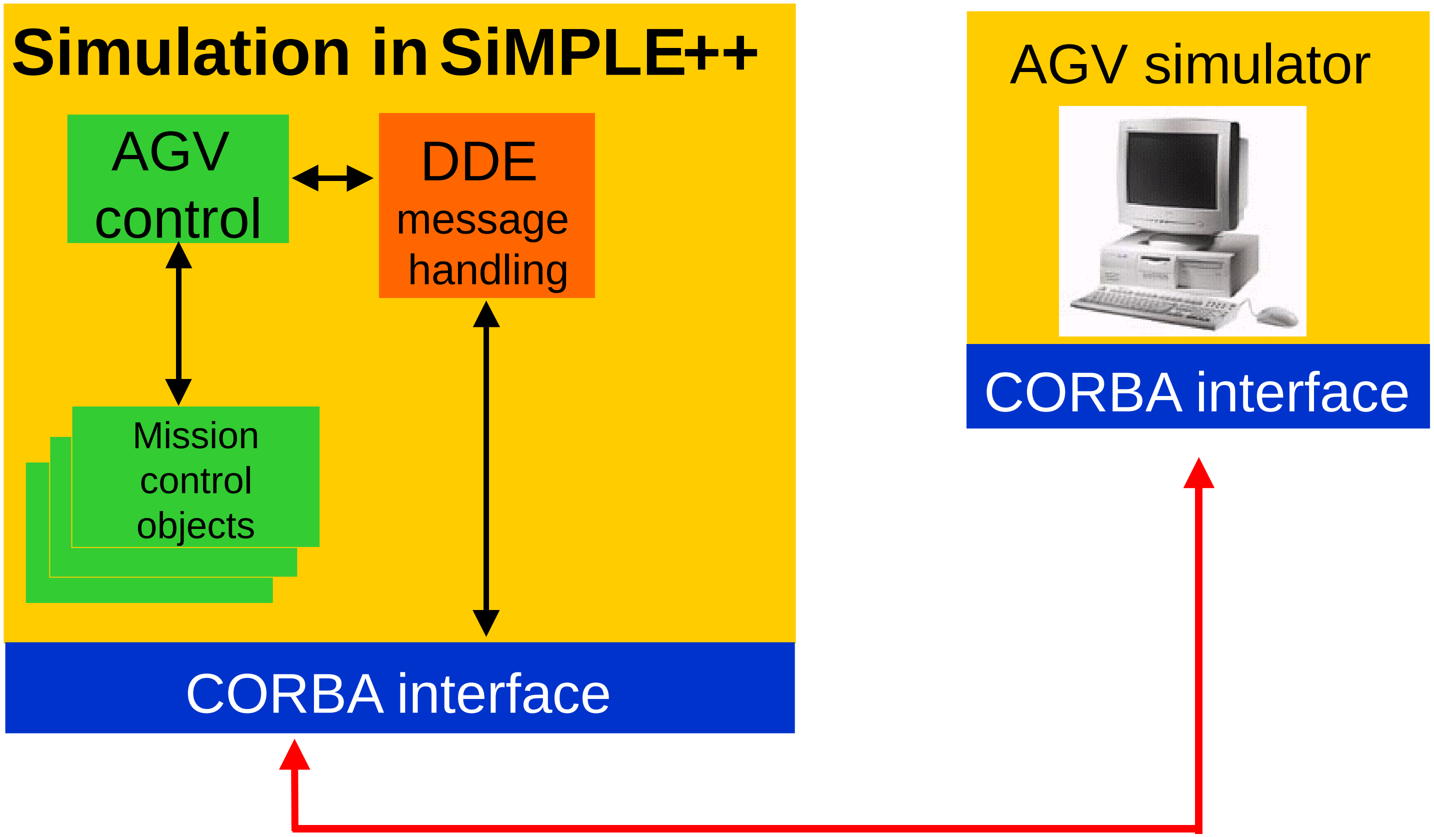
Example: OLS Project AGV control (1)



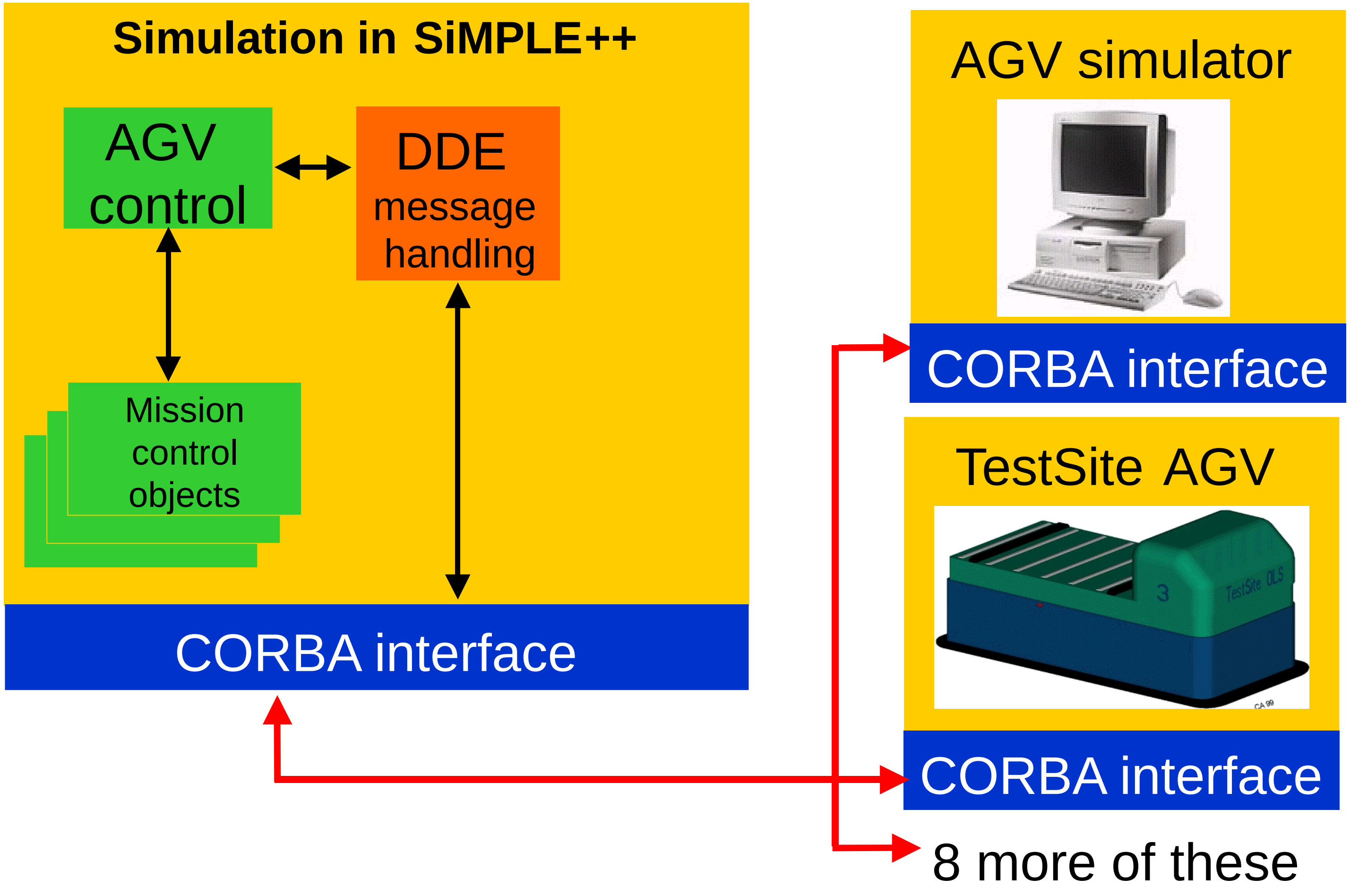
Example: OLS Project AGV control (2)



Example: OLS Project AGV control (3)



Example: OLS Project AGV control (4)



Challenges

- State (see Distributed Simulation class and papers)
- Time (see Distributed Simulation class and papers)

- Extra requirement for time: synchronize with wall clock
- Can we realize that?
- How can we realize that best?

- Extra requirement: data should never lag behind
- Can we realize that?
- How can we realize that best?

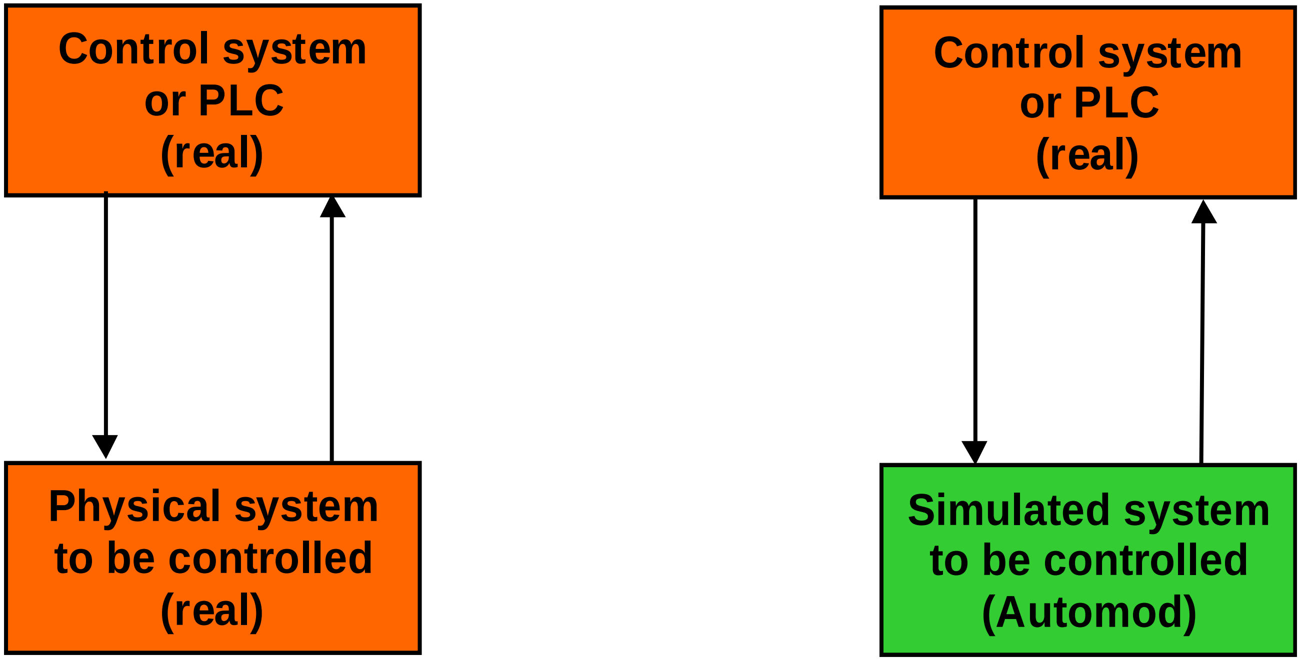
2.

Real-Time Control with Simulation

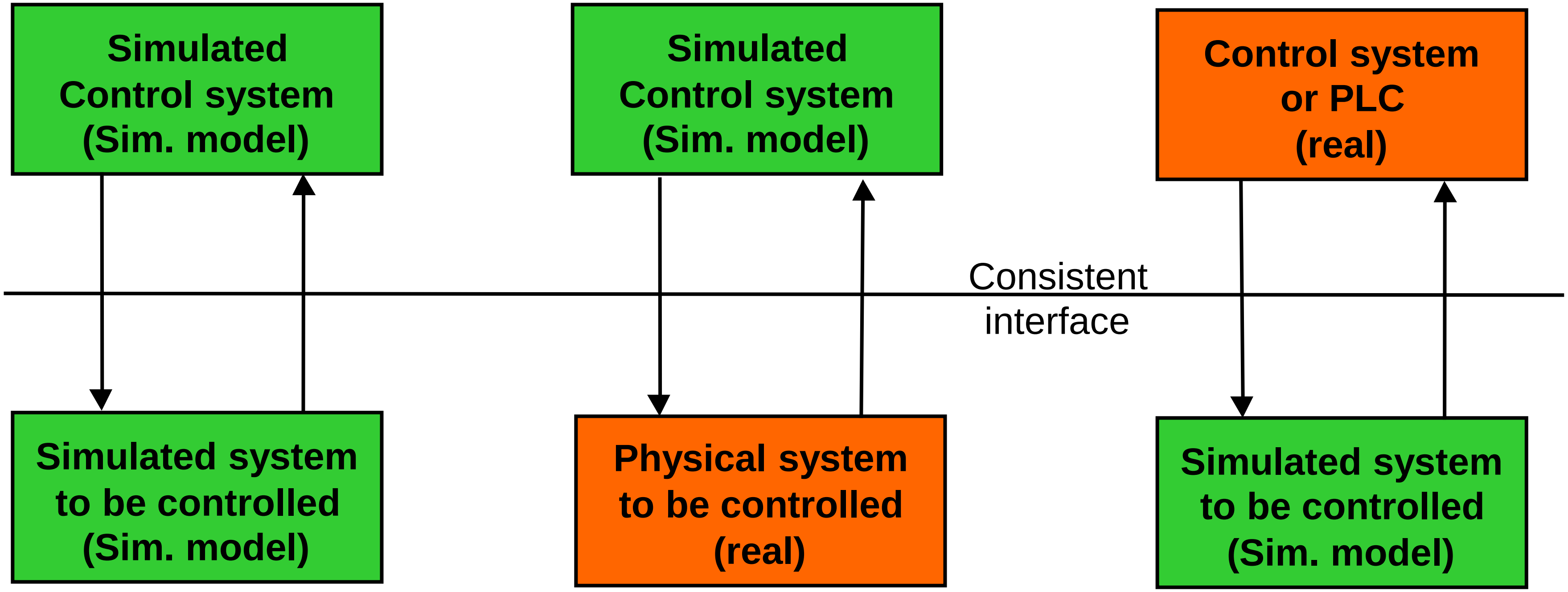
See background paper: C. Versteegt, A. Verbraeck. "Evaluating the design of fully automated logistic systems using a combination of simulation, emulation, and prototyping". In: E. Yücesan, C.-H. Chen, J.L. Snowdon and J.M. Charnes (Eds.). Proceedings of the 2002 Winter Simulation Conference, San Diego, 8-11 December 2002. pp. 1659-1666. (plus video on Brightspace)

Emulation according to Brooks (AutoMod)

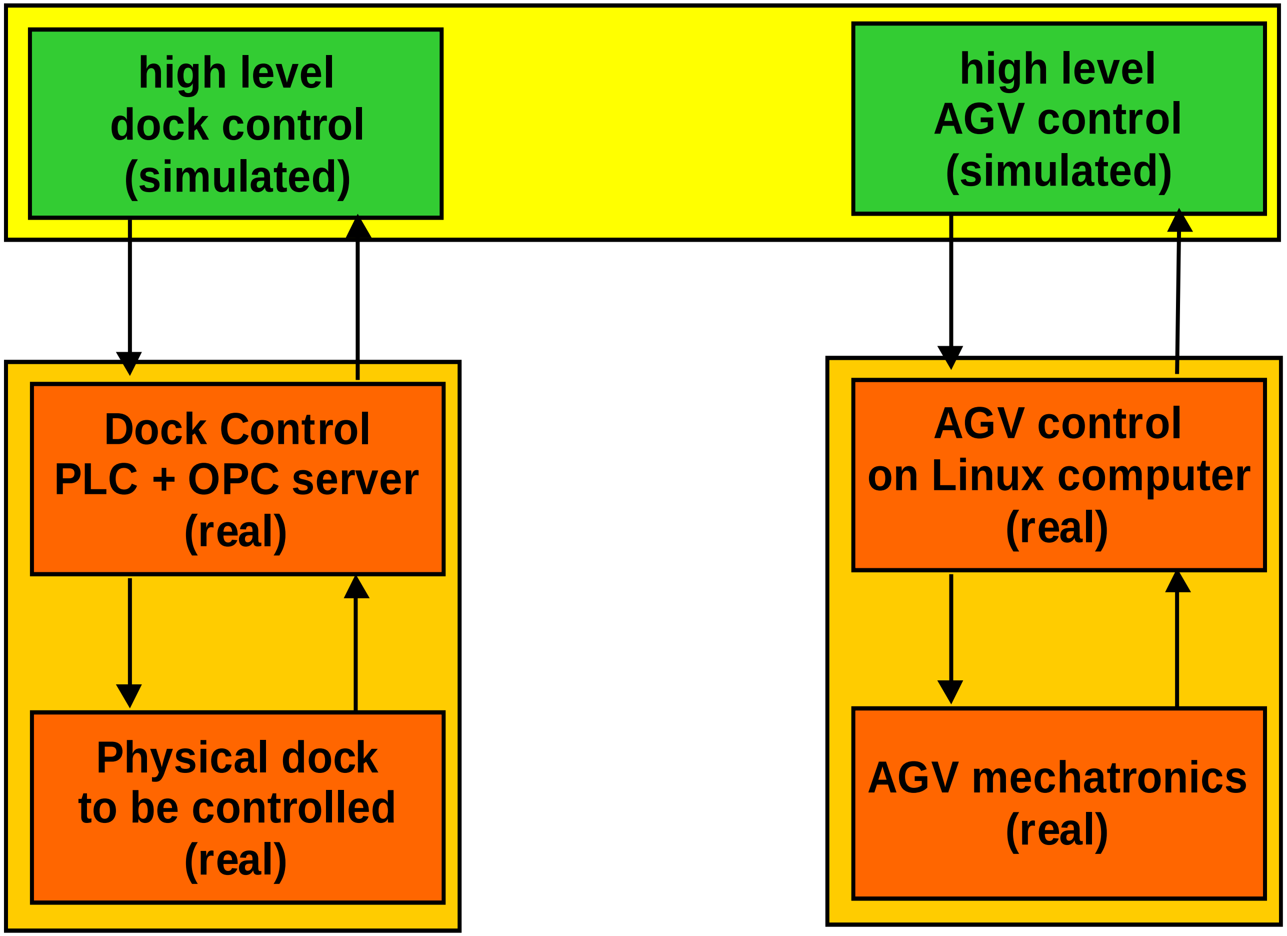
Emulation = testing control systems off-line



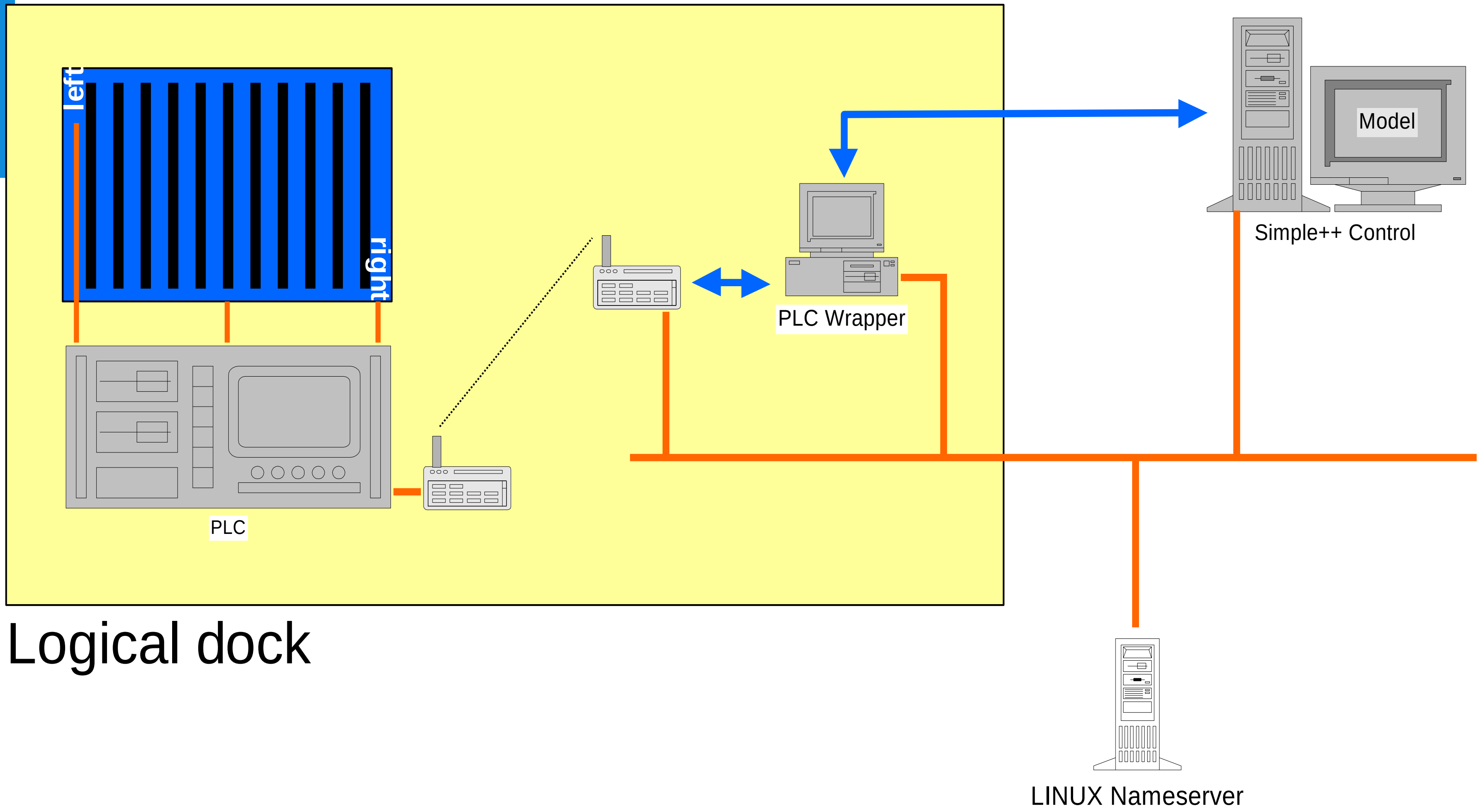
Real-Time simulation possibilities



Real-time simulation in OLS project



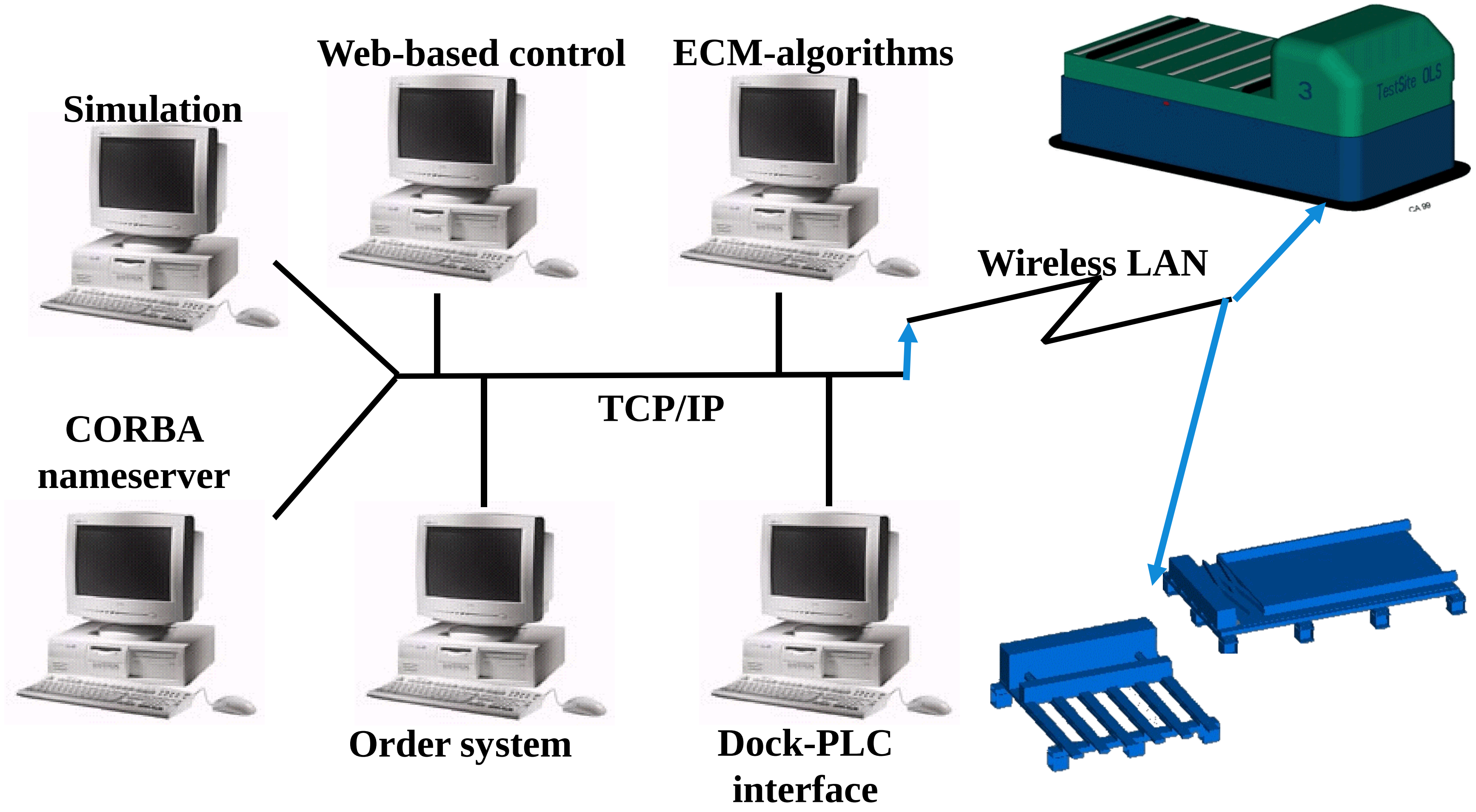
Dock control at OLS TestSite



Logical dock

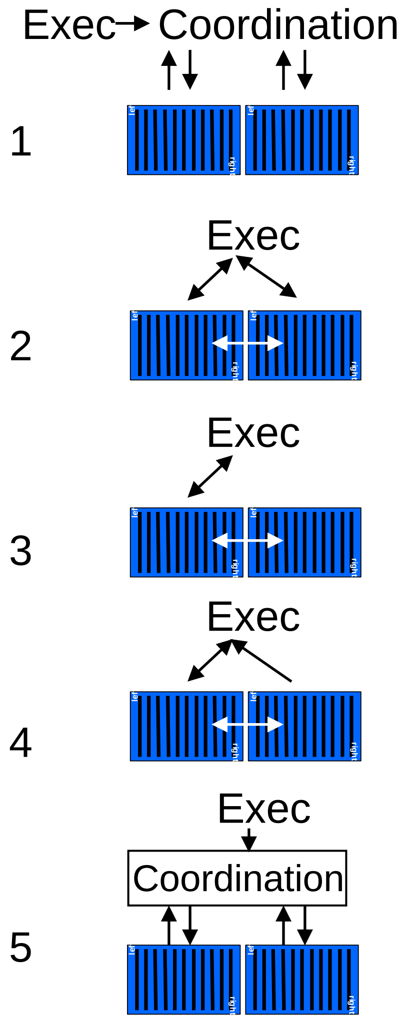
Realized: Communication

Vehicle and controlling system interface



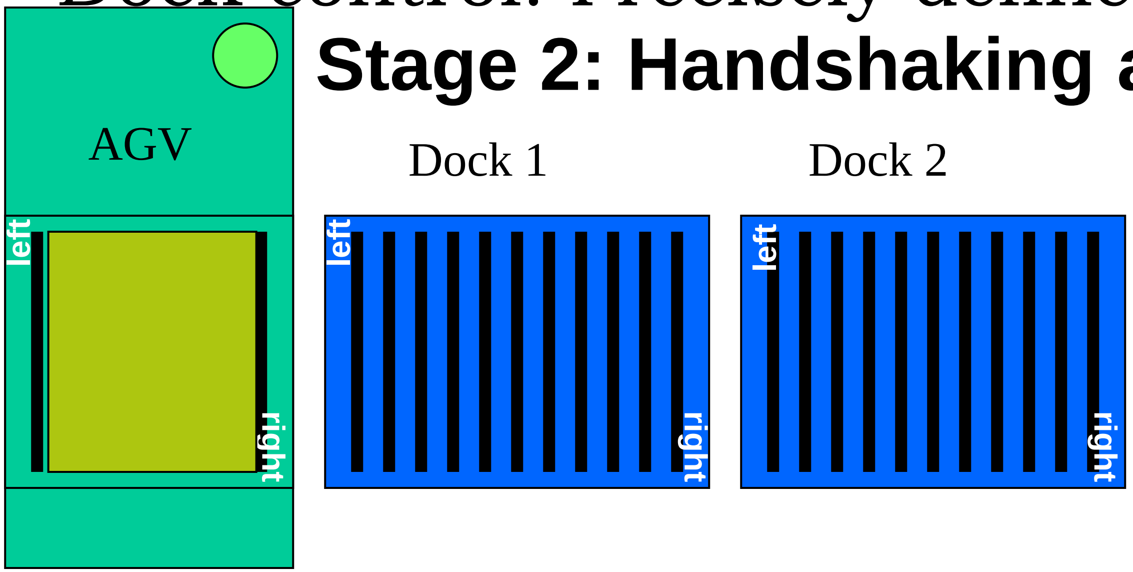
2 stations per move: who is in charge?

- Solution 1: station interaction and handshaking handled on the TRACES level
- Solution 2: 'exec' command to move a load sent to both stations, who can work out handshaking further
- Solution 3: master/slave: one station gets the command and works it out with the other station; TRACES only talks with the master
- Solution 4: master/slave; TRACES gives the command to the master, but also gets callbacks from the slave
- Solution 5: a coordinating object is responsible for the coordination of activities of the two stations.



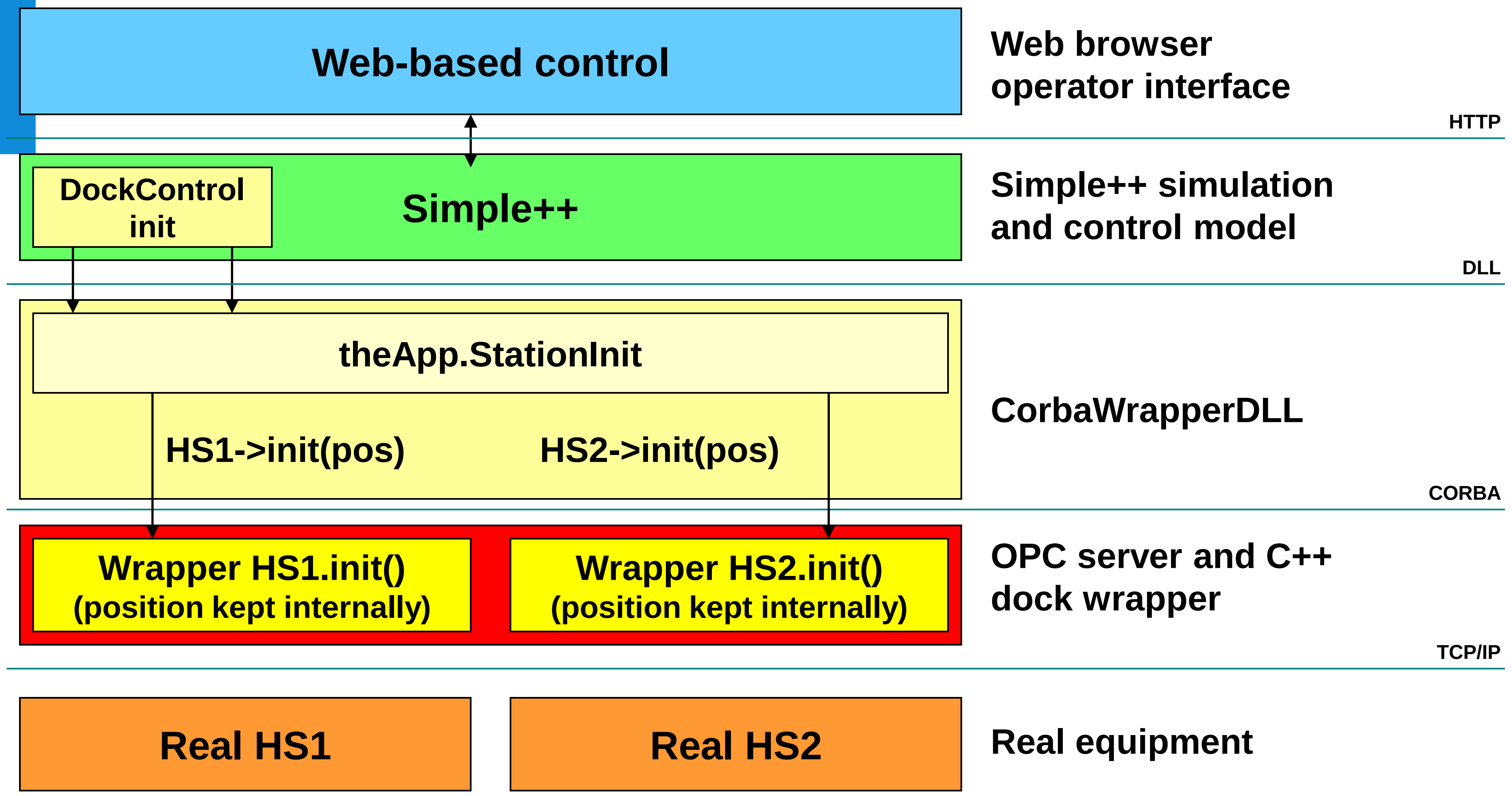
Dock control: Precisely defined interaction

Stage 2: Handshaking and start

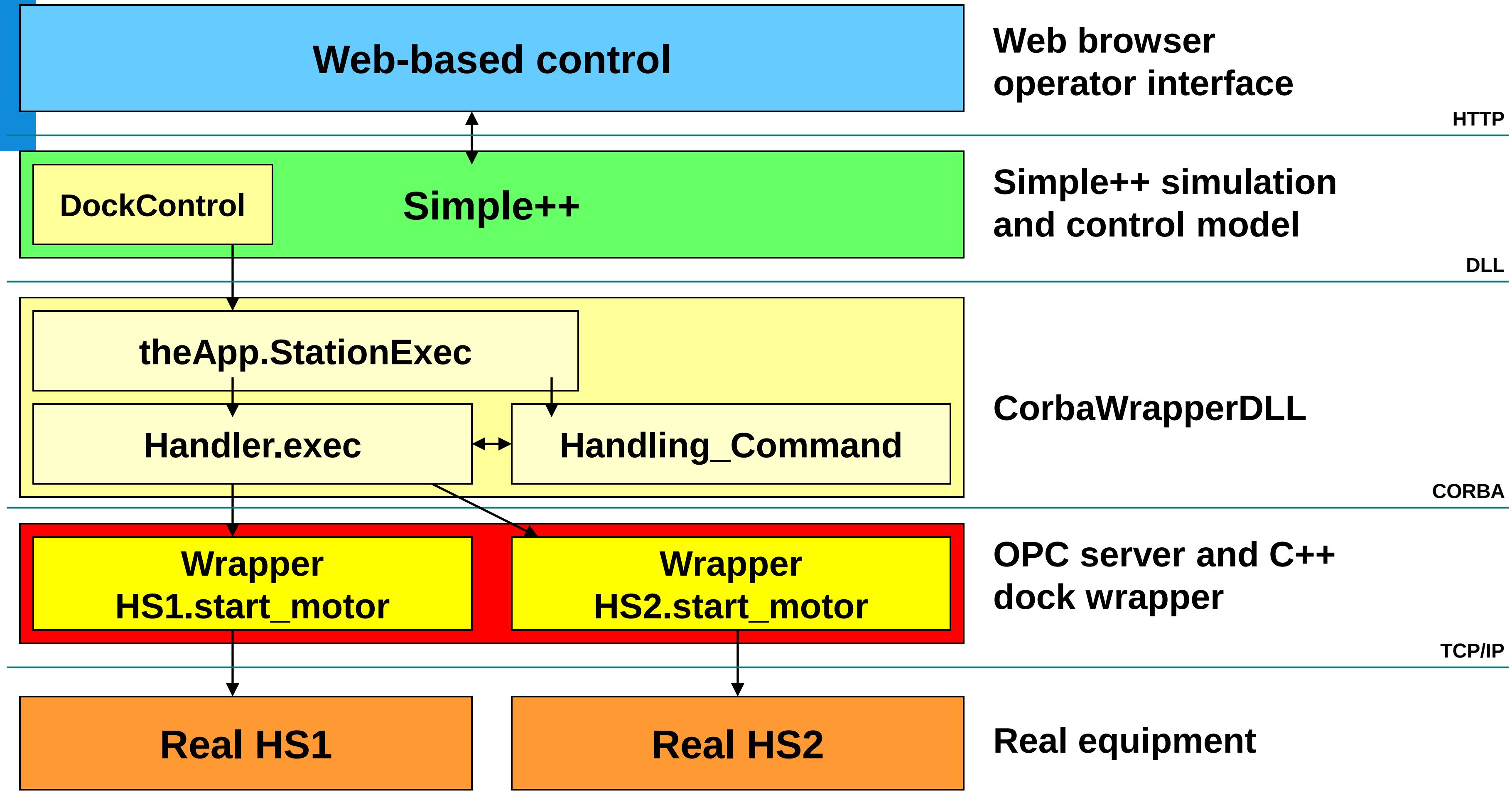


TRACES :	Control internal	=> Can the load be transferred to Dock 1?	
TRACES :	Control internal	=> Load can go to Dock 1.	
TRACES:	Control -> Dock1	=> is_loaded ()	= Do the sensors see a load on the dock?
softPLC:	Dock1 -> Control	=> boolean	= Result of the sensors, false is OK
TRACES:	Control -> Dock1Left	=> can_handle (Load1)	= Does the load fit the interface?
softPLC:	Dock1Left -> Control	=> boolean	= Result of “can_handle”, true is OK
TRACES:	Control -> Dock1Left	=> exec (load, Load1)	= Start interface Dock1Left
Physical:	Dock1Left start conveyor AGVdock		
softPLC:	Dock1Left -> Control	=> notify_moving ()	= Message that conveyor has started
TRACES:	Control -> AGVdockRight	=> exec (unload, Load1)	= Start interface AGVdockRight
Physical :	AGVdockRight start conveyor Dock1		
softPLC :	AGVdockRight -> Control	=> notify_moving ()	= Message that conveyor has started

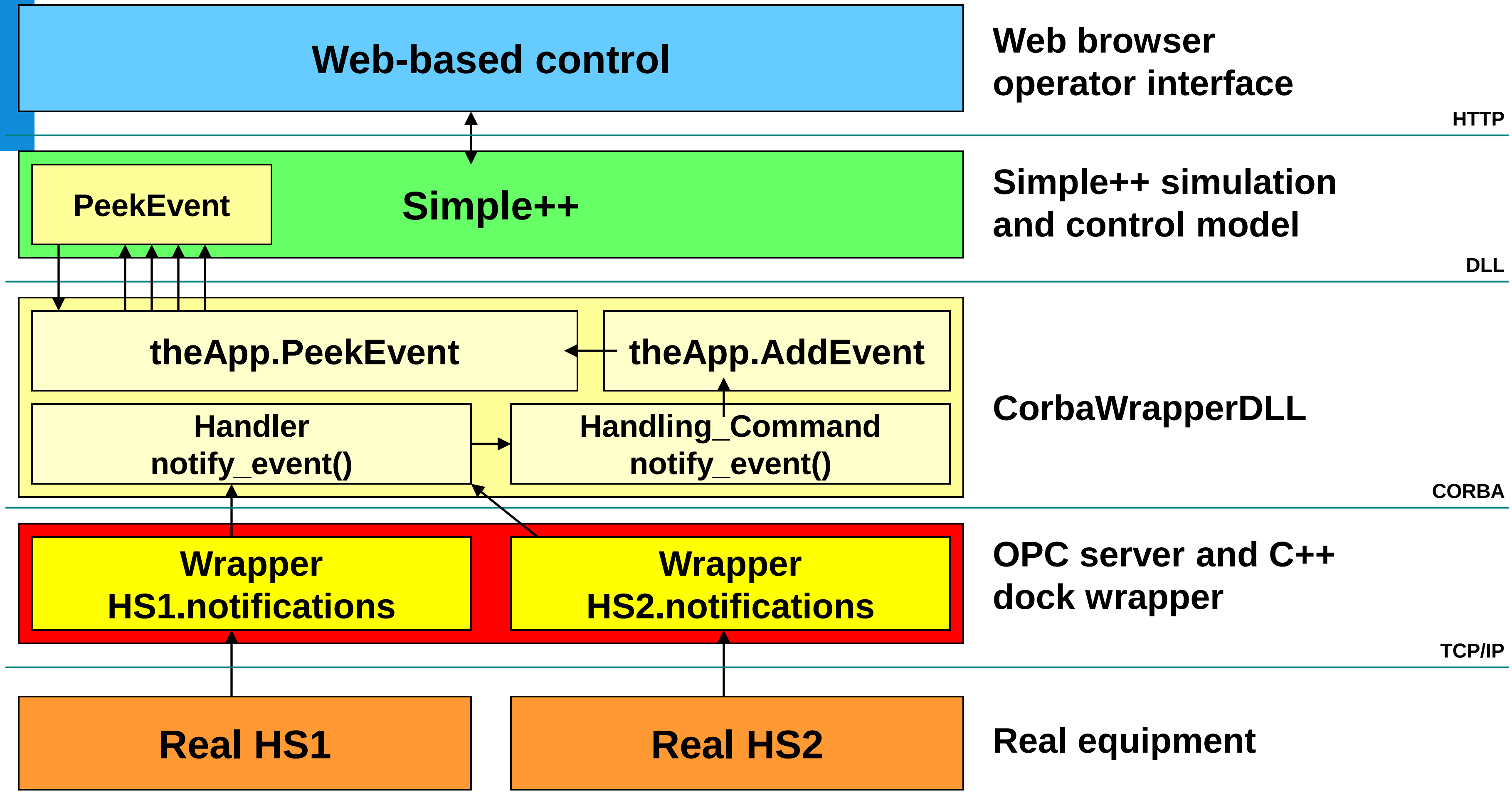
Control of real station
Stage 0: setting up



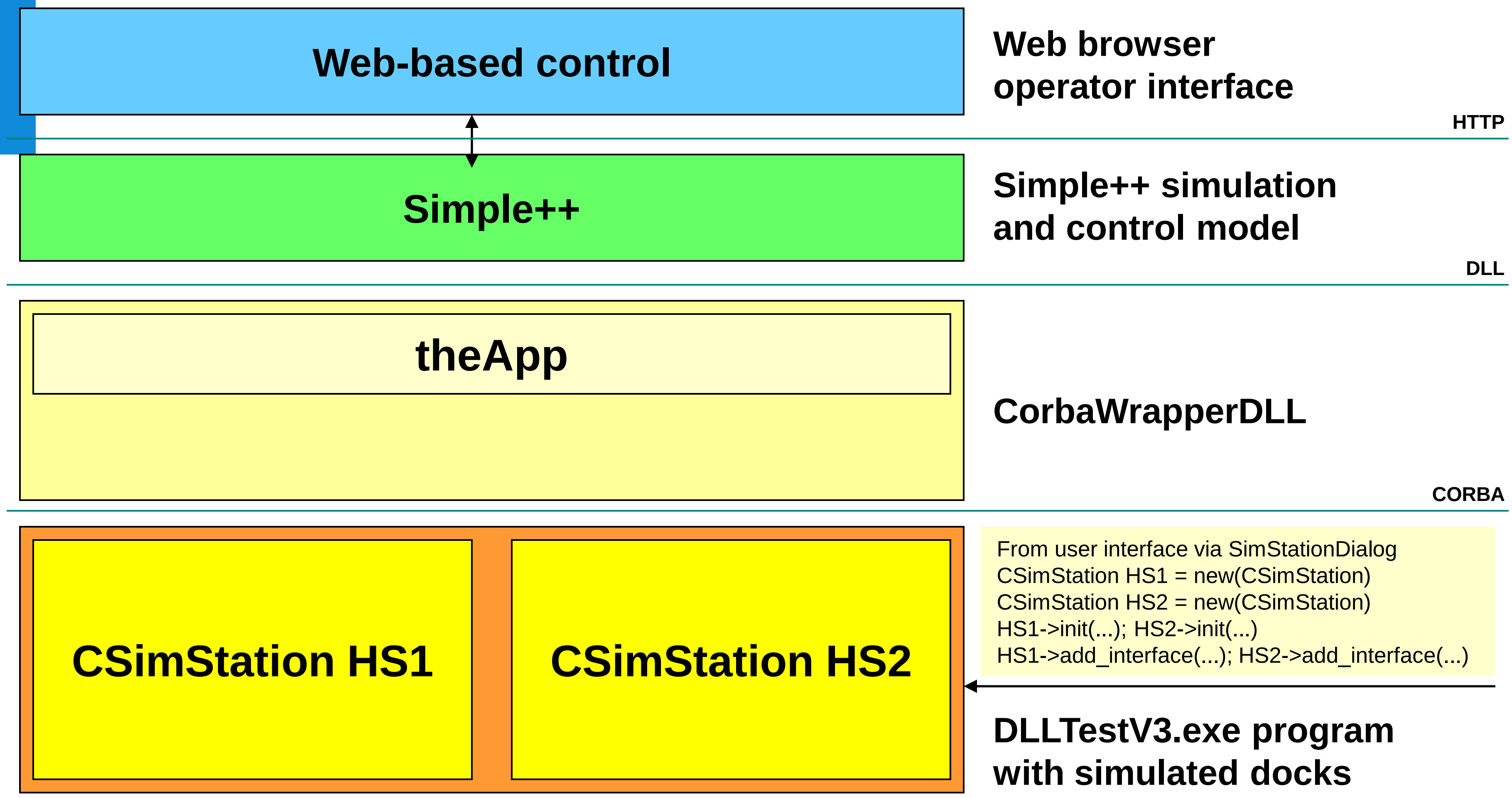
Control of real station
Stage 1: starting



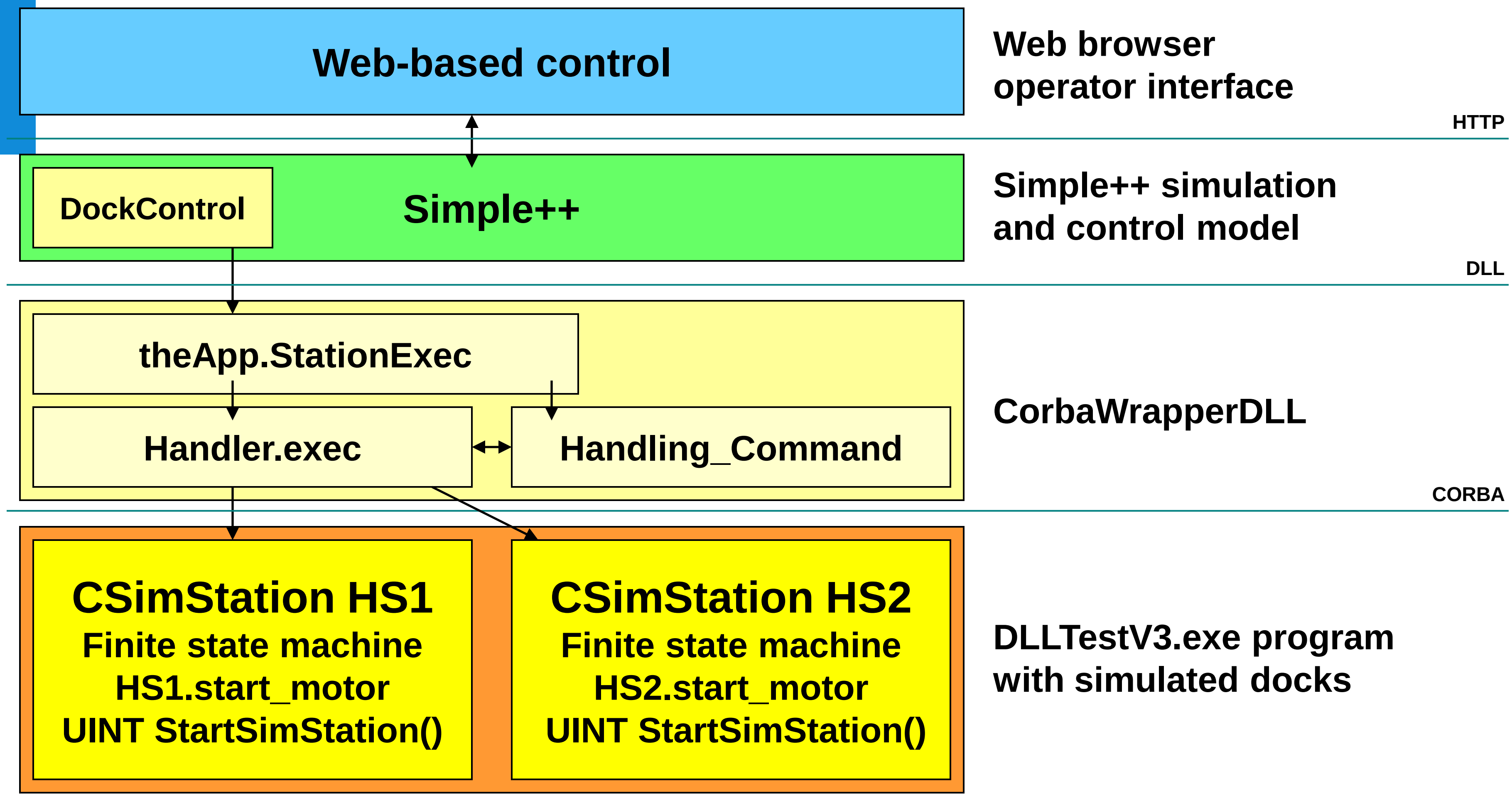
Control of real station
Stage 2: notification



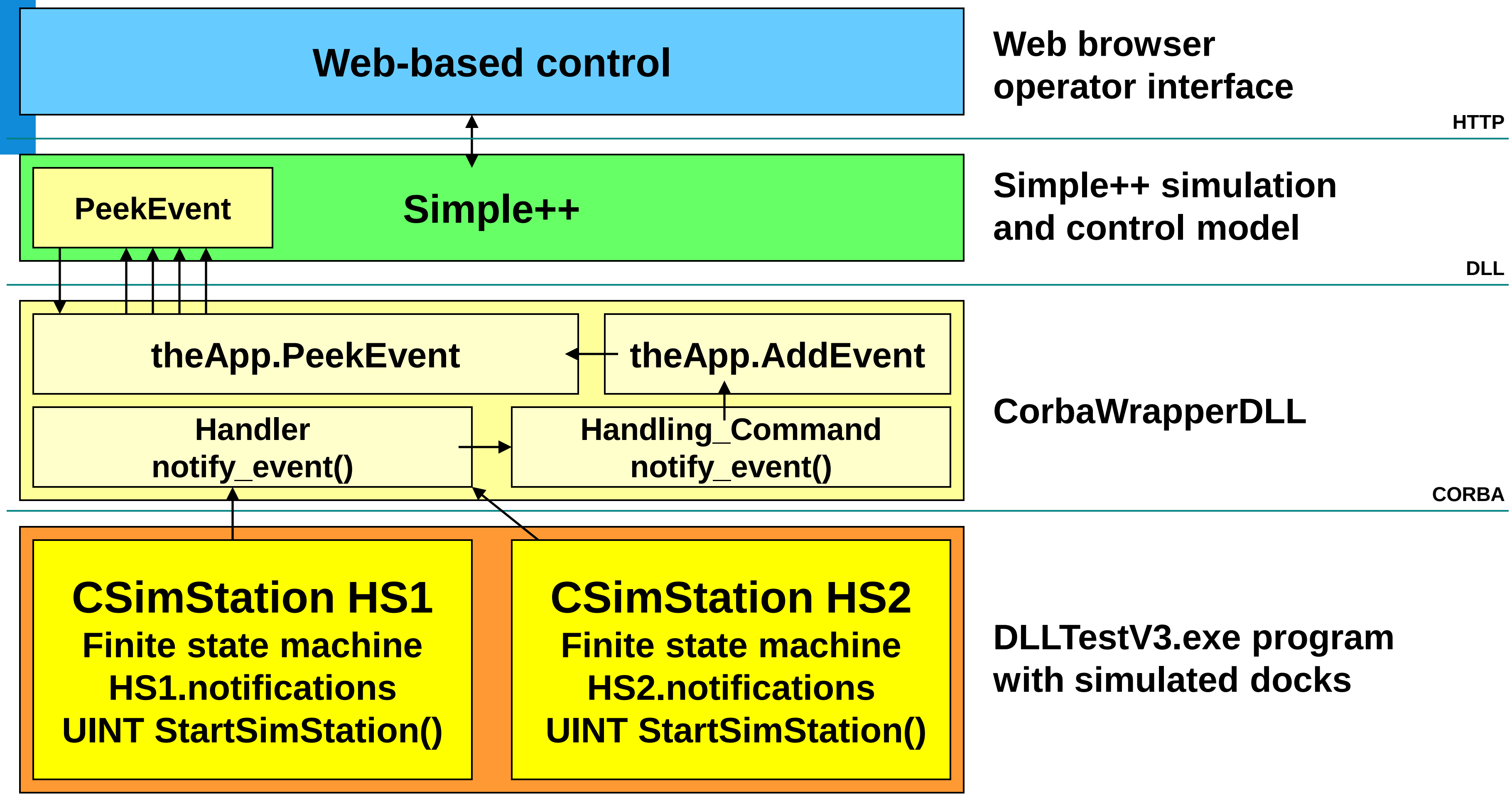
Simulated station
Stage 0: setting up



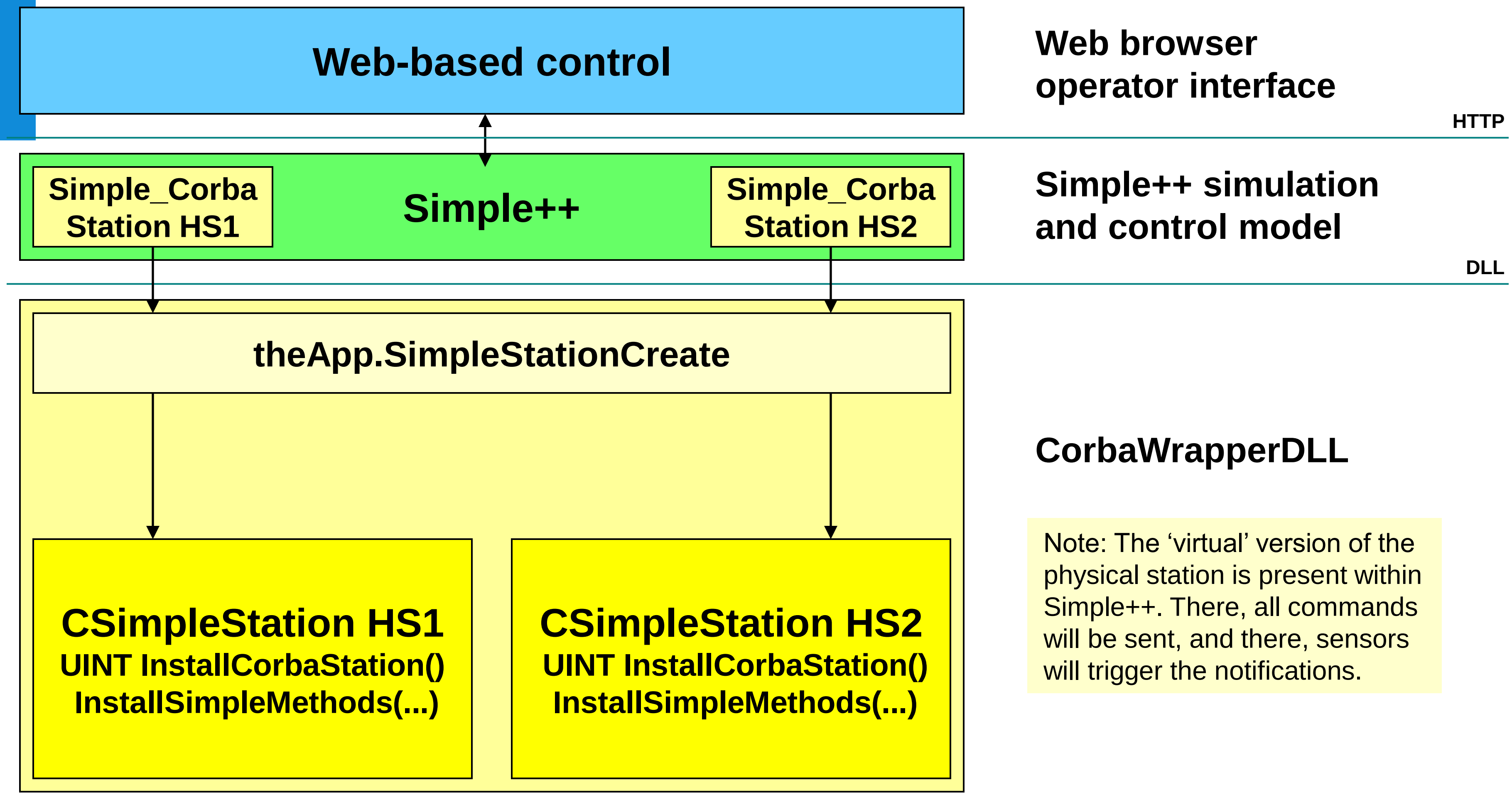
Simulated station
Stage 1: starting



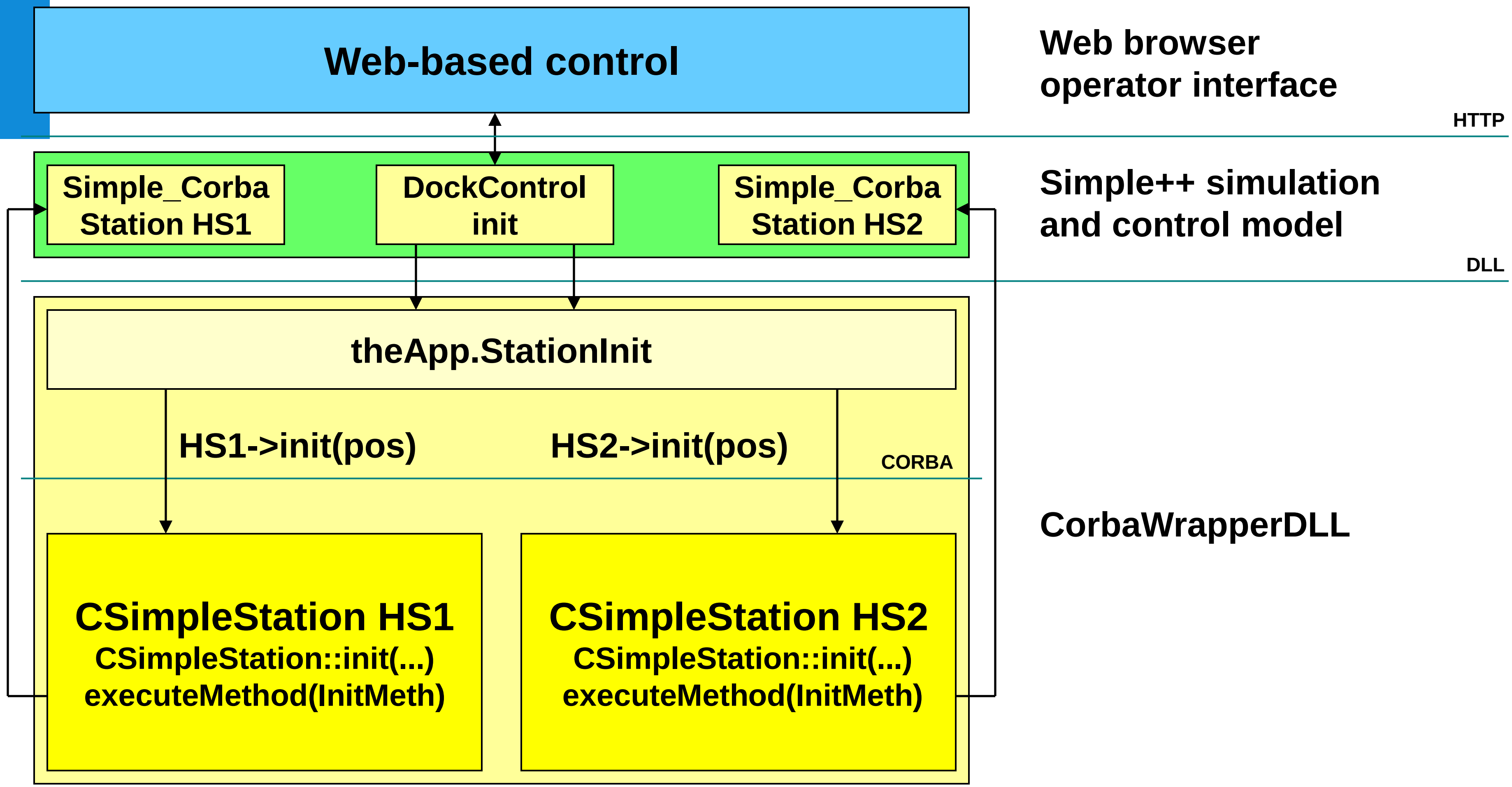
Simulated station
Stage 2: notification



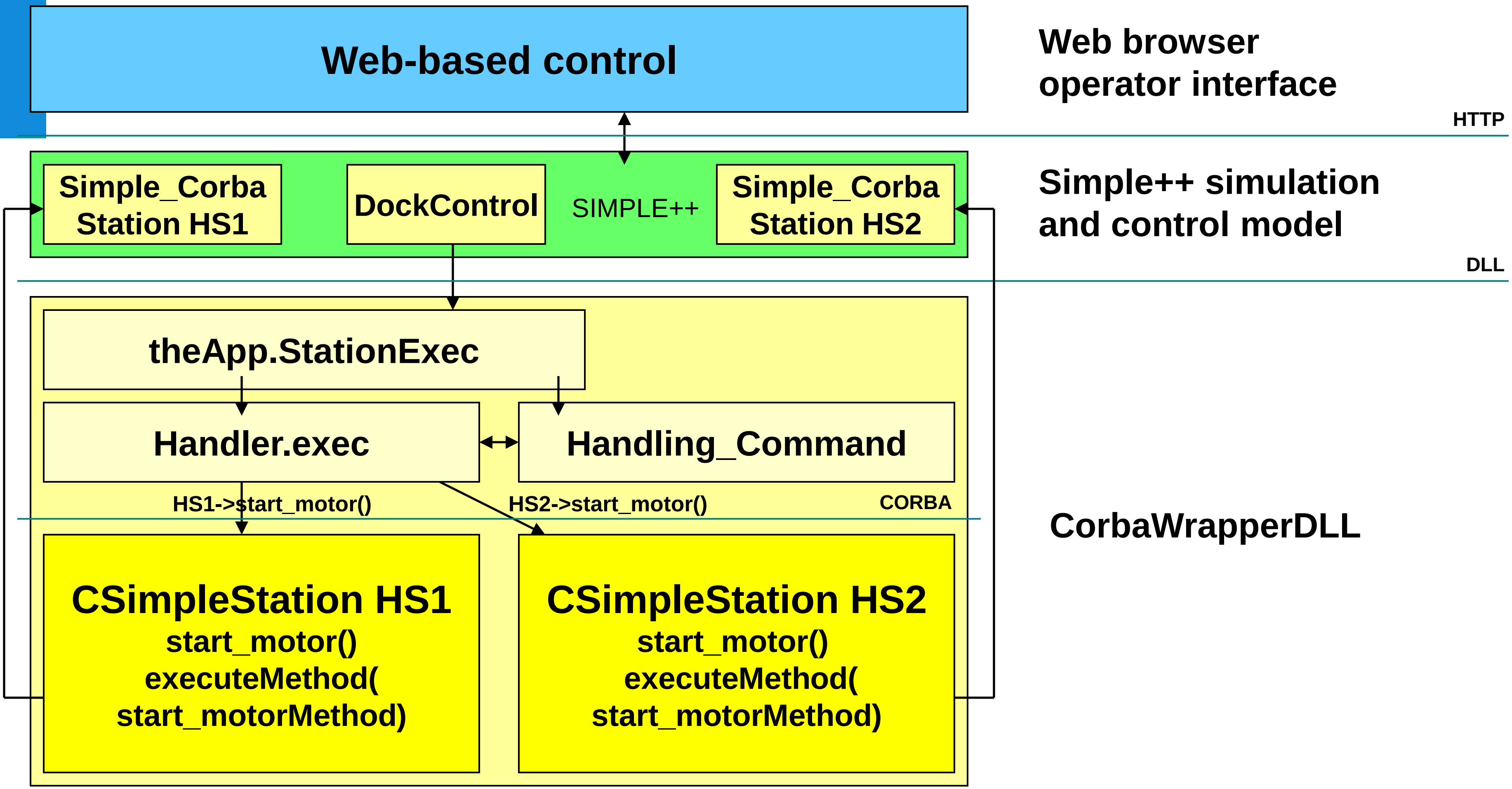
Simple++ Corba station
Stage 0: setting up



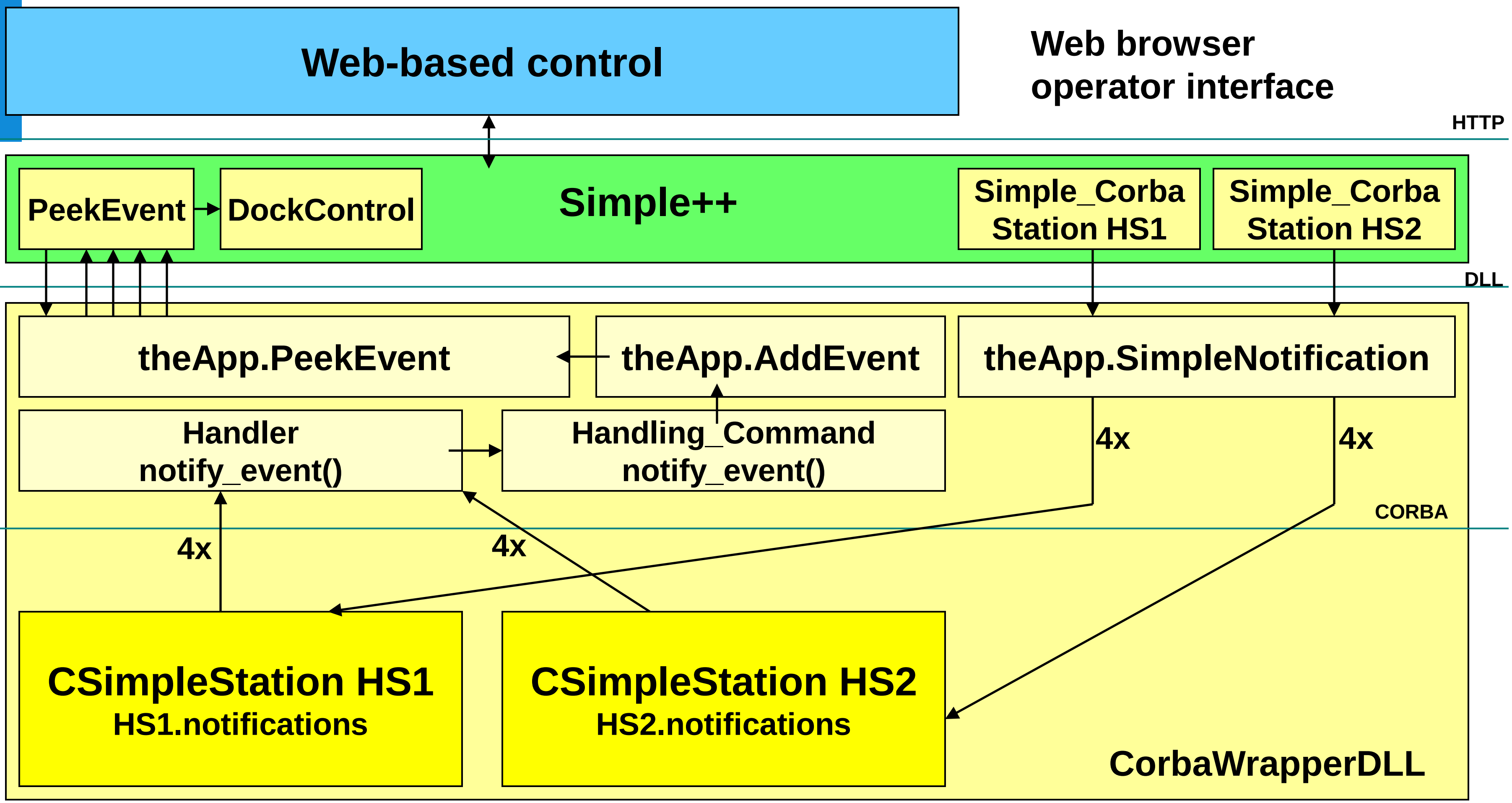
Simple++ Corba station
Stage 0: setting up



Simple++ Corba station
Stage 1: starting

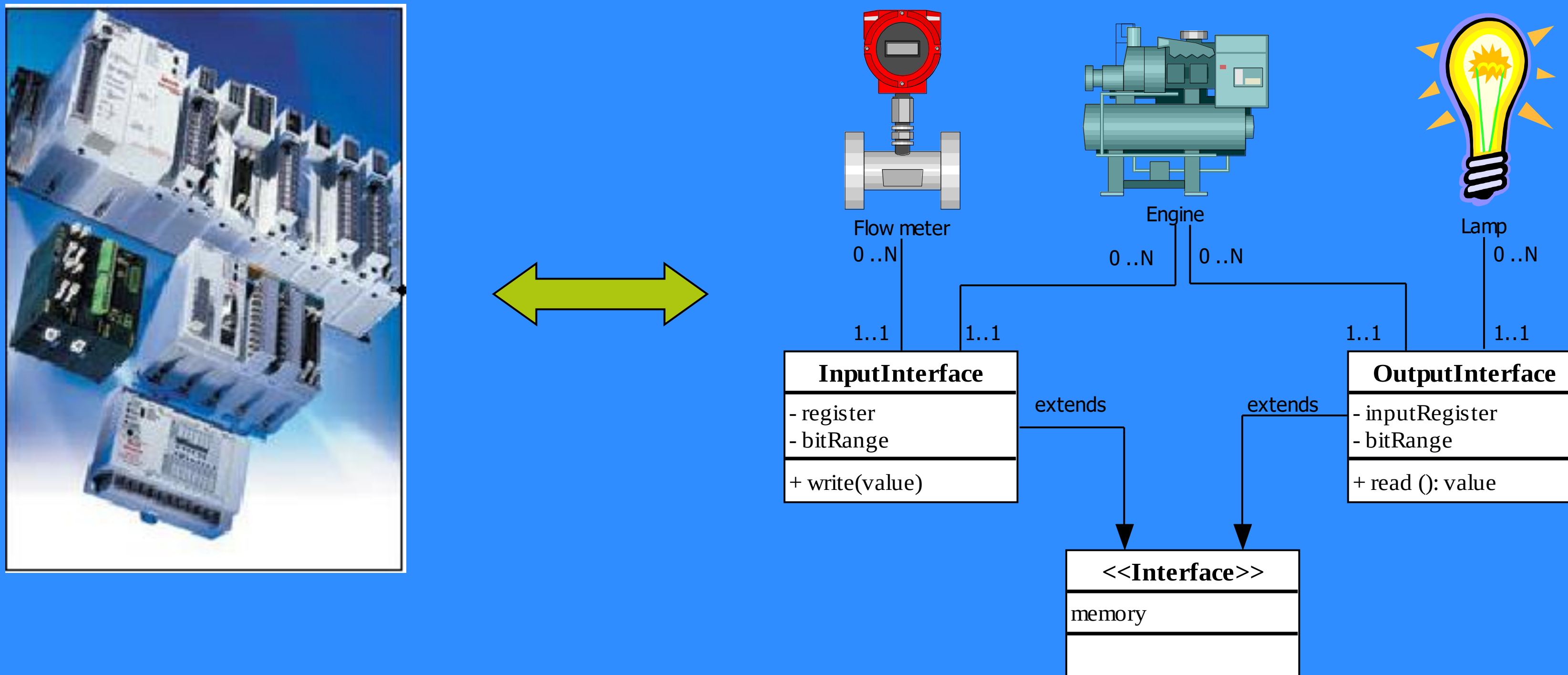


Simple++ Corba station
Stage 2: notification



Conclusions and Further Research

- Real time control with simulation works
- Developing systems with simulation helps to shorten project lead time
- Further research looked at Arena RT and AutoMod to control the automated vehicles
- Further research looked at HLA and lightweight architectures to link distributed models
- Further research focused on libraries for robust control of AGVs and on other control concepts, to be tested using simulations and emulations
- Further research was expanded into live gaming (see class 6.2)



3. Emulation with DSOL and eM-Plant

Paper: Peter H.M. Jacobs, Alexander Verbraeck, William Rengelink. Emulation with DSOL. In: M.E. Kuhl, N.M. Steiger, F.B. Armstrong, and J.A. Joines, (Eds.). Proceedings of the 2005 Winter Simulation Conference. IEEE, 2005. pp. 1453-1462.

Dycore concrete floor manufacturer

Context

- Dycore is a Dutch concrete floor manufacturer
- Dycore employs approximately 500 people
- Produces 3,000,000 m^3 of concrete floors annually
- Emulation case concerns the reinforcement gallery of sheet piling floor production

Sheet piling floors

Production is fully automated

What is emulation and why does Dycore need it?

What is emulation?

- Emulation is a hardware in the loop approach. Simulation is linked up with hardware used in daily operation
- It is an approach to test the behavior of a Programmable Logic Controller (PLC)
- Emulation implies that **all** inputs and outputs of a PLC are connected to simulated components (e.g. devices)
- Emulation enables a tester to reproduce interaction of various parts in a system

Why does Dycore need it?

- Dycore wanted to replace its PLC responsible for automatic guided vehicles, welding, sensors, emergency circuits, etc.
- The impact of failures in the PLC is high because in case of misalignment, the mass of the floors might bring significant damage to the infrastructure and potentially to people

The case was conducted in a *competitive* setting

TU Delft

- Delft University of Technology; Department of Systems Engineering
- Used DSOL simulation environment

TBA

- TBA Netherlands; Company specialized in emulation, simulation of logistic systems
- Used eM-Plant environment

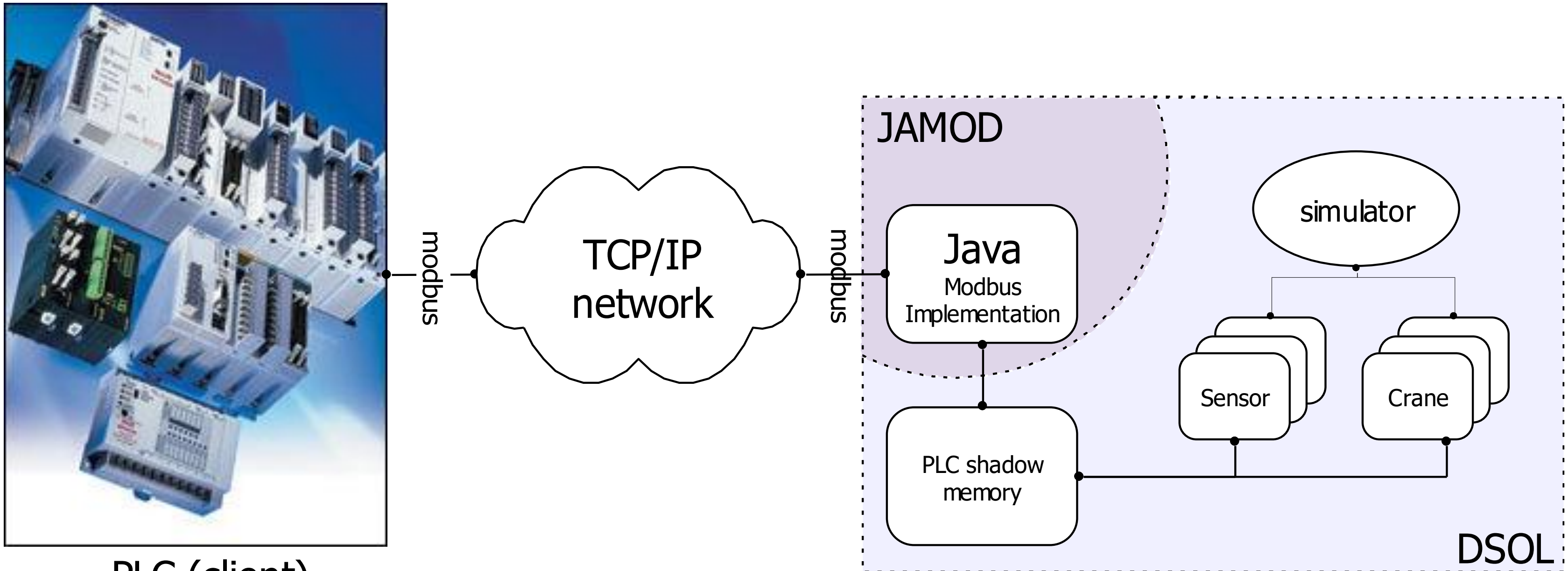
Why did we expect to outperform eM-Plant?

- The service oriented, open architecture underlying DSOL should make the deployment in a distributed, networked environment more straightforward
- The multi-threaded, scalable characteristics of the Java programming language should make DSOL more effective in the performance-defiant domain of emulation
- Support for CAD drawings and Java 3D library makes infrastructure modeling more straightforward

Requirements for the case

- The emulation model may not impose any modifications to the PLC for testing purposes
- The emulation model should support the industrial data exchange protocol used by Dycore's PLC (Modbus)
- The emulation model should meet the real-time period of the PLC (30 ms, max. 35 ms)
- Whenever the emulation model is deployed on a non-realtime operating system (e.g. Microsoft Windows), the model should report backlog when this occurs
- The emulation model should animate all devices, sensors, etc. on top of the CAD drawing which is well known to the controllers of the physical system
- All simulated devices should be controllable at runtime through a graphical user interface

Overview of the architecture



PLC (client)

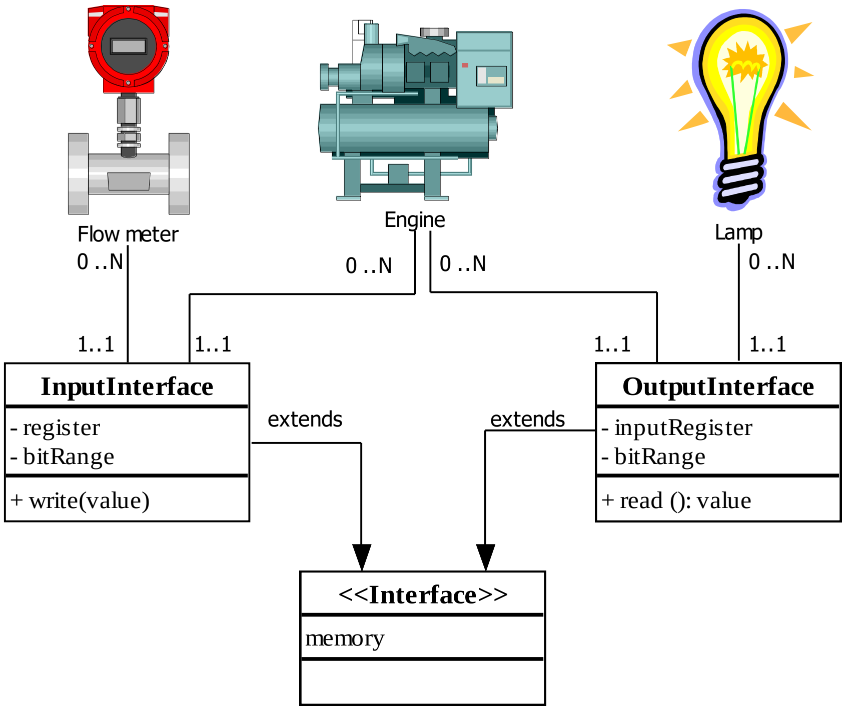
Java based emulation model

- Open source Java based Modbus communication service (JaMod)
- Open source Java based simulation service (DSOL)
- DSOL model contains PLC shadow memory
- Every PLC period (30 milliseconds) shadow memory is synchronized over Modbus protocol

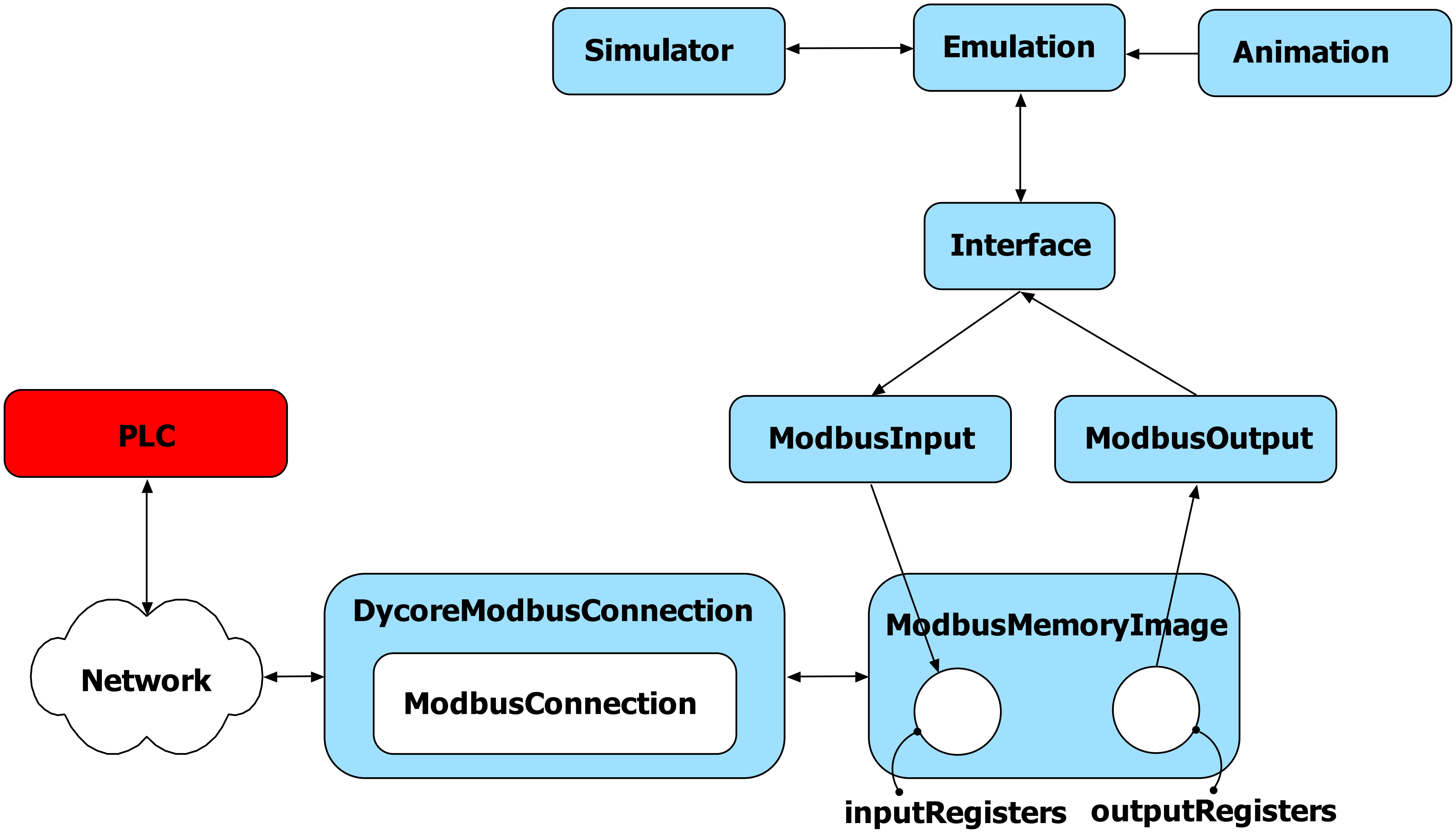
Three different devices used in emulation

Devices

- **Input devices** which only send data to the PLC (e.g. sensor, emergency button and GPS device)
- **Output devices** which only receive data from the PLC (e.g. a lamp or a siren)
- **Combined devices** which both receive and send data to the PLC (e.g. an automated guided vehicle which sends position and receives orders for action)



Architecture



Specification of the emulation case

DSOL

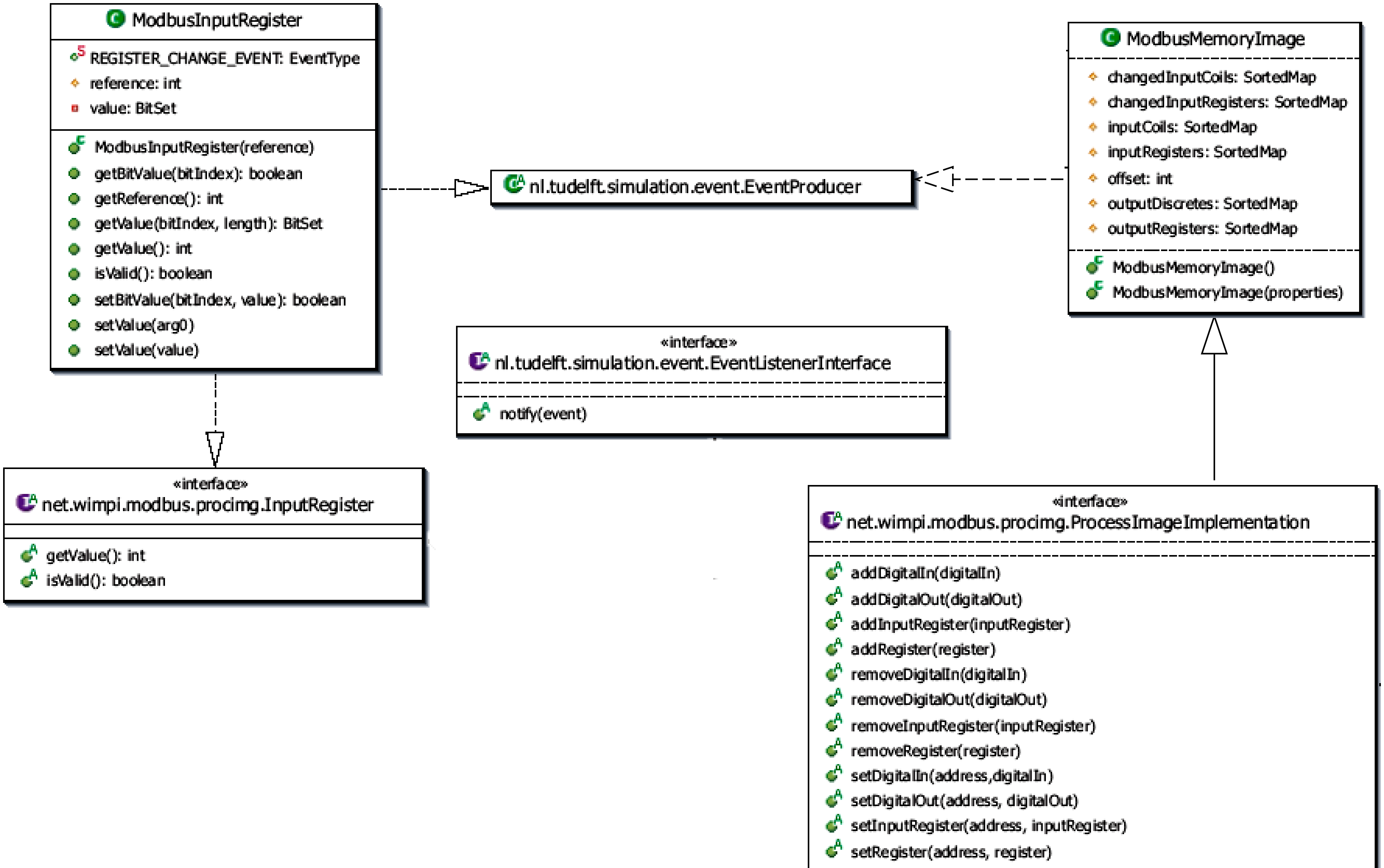
- External service named POI was used for Excel input/out specification
- External service Gisbeans was used to render CAD drawings. Zooming, panning etc was therefore possible
- External service JaMod was used for the Java-Modbus communication. PLC communication was established within 24 hours

eM-Plant (Plant Simulation)

- No external services used. Excel input/output is natively supported
- Cad rendering was not possible. The background of the model was a screenshot
- Custom, tailored eM-Plant Modbus communication was programmed. This took 3 weeks of dedicated programming in C++

Sensor and device classes

Register classes



Specifying the action sequence

DSOL

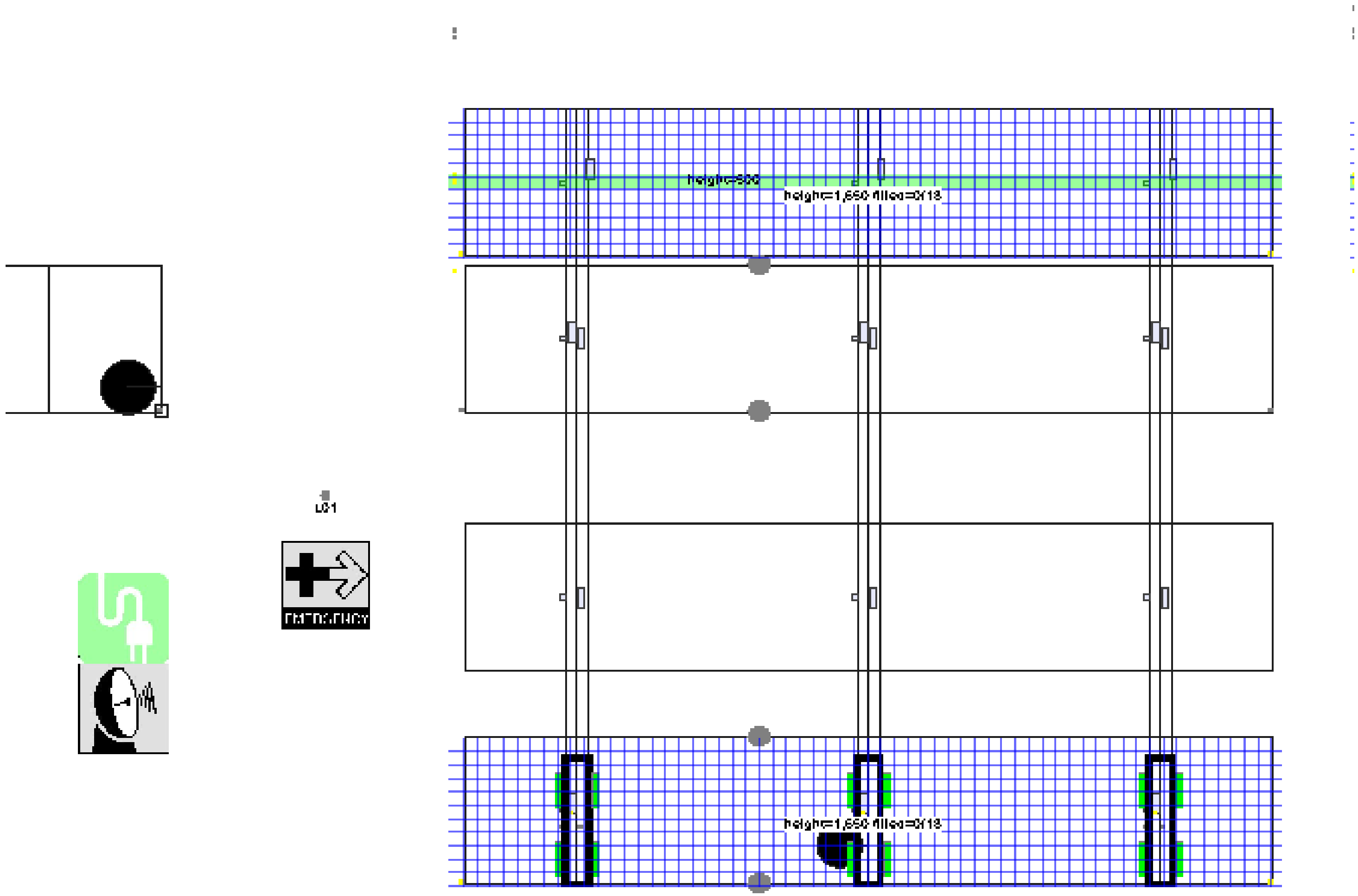
- One high priority thread is used to synchronize the shadow PLC memory and the actual PLC every 30 milliseconds. This thread also updates the state of the model
- One low priority thread was used to update the animation (i.e. the CAD drawing every 200 milliseconds
- One normal priority thread was used to capture user input and update the state of the model accordingly

eM-Plant (Plant Simulation)

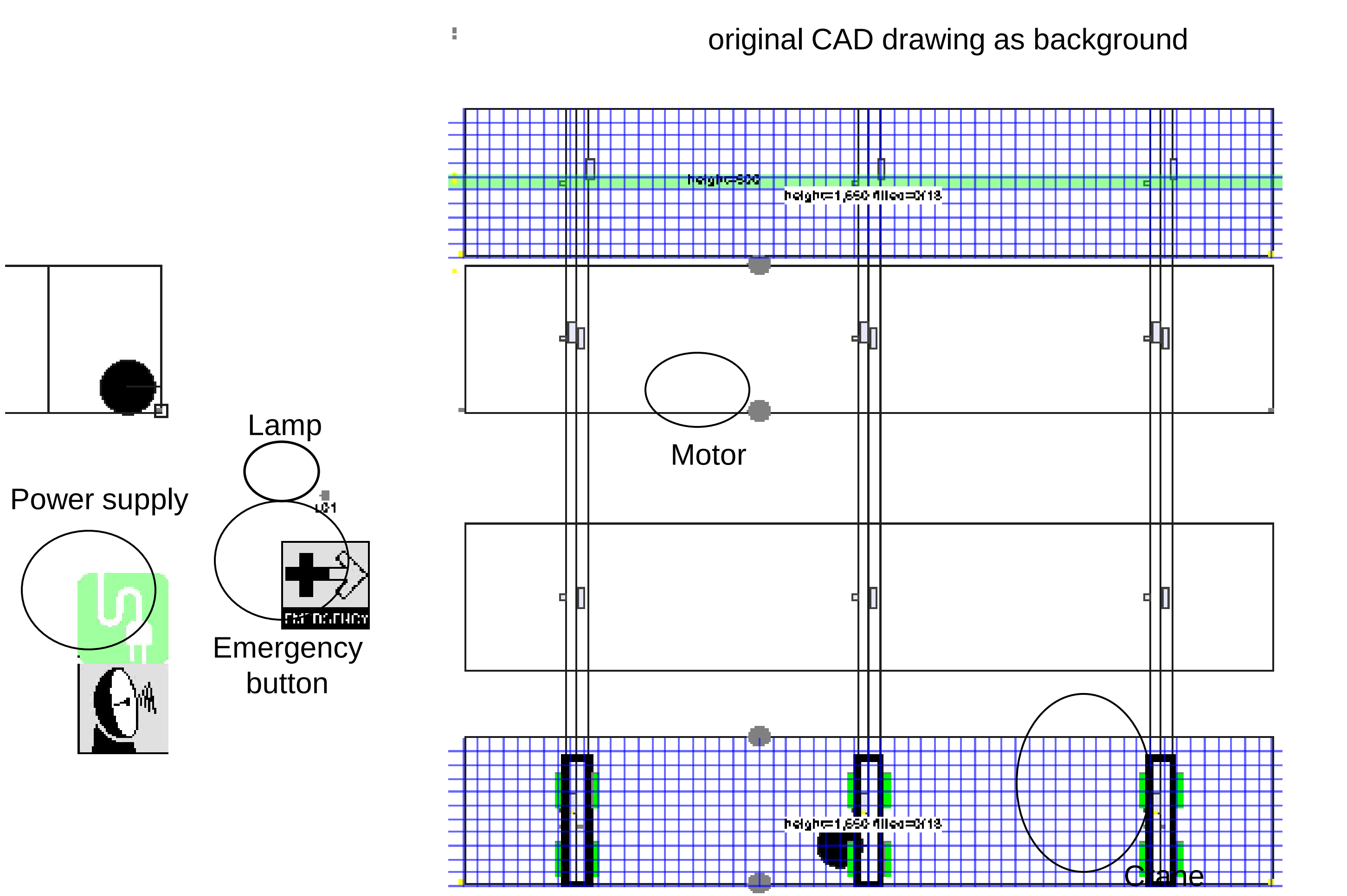
- eM-Plant only uses 1 thread for the simulation, the animation and the user input

- The emulation model programmed in DSOL succeeds in the performance defiant domain of simulation where 30 milliseconds was not achieved with eM-Plant

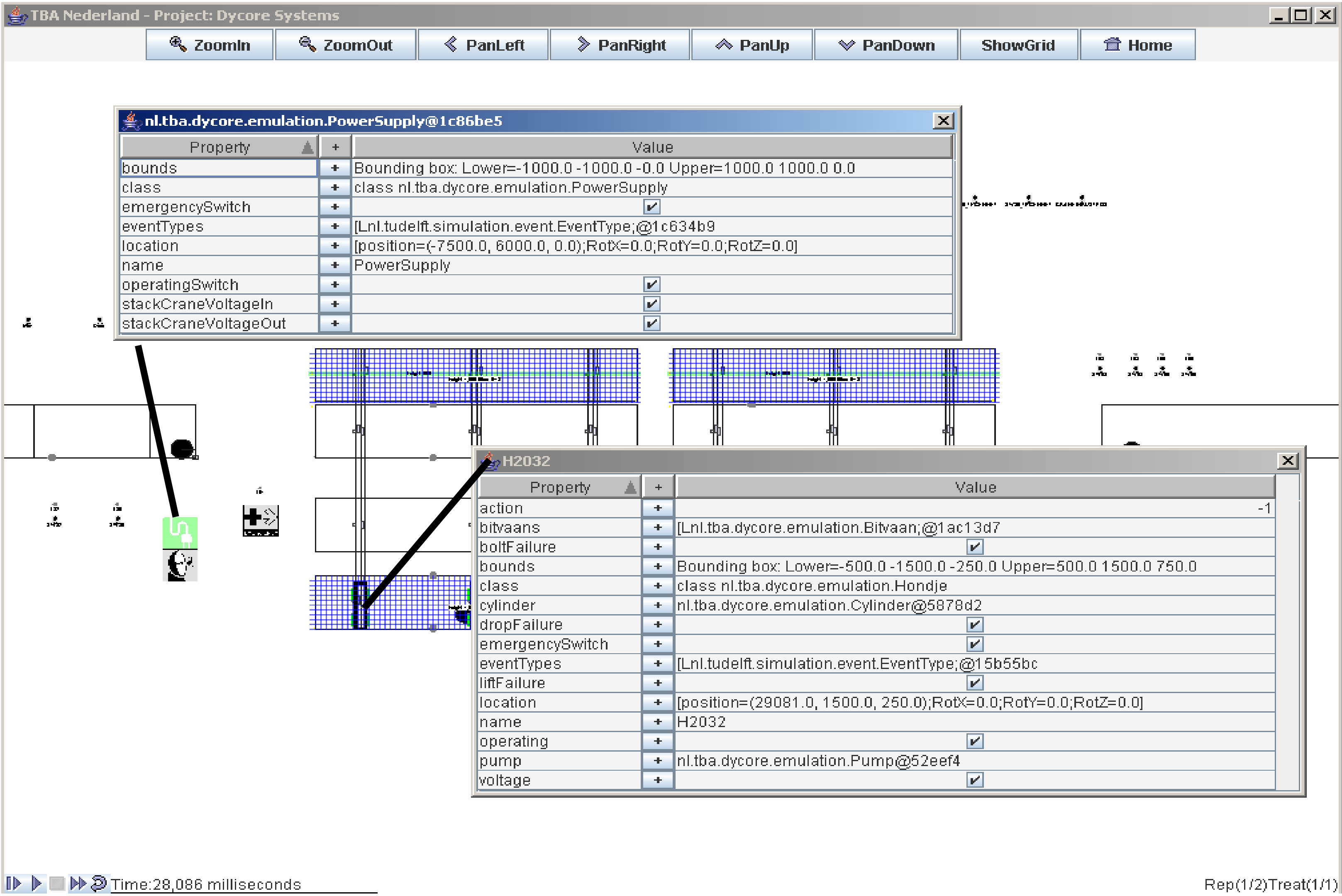
Proof of concept: animation



Proof of concept



Proof of concept: introspection



Conclusions and recommendations

Conclusions

- The ability to integrate external services (e.g. Excel input-output, Modbus, CAD) resulted in very efficient model specification
- The ability to use different threads for animation and simulation resulted in a well performing emulation model

- DSOL is Java-based, open source (BSD-3) environment for multi-formalism modeling.
  see: www.simulation.tudelft.nl or github

Recommendations

- To integrate the architecture with a real-time operating system
- To develop new formalisms for trackless infrastructure modeling