# Specification, Analysis and Verification of an Automated Parking Garage

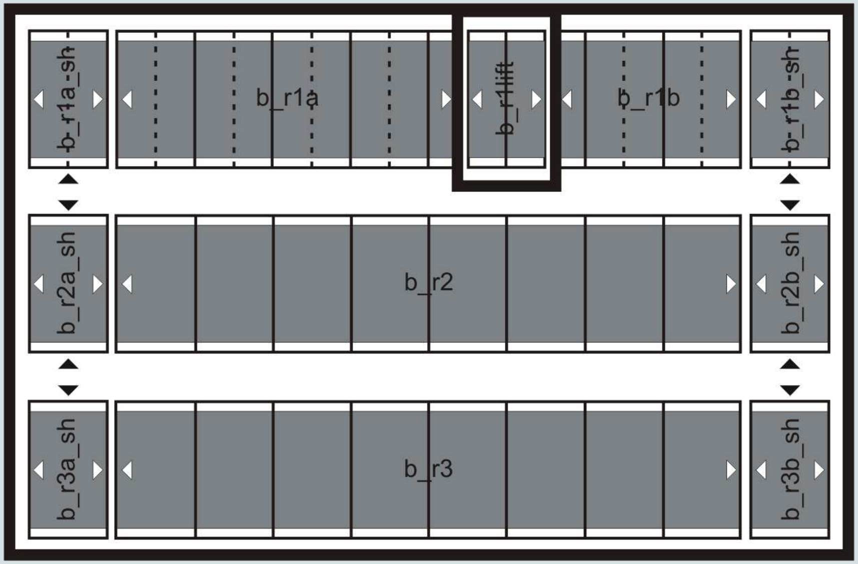
To be realised by CVSS BV in Bremen, Germany

Aad Mathijssen Hannes Pretorius

30th June 2005

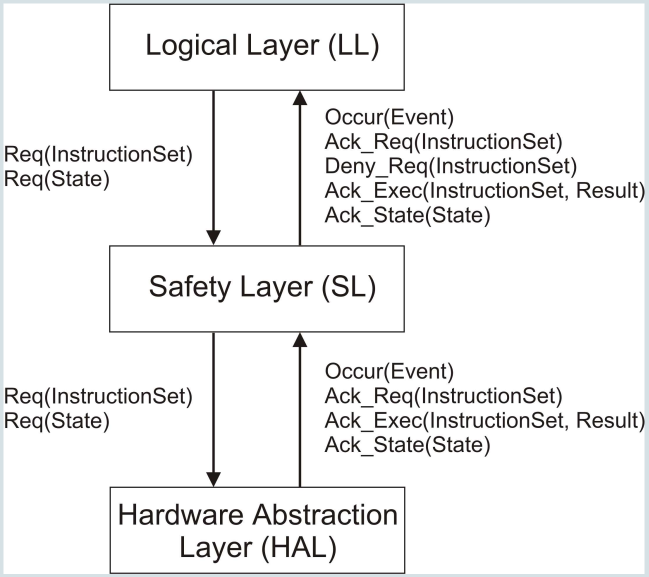
## Problem description

Most complex part of system: parking garage and lift



## Conceptual system design: architecture

Distinction between three layers in order to maintain separation of concerns:



## Conceptual system design: architecture (2)

Role of different layers:

- *logical layer*: the parking/retrieval algorithm
- *safety layer*:
  - pass messsages between the logical and hardware layer
  - **only** if they are safe, deny otherwise
- *hardware abstraction layer*:
  - receive and execute instructions
  - give feedback on results
  - issue events to the safety layer

## Conceptual system design: data

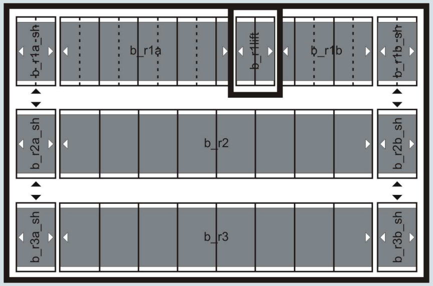
Data communicated between layers:

- *events*: addition and removal of cars
- *instruction sets*: sets of operations to be executed *simultaneously* on belts, shuttles and lift
- *results*: success or failure of execution of an instruction set
- *states*: physical system state

## Conceptual system design: data (2)

Instructions:

- move_belts(bs: BeltSet, d: Direction, ms: MoveSize)
- move_shuttles(shs: ShuttleSet, o: ShuttleOrientation, d: Direction)
- tilt_shuttle(sh: Shuttle, o: ShuttleOrientation)
- move_lift(h: Height)
- rotate_lift

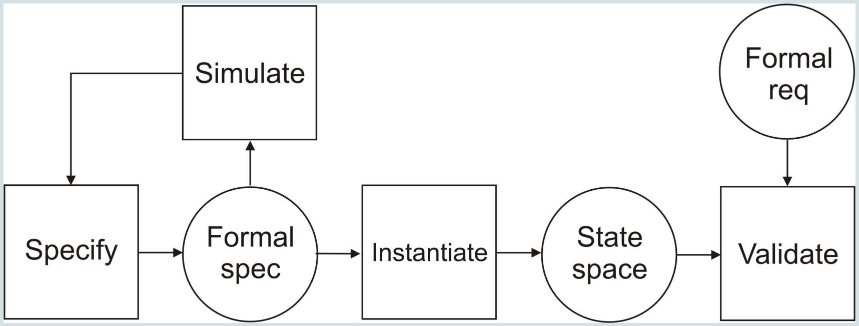


## Specification approach

Describe *behaviour* (not implementation)

Focus on *safety*

Modelling process:



## Specification of the safety layer

Requirements: guarantee safe behaviour of belts, shuttles and lift

Complexity:

- due to lift position, cars should be able to move in half positions
- shuttles can be tilted

Result:

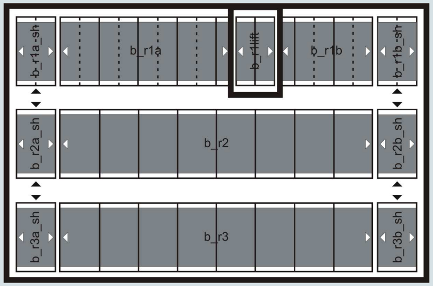
- components cannot be viewed in isolation
  e.g. a car can be on both the lift and the bordering conveyor belt
- more checks are needed
- complex checks are needed

Satisfy requirements by introduction of an *allow* function:
check safety of an instruction in the current state

## Specification of the safety layer: *allow* example

Instruction move_shuttles(*shs*, *o*, *d*) is allowed if:

- at least one shuttle is specified
- all specified shuttles border each other
- all specified shuttles must be in orientation *o* (lowered or tilted)
- there is an open position for the shuttles to move to
- there must be no car suspended between one of the shuttles and its bordering conveyor belt



## Simulation

Simulation allows us to walk through the specification:

- for current state all possible actions are provided
- choose an action and go to next state

Infeasible in practise due to enormous amount of possible actions (requests, events, message passing)

Restrict behaviour:

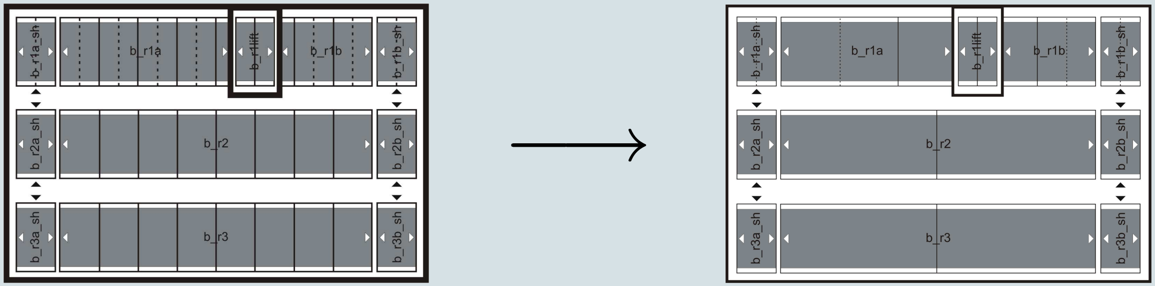
- reduce instruction sets to single instructions
- abstract from non-essential communication (message passing of events and states)

## Toward verification

Verification: formalise requirements and check them on every possible system state

Unfeasible with the previous model: 640 billion states $(6,4 * 10^{11})$

Restrict the number of *positions*:



This system has 3,3 million states $(3,3 * 10^{6})$

## Conclusions

- Based on simulation we are confident that the safety requirements are met
- With the aid of visualization we discovered a number of errors in the specification
- However, to be completely sure verification has to be applied
- This will be added to the report, which also includes:
  - detailed description of the approach
  - all conditions for which instructions are allowed
  - description of the specification language used
  - formal specifications
- The specifications may serve as a starting point for implementation
- The architecture enables the implementor to separate concerns