HIGHLY COMPETITIVE COMPETITION

- my new car!

- my new software toolkit?

Functional safety of electrical/electronic/programmable electronic safety-related systems

IEC 61508
**FAULT PREVENTION**

- Gamma, E. et al.: 
  *Design Patterns: Elements of Reusable Object-Oriented Software*; Addison Wesley 1994.

- Fowler, M.: 
  *Analysis Patterns*, Reusable Object Models; 
  Addison-Wesley 1997.

- Grand, M.: 
  *Patterns in Java*, Vol. 1, A Catalog of Reusable Design Patterns Illustrated with UML; Wiley 1998.

- Rising, L. (ed): 
  *Design Patterns in Communication Software*; 

- Pont, M. J.: 
  *Patterns for Time-Triggered Embedded Systems*; 
  Addison-Wesley 2001.

**SOFTWARE ENGINEERING & MODELS, TWO APPROACHES**
CASE STUDIES:

ACADEMIC:
- low-level mutex algorithms
- Dijkstra’s philosophers
- Milner’s schedulers
- solitaire
- ...

MORE REALISTIC
- production cell, Karlsruhe
- production cell, Cottbus
- concurrent pushers
- 2-hand switch
- plc press controller
- ...

PRODUCTION CELL:
- feed belt (belt 1)
- deposit belt (belt 2)
- elevating rotary table
- press
- travelling crane
- arm 1
- arm 2
- 14 sensors
- 34 commands
INFORMAL SAFETY REQUIREMENTS (21):

- The press must not be moved downward, if sensor 1 is true, and it must not be moved upward, if sensor 3 is true.
  - Restrictions of machine mobility.

- The press may only be closed, when no robot arm is positioned inside it.
  - Avoidance of machine collisions.

- The feed belt may only convey a blank through its light barrier, if the table is in loading position.
  - Blanks are not dropped outside safe areas.

- Blanks may not be put into the press, if it is already loaded.
  - Insurance of a sufficient distance between consecutively processed blanks.

additional requirements related to design consistency:

- The robot swivel is either stopped or moves in exactly one direction.
- ...
**COOPERATION MODEL, BASIC DESIGN PRINCIPLES:**

- production cell = pipeline of machines
- each machine takes plates from some input places; processes them; puts plates on some output places;
- cooperation region between two consecutive machines
- mutual exclusive shared resources
  - robot swivel (to rotate both arms)
  - physical regions (intersection of trajectories of different machines)

**BOUNDED PRODUCER CONSUMER PATTERN:**

**PRODUCER**

- ready_to_produce
- input_area_free
- input_available
- processing
- produce
- controller
- ready_for_processing

**CONSUMER**

- ready_for_processing
- output_area_free
- output_available
- consume
- processing
- ready_to_consume

**THREE TYPES OF COOPERATION PATTERN**

- input_area_free
- output_area_free
- table/press
- input_available
- output_available
- arms/crane
- input_area_free
- output_area_free
- belts
- input_available
- output_available
(A) INDEPENDENT INPUT/OUTPUT

- arms/crane: step-wise synchronization with only one of the adjacent controllers,

- pattern property, e.g.
  \[ G_A(\text{step1} \rightarrow \neg (\text{input\_available} \lor \text{input\_area\_free})) \]

(B) DEPENDENT INPUT/OUTPUT

- belts: simultaneous control of input and output region necessary,

- pattern property
  \[ G_A(\text{step2} \rightarrow \neg (\neg (\text{input\_available} \lor \text{input\_area\_free}) \lor \text{output\_area\_free} \lor \text{output\_available})) \]
(C) Mutually Exclusive Input/Output

- table/press:
  the controller must always hold a lock on one of its cooperation regions;

- pattern property
  \[ G_A(\neg(input \_available \lor input \_area \_free) \lor \neg(output \_available \lor output \_area \_free)) \]

- arm version 2:
**ARM VERSION 3**

- **Waiting for swivel**
  - Input area free
  - Lock input area
  - Load
  - Unlock input area
  - Unlock swivel
  - Storing
  - Output area free
  - Lock output area
  - Wait for swivel
  - Unlock swivel
  - Unloading
  - Unlock output area
  - Having swivel
  - Unlock swivel

**CONTROLLER ANALYSIS:**

- **Producer**
  - Ready to produce
  - Input area free
  - Processing
  - Produce
  - Controller
  - Output area free
  - Consumer
  - Ready for processing

- **Arms**
  - ORD HOM NBM PUR CSV SCF CON SC F10 IF0 pF0 MG SM FC EFC ES
  - Y Y Y Y N N Y Y N N N N Y Y Y Y
  - DTP SMC SMD SMA CPI CTI B SB REV DSt BSt DTr DCF L LV L&S
  - Y Y Y Y Y N Y Y N N Y Y Y Y

- **Else**
  - ORD HOM NBM PUR CSV SCF CON SC F10 IF0 pF0 MG SM FC EFC ES
  - Y Y Y Y N N Y Y N N N N Y Y Y Y
  - DTP SMC SMD SMA CPI CTI B SB REV DSt BSt DTr DCF L LV L&S
  - Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y
STEP-WISE COMPOSITION:
E.G. SUBSYSTEM: ARM1 - PRESS - ARM2
(ARMS: VERSION2):

ORD  HOM  NBM  PUR  CSV  SCF  CON  SC  F0  tF0  Fp0  pF0  MG  SM  FC  EFC  ES
Y  Y  Y  Y  Y  N  N  N  N  N  N  N  N  N  N  Y
DTP  SMC  SMD  SMA  CPI  CTI  B  SB  REV  DSt  BSt  DTr  DCF  L  LV  L&S
Y  Y  Y  N  Y  Y  N  Y  N  N  ?  N  N  N  N

STEP-WISE COMPOSITION:
E.G. SUBSYSTEM: ARM1 - PRESS - ARM2
(ARMS: VERSION3):

ORD  HOM  NBM  PUR  CSV  SCF  CON  SC  F0  tF0  Fp0  pF0  MG  SM  FC  EFC  ES
Y  Y  Y  Y  N  N  Y  N  N  N  N  N  N  N  N  Y
DTP  SMC  SMD  SMA  CPI  CTI  B  SB  REV  DSt  BSt  DTr  DCF  L  LV  L&S
Y  Y  Y  N  Y  Y  Y  Y  Y  Y  Y  N  N  N  N  Y
**BASIC MOTION STEP + ENVIRONMENT:**

- **css**: change sensor state
- **stop_**: stop
- **wait_**: wait
- **start_**: start
- **start_con**: start command
- **stop_con**: stop command
- **in**: input
- **out**: output
- **ready_to_stop**: ready to stop
- **start_command**: start command
- **running**: running

**MACRO NET OF BASIC MOTION STEP:**

- **in**: input
- **start_con**: start command
- **stop_con**: stop command
- **out**: output

**fusion nodes:**
- **interface**
- **actuator states**
- **sensor states**

**formal parameters**

**actual parameters, e.g.:**

<table>
<thead>
<tr>
<th>press_forge</th>
<th>press_lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>press_at_middle_pos</td>
<td>press_at_lower_pos</td>
</tr>
<tr>
<td>press_upward</td>
<td>press_up</td>
</tr>
<tr>
<td>press_stop</td>
<td>press_stop</td>
</tr>
<tr>
<td>press_at_upper_pos</td>
<td>press_at_middle_pos</td>
</tr>
</tbody>
</table>
**Main Lessons Learnt**

- management of medium-sized Petri nets
  - hierarchical structure + fusion nodes;

- the whole model is composed of a few patterns
  - bounded producer/consumer pattern
  - communication patterns for producer/consumer pipeline
    - independent input/output
    - dependent input/output
    - mutually exclusive input/output
  - mutex pattern
  - basic motion step pattern
    - sequence
    - alternative

- new editor feature: parameter substitution
  - library of reusable Petri net components;

- interleaving rule of communication and mutex synchronisation
  - lock a mutex resource always as late as possible

- pattern properties
  - model consistency criteria
  - to be generated for each instance

**Other Concurrency Patterns**

- **n mutex resource pattern**
  - n=2: dining philosophers
  - pattern property (deadlock freedom):
    - acyclic access structure,
    - hierarchical resource organisation [Brinch Hansen]

- **unbounded producer/consumer pattern**

- **client/server communication pattern**

- **n layered client/server pattern**
  - pattern property (deadlock freedom):
    - acyclic communication structure

- **fault-tolerant basic structures**
  - n version programming
  - recovery block scheme
**N Version Programming**

-> parallel execution of n program versions, followed by majority test

**Recovery Block Scheme**

-> alternative execution of n program versions, each followed by acceptance test
CONCLUSIONS

- catalogue of concurrency patterns
- step-wise system development
  + step-wise specification of system properties

- properties taxonomy

  taxonomy I
  - general properties
    - boundedness
    - liveness
  - special properties
    - safety properties
    - progress properties
  - model consistency properties

  taxonomy II
  - “must” properties -> fatal errors
  - “maybe” properties -> warnings
  - “fun” properties -> insights

PROPERTY TAXONOMY II

- FATAL ERRORS
  e.g. safety properties
  If a robot arm is loaded, its magnet is not deactivated until the robot is in its unloading position.

\[
G(\varphi \rightarrow \neg \chi U \psi), \text{ where}
\]

\[
\varphi = \text{arm1\_mag\_on} \land \text{arm1\_pickup\_angle} \land \text{arm1\_pickup\_ext} \\
\chi = \text{arm1\_mag\_off} \\
\psi = \text{arm1\_release\_angle} \land \text{arm1\_release\_ext}
\]

- WARNINGS
  e.g. liveness

\[
\text{AG}(\text{EF en}(t)) \text{ for each transition } t
\]

- INSIGHTS
  Is it possible, that both robot arms carry a plate at the same time?

\[
\text{EF}(\text{arm1\_mag\_on} \land \text{arm2\_mag\_on})
\]
CHALLENGES

- structured (sequential) programming [Dijkstra]
  -> to avoid uncontrolled use of goto’s

  STRUCTURED CONCURRENT PROGRAMMING
  -> to avoid uncontrolled use of synchronisation / communication

- problem frames
  architecture styles
  design patterns
  idioms