A model-based monitoring approach for safety-critical cyber-physical systems

Federico Aromolo, Cosimo Antonio Prete, Pierfrancesco Foglia, Gabriele Antonio De Vitis
Department of Information Engineering – University of Pisa, Italy

IWES 2017 – 2nd Italian Workshop on Embedded Systems
Computer Science Department – Sapienza University of Rome, Italy
September 7-8, 2017

UNIVERSITÀ DI PISA

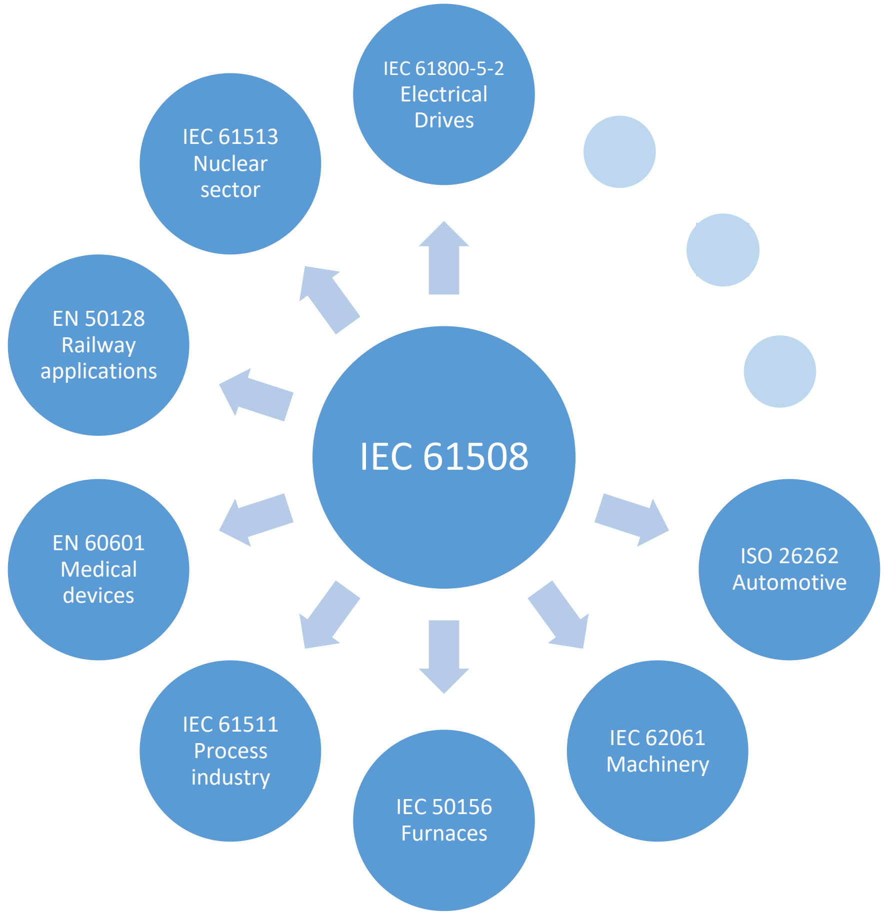
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Motivations

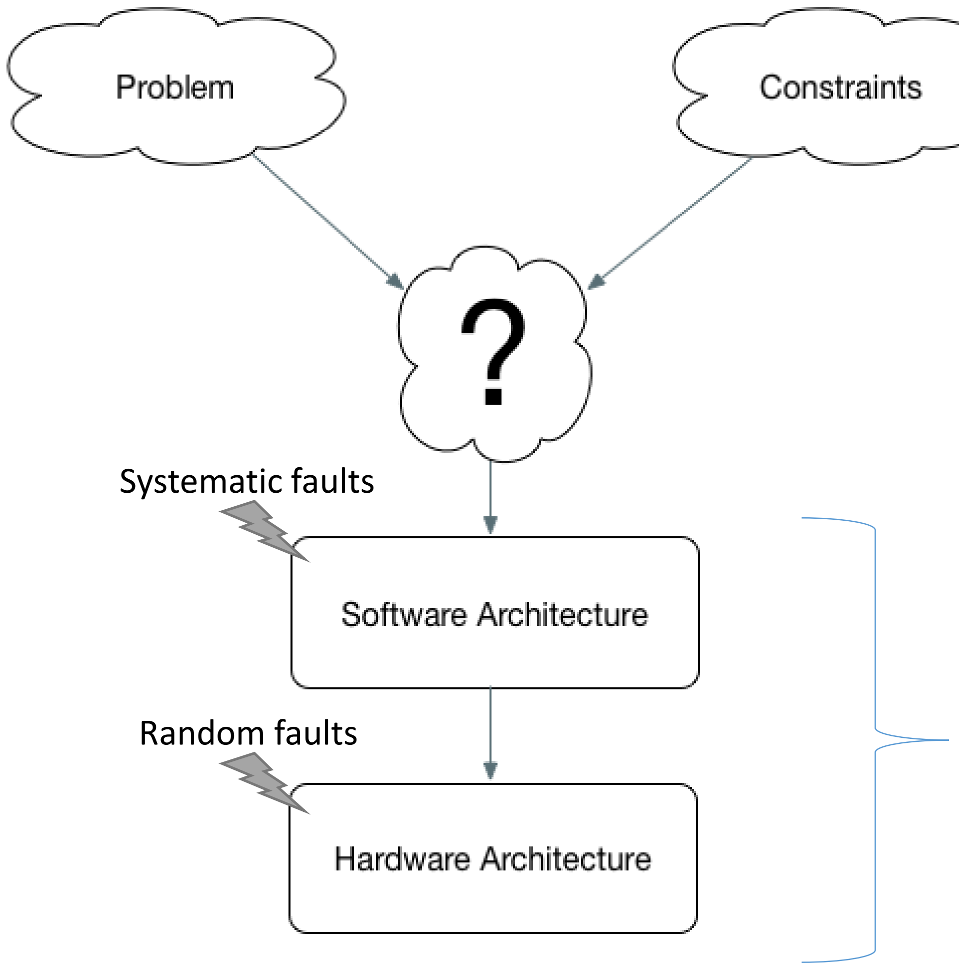
- The continuous technological advancements in the domain of cyber-physical systems allow designers to devise highly integrated systems of increasing complexity exhibiting intelligent and adaptive behaviors
- These systems are able to replace the humans-in-the-loop component to integrate higher-level logic in real-time control
  - E.g., autonomous vehicles, industrial automation, medical systems, …
  - Operation in open and constantly changing environments
- Safety is one of the key concerns in the development of such systems
  - Requires increased development and verification efforts

Motivations

- The concept of functional safety was introduced to deal with the impossibility of complete system testing, while providing safety guarantees in the development of critical systems
- Based on a quantitative measure of dependability
  - E.g., probability of failure per hour
- Iterative refinement procedure based on the application of well-known techniques



Motivations



Safety functions are typically expressed in qualitative or quantitative terms concerning high level behaviors

Most of the standardized techniques for functional safety rely on low level solutions, aimed at the reduction of the probability of safety requirements violation due to random failures in the hardware

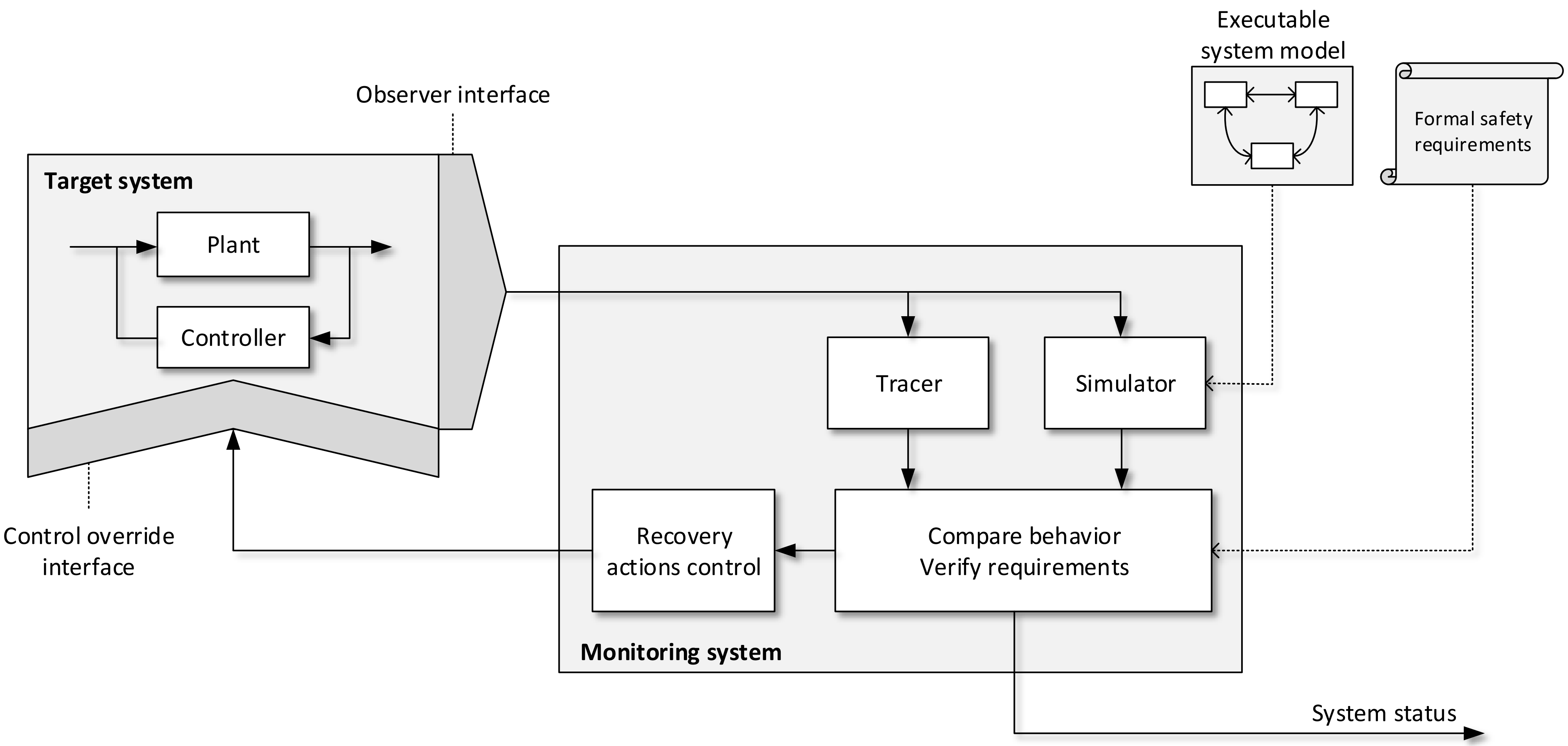
Background

- Functional safety
- Model-based systems engineering
- Formal verification
  - Model checking
- Runtime verification
- Simulation
- PLC design and implementation for industrial systems
- Supervisory control theory and its derivatives
  - Supervisor synthesis for discrete control systems
- Model-predictive control
- Autonomous guided vehicles and multi-agent systems

Objectives

- Improve system reliability with online simulation-based system monitoring in the context of a strongly automated development environment
  - Verification of behavioral consistency with respect to the models used for code generation and implementation
  - Verification of safety properties at a high level of abstraction
  - Intercept both random and systematic faults by analyzing high-level and system-level behaviors
    - E.g., erroneous subsystem interaction, faulty actuator or sensor, software bug
  - Used for both static and runtime system-level verification
- Analyze the possible applications of predictive monitoring approaches for advanced control schemes

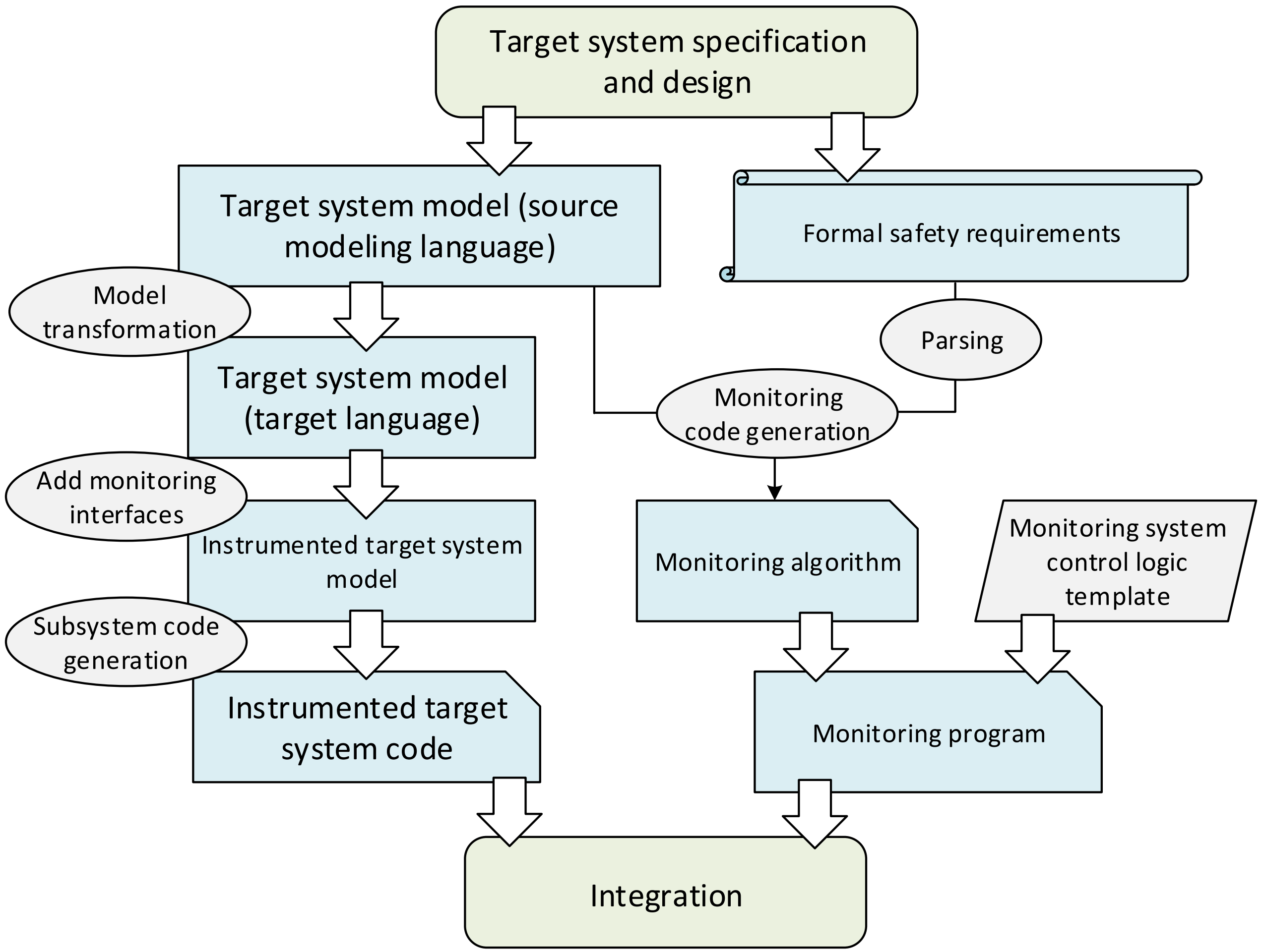
Simulation-based monitoring approach



Simulation-based monitoring approach

- At each time step:
  1. Extract the target system states and variables
  2. Initialize a simulation instance with the observed state as initial conditions
  3. Perform one or more simulation steps of an executable system model
  4. Compare the expected behavior with the actual system behavior and verify safety properties
  5. If necessary, perform a recovery action
     - E.g. modify control parameters, perform an emergency stop, notify the operator
  6. Store execution trace and logging data

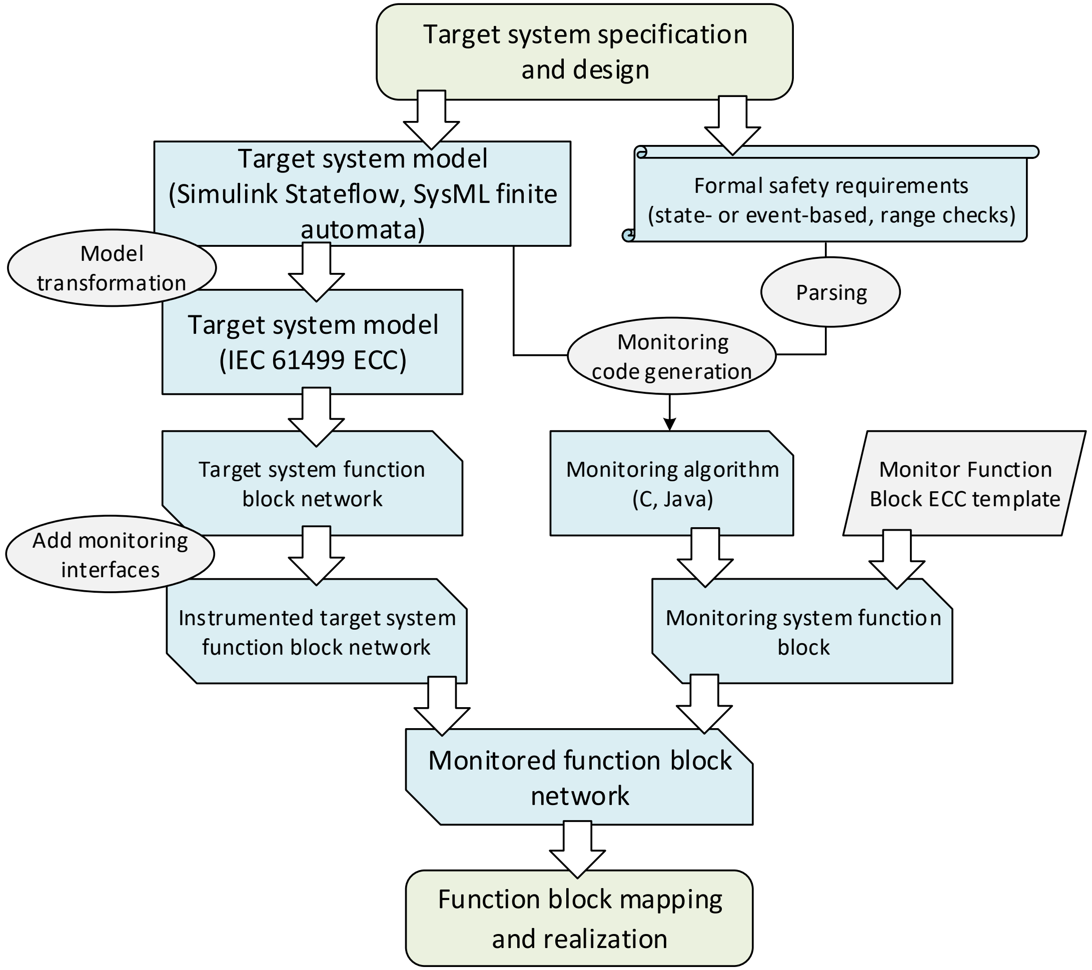
Overview of the general development process



Development process instantiation: IEC 61499

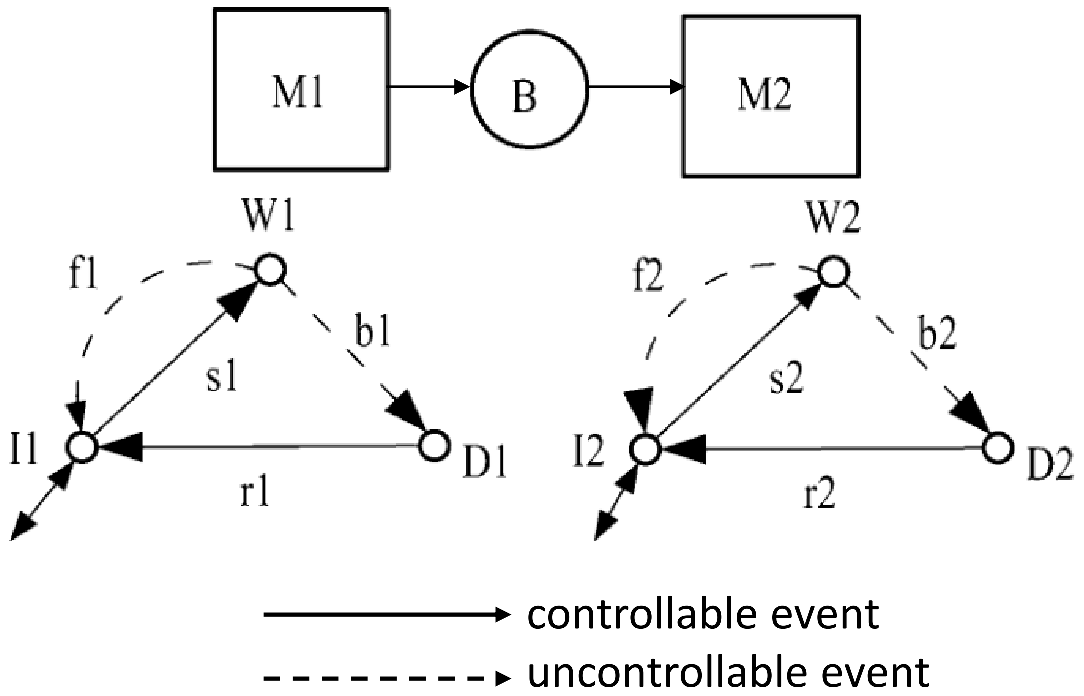
- IEC 61499 is a standard for PLC systems engineering which is widely adopted in the industrial field
  - Support for distributed discrete-event control systems
- The proposed approach can be easily adapted for use with IEC 61499
  - Fitting model of computation
    - Support for Execution Control Charts (ECC), closely related to finite automata
    - Manages synchronization, concurrency and event dispatching between subsystems
  - Automated integration and implementation phases
  - Support for custom-coded modules
  - Can be complemented with supervisor synthesis and traditional reliability techniques

IEC 61499 development workflow



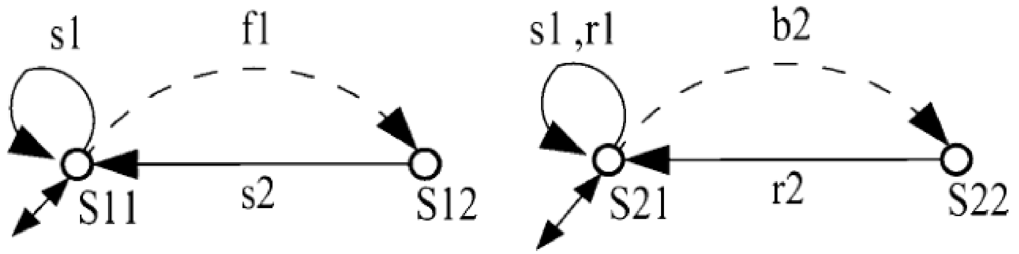
Example: the Small Factory extended process

- Two locally-controlled machines:
  - M1 takes a workpiece from an infinite input bin and puts it into the buffer after performing its work
  - M2 takes a workpiece from the buffer and places it into an infinite output bin after performing its work
  - Both M1 and M2 can break down while performing their work, and can be repaired
- Can be generalized to *n* machines
- Transformed into ECC models



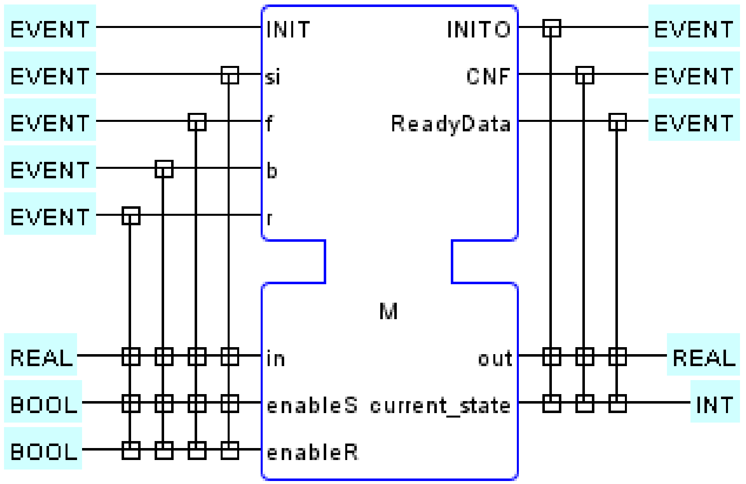
Example: specifications

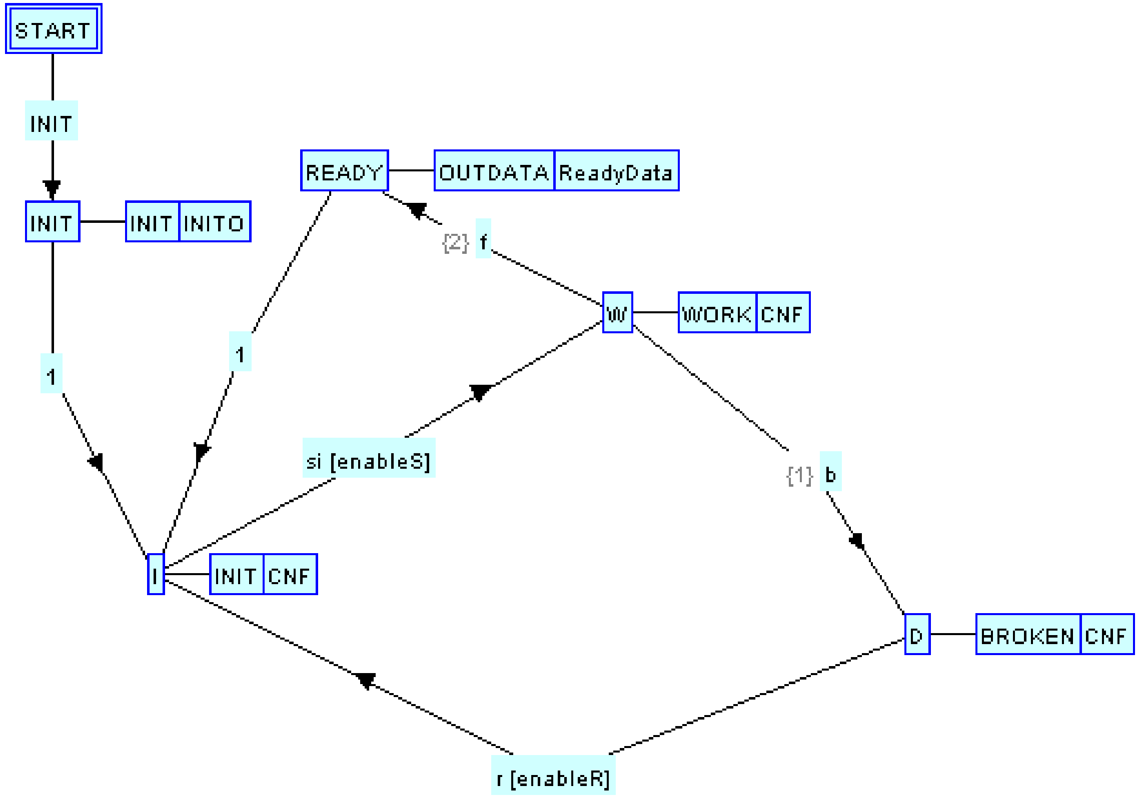
- The buffer has one slot, and it must not overflow nor underflow
- If M2 is broken down, M1 cannot start a work cycle and, if M1 is also broken down, M2 has to be repaired before M1
  - A simple supervisor for these specifications is given by the parallel composition of the two automata



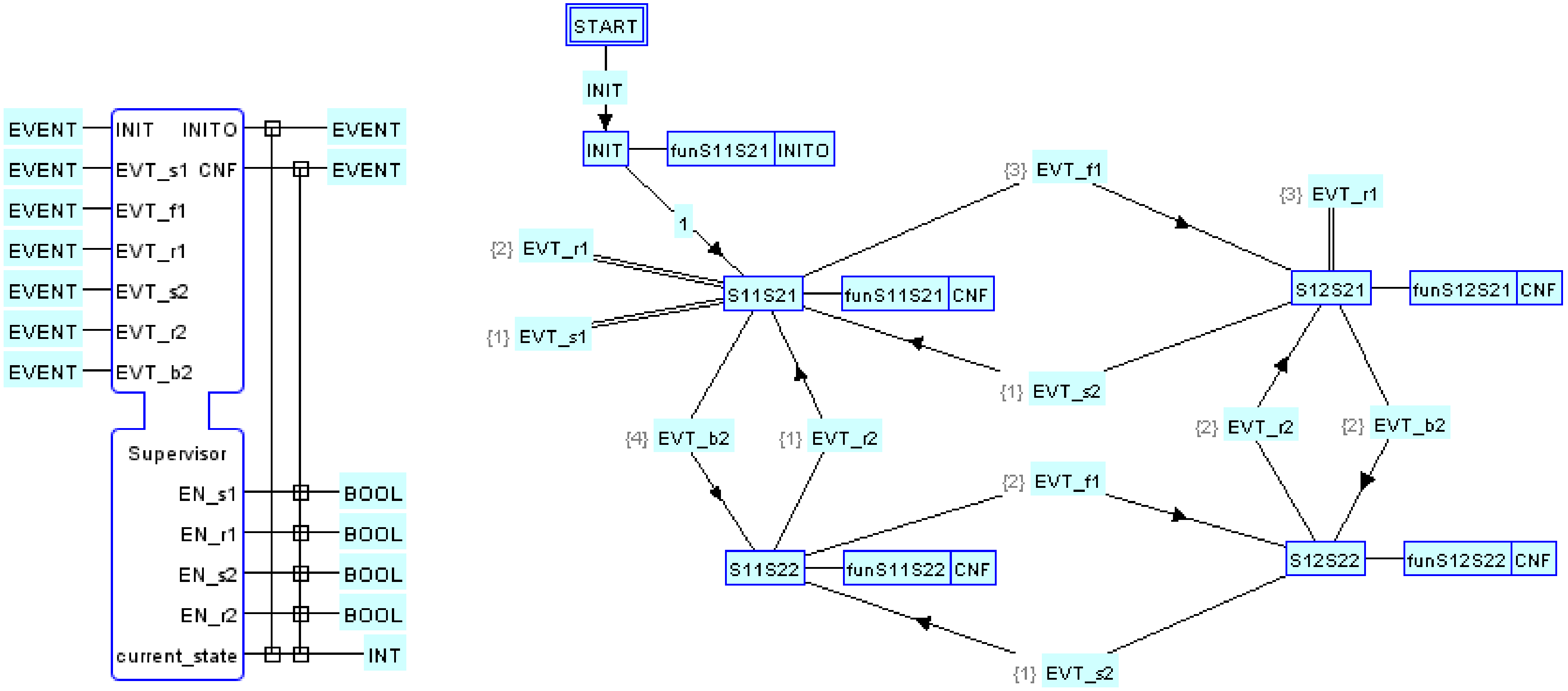
controllable event
uncontrollable event

Example: machine function blocks (ECC)



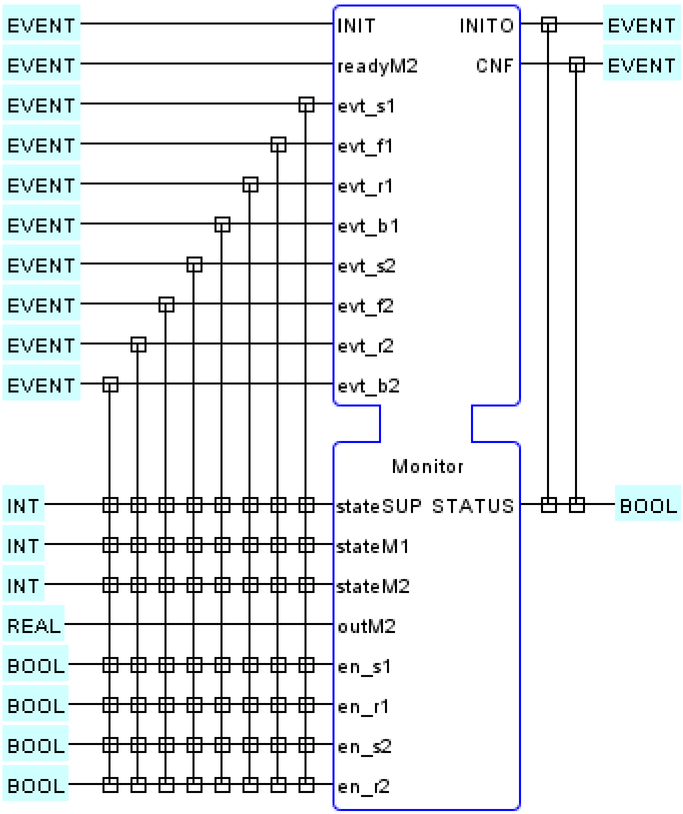


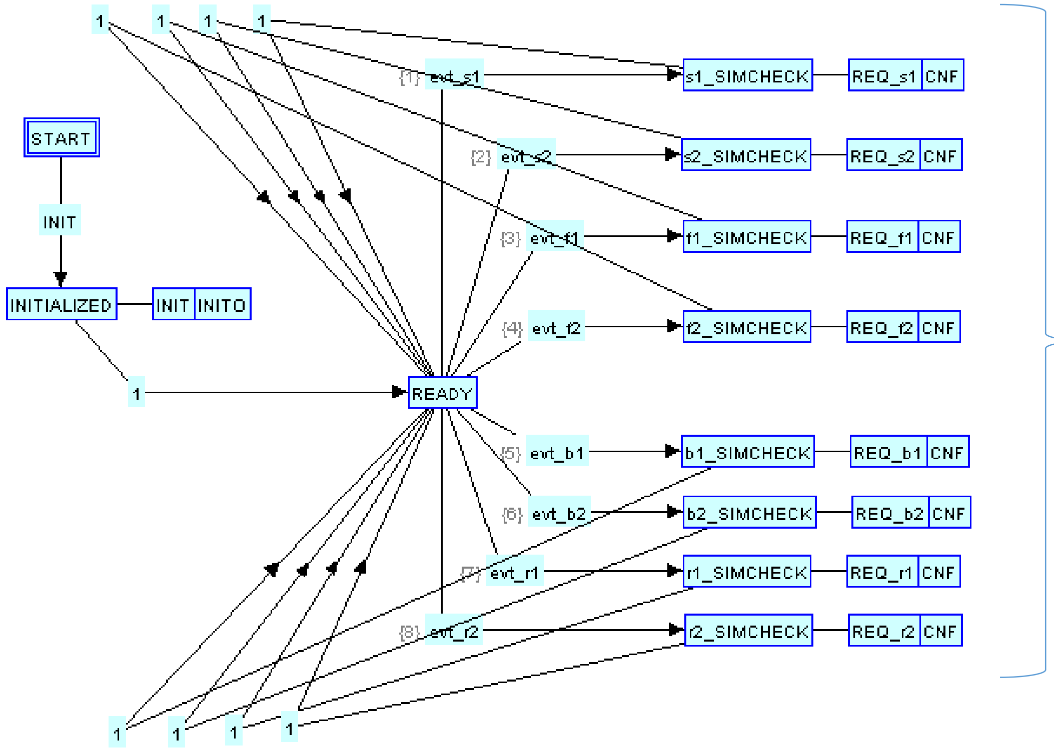
Example: supervisor synthesis (ECC)



State transitions set the control flags according to the specifications

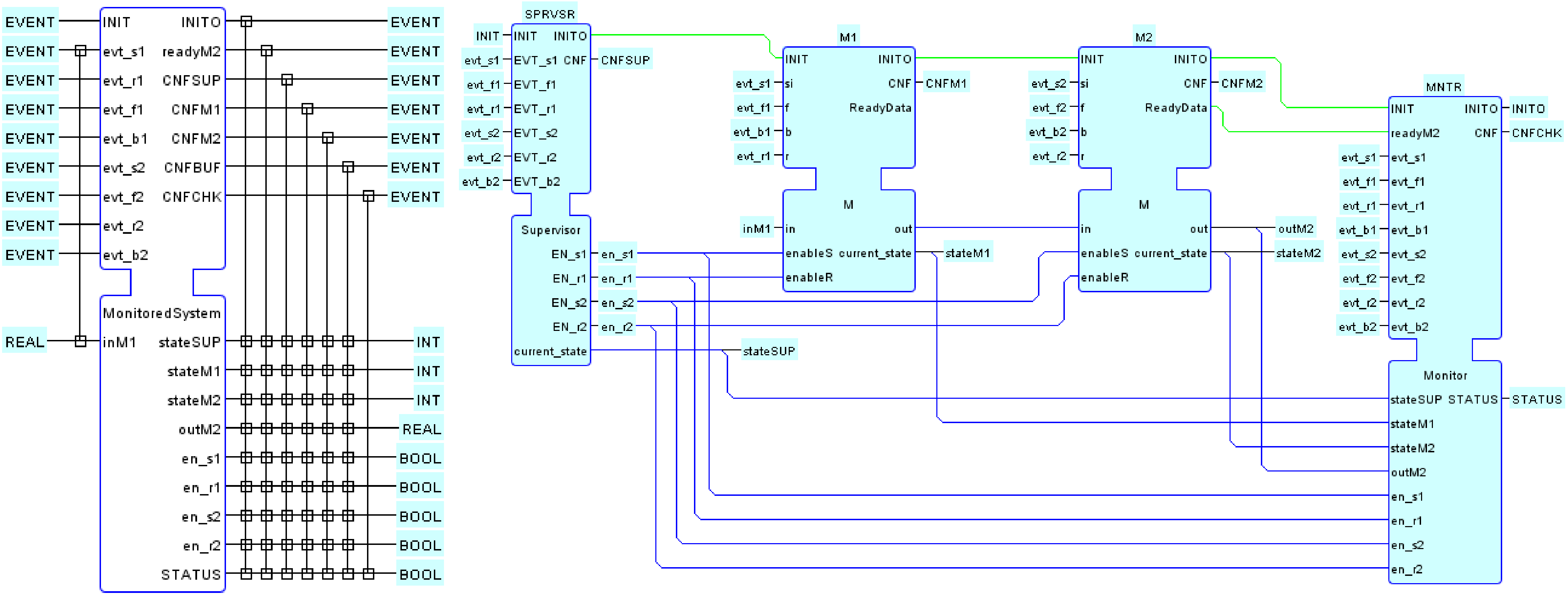
Example: monitor block (ECC + custom code)





Invocation of a custom Java module for simulation at each event trigger

Example: monitored system (FB network)



Future works and challenges

- Complete the IEC 61499 instantiation
  - Extend the support to well-known formal specification languages
    - E.g. linear temporal logic for quantitative safety properties
  - Remove the dependency from the specific RTSS
    - Use of fixed execution semantics
  - Performance and safety evaluation
    - Known functional safety analysis techniques for IEC 61499
- Experiment with continuous systems
  - Time model and synchronization, sampling, parameters selection, …
- Extend the monitoring system to support predictive monitoring
  - Advanced simulation and control techniques
  - Predictive simulations based on a number of possible future scenarios