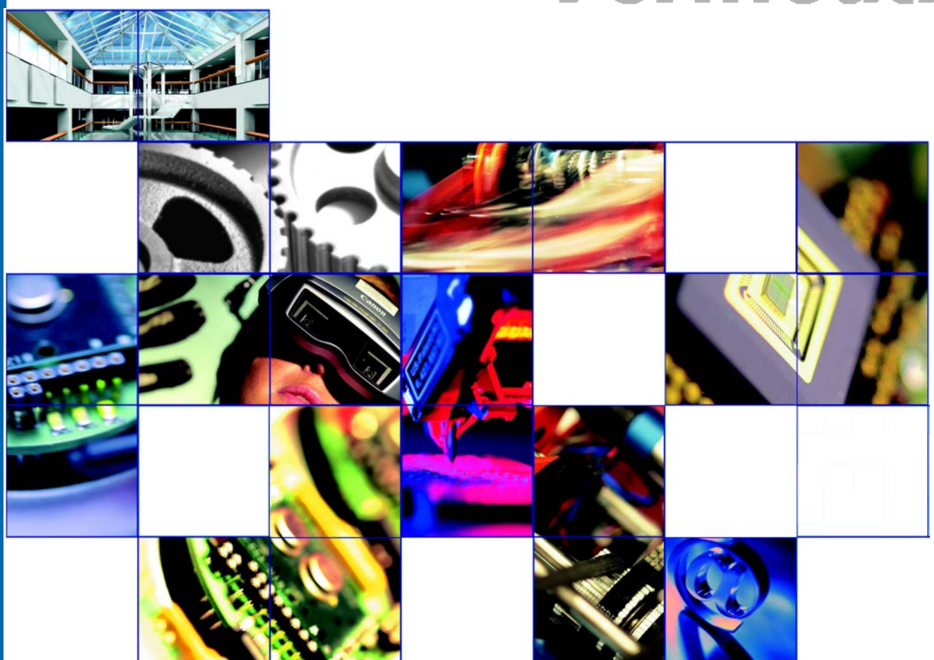
HEINZ NIXDORF INSTITUTE
University Paderborn
Design of Parallel Systems
Prof. Dr. rer. nat. Franz J. Rammig

Model-based Runtime Verification Framework



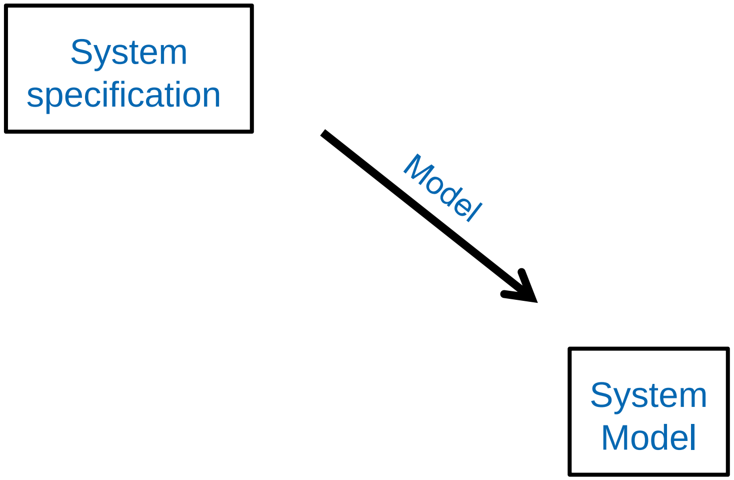
Yuhong Zhao
Heinz Nixdorf Institute
University of Paderborn
Germany

Outline

- Motivation
  - ✓ Model-driven Engineering
  - ✓ Verification and validation techniques
- Model-based runtime verification framework
  - ✓ Problem Statement
  - ✓ Pipelined working principle
  - ✓ Model Checking Methodology
  - ✓ Game between Runtime Verification and System Execution
    - ➢ Pre-checking and post-checking
    - ➢ Speedup strategies
      - Enrich system model with probabilities
      - Enrich system model with additional information
- Conclusion

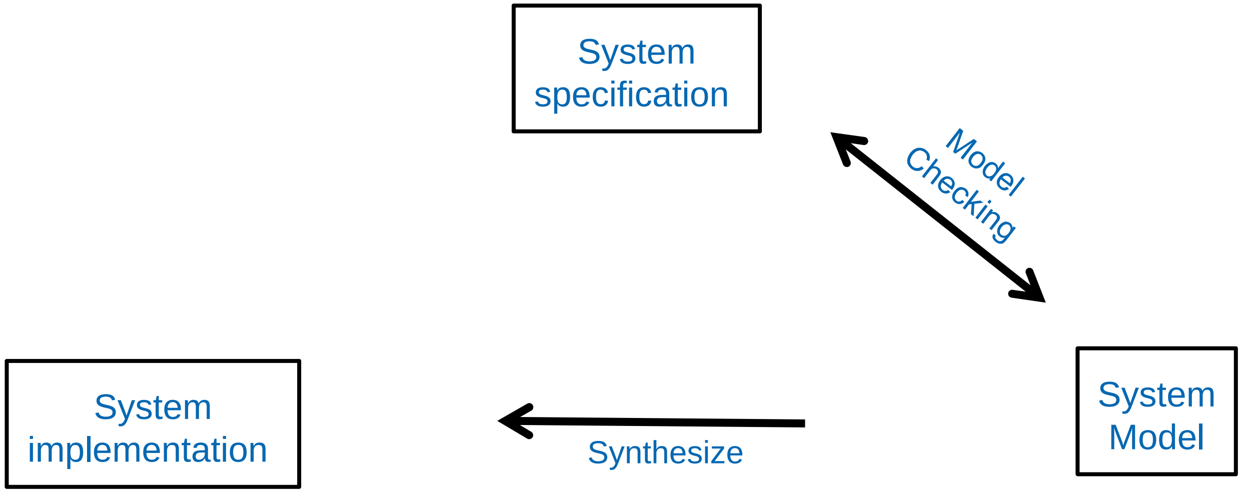
Motivation

- Model-driven Engineering (MDE)
  - ✓ Model system according to system specification
  - ✓ Verify system model against system specification
  - ✓ Synthesize system implementation (source code) from system model

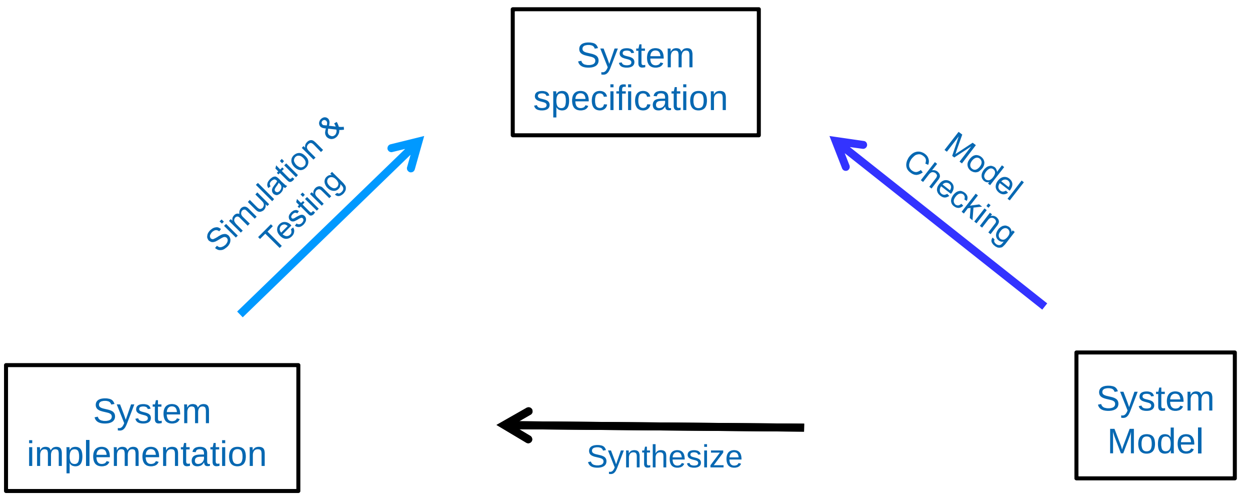


HEINZ NIXDORF INSTITUTE
University Paderborn
Design of Parallel Systems
Prof. Dr. rer. nat. Franz J. Rammig

- Model-driven Engineering (MDE)
  - ✓ Model system according to system specification
  - ✓ Verify system model against system specification
  - ✓ Synthesize system implementation (source code) from system model

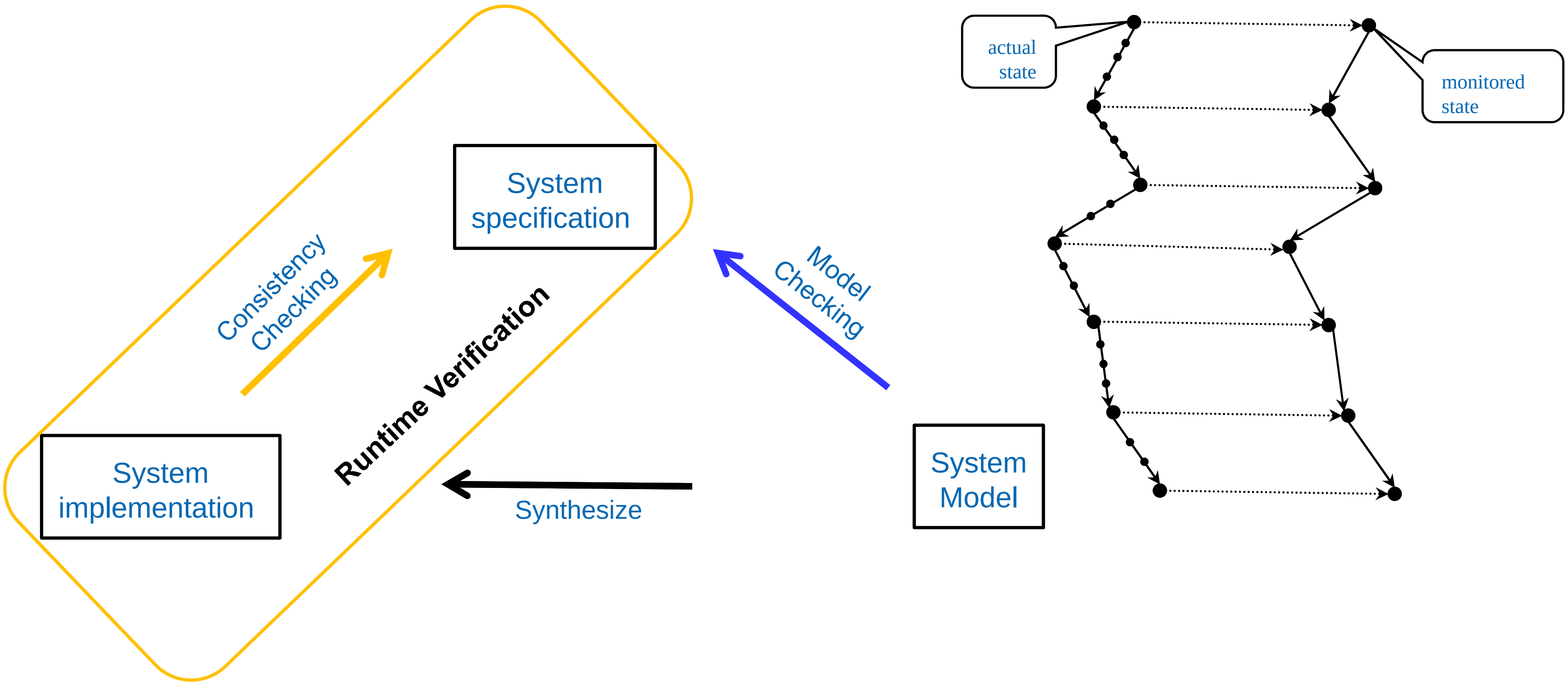


- Verification and Validation Techniques
  - ✓ **Off**-line Methods:
    - ➢ Model Checking (theorem proving)
      - Check all of the system behaviors
    - ➢ Simulation and Testing
      - Check some of the system behaviors



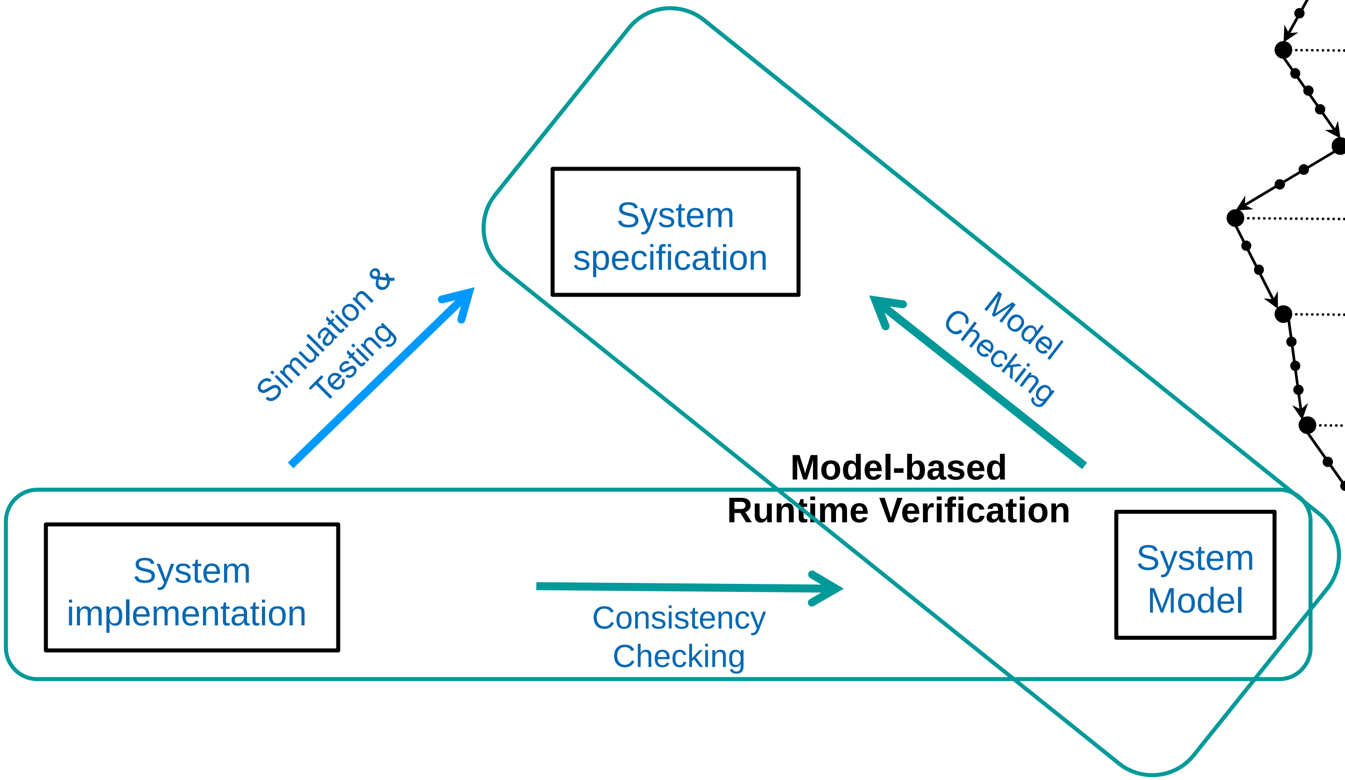
HEINZ NIXDORF INSTITUTE
University Paderborn
Design of Parallel Systems
Prof. Dr. rer. nat. Franz J. Rammig

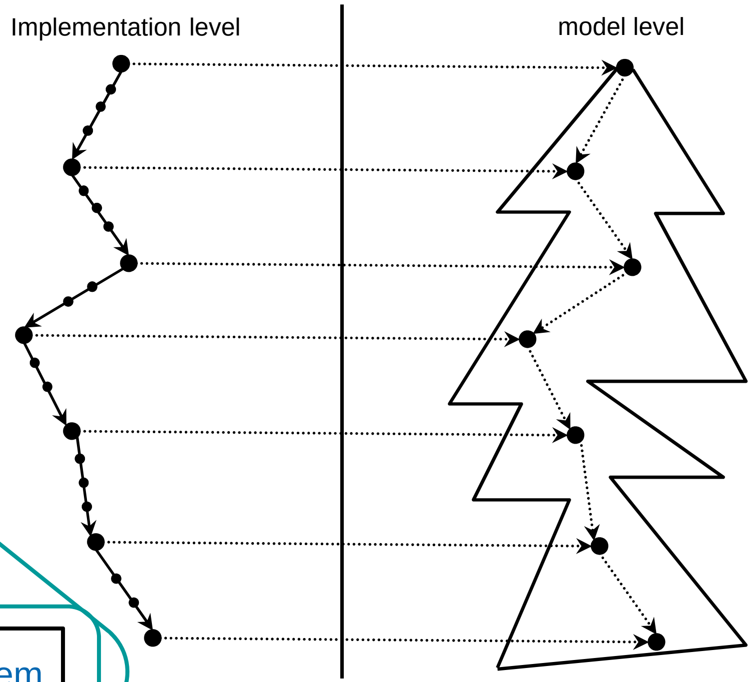
- Verification and Validation Techniques
  - ✓ **On**-line Methods:
    - ➢ State-of-the-art runtime verification



HEINZ NIXDORF INSTITUTE
University Paderborn
Design of Parallel Systems
Prof. Dr. rer. nat. Franz J. Rammig

- Verification and Validation Techniques
  - ✓ **On**-line Methods:
    - State-of-the-art runtime verification
    - Model-based runtime verification





HEINZ NIXDORF INSTITUTE
University Paderborn
Design of Parallel Systems
Prof. Dr. rer. nat. Franz J. Rammig

- As service of Real-time Operating System (RTOS)
- Application scenario

Consider a real-time system model M that

- ✓ contains n modules: M_1, M_2, ..., M_n working in parallel
- ✓ does reconfiguration at runtime by
  - case 1: M - M_i (remove an existing module M_i)
  - case 2: M + M_i' (add a new module M_i')

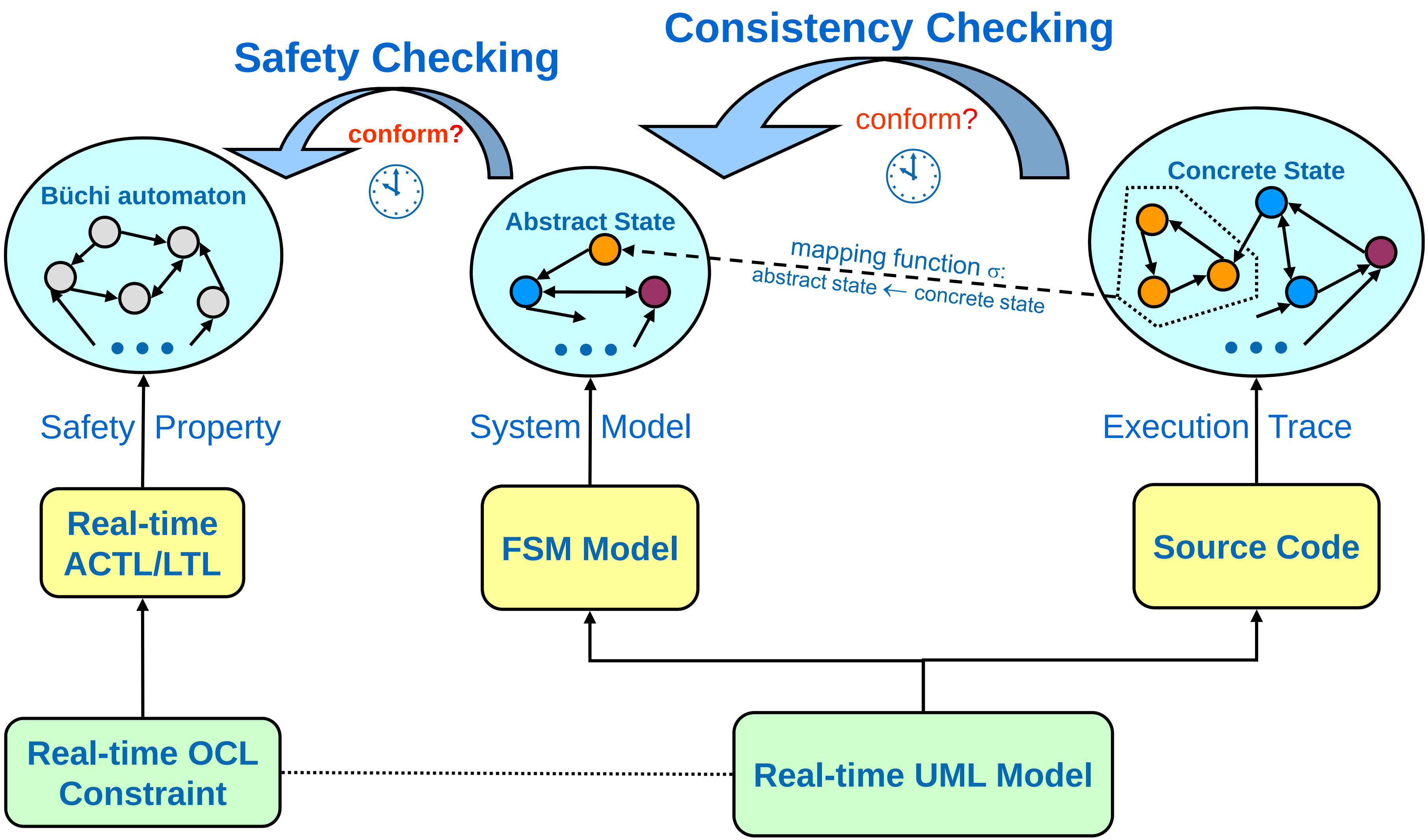
Requirements:

- Send reconfiguration request in advance to RTOS (at time instant t_r)

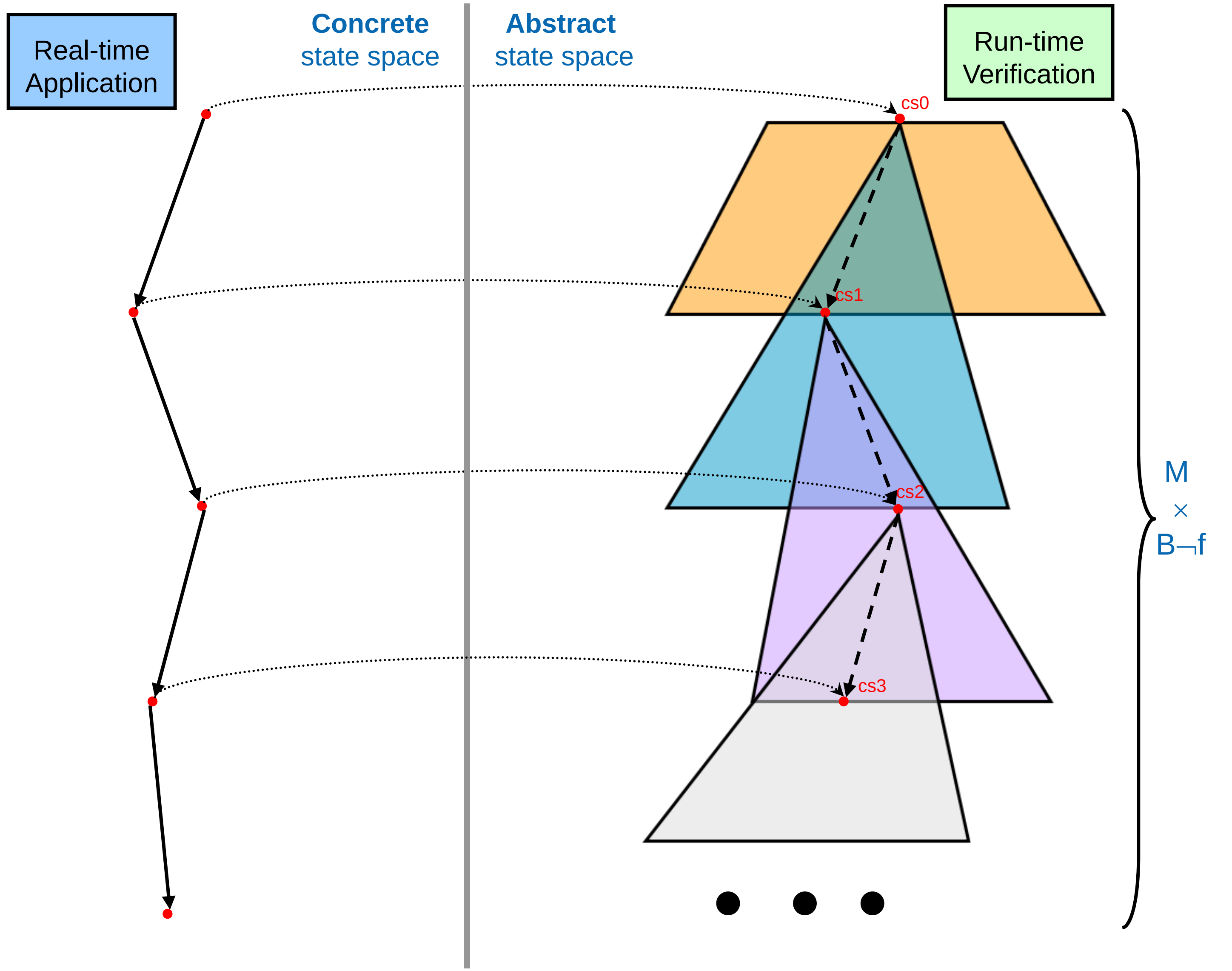
Goal:

- Get answer before the reconfiguration is really done (at time instant $t_0 > t_r$) from runtime verification service

HEINZ NIXDORF INSTITUTE
University Paderborn
Design of Parallel Systems
Prof. Dr. rer. nat. Franz J. Rammig



Model-based Runtime Verification Framework
Basic Idea

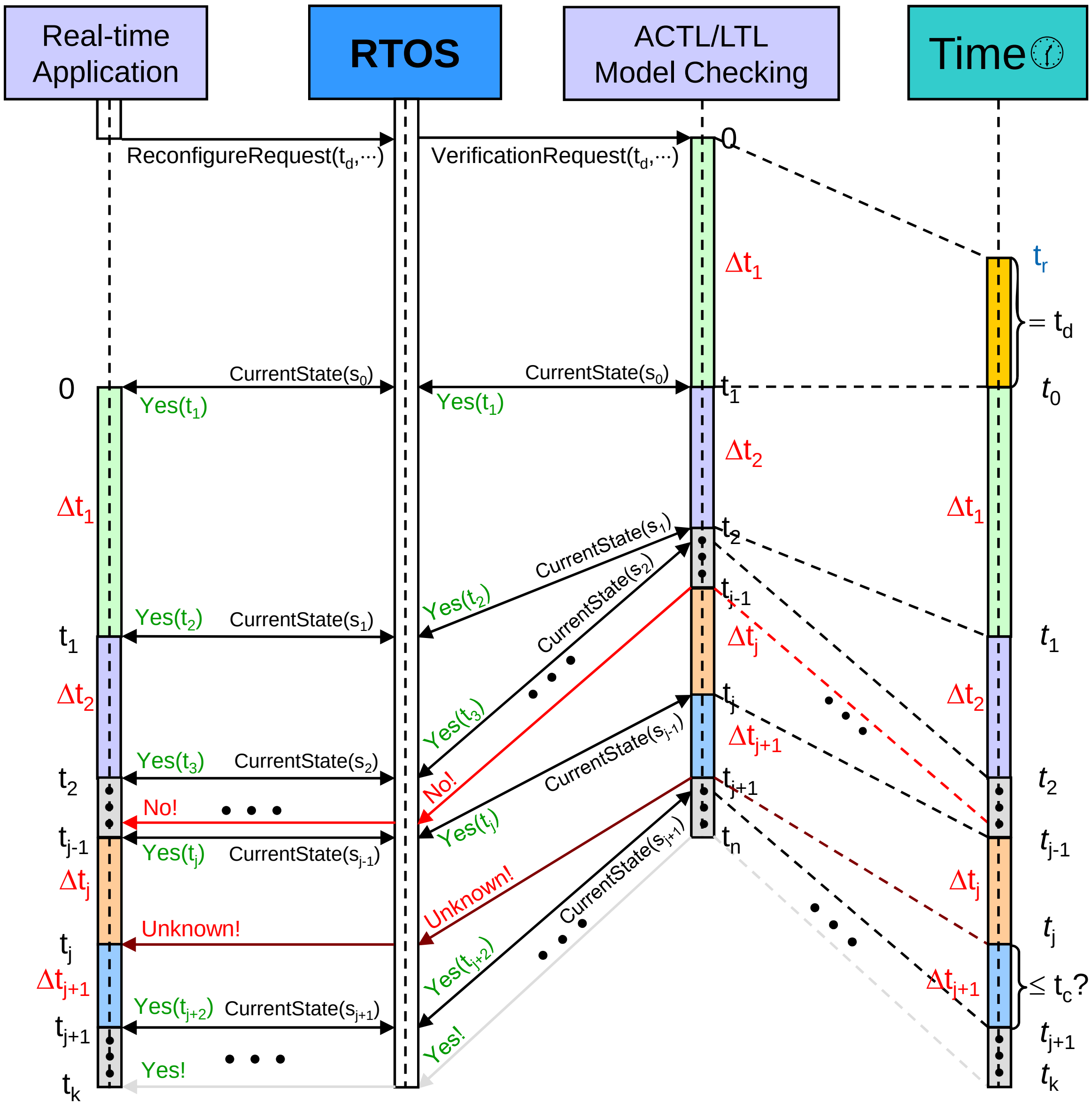


Goal:

checking safety and consistency by

looking ahead a subspace in system model that covers execution trace

Model-based Runtime Verification Framework
Pipelined Working Principle

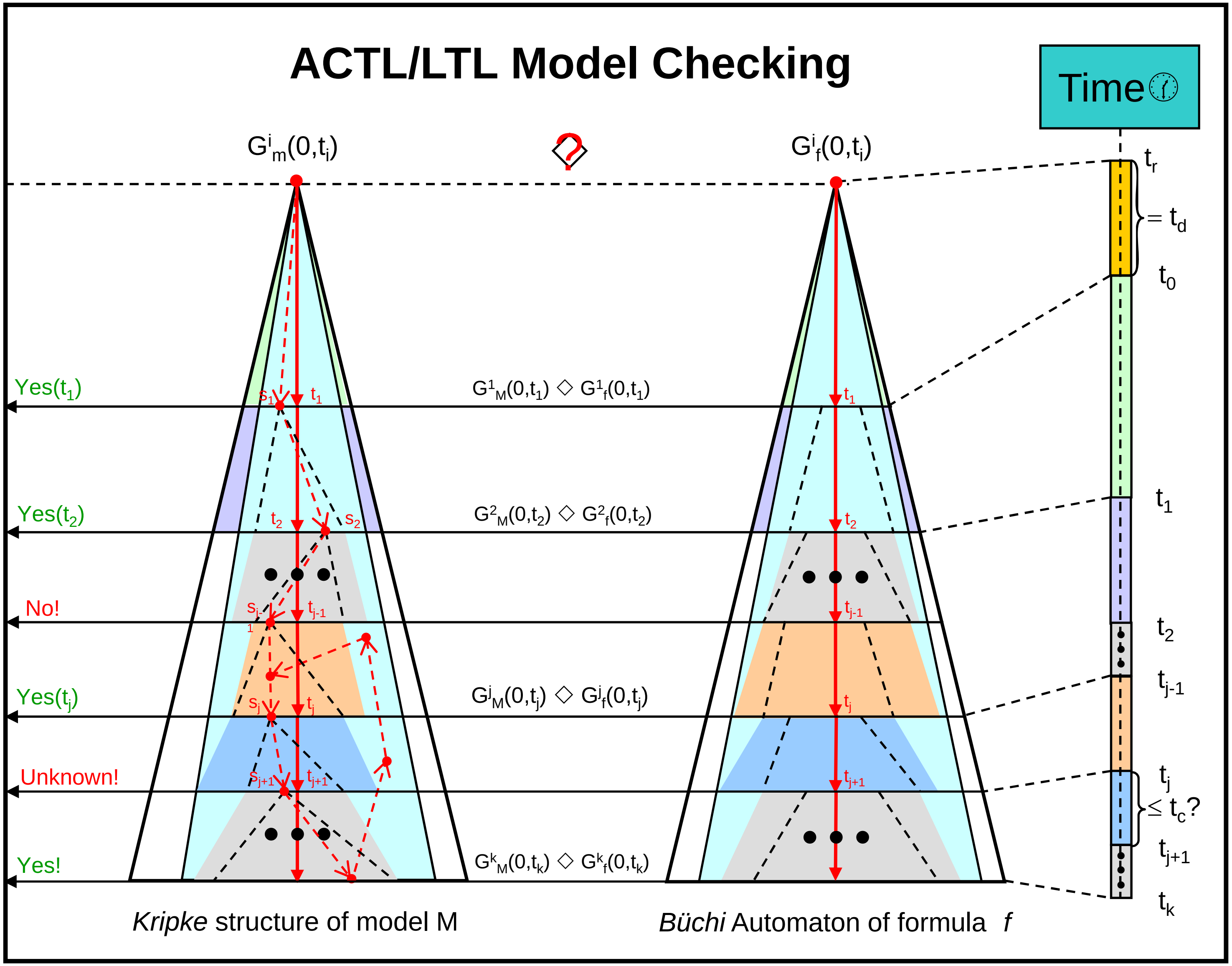


Suppose

- Components and Protocols between Components are checked correct at design phase
- Implementations of the systems conform to the corresponding models
- Properties to be checked are ACTL and LTL formulas
- Processing speed of the verification is faster enough than that of the application

Model-based Runtime Verification Framework

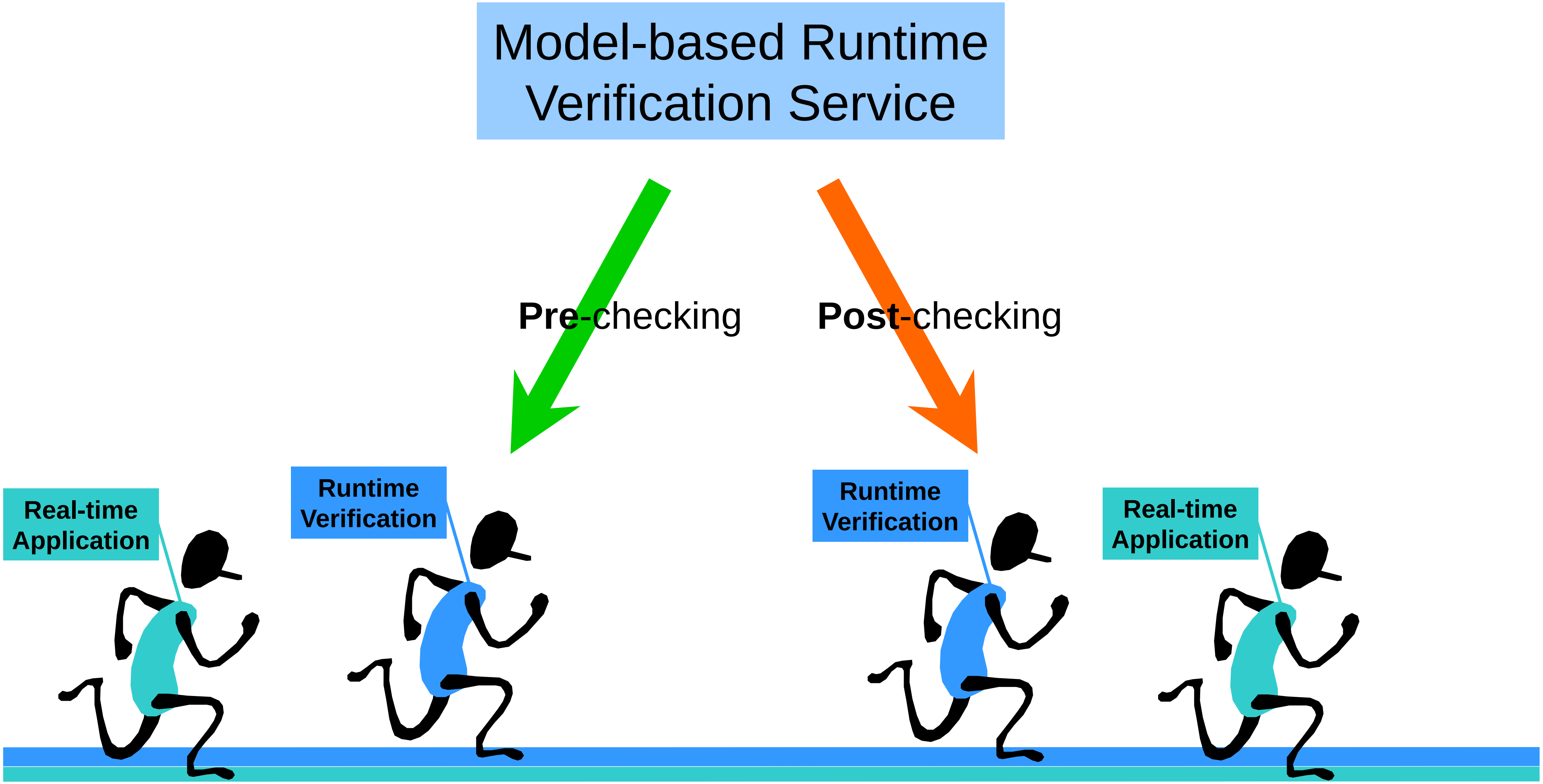
HEINZ NIXDORF INSTITUTE
University Paderborn
Design of Parallel Systems
Prof. Dr. rer. nat. Franz J. Rammig



Note:

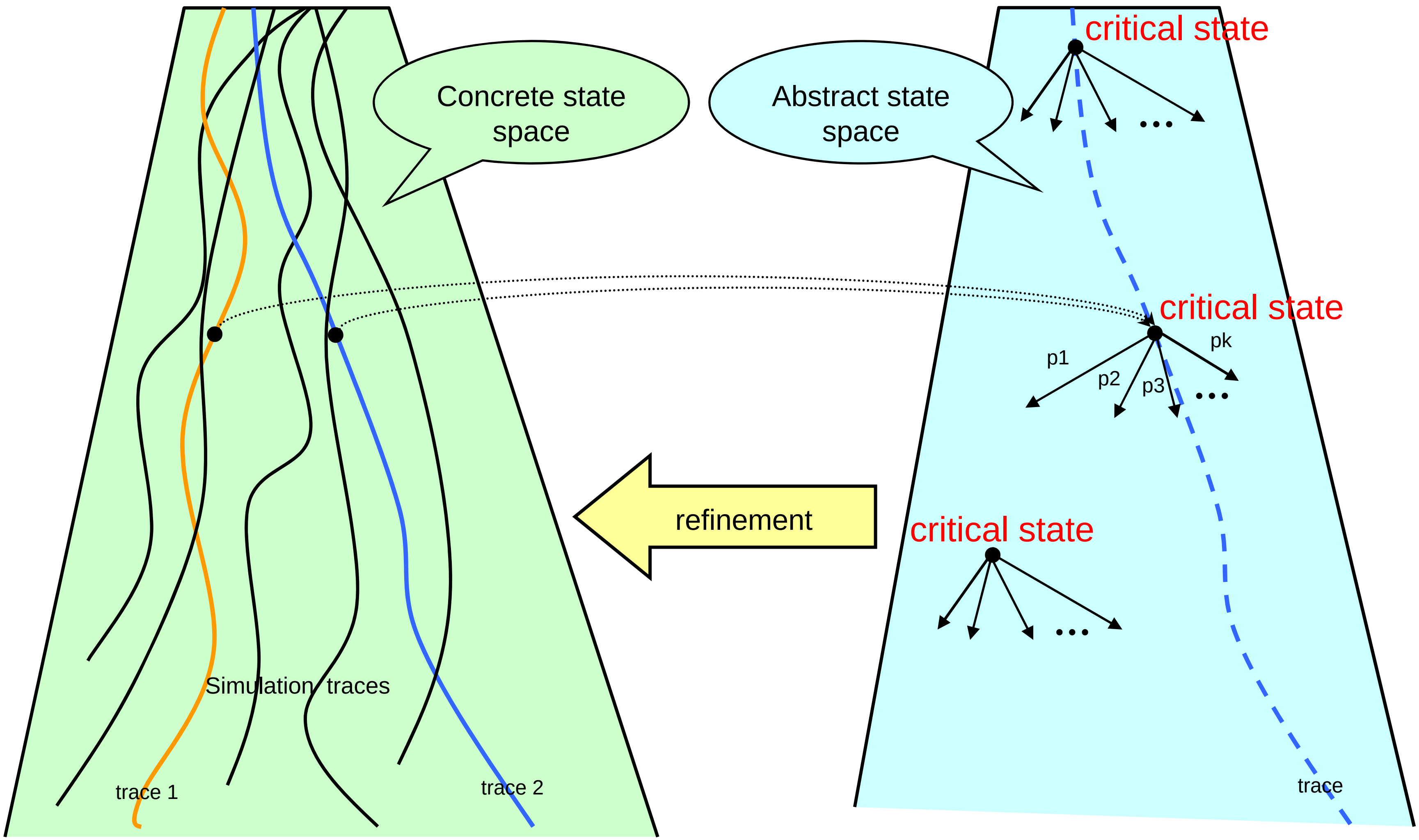
1. "◇" stands for "≤" (Simulation relation) for ACTL Model Checking; "◇" stands for "|=" (Satisfaction relation) for LTL Model Checking.
2. $G^i_m(0,t_i)$ $(1 \leq i \leq k)$ denotes the subgraph of the *Kripke* structure (system model) reachable from initial states within Δt_i steps.
3. $G^i_f(0,t_i)$ $(1 \leq i \leq k)$ denotes the subgraph of the *Büchi* automaton (ACTL/LTL formula) composible with $G^i_m(0,t_i)$.
4. t_d is the timing constraint required for verification.
5. t_c is the minimum time difference between verification and application.

Game between Runtime Verification and System Execution: Pre-checking and Post-checking



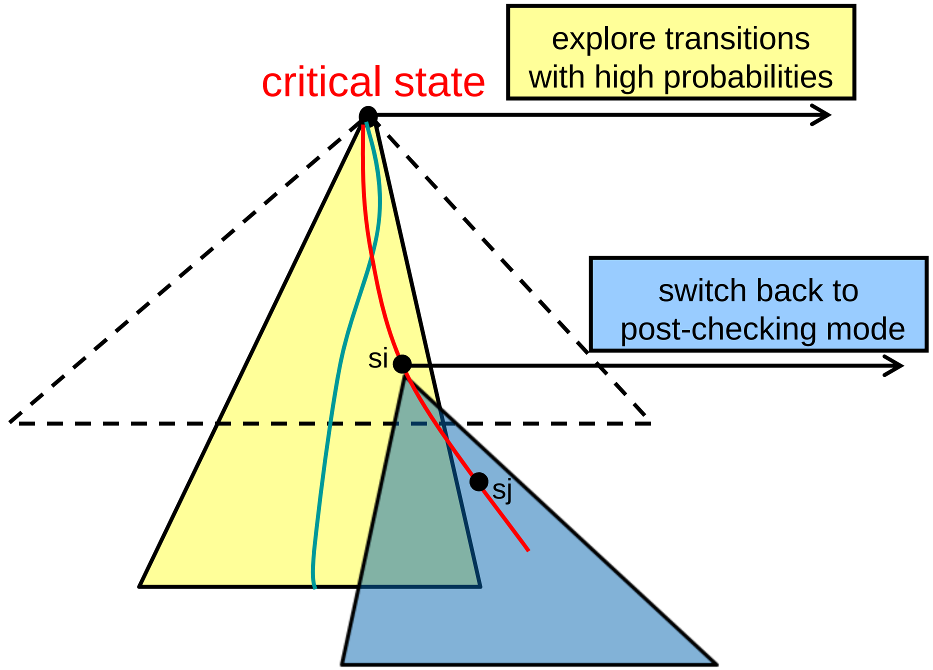
- **Goal:** make runtime verification in pre-checking mode for **as long time as possible** in course of system execution

- Enrich system model with probabilities



Game between Runtime Verification and System Execution: Speedup Strategies

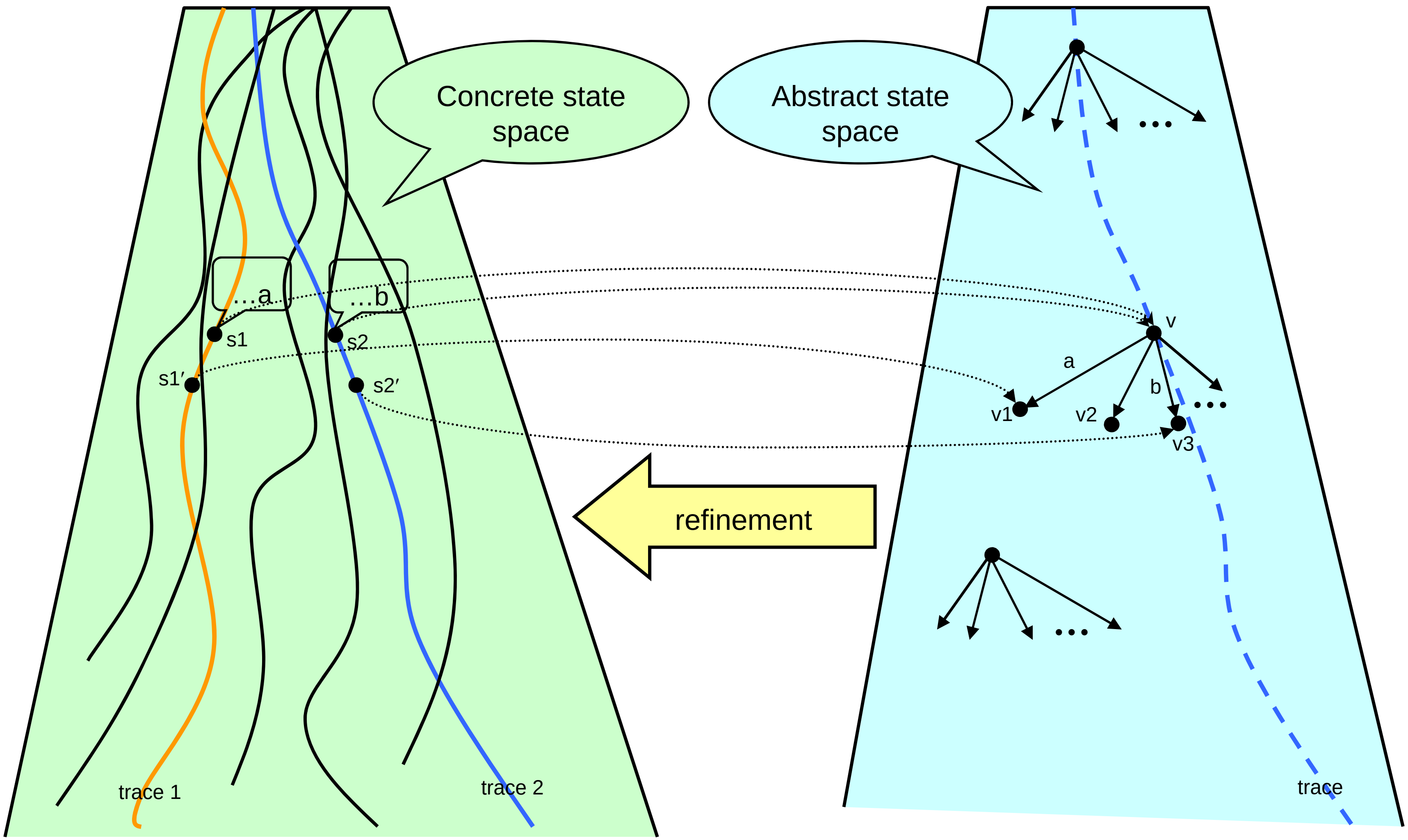
- Enrich system model with probabilities
  - ✓ Intentionally reduce state space to be explored



Game between Runtime Verification and System Execution: Speedup Strategies

HEINZ NIXDORF INSTITUTE
University Paderborn
Design of Parallel Systems
Prof. Dr. rer. nat. Franz J. Rammig

- Enrich system model with additional information



Conclusion

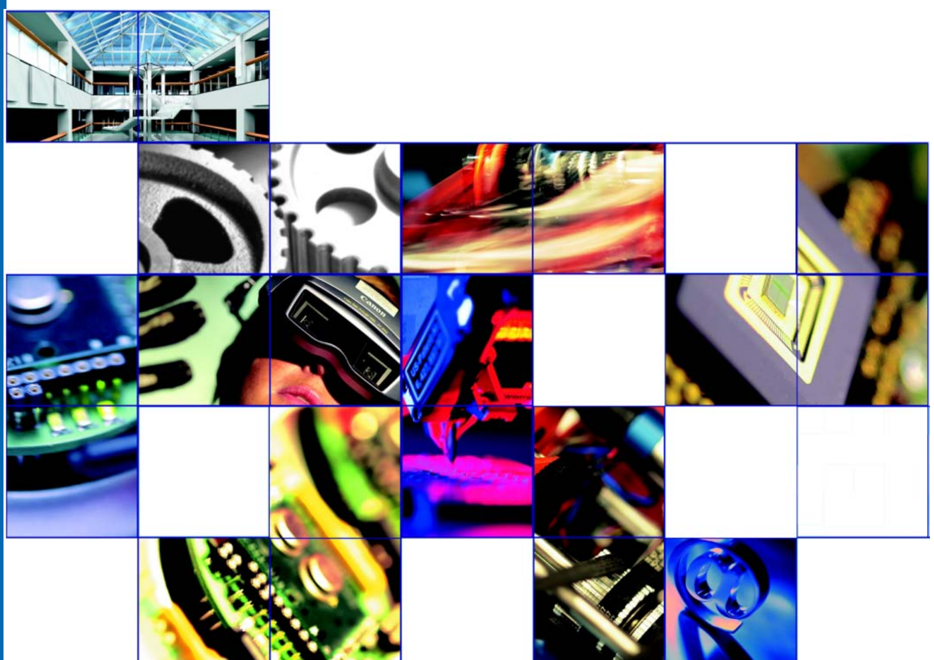
- Model-driven Engineering
  - ✓ System specification → System model →System implementation
- Verification and Validation Techniques
  - ✓ **Off**-line methods
    - ➢ Model-checking
      - Check all of the system behaviors
    - ➢ Simulation and testing
      - Check some of the system behaviors
  - ✓ **On**-line methods
    - ➢ State-of-the-art runtime verification
      - System implementation → System specification
    - ➢ Model-based runtime verification
      - System implementation → system model → system specification

Thank You for Your Attention

Question?

Advice?

……



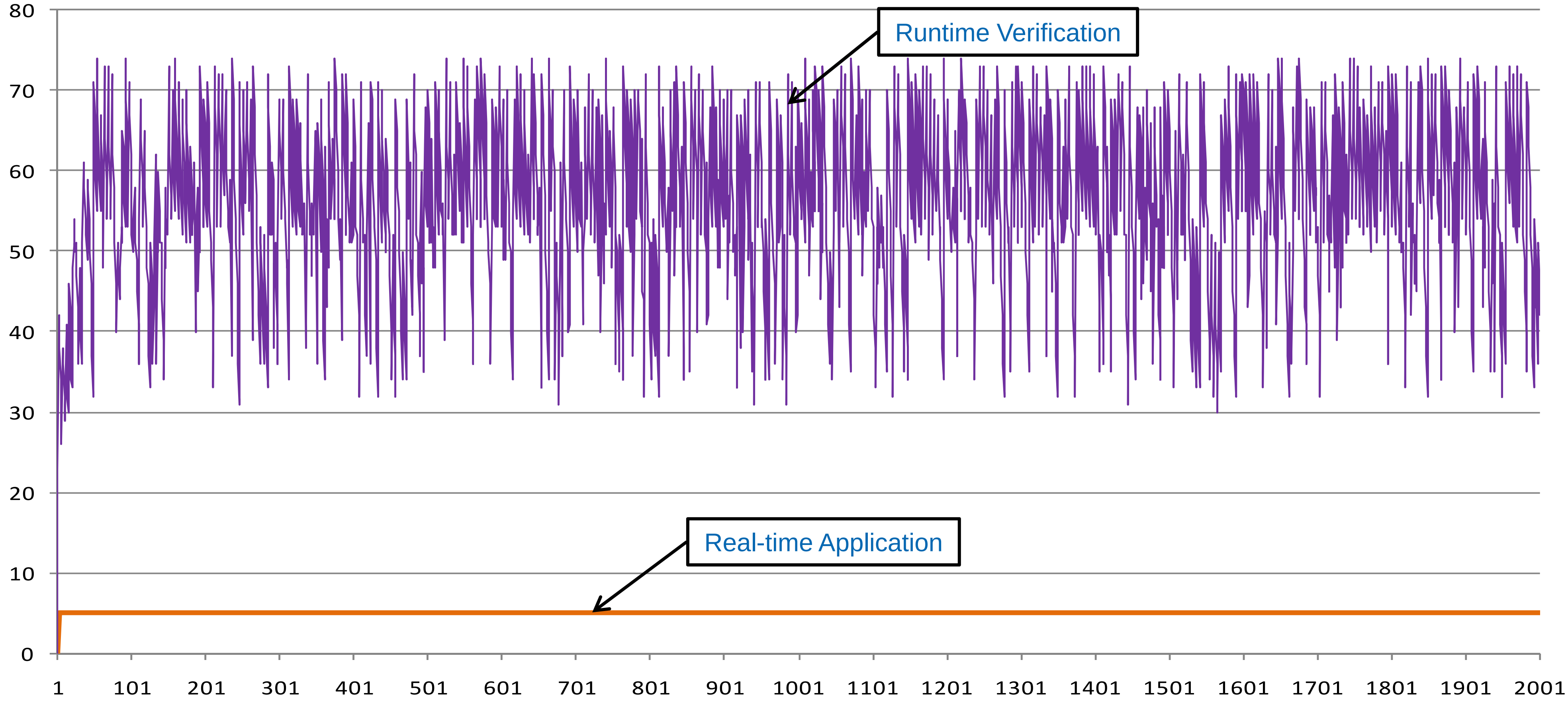
Runtime invariant checking

Modell	Typ	Zustand	Transition	Durch. Grad	Maximaler Ausg.-Grad	BFS Höhe	Max. Stack	Boolsche Variablen	Transition Schritt (ms)	Min. Vor-ausschau	Max. Vor-ausschau	Dur. Vor-ausschau
sorter_1	Controller	20544	30697	1,5	5	198	617	36	1	40	299	103
collision_1	Kommunikations protokoll	5593	10792	1,9	5	57	617	25	1	26	81	48,7
synapse_2	Protokoll	61048	125334	2,1	18	41	2349	46	1	7	28	21,5
driving_phils_2	Mutual exclusion algorithm	33173	81854	2,5	9	150	3702	27	1	31	97	65,7
blocks_1	Planung und Scheduling	7057	18552	2,6	6	19	4263	23	1	8	21	14
peterson_1	Mutual Exclusion Algorithmus	12498	33369	2,7	5	54	1862	30	1	13	39	31,7
szymanski_1	Mutual Exclusion Algorithmus	20264	56701	2,8	3	72	2064	27	1	13	90	49,7
hanoi_1	Puzzle	6561	19680	3	3	256	4376	36	1	56	103	75,9
iprotocol_2	Kommunikations protokoll	29994	100489	3,4	7	91	443	39	1	18	451	50
phils_3	Mutual Exclusion Algorithmus	729	2916	4	6	17	518	18	1	156	357	265
cyclic_scheduler_1	Protokoll	4606	20480	4,4	8	55	1819	40	1	23	437	278
rushhour_1	Puzzle	1048	5446	5,2	9	73	535	28	1	66	248	150,7
rushhour_2	Puzzle	2242	12603	5,6	10	80	906	32	1	36	408	116,4
pouring_1	Puzzle	503	4481	8,9	9	13	348	16	1	42	101	71,9
reader_writer_2	Protokoll	4104	49190	12	19	13	4097	25	1	4	16	9,9
pouring_2	Puzzle	51624	1232712	23,9	25	15	44509	18	1	1	4	2

Note:
1) Transition in system model represents 1 millisecond;
2) Platform: Linux, Pentium 4 processor 3.00 Ghz, 1 G RAM.

HEINZ NIXDORF INSTITUTE
University Paderborn
Design of Parallel Systems
Prof. Dr. rer. nat. Franz J. Rammig

- Model: the driving philosophers
  - ✓ LTL formula: **G**(ac0 → **F**gr0) ---If process 0 requests a resource it will be granted to him eventually



Note:

1) Transition in system model represents 1 millisecond;
2) Platform: Linux, Pentium 4 processor 3.00 Ghz, 1 G RAM.