Model Based Development of Embedded Control Software

Part 7: TDL Time-Safety Checking

Claudiu Farcas

Credits: MoDECS Project Team, Giotto
Department of Computer Science
cs.uni-salzburg.at

UNIVERSITÄT SALZBURG

Contents

- Goals
- Concepts
- Optimal Schedulers
- Notations and basic notions
  - Constraints, attributes, wcet, task models,
- Approaches to real-time scheduling
- Deadline = Period
  - Fixed Priority Scheduling – RM
  - Dynamic Priority Scheduling - EDF
- Deadline < Period
  - Processor Demand for EDF
  - Critical Instant and Response Time Analysis for RM
- Single Processor, Single/Multiple TDL Modules
  - Composability Analysis Tool

UNIVERSITÄT SALZBURG

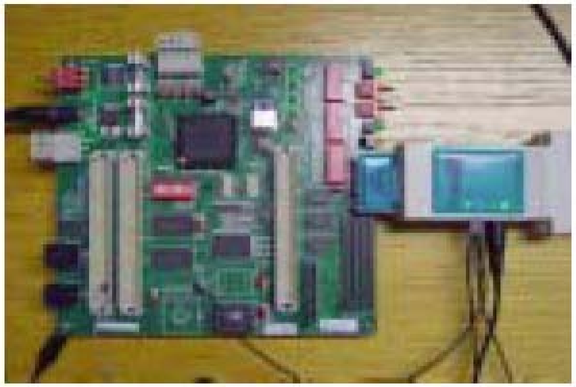
Time-Safety Checking– The Goals!

Applications

Automatic
assignment
to processors

?

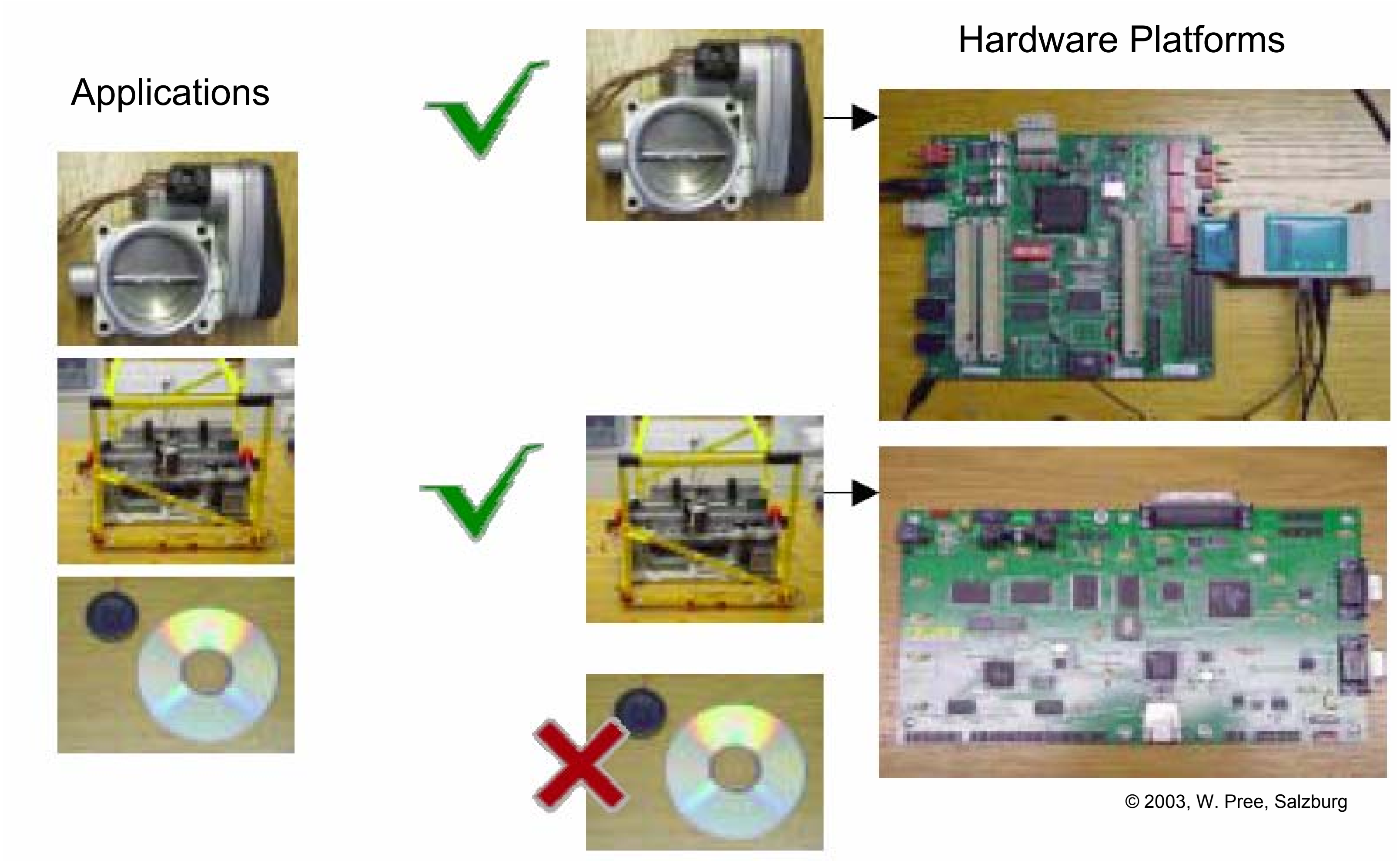
Hardware Platforms



© 2003, W. Pree, Salzburg

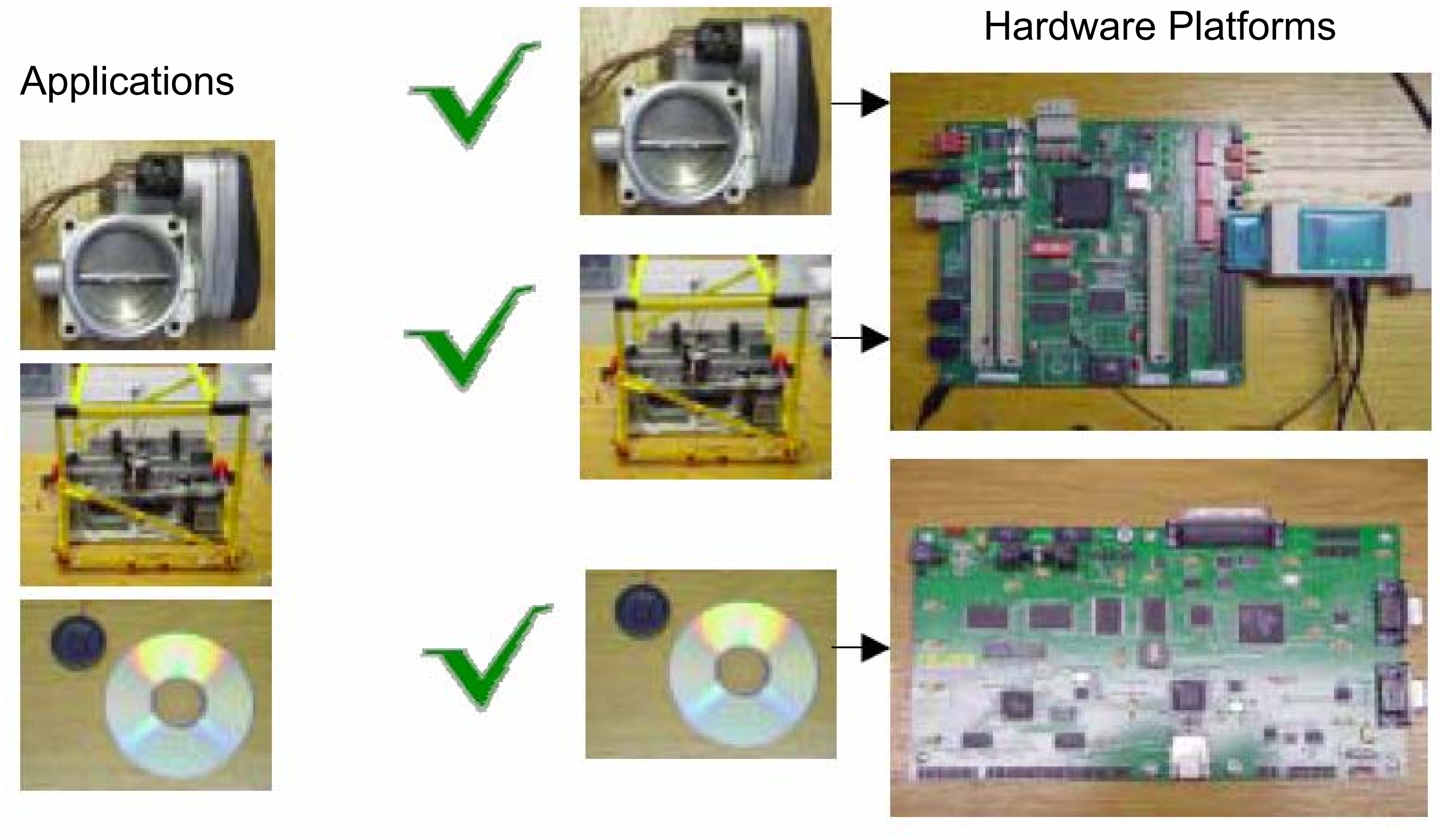
UNIVERSITÄT SALZBURG

Time-Safety Checking - Failure



UNIVERSITÄT SALZBURG

Time-Safety Checking - Success



Concepts

- Scheduler
  - provides the order in which tasks will get access to the processor
- Schedule
  - Is feasible if the execution of all tasks meets given constraints
  - set of tasks is *feasible*, i.e. *schedulable* if there is at least one feasible schedule of those tasks

Optimality for schedulers

A scheduling algorithm A is **optimal** among a *category* of scheduling algorithms if:

Any systems that A cannot schedule cannot be scheduled by any other scheduling algorithms in the same category

Timing constraints of task instances

- *release time $r_{i,j}$* – the instant in time when a task instance becomes available for execution (task i, instance j)
- *deadline $d_{i,j}$* – the instant in time by which the execution of the task instance is require to complete. It is an absolute deadline.

- *Hard real-time constraint* - failure to meet any deadline is considered a fatal flaw

Task attributes

- Task τ_i
- period T_i
- worst-case execution time C_i - difficult to estimate
- *relative deadline* D_i - the maximum allowable response time, $d_i = r_i + D_i$
- phase ϕ_i – the release time $r_{i,1}$ of first task instance $\tau_{i,1}$
- R_i – worst-case response time
- *processor utilization* $$U_i = \frac{C_i}{T_i}$$

Worst case execution time analysis

- Measuring
  - Ifs, calculating cycles (for), bounding loops (while), hashes, ...
  - Analysis tools – statical code analysis + profiling + heuristics
- Complications
  - Caches, Memory latency, Interrupts, DMA, compiler optimizations, function pointers, ...

Complex Task Models

- Dependencies among tasks
  - Precedence graph
- Resource requirements of tasks
  - Blocking due to mutual exclusion
- Aperiodic tasks
  - Event driven

=> not used in the TDL model. We use the schedulability analysis for **periodic, independent, preemptable** tasks.

Approaches to Hard Real-Time Scheduling

- Clock-driven approach
  - Decisions on what tasks execute are made at specific time instants
  - Static off-line scheduling

- Weighted round-robin approach
  - Time sharing
  - Size of time slice given to task depends on its weight
  - Appropriate for traffic scheduling

- Priority-driven approach
  - Task instances have fixed or dynamic priorities
  - Scheduler assigns the processor to highest priority task instance
  - On-line greedy scheduling

Processor Utilization Metric

- The *processor utilization* factor is the fraction of the processor time spent in the execution of the task set:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

Time safety check: for a given algorithm A, we can compute the ***least upper bound*** U_{lub} (A)

- If U > 1 no scheduling algorithms can guarantee the schedulability
- If $U \leq U_{lub}(A)$ the tasks are schedulable by algorithm A

  This condition is *sufficient but not necessary.*

- If $U_{lub}(A) < U \leq 1$, **nothing** can be said on the feasibility of the task set.

Fixed-priority scheduling

Rate Monotonic Scheduling (RM)

- Priority = rate = 1/period
- Tasks with smaller periods have higher priorities
- Optimal among all **fixed-priority** algorithms for task sets with $D_i = T_i$

Processor Utilization Bound for RM

RM Utilization Bound for $D_i = T_i$

- for n tasks:
- $U_{lub}(2) = 0.828$
- $U_{\mathrm{lub}}(n) = n(2^{1/n} - 1)$ $\lim_{n\to\infty} U_{\mathrm{lub}}(n) = \ln 2 = 0.693$

- Time Safety Check: $U \le U_{lub}$
- Only sufficient test
- How do we test schedulability for RM when $U_{lub} < U \le 1$?

UNIVERSITÄT SALZBURG

Dynamic-priority Scheduling

Earliest Deadline First (EDF)

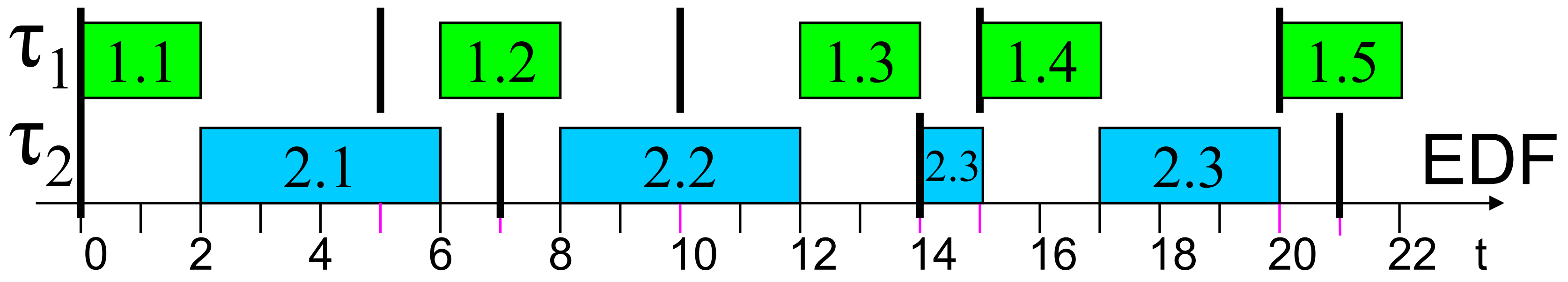
- Priority = **absolute** deadline
- Tasks with earlier deadlines have higher priorities
- Optimal among **all** scheduling algorithms

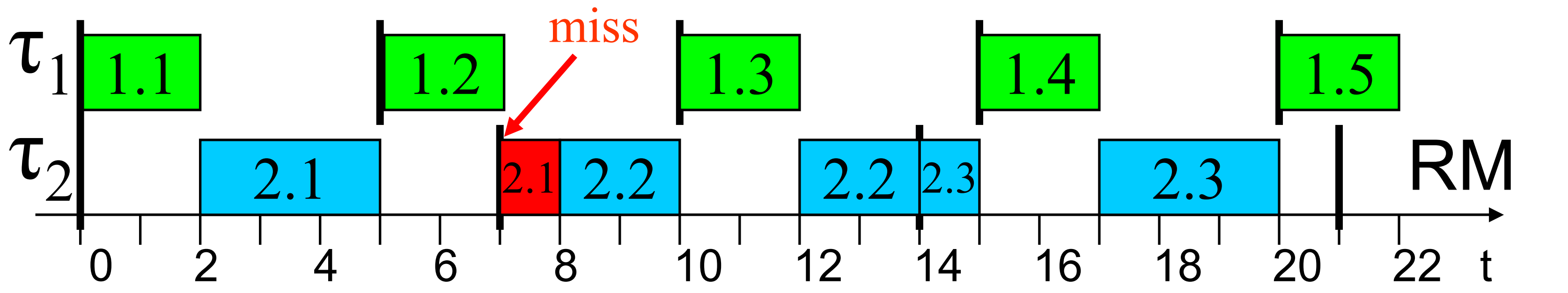
EDF Utilization Bound for $D_i = T_i$

- $U_{lub}=1$
- TSC: $U \leq 1$
- Necessary and sufficient test

EDF vs. RM example

- Two tasks τ_1, τ_2
  - $C_1=2$, $T_1=D_1=5$
  - $C_2=4$, $T_2=D_2=7$





Next Steps – Deadlines less than periods

Important in distributed applications when task must terminate before the period, to allow the communication on the bus

EDF -> EDF with deadlines less than periods

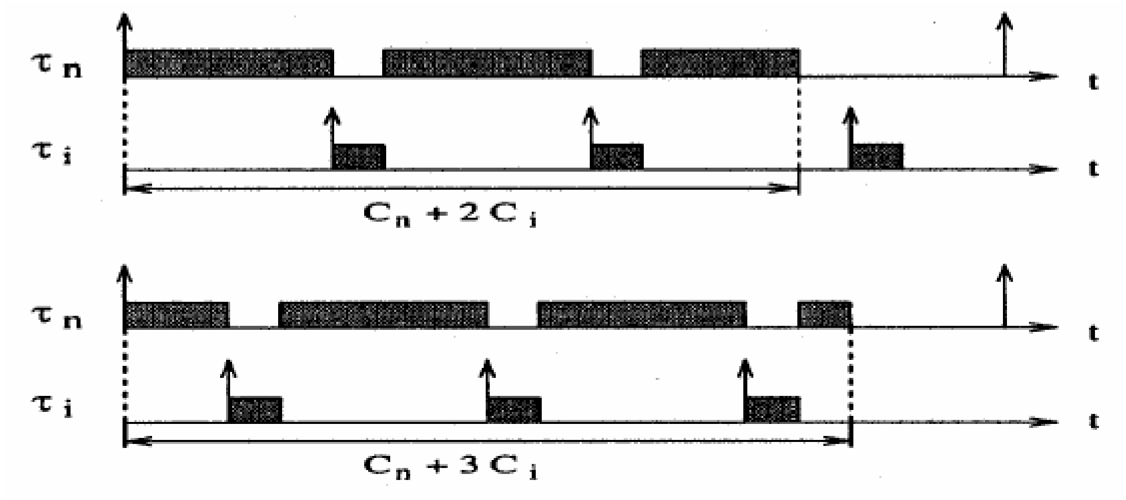
RM -> Deadline Monotonic Scheduling (DMS)

- priority = **relative** deadline
- optimal fixed priority scheduling algorithm when $D_i \leq T_i$

Fixed-priority scheduling - Critical Instant

- Critical Instant = instant at which a request for that task will have the maximum response time

- Example of the increased response time due to task interference (C_x=$wcet_x$)



UNIVERSITÄT SALZBURG

Response Time Analysis for fixed-priority

- *Theorem : In a system with $D_i \le T_i$, a critical instant for any task occurs whenever the task is requested simultaneously with requests of all higher priority tasks*

- *A sufficient and necessary test for RM is*:

$$\forall i : 1 \le i \le n \qquad R_i \le D_i$$

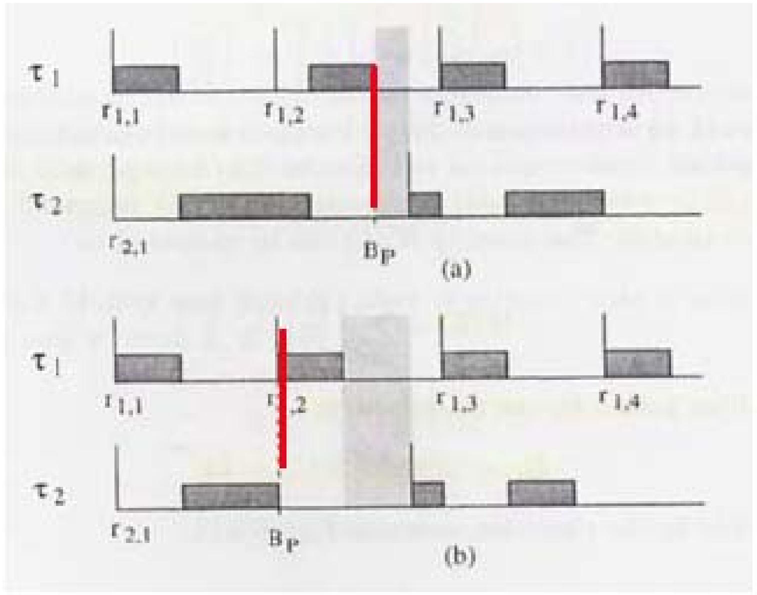
$$R_i = C_i + I_i$$

I_i = Interferences from higher priority tasks

$$R_i^{k+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil C_j$$

Processor demand approach for EDF

- New TSC based on the busy period:
- Busy Period = the first time instant when all the released tasks are completed



Single Processor,
Single / Multiple TDL Modules

Reuse of Processor Utilization formulas

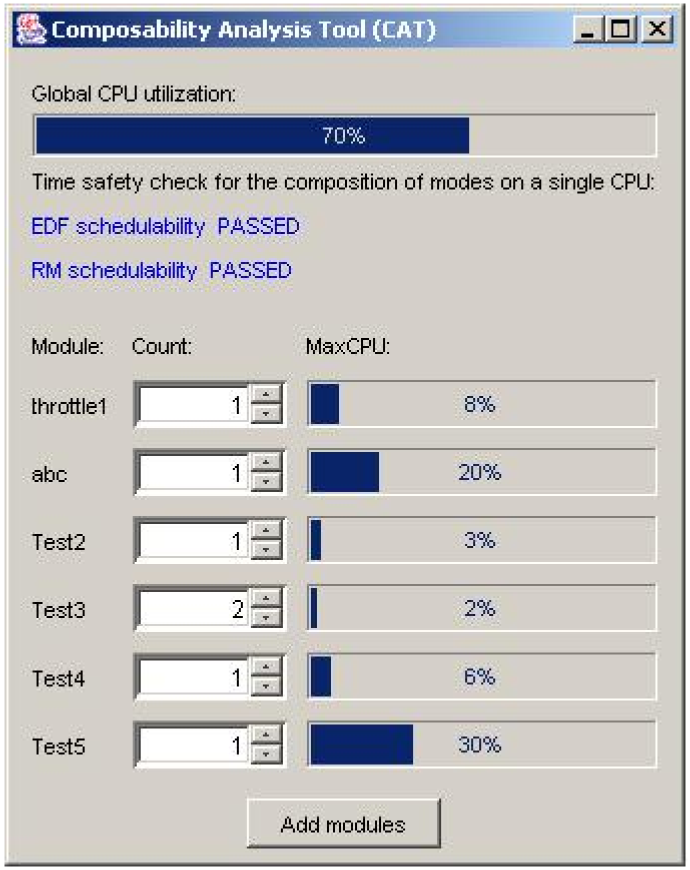
Given a set of modules $M_1 \dots M_i \dots M_n$, each one having a **single** start mode, and a number of modes #modes(M_i), the time safety check:

- The Processor Utilization test for all combination of modes that could run in parallel:

$$\prod_{i=1}^{n} \#\mathrm{modes}(\mathrm{M}_{\mathrm{i}})$$

- Improved: we select from each application, the mode that utilize the processor the most.
- TSC for EDF and RM

Composability Analysis Tool



Composability Analysis Tool (CAT)

Global CPU utilization:

70%

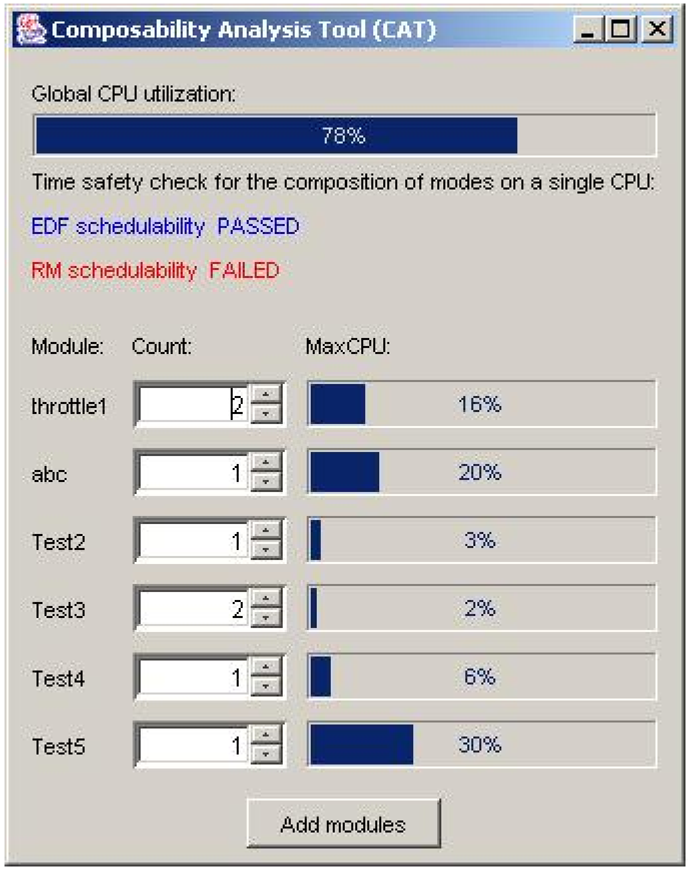
Time safety check for the composition of modes on a single CPU:

EDF schedulability PASSED

RM schedulability PASSED

Module:	Count:	MaxCPU:
throttle1	1	8%
abc	1	20%
Test2	1	3%
Test3	2	2%
Test4	1	6%
Test5	1	30%

Add modules



Composability Analysis Tool (CAT)

Global CPU utilization:

78%

Time safety check for the composition of modes on a single CPU:

EDF schedulability PASSED

RM schedulability FAILED

Module:	Count:	MaxCPU:
throttle1	2	16%
abc	1	20%
Test2	1	3%
Test3	2	2%
Test4	1	6%
Test5	1	30%

Add modules

Remaining issue?

How do we test schedulability for RM when $U_{lub} < U \leq 1$?

- we cannot select from each application, the mode that utilize the processor the most
- Response Time Analysis test run for all combinations of nodes
- Work in progress in implementation and proving that this test is *sufficient*.