Mixed-criticality scheduling theory in safety-critical real-time systems

Ivan Pavić, MSc
Faculty of Electrical Engineering and Computing
University of Zagreb
Zagreb, Croatia
Email: ivan.pavic2@fer.hr

***Abstract*—This paper brings an overview of mixed-criticality scheduling theory and its link to industrial applications which have to comply to industrial safety standards. Mixed-criticality scheduling theory devises feasibility tests, priority assignments, task allocations and various models for task systems which consist of tasks with different criticality levels. This theory emerged as the response to the growth in processing power of embedded computer systems. These systems are based on complex hardware which causes uncertainty in system response time. Additionally, complex multi-core architectures enable running general-purpose operating system alongside safety-critical applications. To resolve issue of uncertainty of system response time and appropriate utilization of system resources, mixed-criticality scheduling theory was developed by academia.**

I. Introduction

Mixed-criticality scheduling theory is part of real-time scheduling theory which deals with mixed-criticality task systems. These systems consist of tasks with different criticality level. Classical sporadic task system model is extended with multiple worst case execution times for each job that some task in a system can dispatch. In last ten years, from the initial Vestal paper [1], significant effort has been made to create a model for mixed-criticality system which could be used in the system certification process [2]. Mixed-criticality scheduling theory attempts to model software for safety-critical systems which should reduce time for verification of safety-critical real-time embedded systems. However, as stated in [3], one must make a persuasive argument that these models represent the actual runtime behavior of the system that is being modeled.

In this paper, brief overview of mixed-criticality scheduling theory is made, explaining advantages, drawbacks and potential applications. The paper consists of several sections. Section II contains a brief introduction to criticality concept in the context of safety standards. Section III explains the motivation and purpose of mixed-criticality scheduling theory. Formal aspects and models of mixed-criticality scheduling theory are explained in the section IV. In the section V applications and links to different research topics are presented. Section VI explains the limitations and perspective of mixed-criticality systems in real-life applications.

II. Industrial context of mixed-criticality systems

In system design of safety-critical systems, non-functional characteristics such as safety, security and performance must be taken into account as well as system's operative functions. Process for ensuring safety properties of systems is called *system safety assessment process* [2]. Typically, *system safety assessment process* of the embedded software starts with hazard analysis. In the next subsections, definitions of basic concepts in dependable and secure computing are introduced and connected with *system safety assessment process*.

A. Concepts of dependable and secure computing

Definitions which are introduced here are based on the work of Aviženis et al. [4]. These concepts are common and defined similarly in dependable computer systems literature [5], [6] and used throughout safety standards.

Definition II.1. Service. *A service is behaviour of a system as it is perceived by a user. A service is **correct** when it implements its system function.*

Definition II.2. Service failure. *A service failure or just failure, is an event that occurs when the delivered service deviates from correct service. It is a transition from correct service to incorrect service. There can be different forms of failure, which are referred to as **failure modes**. And each failure mode has its **failure severity**.*

Definition II.3. Error. *An error is deviation of a system service from correct service.*

Definition II.4. Fault. *Fault is the adjudged or hypothesized cause of an error.*

With these definitions causal connection between these concepts is self-evident. When a fault in a system is active, it produces an error. Error causes transition from correct service to incorrect service (service failure). This is illustrated on Fig. 1.

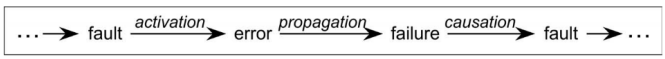


Fig. 1. Causal connection between fault, error and failure [4]

B. Hazard analysis and fault analysis

Software hazard analysis is a procedure which should ensure that the software does not interfere with the objectives and correct operation of the system and if interference cannot be totally avoided, then it must also evaluate and make recommendation to mitigate how the software can hinder the objective or operation of system [2]. Software hazard analysis is classic technique which is used often as a base for fault analysis [7].

Hazard analysis is a base for fault analysis, which is described by certain standards. There are several types of fault analysis as found in [8], [2]:

- Fault Tree Analysis (FTA) [9],
- Failure Modes and Effects Analysis (FMEA) [10],
- Failure Modes and Effect Criticality Analysis (FMECA) [10].

These techniques are used for analysis of possible software failures in the system. They are performed after hazard analysis and can discover other failures which were not discovered by hazard analysis. Every failure of system function has a corresponding failure mode. In this context, failure mode can be:

- non-execution,
- late execution,
- incorrect execution [2].

To every failure, severity is assigned based on effect analysis. The last step in FMEA process is identification of existing compensating provisions that can mitigate the effects of failure. FMECA process assigns "criticality" level to failure mode based on the effect analysis and compensating provisions. This "criticality" level is referred to as *development assurance level* (DAL) [2]. This terminology is specific to DO-178C [11] (avionics domain). Terminology in other standards is different, but the basic concept remains preserved (e.g. IEC61508 uses SIL [12], ISO 26262 uses ASIL [13]).

C. Introduction of criticality levels

Nowadays, there are lots of certifiable embedded systems for safety operation in avionics, automotive and railway industry. Their software and hardware development follows propositions of IEC 61508 standard and other domain specific standards such as DO-178, ISO 26262 and EN 5021X respectively. Classic approach to the development of embedded systems follows a federated approach with "one function = one computer" paradigm [14].

Growth in processing power in embedded systems created the possibilities of embedding more functions on one application processor or even multi-core processors. Such approach can significantly reduce power consumption and cost. However, these systems are not suitable for certification as easy as "simple" microcontroller units. The main reason is their inherent complexity. However, as investigated by many papers, MPSoC systems can be used to increase redundancy in systems which is one of the prerequisites for achieving higher *Safety Integrity Level* (SIL) as proposed by the standards.

There are many functions that embedded systems must implement to ensure correct operation of the system as whole. As proposed in many papers and implied by safety standards, there are certain levels of criticality of embedded system functions. Papers [15], [16] propose a simple model of two levels of criticality in the airplane embedded system:

- *mission* critical functionalities (low criticality),
- *flight* critical functionalities (high criticality).

Combining the functionalities of different level of criticality on same computing platform (single core or multi-core processor) creates **mixed-criticality system**.

Based on results of *system safety assessment process*, which identifies failures, failure modes and compensating provisions, certain level of assurance is assigned to each system function. These assurance levels motivated researchers and academia to introduce criticality levels in real-time scheduling theory and create mixed-criticality scheduling theory. This motivation created debate in academic and industrial circles [3], [2], [8] about many misconceptions that were identified in academic work regarding mixed-criticality scheduling theory.

III. Mixed-criticality scheduling motivation and basis

This section explains common goals and motivational example, which engaged academia to create different models and the new area in real-time scheduling theory.

A. Goals of mixed-criticality scheduling theory

In safety-critical systems, safety assurance is often achieved through partitioning of system resources following already mentioned federated approach. This conservative approach guarantees isolation of system functions with different criticality level. However, this approach does not guarantee efficient resource usage. Nowadays, embedded systems have greater computing capabilities which make resource usage even more inefficient. As it can be seen in [14], system can be optimized by mixing different criticality functions. Nevertheless, as it is stated in [12], sufficient independence must be demonstrated between functions of different criticality both in the spatial and temporal domain.

According to [3], mixed-criticality scheduling tries to reconcile contrasting goals of a priori system verification and resource-efficient implementation. Similarly, authors in [17] point out conflicting requirements of partitioning resources for safety assurance and sharing resources for efficient usage.

Motivational example which is often used in mixed-criticality scheduling papers (especially by Baruah [18], [15] etc.) is briefly presented here.

Example 1. *A system consists of three jobs J_1, J_2 and J_3. All three jobs are released simultaneously on a preemptive fixed priority processor. Deadline of J_1 is 2, while deadline of J_2 and J_3 is 3.5. J_2 and J_3 are high criticality tasks and have to be certified, while J_1 is not critical and it is not subject to certification.*

The system designer estimates WCET to be no greater than 1. Therefore, all jobs can be executed as long as J_1 is not assigned the lowest priority.

System certification in the most cases requires more conservative estimations of WCETs. Safety assessor requires 1.5 time units for J_2 and J_3. System designer has to assign higher priority to J_2 and J_3 for J_2 and J_3 to be schedulable, but then J_1 is not schedulable even if J_1 and J_2 execute for 1 unit. The system seems to be unschedulable.

However, if priority ordering is J_2, J_1, J_3 system is schedulable in case J_3 and J_2 execute for 1 time unit. If for instance J_2 executes for longer than 1 time unit, system changes mode and discards low-criticality job J_1 and can successfully schedule J_2 and J_3. Furthermore, if J_2 and J_1 signal completion, J_3 is still schedulable for more pessimistic estimation of WCET. In every case, requirements from system designer and certification authority are satisfied.

Based on the latter example, one of the goals of mixed-criticality scheduling theory is to find appropriate priority assignment and runtime behaviour for mixed-criticality task system.

B. Classical real-time scheduling theory

Foundations for the real-time scheduling theory were made by Liu and Layland in [19]. In mentioned paper, task system model was introduced. A task set Δ consists of m tasks $\tau_1, \tau_2, \tau_3, \tau_i, ..., \tau_m$ with a corresponding set of periods T and worst case execution times C. Often additional deadline parameter set D is added for a task set unless deadline is implicit $Di = Ti$, otherwise $Di \leq Ti$. Generally, every task is represented as a set of 3 parameters:

$$\tau_i = \{Ci, Ti, Di\} \tag{1}$$

In the paper, Liu and Layland introduce RM and DM priority assignments and the processor utilization factor, which is a fraction of processor time spent in the execution of a task set [19]. Considering already introduced notation utilization factor U is:

$$U = \sum_{i=1}^{m} \frac{Ci}{Ti} \tag{2}$$

Utilization factor metric is often used for comparing scheduling algorithms and testing the feasibility of an assignment for a certain task set. Another concept used as metric is resource augmentation (e.g. processor speedup factor) which is also used in scheduling algorithm and policy comparison [15].

C. WCET estimation problem

To demonstrate deterministic scheduling of tasks, one should provide a model of runtime behavior or show deterministic execution times through the component and integration testing of a task system. This is easily done using simple architecture processor by counting cycles of each instruction. On more complex processors, however, this is not as straightforward as on simpler architectures. As stated in [17], uncertainty in the knowledge of WCET value is primarily epistemic (uncertainty in what we know, or do not know, about a system) rather than aleatory (uncertainty in a system itself). To raise assurance in the WCET estimation, one could create more conservative WCET bound by increasing WCET itself. However, conservative over-approximated WCET values degrade utilization of system resources [3], [20]. In the next subsection, the solution for the uncertainty of WCET value estimation is presented in the context of mixed-criticality scheduling theory.

IV. Mixed-criticality scheduling models

A. Basic sporadic mixed-criticality task system model

The seminal mixed-criticality scheduling paper is based on the following conjecture: *the higher the degree of assurance required that actual task execution times will never exceed the WCET parameters used for analysis, the larger and more conservative the latter values become in practice* [1].

According to this conjecture, higher criticality task will have larger worst case execution times than task with lower criticality. This is the consequence of a measurement technique used for determining WCET for certain tasks. WCET value determined by simple measurement in normal operation should be used for lower criticality tasks. WCET of higher criticality tasks will be determined by thorough testing and advanced measuring methods of task execution. Intuitively, exhaustive testing will discover larger WCET values. Therefore, with greater assurance in the precision of the WCET value estimation, comes the more conservative estimation of the WCET value.

Vestal extended task model with set λ which contains criticality levels:

$$\lambda = \{A, B, C, D\} \tag{3}$$

Generally, it can be up n levels:

$$\lambda = \{\lambda_1, \lambda_2, ..., \lambda_n\} \tag{4}$$

Every task in a task set Δ is described with equation:

$$\tau_i = \{\vec{C_i}, T_i, D_i, L_i\} \tag{5}$$

where Li is task criticality level and $\vec{C_i}$ is n-dimensional vector containing WCET values for certain level of safety assurance. Following the main conjecture of Vestal paper, vector $\vec{C_i}$ is monotonic non-decreasing vector:

$$C_i \leq C_j, \forall i < j \tag{6}$$

Here are few definitions which are crucial for formulating mixed-criticality scheduling problem.

Definition IV.1. *Sporadic mixed-criticality task system schedule is feasible if all tasks are schedulable considering their low criticality WCET and if high criticality tasks are schedulable considering their high criticality WCET.*

Based on equations (4), (5), (6) and definition IV.1, classical mixed-criticality scheduling problem is formulated as follows:

Definition IV.2. *Classical mixed-criticality scheduling problem is the problem of finding optimal priority assignment policy for a given task set Δ where each task τ_i is described with monotonic non-decreasing vector of WCETs $\vec{C}_i$, period T_i, deadline D_i and criticality level L_i. Priority assignment policy is considered optimal in sense that if there is a feasible task schedule, optimal policy will find it.*

There are different schemes of priority assignment policies in mixed-criticality systems regarding type of scheduling (fixed priority or dynamic priority). In following subsections, these different schemes are reviewed. In the most papers, only dual criticality systems are considered (tasks with low and high criticality). However, these models can be easily extended to more than two criticality levels.

B. Fixed-priority mixed-criticality scheduling based on response-time analysis

Fixed priority scheduling in mixed-criticality systems based on RTA (response-time analysis) approach was investigated by Baruah et al. in [18] where they define three different scheduling policies based on RTA approach:

- Partitioned Criticality (PC) or Criticality Monotonic Priority Assignment (CrMPO),
- Static Mixed Criticality (SMC),
- Adaptive Mixed Criticality (AMC).

Every scheduling policy has two main properties:

- priority assignment policy,
- corresponding runtime behaviour.

These properties must ensure that scheduling policy obeys definition IV.1.

Response-time analysis is based on classical real-time scheduling analysis defined by Joseph and Pandya in [21]. Response time of task in sporadic task system can be found by solving recurrence relation:

$$R_i = C_i + \sum_{j \in hp(\tau_i)} \lceil \frac{R_i}{T_j} \rceil \cdot C_j \tag{7}$$

where $hp(\tau_i)$ is set of tasks with higher priority than τ_i. A task is schedulable at priority level ρ if $R_i \leq D_i$. This is a conservative analysis which can be relaxed in a sporadic mixed-criticality task system. SMC and AMC priority assignments use modifications of equation (7) which are suited for mixed-criticality system. More precisely, these modifications ensure definition IV.1 is not violated.

Intuitively, a relaxation of equation (7) implies different runtime behaviour with certain scheduling policy. It is important to see the connection between scheduling policy, priority assignment and runtime behaviour. In case of **SMC** and **AMC**, runtime behaviour impacts response time equation. More precisely, for every runtime behaviour rule or policy, there is a corresponding change in response time equation. While runtime behaviour impacts response time equation, feasibility tests in priority assignment algorithms depend on the response time equation.

Algorithm 1 Audsley's algorithm

Input: $\Delta = \{\tau_1, \tau_2, ..., \tau_n\}$
Output: Ψ - priority ordered task set

```
1:  function PRIORITYASSIGNMENT(Δ)
2:      Ψ ← ∅
3:      n ← |Δ|
4:      unassigned ← true
5:      Δ' ← Δ
6:      j ← n
7:      while j ≥ 1 do
8:          unassigned ← true
9:          for each task τ in Δ do
10:             if isFeasible(τ, Δ'/τ) ∧ unassigned then
11:                 Δ' ← Δ'/τ
12:                 Ψ ← Ψ ∪ {τ}
13:                 unassigned ← false
14:             end if
15:         end for
16:         if unassigned then
17:             return ∅      ▷ feasible schedule does not exist
18:         end if
19:         j ← j − 1
20:     end while
21:     return Ψ              ▷ contains priority ordered set
22: end function
```

Criticality Monotonic Priority Assignment assigns higher priority to higher criticality tasks. This is naive approach which results in very poor schedulability of high utilization task sets. It does not consider timing properties of tasks.

Priority assignment for **Static Mixed Criticality (SMC-NO)** [1] approach is based on Vestal's adaptation [1] of Audsley's algorithm [22]. Unmodified Audsley's optimal priority assignment is shown by algorithm 1. Algorithm 1 takes a task set Δ as input and returns priority ordered set Ψ if there is a feasible priority assignment. Algorithm returns an empty set, if there is not any feasible assignment. Function `isFeasible` returns true, if task τ is feasible on priority level j. Set Δ'/τ is set of tasks with higher priority than τ (i.e. $hp(\tau)$).

In [1], Vestal proposed two modifications of Audsley's algorithm:

- feasibility testing is done by using already mentioned modifications of equation (7),
- tie on any priority level is solved by evaluating critical scaling factor.

SMC-NO priority assignment devised by Vestal was proven to be optimal later by Dorin et al. [23]. This approach is modified in [18] to include runtime monitoring (**SMC**) which yields better results. In runtime, **SMC** approach will deschedule any low criticality task which executes for longer than its criticality level WCET. When high criticality task executes for longer than its low criticality WCET, all low criticality tasks will be descheduled [18]. With introduction of runtime

[1]NO suffix indicates that approach does not use runtime monitoring

monitoring, system execution mode Γ is introduced as well. In dual criticality systems, system execution mode can be either LO or HI, $\Gamma \in [LO, HI]$.

Adaptive Mixed Criticality (AMC) approach has two variants. Both variants are made by making response time analysis more precise than **SMC**, but the priority assignment algorithm is similar. **AMC-max** approach gives the best result as it can be seen in Fig. 2. **AMC** approaches have separate model for HI and LO execution modes. Alongside HI and LO modes, one must devise response time analysis for the criticality mode switch event. The criticality mode switch occurs when some task executes for longer than WCET corresponding to its criticality mode. The model must ensure that jobs of HI criticality tasks which were released before the criticality switch are scheduled properly with respect to their HI criticality WCET. Priority assignment is similar as in **SMC** scheme (different response time equations in feasibility tests are used). The main difference between **SMC** and **AMC** regarding runtime behaviour is that **AMC** discards all low criticality tasks in HI criticality mode.

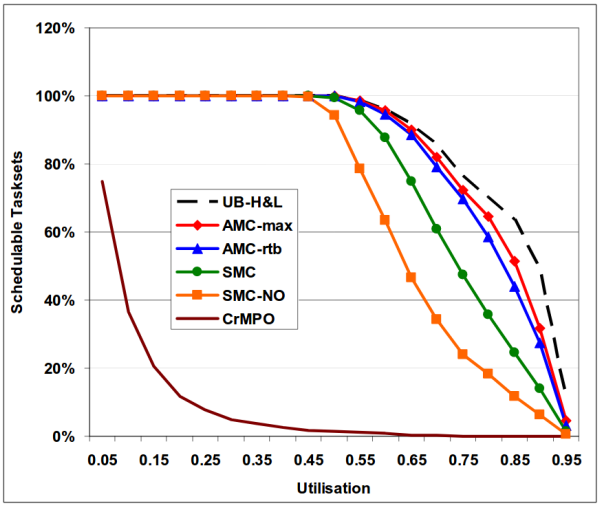


Fig. 2. Percentage of schedulable task sets [18]

UB-H&L on Fig. 2 represents theoretical utilization bound which is constructed from separate deadline monotonic priority assignment for low and high criticality tasks.

C. Exact schedulability test for fixed-priority mixed-criticality systems

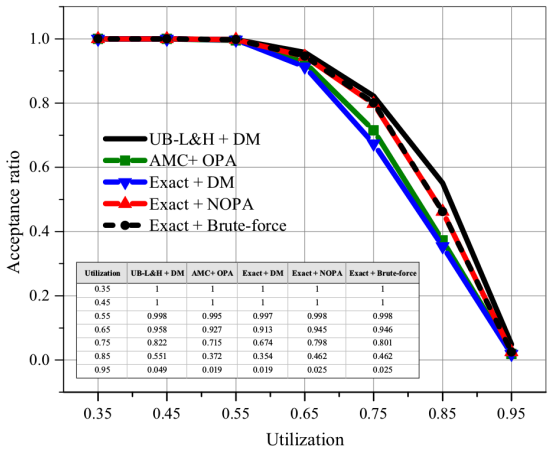
Using state exploration of all possible states in a task system, it is possible to determine feasibility of a schedule for a task set. This was done for real-time multiprocessor task system by Bonifaci and Marchetti-Spaccamela in [24]. Similar approach was recently used in mixed-criticality sporadic task system by Asyaban and Kargahi in [25]. In [25], authors devise an exact schedulability test for mixed-criticality task systems with given priority order by searching state space of a mixed-criticality task system. System state space at time instant t is formulated as:

$$s_t = \langle \Gamma, (c_i, q_i, p_i, \epsilon_i)_{i=1}^{N} \rangle \tag{8}$$

where:

- $\Gamma \in [LO, HI]$ is system criticality mode (LO or HI);
- $c_i \in \{0, 1, ..., C_i(HI)\}$ is remaining execution time for current job of a task τ_i;
- q_i is remaining time to the deadline for current job of a task τ_i;
- p_i is remaining time for the arrival of new job of task τ_i;
- $\epsilon_i \in \{0, 1, ..., C_i(HI)\}$ is actual execution time of a job of a task τ_i;

Authors in [25], emphasize that the devised exact schedulability test is not compatible with the Audsley's OPA algorithm. Conditions for the OPA compatibility of a feasibility test were introduced by Davis and Burns in [26]. More precisely, schedulability of a task τ_i at some priority level depends on any independent properties of tasks with higher priorities but not on their relative priority ordering. To resolve this issue authors in [25] devise non-optimal priority assignment algorithms and brute-force methods for assigning priorities using the exact schedulability test. The results of this approach are better than the previously reviewed static scheduling schemes as shown on Fig. 3. However, this approach is not proven to be optimal.



Utilization	UB-L&H + DM	AMC+ OPA	Exact + DM	Exact + NOPA	Exact + Brute-force
0.35	1	1	1	1	1
0.45	1	1	1	1	1
0.55	0.998	0.995	0.997	0.998	0.998
0.65	0.958	0.927	0.913	0.945	0.946
0.75	0.822	0.715	0.674	0.798	0.801
0.85	0.551	0.372	0.354	0.462	0.462
0.95	0.049	0.019	0.019	0.025	0.025

Fig. 3. Percentage of schedulable task sets [25]

D. Dynamic priority mixed-criticality scheduling

The majority of work in dynamic priority scheduling regards modification of task timing characteristics to be suitable and optimally scheduled by modifications of the EDF scheduling algorithm. As stated in the review [17], first attempt to address mixed-criticality scheduling problem in the context of EDF scheduling was done by Baruah and Vestal in 2008 [27]. In [28], [29], Baruah et al. introduce EDF-VD (EDF with virtual deadlines algorithm). In the EDF-VD scheme, deadlines of high criticality tasks are reduced by the same factor in LO-criticality mode of execution.

In case of dynamic scheduling approaches, task model is changed to include different values of deadlines for different criticality levels and can be represented by equation (9).

$$\tau_i = \{\vec{C_i}, T_i, \vec{D_i}, L_i\} \tag{9}$$

More general approach in EDF scheduling was pursued by Ekberg and Yi in [30]. They use demand bound functions for low and high criticality modes of execution for analysis of the schedulability.

Definition IV.3. (Demand-bound function as defined in [30]). *A demand-bound function $dbf(\tau_i, l)$ gives an upper bound on the maximum possible execution demand of task τ_i in any time interval of length l, where demand is calculated as the total amount of required execution time of jobs with their whole scheduling windows within the time interval.*

As in the fixed-priority schemes, the problem is modeling of task schedule when some task executes longer than WCET corresponding to its criticality level (criticality switch). They define a demand bound function for low criticality execution as:

$$dbf_{LO}(\tau_i, l) = [\![(\left\lfloor \frac{l - D_i(LO)}{T_i} \right\rfloor + 1) \cdot C_i LO]\!]_0 \tag{10}$$

in any time interval of length l. Additionally, $[\![A]\!]_k$ denotes $max(A, k)$.

In a similar manner, one can devise a demand bound function for high criticality mode:

$$dbf_{HI}(\tau_i, l) = full(\tau_i, l) - done(\tau_i, l) \tag{11}$$

where $full$ and $done$ are defined separately for purpose of maximizing scheduling window of τ_i in case of criticality switch.

$$full(\tau_i, l) = [\![(\left\lfloor \frac{l - (D_i(HI) - D_i(LO))}{T_i} \right\rfloor + 1) \cdot C_i LO]\!]_0$$

$$done(\tau_i, l) = \begin{cases} [\![C_i(LO) - n + D_i(HI) - D_i(LO)]\!]_0, \\ D_i(HI) > n \geq D_i(HI) - D_i(LO) \\ 0, otherwise \end{cases}$$

To ensure maximum scheduling windows for τ_i deadlines in low criticality mode need to be tuned. Ekberg in Yiu in [30] propose a greedy algorithm for finding those deadlines in pseudo-polynomial time. The results of their approach clearly dominate different static and dynamic priority assignments which can be seen in Fig. 4.

The EDF-VD approach uses one scaling factor for deadlines of all tasks. Therefore, it is dominated by a greedy approach which finds independent scaling factor for every task in a task system. However, the simplicity of EDF-VD allows schedulability bounds to be derived as stated in [17].

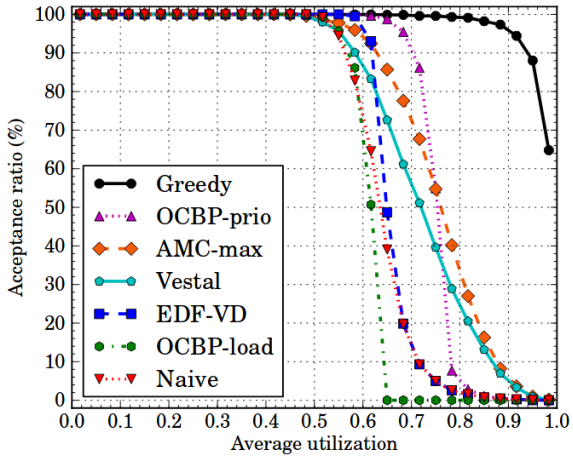


Fig. 4. Percentage of schedulable task sets [30]

E. Priority assignment as an integer linear program (ILP)

Priority assignment in real-time systems can be viewed as an integer linear program (ILP). ILP is a type of linear programming problem where some of the variables are restricted to be integers [31]. Objective function of priority assignment integer linear programming problem for classical real-time systems can be formulated as a weighted sum of task worst case response times:

$$\min_{\pi,\epsilon} \sum_{i=1}^{|\Delta|} |w_i \cdot R_i(\pi_i \cdot \epsilon_i)| \tag{12}$$

Equation (12) depends on binary matrix π where element π_{ij} equals 1 in case τ_i has higher priority than τ_j, 0 otherwise. Variable ϵ comes from linearizing ceiling function in equation (7). Therefore, R_i in (12) is formulated as:

$$R_i(\pi_i, \epsilon_i) = C_i + \sum_{j=1}^{|\Delta|} \pi_{ij}\epsilon_{ij} \cdot C_j \tag{13}$$

The first formulation of priority assignment as bilinear programming problem was done by Lisper in [32] which was not very practical because of the long computation time. However, this formulation can be further linearized because the priority assignment is binary variable. Furthermore, it is not necessary to specify an objective function when additional constraints $R_i(\pi_i, \epsilon_i) \leq D_i$ are specified [32]. This approach was used in mixed-criticality priority assignment by Al-bayati and Meyer in [33] where the problem is formulated as mixed-integer linear programming (MILP) problem [34]. Alongside priority assignment, authors in [33], optimize task allocation on multi-core processors. This formulation of a problem is useful when there are more variables in a system to optimize (e.g. fault-tolerant task allocation) as well as priority assignment.

F. Different mixed-criticality scheduling approaches

In the section IV-B, approaches based on response-time analysis are presented, but there are other fixed-priority approaches in mixed-criticality scheduling. A lot of work has be

done on slack scheduling and period transformation which is in fact a continuation of work on robust real-time scheduling.

As described in [17], in the **slack scheduling** scheme, low criticality jobs are run in the slack generated by high criticality jobs only using their low criticality execution budgets. This was explored by Niz et al. in [35], and by Huang et al. [36].

Adaptation of **period transformation** in the mixed-criticality scheduling was used in Vestal's seminal paper [1]. However, the protocol was introduced before in [37], [38]. The idea is to split a task in several tasks with smaller periods which ensures its higher relative priority. One of the shortcomings in implementation is a more frequent context switch which introduces additional overhead.

V. Applications and link to different research topics

A. Software fault tolerance in mixed-criticality systems

System fault tolerance is an important property of any safety certifiable embedded system. To achieve fault tolerance, system software must be extensively tested as proposed by the standards. Additionally, system should be robust and resilient to software and hardware transient faults. To achieve this, system can be built using safety certified hardware. Furthermore, the software architecture of system can be built in such way that it ensures redundancy, fault state detection, monitoring and graceful degradation of tasks with lower criticality.

Papers [39] and [40] by Huang et al. quantify some system quality properties from standards (IEC 61508 and DO-178) and put them in context of mixed-criticality systems on single core [39] and multi-core processors [40]. In the latter papers, authors propose scheduling task replicas in order to increase redundancy and the success rate of certain function. Similar effort was done by Abayati et al. in [41] where they propose four-mode model which considers QoS (*quality of service*) of LO criticality task alongside issues of robust HI criticality task execution and certification. In [42], Thekkilakattil et al. introduce error burst model in the mixed-criticality system and define MWET (Maximum Wasted Execution Time) for critical tasks which is caused by error bursts. In [43], authors consider using various types of hazard analysis in the context of fault-tolerance in mixed-criticality systems. This approach impacts task timing characteristics, task allocation and feasibility.

B. Multiprocessor analysis of mixed-criticality systems

From the aspect of task allocation on multi-core systems, finding optimal task assignment, is the NP hard problem equivalent to bin packing. Additionally, multi-core approach is often intertwined with fault tolerance mechanisms. Paper [44] creates model based on mixed-criticality system model using task replication and task scheduling on multi-core systems to achieve maximum utilization factor and spatial independence between task and task replicas. The model proposed in the paper is used in automotive applications. Fault-tolerant scheduling on multi-core mixed-criticality systems under permanent failures was investigated in [33].

C. Probabilistic mixed-criticality systems

In the section III-C, importance of WCET estimation was emphasized. Based on results from different WCET estimation tools various WCET and WCRT times can be derived. The worst case execution time in mixed-criticality system is a function of system criticality mode (e.g. $C(LO)$ or $C(HI)$). In this case WCET is expressed as a discrete function with two possible values [17]. However, sometimes is possible to increase the number of discrete values for WCET. Furthermore, WCET can be expressed as a probability distribution, recently referred to as pWCET [45], [46]. However, this approach was used before in the classical real-time systems [47].

The pWCET can be represented with the exceedance probability function which is in fact $1 - F_{cd}(C)$, where $F_{cd}(C)$ is cumulative distribution function [17]. The exceedance probability function represents probability that a job which some task dispatches will exceed certain value. An arbitrary exceedance probability function is shown on Fig. 5. Safety

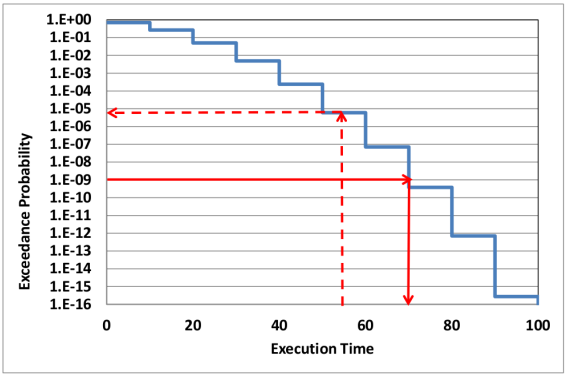


Fig. 5. An exceedance probability function for some task [17]

standards provide failure rates for components in the system based on their level of criticality. Components with higher SIL or DAL have smaller failure rates (e.g. 10^{-9}). On the other hand lower SIL and DAL levels imply higher allowed failure rates (e.g 10^{-6}). An exceedance probability function can be used to determine allowed execution time based on the required failure rate. These statistic properties can be used in advanced probabilistic models such as constrained Markov decision process in [48] for scheduling of mixed-criticality job sets. Probabilistic analysis of fixed-priority uniprocessor scheduling schemes SMC and AMC was done in [49].

VI. Limitations and perspective

There are lots of limitations of the mixed-criticality model as elaborated in [8] and [2]. This is due to similarity in terminology of safety standards and academic papers. Concept of criticality and design assurance level as defined in standards is more dependent on qualitative than quantitative factors. Academic model as presented in IV depends only on WCETs of tasks in task system. The WCET is important parameter, but rarely it is the only parameter. However, as Baruah affirms

in [3], mixed-criticality system model has potential to become part of safety standards if properly created by the academia.

VII. Conclusion

In this paper, brief overview of mixed-criticality scheduling theory was presented. Additionally, mixed-criticality scheduling theory model was described in the context of industrial *system safety assessment process*. It is shown that priority assignment schemes can be very useful in increasing schedulability of task sets while using fixed-priority scheduling and that run-time policies ensure correct behaviour of system. State-of-the-art approaches for fixed-priority mixed-criticality priority assignment based on RTA and state exploration are presented. Generic EDF scheduling work is presented in the context of dynamic scheduling and it is shown how modifying relative deadlines of HI-criticality tasks in LO-criticality mode can increase overall schedulability of mixed-criticality task sets. Additionally, mixed-criticality priority assignment is formulated as ILP problem.

There are a lot of possible applications where mixed-criticality scheduling can be used. As pointed out, important application is *a priori* system verification which is important in *safety assessment process* of fault-tolerant systems. This application creates link between mixed-criticality scheduling theory and other research topics such as probabilistic response-time analysis and task allocation which are typical problems in software design for industrial computer systems. There are a lot of limitations of mixed-criticality scheduling theory and one must be careful while applying purely theoretical work to critical environments. However, the theory provides models which can successfully describe many systems and which can be used as a tool in system design and verification process.

Acknowledgement

This work has been supported by the European Regional Development Fund under the project "System for increased driving safety in public urban rail traffic (SafeTRAM)".

References

[1] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, Dec 2007, pp. 239–243.

[2] A. Esper, G. Nelissen, V. Nélis, and E. Tovar, "How realistic is the mixed-criticality real-time system model?" in *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, ser. RTNS '15. New York, NY, USA: ACM, 2015, pp. 139–148. [Online]. Available: http://doi.acm.org/10.1145/2834848.2834869

[3] S. Baruah, "Mixed-criticality scheduling theory: scope, promise, and limitations," *IEEE Design Test*, vol. PP, no. 99, pp. 1–1, 2017.

[4] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, Jan 2004.

[5] D. P. Siewiorek and R. S. Swarz, *Reliable Computer Systems (3rd Ed.): Design and Evaluation*. Natick, MA, USA: A. K. Peters, Ltd., 1998.

[6] J.-C. Geffroy and G. Motet, *Design of Dependable Computing Systems*. Hingham, MA, USA: Kluwer Academic Publishers, 2002.

[7] L. D. Gowen, J. S. Collofello, and F. W. Calliss, "Preliminary hazard analysis for safety-critical software systems," in *Eleventh Annual International Phoenix Conference on Computers and Communication [1992 Conference Proceedings]*, April 1992, pp. 501–508.

[8] R. Ernst and M. D. Natale, "Mixed criticality systems - a history of misconceptions?" *IEEE Design Test*, vol. 33, no. 5, pp. 65–74, Oct 2016.

[9] "Fault tree analysis (FTA)," International Electrotechnical Commission, Geneva, CH, Standard, 2006.

[10] "Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA)," International Electrotechnical Commission, Geneva, CH, Standard, Mar. 2006.

[11] "DO-178C, Software Considerations in Airborne Systems and Equipment Certification," Radio Technical Commission for Aeronautics, U.S.A, Standard, Jan. 2012.

[12] "Functional safety of electrical/electronic/programmable electronic safety-related-system," International Electrotechnical Commission, Geneva, CH, Standard, 2010.

[13] "Road vehicles – Functional safety," International Organization for Standardization, Geneva, CH, Standard, Nov. 2011.

[14] I. Agirre, M. Azkarate-Askasua, A. Larrucea, J. Perez, T. Vardanega, and F. J. Cazorla, "A safety concept for a railway mixed-criticality embedded system based on multicore partitioning," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, Oct 2015, pp. 1780–1787.

[15] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie, "Scheduling real-time mixed-criticality jobs," *IEEE Transactions on Computers*, vol. 61, no. 8, pp. 1140–1152, Aug 2012.

[16] S. K. Baruah, V. Bonifaci, G. D'Angelo, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie, *Mixed-Criticality Scheduling of Sporadic Task Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 555–566.

[17] A. Burns and R. I. Davis, "Mixed criticality systems - a review," 2015.

[18] S. K. Baruah, A. Burns, and R. I. Davis, "Response-time analysis for mixed criticality systems," in *2011 IEEE 32nd Real-Time Systems Symposium*, Nov 2011, pp. 34–43.

[19] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973. [Online]. Available: http://doi.acm.org/10.1145/321738.321743

[20] P. J. Graydon and I. Bate, "Safety assurance driven problem formulation for mixed-criticality scheduling," 2013.

[21] M. Joseph and P. Pandya, "Finding response time in real-time system," vol. 29, pp. 390–395, 10 1986.

[22] N. Audsley, "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times," 1991.

[23] F. Dorin, P. Richard, M. Richard, and J. Goossens, "Schedulability and sensitivity analysis of multiple criticality tasks with fixed-priorities," *Real-Time Systems*, vol. 46, no. 3, pp. 305–331, Dec 2010. [Online]. Available: https://doi.org/10.1007/s11241-010-9107-4

[24] V. Bonifaci and A. Marchetti-Spaccamela, "Feasibility analysis of sporadic real-time multiprocessor task systems," *Algorithmica*, vol. 63, no. 4, pp. 763–780, Aug 2012. [Online]. Available: https://doi.org/10.1007/s00453-011-9505-6

[25] S. Asyaban and M. Kargahi, "An exact schedulability test for fixed-priority preemptive mixed-criticality real-time systems," *Real-Time Syst.*, vol. 54, no. 1, pp. 32–90, Jan. 2018. [Online]. Available: https://doi.org/10.1007/s11241-017-9287-2

[26] R. I. Davis and A. Burns, "Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems," *Real-Time Systems*, vol. 47, no. 1, pp. 1–40, Jan 2011. [Online]. Available: https://doi.org/10.1007/s11241-010-9106-5

[27] S. Baruah and S. Vestal, "Schedulability analysis of sporadic tasks with multiple criticality specifications," in *2008 Euromicro Conference on Real-Time Systems*, July 2008, pp. 147–155.

[28] S. K. Baruah, V. Bonifaci, G. D'Angelo, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie, "Mixed-criticality scheduling of sporadic task systems," in *Proceedings of the 19th European Conference on Algorithms*, ser. ESA'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 555–566. [Online]. Available: http://dl.acm.org/citation.cfm?id=2040572.2040633

[29] S. Baruah, V. Bonifaci, G. DAngelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in *2012 24th Euromicro Conference on Real-Time Systems*, July 2012, pp. 145–154.

[30] P. Ekberg and W. Yi, "Bounding and shaping the demand of generalized mixed-criticality sporadic task systems," *Real-Time Systems*, vol. 50, no. 1, pp. 48–86, Jan 2014. [Online]. Available: https://doi.org/10.1007/s11241-013-9187-z

[31] A. Schrijver, *Theory of Linear and Integer Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1986.

[32] B. Lisper and P. Mellgren, "Response-time calculation and priority assignment with integer programming methods," in *Proc. Work-in-progress and Industrial Sessions, 13th Euromicro Conference on Real-Time Systems*, E. Tovar and C. Norström, Eds., June 2001, pp. 13–16. [Online]. Available: http://www.es.mdh.se/publications/282-

[33] Z. Al-bayati, B. H. Meyer, and H. Zeng, "Fault-tolerant scheduling of multicore mixed-criticality systems under permanent failures," in *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Sept 2016, pp. 57–62.

[34] G. Cornuéjols, "Valid inequalities for mixed integer linear programs," *Mathematical Programming*, vol. 112, no. 1, pp. 3–44, Mar 2008. [Online]. Available: https://doi.org/10.1007/s10107-006-0086-0

[35] D. de Niz and L. T. X. Phan, "Partitioned scheduling of multi-modal mixed-criticality real-time systems on multiprocessor platforms," in *2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2014, pp. 111–122.

[36] H.-M. Huang, C. Gill, and C. Lu, "Implementation and evaluation of mixed-criticality scheduling approaches for sporadic tasks," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 4s, pp. 126:1–126:25, Apr. 2014. [Online]. Available: http://doi.acm.org/10.1145/2584612

[37] L. Sha, "Solutions for some practical problems in prioritized preemptive scheduling," *Proc. 7th IEEE Real-Time Systems Symposium, 1986*, 1986. [Online]. Available: https://ci.nii.ac.jp/naid/80003373989/en/

[38] A. M. Gruzlikov, N. V. Kolesov, Y. M. Skorodumov, and M. V. Tolmacheva, "Task scheduling in distributed real-time systems," *Journal of Computer and Systems Sciences International*, vol. 56, no. 2, pp. 236–244, Mar 2017. [Online]. Available: https://doi.org/10.1134/S1064230717020101

[39] P. Huang, H. Yang, and L. Thiele, "On the scheduling of fault-tolerant mixed-criticality systems," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2014, pp. 1–6.

[40] L. Zeng, P. Huang, and L. Thiele, "Towards the design of fault-tolerant mixed-criticality systems on multicores," in *2016 International Conference on Compliers, Architectures, and Sythesis of Embedded Systems (CASES)*, Oct 2016, pp. 1–10.

[41] Z. Al-bayati, J. Caplan, B. H. Meyer, and H. Zeng, "A four-mode model for efficient fault-tolerant mixed-criticality systems," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 97–102.

[42] A. Thekkilakattil, R. Dobrin, and S. Punnekkat, "Fault tolerant scheduling of mixed criticality real-time tasks under error bursts," vol. 46, pp. 1148–1155, 12 2015.

[43] ——, "Mixed criticality scheduling in fault-tolerant distributed real-time systems," in *2014 International Conference on Embedded Systems (ICES)*, July 2014, pp. 92–97.

[44] A. Bhat, S. Samii, and R. Rajkumar, "Practical task allocation for software fault-tolerance and its implementation in embedded automotive systems," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2017, pp. 87–98.

[45] R. Davis, A. Burns, and D. Griffin, "On the meaning of pwcet distributions and their use in schedulability analysis," in *Real-Time Scheduling Open Problems Seminar*, June 2017.

[46] S. Altmeyer, L. Cucu-Grosjean, and R. I. Davis, "Static probabilistic timing analysis for real-time systems using random replacement caches," *Real-Time Systems*, vol. 51, no. 1, pp. 77–123, Jan 2015. [Online]. Available: https://doi.org/10.1007/s11241-014-9218-4

[47] G. Bernat, A. Colin, and S. M. Petters, "Wcet analysis of probabilistic hard real-time systems," in *23rd IEEE Real-Time Systems Symposium, 2002. RTSS 2002.*, 2002, pp. 279–288.

[48] B. Alahmad and S. Gopalakrishnan, "A Risk-Constrained Markov Decision Process Approach to Scheduling Mixed-Criticality Job Sets," in *Workshop on Mixed Criticality Systems (WMC 2016)*, Porto, Portugal, Nov. 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01403223

[49] D. I. Maxim, R. Davis, L. I. Cucu-Grosjean, and A. Easwaran, "Probabilistic Analysis for Mixed Criticality Scheduling with SMC and AMC," in *WMC 2016 - 4th International Workshop on Mixed Criticality Systems*, Porto, Portugal, Nov. 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01416310