# Reproducibility and representativity - mandatory properties for the compositionality of measurement-based WCET estimation approaches

C. Maxim[1,2], A. Gogonel[1], I. Asavoae[1], M. Asavoae[1], and L. Cucu-Grosjean[1]
[1]Inria and [2]Airbus
Email: *cristian.maxim@airbus.fr*

***Abstract*—The increased number of systems consisting of multiple interacting components imposes the evolution of timing analyses towards methods able to estimate the timing behavior of an entire system by aggregating timings bounds of its components. In this paper we propose the first discussion on the properties required by measurement-based timing analyses to ensure such compositionality. We identify the properties of reproducibility and representativity as necessary conditions to ensure the convergence of any measurement protocol allowing a compositional measurement-based timing analysis.**

## I. Introduction

Nowadays many real-time embedded systems consist of different interacting components. These components have functional characteristics while timing constraints should be satisfied locally (at the level of each component) as well as globally (by the entire system). While existing timing analyses may be applied at the level of the components, providing guarantees for the global timing behavior of the system requires either new methods, or compositionality properties such that the timing guarantees obtained at the level of the components may be combined into a global timing estimate.

The timing constraints may be ensured at different levels including the estimation of the worst case execution time (WCET) of programs on processors. Existing WCET estimation solutions to such (complex) systems fall into several orthogonal settings:

- the timing analyses may be based on **deterministic** or **probabilistic** grounds;
- the analysis techniques could be of **static** or **dynamic** nature;
- the system under analysis could be specified at **high-level** or **low-level** (e.g., on the model design, without or with limited architecture information or respectively on the binary code with accurate architecture description) etc.

Before addressing the more general problem of compositionality, we enumerate several points of interest for a compositional (deterministic or probabilistic) WCET analysis.

The nature of the system and its components could lead to several decomposition criteria. For example:

- in a typical single-core system, as shown in Figure 2, one possible decomposition is at the structural level, leading to separate analysis for pipeline, instruction cache, data cache;
- in a typical multicore system, as shown in Figure 3, one possible partition is at the (high-) functional level, having separate timing contributions for communication and computation.

These different decomposition criteria indicate that the answer to the question on *how to decompose a system?* is mandatory for any WCET estimation method.

The existing analyses developed at the level of the components should be re-integrated in a compositional way. For instance, in a single-core system, the may- and must-analyses for Least Recently Used (LRU) caches [1] are developed in isolation and integrated in the WCET analysis tool chains. We identify this integration concern as another mandatory property answering to the question - *how to transfer existing analyses in a compositional setting?*

Last but not least, the definition and the analysis of effects/interferences between the components is mandatory for a WCET estimation method to be compositional. For instance, in a multicore system the cache related preemption delay analysis models a type of interference in a preemptive execution environment. Thus, an answer to the question - *how to take into account inter-component interferences?* should be provided by any method for compositional WCET estimation.

These three questions, *how to decompose a system, how to transfer existing analyses in a compositional setting, and how to take into account inter-component interferences*, highlight important aspects on the compositionality of the WCET analysis. In this paper we provide a discussion on the transfer of existing analyses in a compositional setting for measurement-based WCET approaches and more precisely on properties of the WCET estimation method and the associated measurement protocol. To our best knowledge this discussion is the first proposed in this context and the purpose of this paper is to formalize concepts often used in industry such as reproducibility and convergence.

**Paper organization.** The is organized as follows. We have introduced in this section the context of our work. We motivate our interest in probabilistic measurement-based approaches by indicating the limitations of existing work in Section II. In Section III we present main concepts and challenges of static WCET estimation with respect to the compositionality problem and in Section IV a similar investigation of the probabilistic measurement-based WCET estimation methods. The

concepts of reproducibility and representativity are defined in Section V. In Section V we also discuss the impact of these two concepts on both the compositionality of a WCET estimation and the convergence of a measurement protocol. We present our future work as open problems related to our current contribution.

## II. Motivation of our position paper

The measurement-based approaches are widely used in the real-time embedded systems industry where the concept of *high water mark* (HWM) is considered as a safety margin added to the largest observed execution time. Lifting its utilization to systems with different components requires compositionality while two different components may have different values for the HWM of a program executed on those components. Moreover timing anomalies may prevent the HWM of a program to be obtained by the combination of the HWMs of the program on the components.

Lisper and al. [2] have studied the compositionality of the probabilistic measurement-based approaches. Such approaches require an independence hypothesis between the probability distributions in order to allow their combination by convolution operations. The authors of [2] have presented interesting results by indicating that the lack of independence has a low impact on the combination of two sequentially executed programs. Nevertheless the paper does not proceed at statistical testing of the independence of the execution times of the programs under study.

*Why the independence tests are necessary in order to conclude on a low impact of the independence on the compositionality?*

In the absence of appropriate testing, **the compositionality property showed by the authors may actually be introduced by the measurement protocol** producing the execution times. As showed below, the independence of a set of execution times may be obtained with an appropriate measurement protocol even in the presence of dependent programs. Before indicating how such measurement protocol may be proposed, we provide first the (necessary) definitions for independent programs, statistical independence, and probabilistic independence.

*Definition 1:* We consider two programs $Prog_1$ and $Prog_2$ to be *independent* iff any execution of $Prog_1$ may be done before or after any execution of $Prog_2$ without any impact on their execution times.

Two programs, that are in any other situation than those covered by the definition of independent programs given previously, are *dependent*.

Consider for instance the program $Prog_{ex1}$ described in Table I and $Prog_{ex2}$ described in Table II. These two programs are kept simple in order to ease the understanding.

The two programs $Prog_{ex1}$ and $Prog_{ex2}$ are dependent as $Prog_{ex1}$ produces a (positive integer) value for the global variable $var\_global_2$ that is then used as an input by $Prog_{ex2}$. For instance each time $var\_global_1 = 1$, then $Prog_{ex1}$ has an execution time equal to 3 time units and $Prog_{ex2}$ has an execution time equal to 6 time units. For $var\_global_1 = 2$, then $Prog_{ex1}$ has an execution time equal to 4 time units and $Prog_{ex2}$ has an execution time equal to 10 time units.

TABLE I
Body of program $Prog_{ex1}$

| $Prog_{ex1}$ ($var\_global_1$); | |
|---|---|
| $value$ = $var\_global_1$; | // *execution time* = 1 *time unit* |
| **for** i = 1 **to** $var\_global_1$ | // *the loop cost is in the instr* |
| $value$ = $value$ + 1; | // *execution time* = 1 *time unit* |
| **endfor** | |
| $var\_global_2$ = 2* $value$; | // *execution time* = 1 *time unit* |

TABLE II
Body of program $Prog_{ex2}$

| $Prog_{ex2}$ ($var\_global_1$, $var\_global_2$); | |
|---|---|
| $value$ = $var\_global_2$; | // *execution time* = 1 *time unit* |
| **for** i=1 **to** $var\_global_2$ | // *loop cost is in the instr* |
| $value$ = $value$ + 1; | // *execution time* = 1 *time unit* |
| **endfor** | |
| $avg\_global = \frac{value+var\_global_1}{2}$; | // *execution time* = 1 *time unit* |

For these two programs we may obtain both statistical dependent execution times or statistical independent execution times.

*Definition 2:* Two probability distributions $\mathcal{C}_1$ and $\mathcal{C}_2$ are *independent* iff

$$P(\{\mathcal{C}_1 = c_1\} \cap \{\mathcal{C}_2 = c_2\}) = P(\mathcal{C}_1 = c_1) \cdot P(\mathcal{C}_2 = c_2)$$

For instance the probability distributions of the execution times $pET(Prog_{ex1})$ and the probability distributions of the execution times $pET(Prog_{ex2})$ are *probabilistically dependent* as there is a relation between the probability of appearance of an execution time for $Prog_{ex1}$ and the probability of appearance of an execution time for $Prog_{ex2}$. If one would like to estimate the probability distributions of the execution times of these two programs executed sequentially then, given their probabilistic dependence, a complex probabilistic operation is necessary to take into account the conditional probabilities.

*Definition 3:* A set $A$ is *statistically independent* iff its elements are generated in a random manner (i.e., the value generated at one instant only depends on the generator and not on the values generated before).

For instance $A_{ex} = \{8, 18, 21, 24, 28, 30\}$ is statistically independent. We have generated $A_{ex}$ using an on-line random generator[1].

**Statistically independent execution times for dependent programs.** We consider the independent set $A_{ex}$ as input to obtain independent execution times for our two programs, $Prog_{ex1}$ and $Prog_{ex2}$. If we consider $var\_global_1$ to take the values from $A_{ex}$ then the set of execution times of $Prog_{ex1}$ is $\mathcal{C}_{Prog_{ex1}} = \{10, 20, 23, 26, 30, 32\}$ which is statistically independent.

[1]http://www.infowebmaster.fr/outils/generateur-nombre-aleatoire.php, (on-line form), but the reader may use any other such generator.

In order to obtain, for $Prog_{ex2}$, a set of statistically independent execution times we consider the values of $var\_global_2$ to take the values from (another) statistically independent set $A_{bis} = \{1, 45, 59, 75, 88, 90\}$. We obtain a set of statistically independent execution times for $Prog_{ex2}$ equal to $\mathcal{C}_{Prog_{ex2}} = \{3, 47, 61, 77, 90, 92\}$. These two sets of execution times are statistically independent, while the programs are dependent.

**Statistically dependent execution times for dependent programs.** Moreover if we use a set of dependent elements like $B = \{1, 2, 3, \cdots, 8\}$, then we may obtain statistical dependent sets for the execution times of $Prog_{ex1}$ and $Prog_{ex2}$. In this case the execution times of $Prog_{ex1}$ are $\{3, 4, 5, 6, 7, 8, 9, 10\}$ and the execution times of $Prog_{ex2}$ are $\{6, 10, 14, 18, 22, 26, 30, 34\}$. These two sets are statistically dependent, while the programs are dependent.

In conclusion **the measurement protocol has a direct impact on the statistical independence of the execution times** while this independence is mandatory for the compositionality of probabilistic WCET estimates (see Section IV for a definition of a pWCET estimation). For two programs with pWCET estimates we may proceed at a convolution (or other composition) of their bounds [3]. Nevertheless, the existence of this bound is not only requiring statistical independence but also the convergence of the WCET measurement-based estimation (see Section V for more details). We present first some preliminary considerations on the static WCET and probabilistic WCET estimations in order to ease the understanding of Section V.

## III. Considerations on existing static WCET analysis

One of the challenges of static WCET analysis is to analyze individual components and then to safely compose their results. It becomes crucially important how to manipulate the system state, at a structural level, in order to safely capture the system functionality over time from the functionality of sub-components. The implicit path enumeration technique (IPET), which translates the timing analysis problem into integer linear programming (ILP) solving, became the standard for static WCET analysis, as in [4]. The WCET analysis workflow, shown in Figure 1, consists of several distinct phases (e.g., control-flow graph (CFG) reconstruction, control-flow analysis, micro-architectural analysis, etc) and outputs safe and (hopefully) tight WCET bounds of the given program on the specified architecture. We elaborate next on the WCET analysis workflow using IPET and how this is positioned w.r.t. compositionality.

### A. Software Representation - static WCET estimation

The program under analysis is considered at the binary level and the first phase is to reconstruct its CFG from the binary. The CFG reconstruction phase outputs a safe over-approximation of the actual CFG, with some precision being lost for indirect jump instructions. The CFG, whose nodes are basic blocks, becomes *the working structure* for the program semantics transfer.

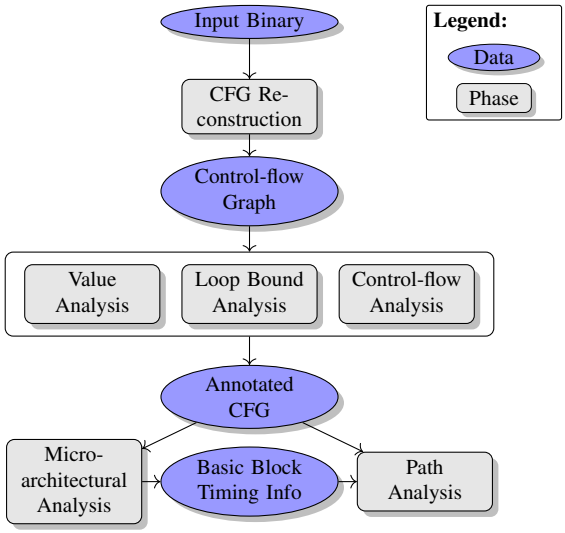


Fig. 1. Typical workflow for static WCET analysis

The WCET analysis workflow considers several standard program analyses: value analysis, loop bound analysis, and control-flow analysis. Value analysis is a variant of interval analysis and computes address ranges of load/store instructions. Loop bound analysis computes safe bounds for the program loops while the control-flow analysis is specialized in infeasible path detection (i.e., those program paths which cannot be executed under any given input).

*The method* of choice is abstract interpretation (AI), a technique which soundly computes abstract properties of the program [5]. It is difficult to compose abstractions in AI [6], the main impediment consisting in *how to handle the overlapping functionalities of the components*. A suitable decomposition should exploit the key concept of a state configuration. The state configuration is the set of all semantic entities which are necessary to characterize the program execution. For example, the state configuration of a binary program is given by the general registers (i.e., integer and floating point) and special registers (e.g., program counter, the stack pointer).

Strictly from a software representation point of view, the WCET analysis based on IPET does not have compositionality issues because each of the aforementioned analyses consider the entire program semantics (and hence the complete state configuration). The results of all three analyses are represented on the program CFG. The compositionality becomes an issue when the program is executed on a hardware model, which has its own state configuration. We address this point next.

### B. Hardware Representation - static WCET estimation

The WCET analysis workflow considers several static analyses of architectural elements (e.g., instruction cache, data cache, branch speculation, etc), grouped under the name of *processor behavior analysis*. The results of each of these analyses, i.e., how the architecture behaves, are projected on the program CFG, at the node (i.e., basic block) level. Standard examples of processor behavior analyses are **may**, **must**, and

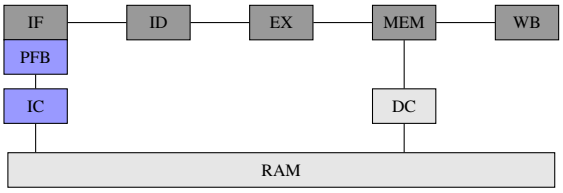


Fig. 2. Standard design of a single-core system where a timing anomaly could appear because of prefetching (i.e., interaction between PFB and IC)

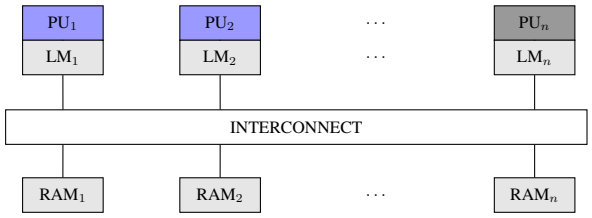


Fig. 3. Standard design of a multicore system where a timing anomaly could appear when scheduling tasks on processing components $\mathrm{PU}_1$ and $\mathrm{PU}_2$

**persistency** analyses for Least Recently Used (LRU) caches, surveyed in [1].

There are two hardware platforms of interest, single-core and multicore, which we briefly present next. The state configuration of each platform (that, we remind, is the set of semantics entities to describe an execution on a particular platform) consists of the state configuration to describe each component of the platform. One challenge in WCET static analysis w.r.t. compositionality is introduced by *timing anomalies* that appear when, due to component interaction, the local (i.e., component level) worst-case behavior does not produce the global (i.e., system level) worst-case.

**Single-core.** A typical single-core, depicted in Figure 2, consists of several interacting components. One component example is an in-order pipeline with 5-stages: fetch (IF), decode (ID), execute (EX), memory access (MEM), and write-back (WB). The pipeline interacts with the memory components such as caches for instruction (IC) and data (DC) and the RAM memory. Note that the pipeline has a prefetch buffer (PFB), which could be polluted with wrongly speculated instructions from the instruction cache and, in consequence, may produce timing anomalies, as shown in [7].

**Multicore.** A multicore system, Figure 3, has as components the processors (or processing units $\mathrm{PU}_i$), their local memories ($\mathrm{LM}_i$), one or more bus/interconnect, and the memory banks ($\mathrm{RAM}_i$). If two processing units, say $\mathrm{PU}_1$ and $\mathrm{PU}_2$, use overlapping resources (e.g., they access the same memory bank, say $\mathrm{RAM}_1$, or they send requests on the bus in the same time) the entire system is prone to have timing anomalies, as noted in [8].

## IV. Preliminaries on probabilistic WCET analysis

The measurement-based approaches, in general, and the probabilistic measurement-based approaches, in particular, propose WCET estimates using the execution times of the program on the given platform (see Figure 4). More precisely

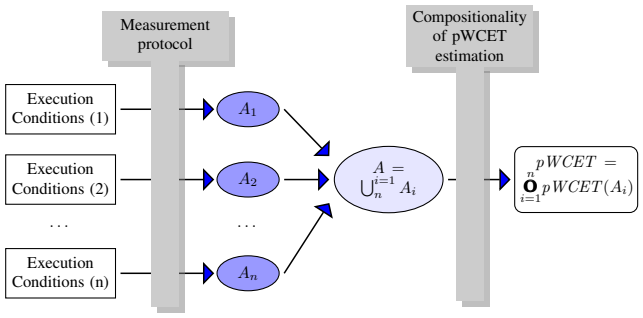


Fig. 4. The protocol of a (p)WCET estimation from different scenarios of execution conditions

let $C_1^i, C_2^i, \cdots, C_n^i$ be $n$ consecutive executions of a program on a processor starting from a given scenario of execution $S_i$. A scenario of execution for a program on a processor is defined by a set of states corresponding to different execution time variability factors. A scenario of execution could correspond for instance to the pair (path of the program, state of the cache) or any other information related to the execution of the program.

For a scenario $S_i$ we may define a probabilistic execution time $\mathcal{C}_i$ as an empirical probability distribution of the execution time of that program for the given processor. For instance we have $\mathcal{C}_i$ defined as follows:

$$\mathcal{C}_i = \begin{pmatrix} 2 & 3 & 5 & 6 & 105 \\ 0.7 & 0.2 & 0.05 & 0.04 & 0.01 \end{pmatrix} \quad (1)$$

The probabilistic worst-case execution time (pWCET) $\mathcal{C}$ of that program is an upper bound on all possible probabilistic execution times $\mathcal{C}_i$ for all possible execution scenarios $S_i, \forall i \geq 1$. The relation $\succeq$ describes the relation between the probabilistic execution times (pETs) of a program and its probabilistic worst case execution time (pWCET), i.e., $\mathcal{C} \succeq \mathcal{C}_i$, $\forall i$, defined as follows:

*Definition 4:* We say that $\mathcal{C} \succeq \mathcal{C}_i$ or $\mathcal{C}$ is worse than $\mathcal{C}_i$ if its complementary cumulative distribution function (1-CDF) has a higher or equal probability associated to each possible value, i.e., $P(C \geq c) \geq P(C_i \geq c)$, $\forall c$.

For instance, the probability distribution defined in Figure 5 has its 1-CDF provided in Figure 6 (blue). Moreover, the probability distribution defined in Figure 5 is upper bounded by the probability distribution described by Figure 6 (in red). Note, however, that there may exist two random variables that are not comparable with respect to the relation $\succeq$.

In the remainder of this paper we consider that the processor is fixed in sense that we estimate the pWCET of a program on a processor from the execution time measurements of the program on that given processor.

### A. Software Representation - Probabilistic

From a probabilistic point of view, a program can be represented as a generator of execution time profile sets (ETPs). Combining all these ETPs we obtain an absolute

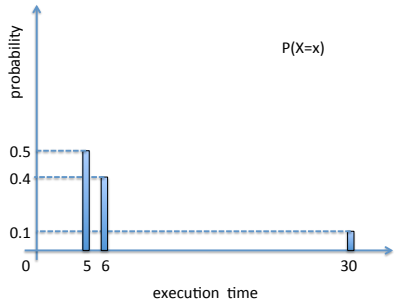


Fig. 5. A probability distribution function (PDF)

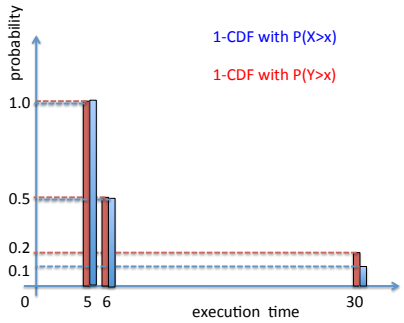


Fig. 6. 1-CDF associated to the PDF defined in Figure 5 and 1-CDF that upper bounds the random variable described in Figure 5 (red)

domain of execution times. In practice, such a domain is hard to impossible to determine through measurement for complex programs running on non-deterministic hardware. Discovering which ETP sets have a higher influence on the pWCET estimation would allow us to concentrate on their analysis in order to produce a reliable pWCET without knowing the entire domain of execution time.

The key in highlighting the influential ETP sets stays in the structure of the program and the representation of the input-output relations. Every probabilistic analysis should start with the definition of the domain of analysis and the decision of the interval in which our program's inputs appear. The choice of this interval defines the number of ETPs we need to consider for analysis. The program semantics is afterwards projected on probabilities, each path and its weight are processed in order to compute the ETP sets and its influence on the total domain of execution times.

The method we may choose for maximizing the domain coverage of the measurement protocol is path coverage testing and variants of static analyses with probabilistic abstract domains. In [9] authors propose an abstract interpretation of probabilistic systems, in [10] a reasoning on probabilistic execution times, and in [11] imprecise probabilities are used to define a program input. [12] presents a tool support for determining the compliance with probabilistic WCET analysis.

In the followings we assume having an ETP obtained with a measurement protocol adequate for maximizing the domain coverage (e.g., the above exemplified). Hence, we lift the WCET analysis from the level of program/architecture and we focus on the compositional features of the ETP processing for obtaining pWCET.

## V. Reproducibility and Representativity of Measurement-Based Approaches

In this section we identify and characterize the *convergence*– a key feature for the compositionality of a measurement-based WCET estimation process. Any measurement-based WCET estimation process has two main parts: (i) the measurement protocol and (ii) the WCET estimation method.

The convergence of a measurement-based WCET estimation process for a program on a processor is defined by the existence of a finite set of execution times provided by a measurement protocol such that the associated measurement-based WCET estimation method provides a unique WCET estimation of that program on the given processor. A more formal definition is provided in Definition 5.

*Definition 5:* Given $A$ an absolute domain of execution times, a measurement-based WCET estimation process $pWCET$ is *convergent* if for any ascending chain of subsets $A_i \subset A$ (obtained using its measurement protocol) converging to $A$ (i.e., $A_i \subseteq A_{i+1}, \forall i$ and $\lim_i A_i = A$) the associated chain of WCET estimations (obtained using its WCET estimation method) is almost constant, equal (or sufficiently close) to the WCET estimation of $A$ (i.e., $\exists t \geq 0\ \forall j \geq t$ such that $pWCET(A_j) \approx pWCET(A)$).

The convergence of a measurement-based WCET estimation requires several properties to be satisfied. We identify in this document a (non-exhaustive) list of these mandatory properties (their order of presentation is not relevant):

- The reproducibility of the WCET estimation method (defined Section V-A);
- The reproducibility of the measurement protocol (defined Section V-B);
- The representativity of the measurement protocol (defined Section V-C).

We present in Section V-D the relations between these three properties and the convergence.

### A. The reproducibility of the WCET estimation method

We may note that the measurement-based WCET estimation method is used several times over subsets of ETPs, e.g., $A_i$ in Definition 5. If two different utilizations of the estimation method on (exactly) the same subset $A_0$ of execution times provide different WCET estimates then the measurement-based WCET estimation diverges and it cannot provide a reliable result.

*Definition 6:* A measurement-based WCET estimation method $pWCET$ is *reproducible* iff for any two utilizations $i$ and $j$ the estimates $pWCET^i(A_0)$ and $pWCET^j(A_0)$ $(i \neq j)$ are the same.

In Figure 7 we depict the notion of reproducibility of a pWCET estimation. For example, the EVT-based pWCET estimation method (introduced in [13]) is reproducible as long as the order of the elements in $A_0$ of the execution times is not modified.

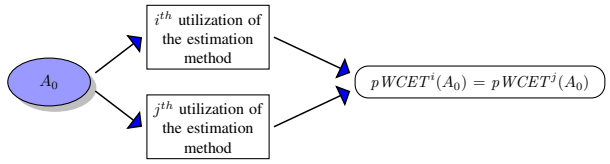


Fig. 7. The WCET estimation is the same for different utilizations $i$, $j$ of a reproducible WCET estimation method from the exactly same ordered set of execution times

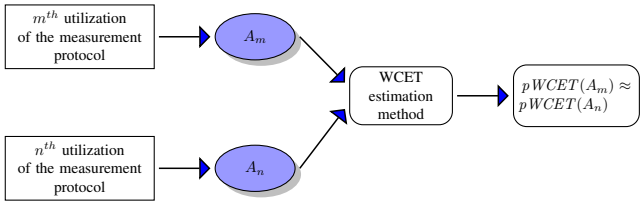


Fig. 8. Different utilizations of a reproducible measurement protocol provides WCET estimates that are equal (or sufficiently close)

### B. The reproducibility of the measurement protocol

The measurement protocol is an essential step in the measurement-based WCET estimation. We now focus on characterizing this step w.r.t. the convergence property.

*Definition 7:* A measurement protocol $P$ is *reproducible* iff for two different utilizations $P_m$ and $P_n$ with the same execution conditions (status of the processor, program and external factors) the obtained set of execution times $A_m$ and respectively $A_n$ correspond to equal (or sufficiently close) WCET estimates for the utilization of the same WCET estimation method $pWCET$, i.e., $pWCET(A_m) \approx pWCET(A_n)$.

In Figure 8 we depict the reproducibility of a measurement protocol. Note that a completely randomized measurement protocol may not be reproducible with respect to the EVT-based pWCET estimation method. For instance, given a randomized cache replacement policy, if both the seed of the random generator and the places in caches are randomly picked, then the architecture execution times may not be equivalent as different associated pWCET estimates may be obtained with EVT-based pWCET estimation methods.

We also note that the randomization of only the input values for a program is not a reproducible measurement protocol either when considering an EVT-based pWCET estimation method [14], [15]. This absence of the reproducibility is due to the sensitivity of EVT-based pWCET estimation method to the order of the execution times.

### C. Representativity of a measurement protocol

We now present a second feature of the measurement protocol which contributes to ensuring the convergence property.

*Definition 8:* A measurement protocol is *representative* iff there exists a number $k$ of execution times for a measurement protocol such that

$$pWCET(A_{k'}) \approx pWCET(A), \forall A_{k'}: A_k \subseteq A_{k'} \subseteq A \qquad (2)$$

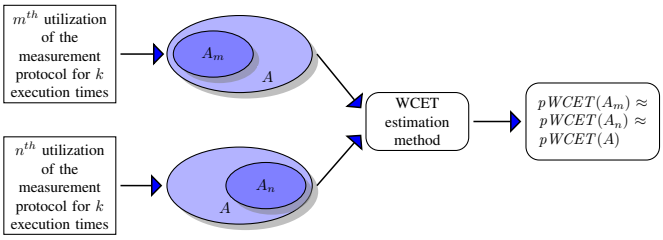


Fig. 9. A representative measurement protocol provides equivalent subsets of execution times

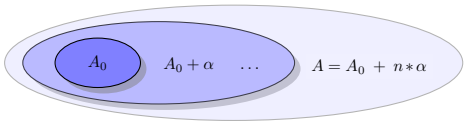


Fig. 10. The absence of the reproducibility of a measurement protocol may prevent $A_0$ to converge to $A$

for any $A_k \subseteq A$ with $|A_k| = k$.

In Figure 9 we depict the representativity of a measurement protocol, where we denote by $A$ the ETP of the WCET estimation, while $A_m$ and $A_n$ denote some subsets of execution times of cardinal $k$.

Using the notations of Figure 9, we may indicate that measurements obtained using randomized replacement policies (with the method provided in [16]) seem to present a representativity of the HW-randomization measurement protocol for $m = 6$ utilizations of the protocol for $k = 1000$. Note that the original set has 500 execution times in the presence of Mälardalen benchmarks [17]. Nevertheless there is currently no proof that such representativity may be extended to other classes of programs.

Note that the random picking of program inputs is not by default a representative measurement protocol. However, such protocol should define a representativity property with respect to the pWCET estimation method.

To our best knowledge, there exists no proof for the representativity of a measurement protocol for any given set $A$.

### D. Relations between reproducibility, representativity and convergence

We enumerate the relations between the concepts defined previously:

- The reproducibility of the WCET estimation method is a mandatory property for the reproducibility of the measurement protocol. Indeed if the WCET estimation method is not reproducible than for two same sets of execution times provided by a measurement protocol, the WCET estimates could be non-equal.
- The reproducibility of a WCET estimation process requires both the reproducibility of the WCET estimation method and the reproducibility of measurement protocol. Indeed if the WCET measurement protocol is not reproducible, the WCET estimate will be modified for each new measurement even in presence of a reproducible WCET estimation method.

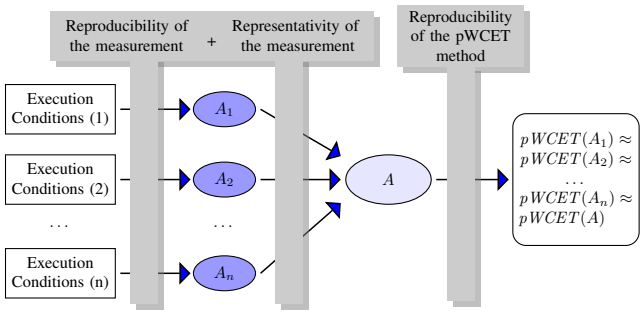


Fig. 11. The impact of the reproducibility and the representativity on the convergence of a measurement-based WCET estimation

- The reproducibility of the WCET estimation process and the representativity of the measurement protocol are *mandatory properties* for the convergence of a WCET estimation process. In Figure 10 we illustrate a convergence principle by slowly increasing an initial set of execution times by $\alpha$ elements. The absence of the reproducibility of the measurement protocol makes the measurement-based WCET estimation process unable to converge. Namely, let $X$ and $Y$ with $|X| = |Y| = \alpha$ be two disjoint input sets produced by the measurement protocol at step $n-1$. If $pWCET(A_0+(n-1)\alpha)$ produces two different results (when adding to $A_0+(n-2)\alpha$ either $X$ or $Y$) then the set $A = X \cup Y \cup A_0+(n-2)\alpha$ diverges since $pWCET$ may produce two different WCET estimates.

In Figure 11 we illustrate a measurement-based WCET protocol with relations between the three properties. For instance from execution conditions (1) the measurement reproducibility ensures that an unique set of execution times $A_1$ is obtained.

The relations described previously are stated in presence of any WCET estimation method. If the WCET estimation method is transitive, then a stronger relation between the representativity and the reproducibility of a measurement protocol may be established.

*Theorem 1:* If a measurement protocol is representative, then the measurement protocol is reproducible for any set of execution times with a cardinal larger or equal to $k$.

*Proof:* We prove the reproducibility of a measurement protocol by contradiction. We suppose that the measurement protocol is not reproducible for any set of execution times with a cardinal larger or equal to $k$. This implies that there exist two utilizations $i \neq j$ of the measurement protocol $A^i$ and $A^j$, with $|A^i| \geq k$ and $|A^j| \geq k$, such that

$$pWCET(A^i) \neq pWCET(A^j) \tag{3}$$

From the definition of the representativity we have that

$$pWCET(A^i) \approx pWCET(A) \tag{4}$$

and

$$pWCET(A^j) \approx pWCET(A) \tag{5}$$

From the transitivity of the relation $\approx$ and Equations (4) and (5) we obtain

$$pWCET(A^i) \approx pWCET(A^j) \tag{6}$$

We obtain the contradiction between Equation (3) and Equation (6) indicating that our initial hypothesis is not correct, thus we prove that the measurement protocol is reproducible.

## VI. Related Work

Compositionality together with abstraction are key principles in developing complex systems. A system consists of interacting components and the behavior of each component is captured by an input-output interface [18]. While much of the existing work focuses on the functionality aspects, the real-time embedded systems require reasoning about timing. Time is then explicitly represented at the level of input-output interfaces, as in [19], and conveniently enables compositionality via a time-centric abstraction-refinement procedure, as in [20].

The static WCET analyses are surveyed in [4] and timing compositionality in static WCET analysis is defined in [21]. One of the very few solutions based on static methods for compositionality is proposed in [22], as *MRTA* - a framework for response time analysis on multicores. MRTA considers the response time calculation in the presence of *interferences* but it does not focus on how interferences may be estimated.

Timing anomalies are a challenging issue for compositionality in static analysis. These are introduced in [23] where the necessary conditions for timing anomalies to appear in dynamically scheduled processors are investigated. The presence of timing anomalies is a culprit for non-scalability or even unsoundness in WCET static analysis. For example, timing anomalies exhibit, in a degenerate case, the so-called domino effect which causes unboundedness in the underlying state space. The domino effect is studied in [7] where the solution proposed is a *state-based time variation bound*. This bound is precomputed for each architecture and helps pruning the next-state set at each step during the static analysis.

The first probabilistic approaches [24], [25], [26] for real-time systems have been proposed in the late 90s to support the *soft real-time constraints* satisfaction. Even if those approaches were using the commonly accepted hypothesis that larger values of the execution times of a program have lower probability of appearance, these first results concern average timing behaviors. Indeed the real-time community considers at that moment that the probabilistic approaches are only able to ensure constraints that are defined using a ratio of non-satisfied timing constraints within a given time interval (mainly multimedia systems).

Few years later two seminal papers [13], [27] introduce almost simultaneously two major concepts for the probabilistic approaches. The first one is that the worst execution times of a program are probably rare events, [13], and is related to the first utilization of EVT [28] in the context of execution times estimation. The second concept is that the probabilistic analysis may cover worst case scenarios - the first probabilistic

worst case reasoning considers the program instances with the (proved) largest response time to provide probabilistic upper bounds on the response time of any instance.

The follow-up works of [16], [29], [30], [31], [32] enforce the impact of the two concepts by underlining that the probabilistic approaches may be used in different contexts of *hard real-time industries*.

## VII. Conclusion

In this paper we have provided the first intuitive mandatory properties for the convergence of measurement-based WCET estimation processes: reproducibility and representativity. The reproducibility describes the stability of the result w.r.t. different executions of the process. The representativity describes the existence of a (small enough) number of input measures that leads to the correct global result of the process. The provided examples are described in the context of probabilistic approaches but we expect these properties to remain true for any measurement-based WCET estimation process.

We identify an important thread of future work in providing proofs of compositionality for the existing measurement-based methods following the framework we have introduced in this paper. In particular, we would like to study methodologies for proving and detecting convergence via reproducibility and representativity of measurement-based WCET estimation.

## References


[1] J. Reineke, "Caches in WCET analysis: Predictability - competitiveness - sensitivity," Ph.D. dissertation, Saarland University, 2009.

[2] M. Santos, B. Lisper, G. Lima, and V. Lima, "Sequential composition of execution time distributions by convolution," in *Proc. 4th Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS 2011)*, November 2011, pp. 30–37.

[3] L. Cucu-Grosjean, "Independence - a misunderstood property of and for (probabilistic) real-time systems," in *"Real-Time Systems: the past, the present, and the future", the 6Oth birthday of Prof. Alan Burns*, 2013.

[4] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. B. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. P. Puschner, J. Staschulat, and P. Stenström, "The worst-case execution-time problem - overview of methods and survey of tools," *ACM Trans. Embedded Comput. Syst.*, vol. 7, no. 3, 2008.

[5] P. Cousot and R. Cousot, "Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages (POPL)*, 1977, pp. 238–252.

[6] ——, "Modular static program analysis," in *Compiler Construction, 11th International Conference, CC 2002*, 2002, pp. 159–178.

[7] J. Reineke and R. Sen, "Sound and efficient WCET analysis in the presence of timing anomalies," in *9th Intl. Workshop on Worst-Case Execution Time Analysis, WCET*, 2009.

[8] I. Wenzel, R. Kirner, P. P. Puschner, and B. Rieder, "Principles of timing anomalies in superscalar processors," in *Fifth International Conference on Quality Software (QSIC)*, 2005, pp. 295–306.

[9] P. Cousot and M. Monerau, "Probabilistic abstract interpretation," in *Programming Languages and Systems - 21st European Symposium on Programming, ESOP*, 2012, pp. 169–193.

[10] L. David and I. Puaut, "Static determination of probabilistic execution times," in *16th Euromicro Conference on Real-Time Systems (ECRTS)*, 2004, pp. 223–230.

[11] A. Adjé, O. Bouissou, J. Goubault-Larrecq, E. Goubault, and S. Putot, "Static analysis of programs with imprecise probabilistic inputs," in *Verified Software: Theories, Tools, Experiments - 5th International Conference, VSTTE 2013*, 2013, pp. 22–47.

[12] F. Guet, L. Santinelli, and J. Morio, "On the Reliability of the Probabilistic Worst-Case Execution Time Estimates," in *8th European Congress on Embedded Real Time Software and Systems (ERTS)*, Jan. 2016.

[13] S. Edgar and A. Burns, "Statistical analysis of WCET for scheduling," in *the 22nd IEEE Real-Time Systems Symposium*, 2001.

[14] G. Lima, D. Dias, and E. Barros, "Extreme value theory for estimating task execution time bounds: A careful look," in *28th Euromicro Conference on Real-Time Systems*, 2016, pp. 200–211.

[15] L. Yue, I. Bate, T. Nolte, and L. Cucu-Grosjean, "A new way about using statistical analysis of worst-case execution times," *ACM SIGBED Review*, September 2011.

[16] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzeti, E. Quinones, and F. Cazorla, "Measurement-based probabilistic timing analysis for multi-path programs," in *the 24th Euromicro Conference on Real-time Systems*, 2012.

[17] J. Gustafsson, A. Betts, A. Ermedahl, and B. Lisper, "The Mälardalen WCET benchmarks – past, present and future," in *the International Workshop on Worst-case Execution-time Analysis*, 2010.

[18] N. A. Lynch and M. R. Tuttle, "An introduction to input/output automata," *CWI Quarterly*, vol. 2, pp. 219–246, 1989.

[19] L. d. Alfaro, T. A. Henzinger, and M. Stoelinga, "Timed interfaces," in *Proceedings of the Second International Conference on Embedded Software*, ser. EMSOFT '02, 2002, pp. 108–122.

[20] M. Geilen, S. Tripakis, and M. Wiggers, "The earlier the better: A theory of timed actor interfaces," in *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '11, 2011, pp. 23–32.

[21] S. Hahn, J. Reineke, and R. Wilhelm, "Towards compositionality in execution time analysis: definition and challenges," *SIGBED Review*, vol. 12, no. 1, pp. 28–36, 2015.

[22] S. Altmeyer, R. I. Davis, L. S. Indrusiak, C. Maiza, V. Nélis, and J. Reineke, "A generic and compositional framework for multicore response time analysis," in *Proceedings of the 23rd International Conference on Real Time Networks and Systems, RTNS*, 2015, pp. 129–138.

[23] T. Lundqvist and P. Stenström, "Timing anomalies in dynamically scheduled microprocessors," in *Proceedings of the 20th IEEE Real-Time Systems Symposium, 1999*, 1999, pp. 12–21.

[24] T. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L. Wu, and J. Liu, "Probabilistic performance guarantee for real-time tasks with varying computation times," in *the 2nd IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS95)*, 1995, pp. 164–174.

[25] M. Gardner and J. Lui, "Analyzing stochastic fixed-priority real-time systems," in *the 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS99)*, 1999, pp. 44–58.

[26] L. Abeni and G. Buttazzo, "Integrating multimedia applications in hard real-time systems," in *the 19th IEEE Real-Time Systems Symposium (RTSS98)*, 1998, pp. 4–13.

[27] J. Díaz, D. Garcia, K. Kim, C. Lee, L. Bello, L. J.M., and O. Mirabella, "Stochastic analysis of periodic real-time systems," in *the 23rd IEEE Real-Time Systems Symposium (RTSS02)*, 2002.

[28] B. Gnedenko, "Sur la distribution limite du terme maximum d'une seris aleatoire," *Annals of Mathematics*, vol. 44, pp. 423–453, 1943.

[29] F. Wartel, L. Kosmidis, C. Lo, B. Triquet, E. Quinones, J. Abella, A. Gogonel, A. Baldovin, E. Mezzetti, L. Cucu, T. Vardanega, and F. Cazorla, "Measurement-based probabilistic timing analysis: Lessons from an integrated-modular avionics case study," in *the 8th IEEE International Symposium on Industrial Embedded Systems*, 2013.

[30] F. Wartel, L. Kosmidis, A. Gogonel, A. Baldovin, Z. R. Stephenson, B. Triquet, E. Quiñones, C. Lo, E. Mezzetti, I. Broster, J. Abella, L. Cucu-Grosjean, T. Vardanega, and F. J. Cazorla, "Timing analysis of an avionics case study on complex hardware/software platforms," in *the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE*, 2015, pp. 397–402.

[31] K. Berezovskyi, L. Santinelli, K. Bletsas, and E. Tovar, "WCET measurement-based and extreme value theory characterisation of CUDA kernels," in *22nd International Conference on Real-Time Networks and Systems*, 2014, p. 279.

[32] L. Santinelli, J. Morio, G. Dufour, and D. Jacquemart, "On the sustainability of the extreme value theory for WCET estimation," in *14th International Workshop on Worst-Case Execution Time Analysis*, 2014, pp. 21–30.