# Contract-Based Reusable Analysis for Software Components with Extra-Functional Properties

Johan Fredriksson*, Thomas Nolte
Dept. of Computer Science and Electronics
Mälardalen University, Västerås, Sweden

## Abstract

*Component-based software engineering (CBSE) for embedded systems is currently gaining ground because of shortened time-to-market, reduced development costs and increased software quality. One main characteristic of CBSE that enable these benefits is its facilitation of component reuse. However, existing tools and methods do not consider reuse of extra-functional properties in these systems.*

*In this paper we extend our previous work on contract-based reusable execution time predictions for software components with additional extra-functional properties, such as memory and energy consumption.*

## 1 Introduction

In this paper we present extensions to our previously developed method that allows reuse of software component WCET analysis [1, 2]. We extend the method with reusable analysis for additional extra-functional properties and also propose relationships between the properties.

In component-based software engineering (CBSE) reuse is a central concept for facilitating benefits such as shorter development times and higher software quality. There has been much focus on reuse of functional artifacts such as code, interfaces etc. For systems with requirements on predictability and resource efficiency extra-functional properties (EFPs) such as worst-case execution time and memory consumption are equally important for predicting the system behaviour. Still there has been little focus on reuse of EFPs.

Resource constraints and predictability requirements are especially common in many embedded-systems sectors, such as automotive, robotics and other types of computer controlled equipment. Because of the intrinsically non-linear behavior of software, it is often hard to make accurate predictions of extra functional properties (EFPs). The problem is worsened in component-based development where components are kept free of context to facilitate reuse. To make analysis more accurate, and thereby systems more predictable, it is desirable to have high accuracy of the predictions. This can be achieved by considering the context in which the software is used.

The contribution if this paper is the discussion and proposal of extensions to previously developed techniques for reusable WCET analysis. We extend the reusable analysis with additional extra-functional properties, such as memory and energy consumption. We also discuss future work and potential of the techniques.

## 2 Related work

Component contract with respect to extra-functional properties and their composition has been proposed in, e.g., [3]. In [4] a usage-scenario parameterized component contract is proposed with respect to reliability. Other approaches to contract-based preformance analysis are [5, 6, 7]; however, usage scenarios are not considered in these papers. In [8, 9, 10] parameterized WCET analysis has been proposed; however, neither software components nor reuse is considered.

## 3 Usage scenario

In the "real" physical world, distinct modes exist and are often engineered into systems, for example, as *modes of operation*. We hypothesize that modes are significant discriminators of WCET and can be utilized for more accurate WCET modeling. Thus we define a *usage scenario* as $\mathbf{U} = \langle X_0, ..., X_{n-1} \rangle$, where the $X_i (0 \leq i < n)$ are input variables, each with bounds on values, a given type, and a probability distribution $P_i : X_i \rightarrow [0, 1]$ for the occurrence of these values in the input. (See Figure 1 for an illustration of these concepts). The input domain $M$ is then defined as $M := X_0 \times \cdots \times X_{n-1}$. The probability distributions $P_i (0 \leq i < n)$ extend uniquely to a probability distribution $P : M \rightarrow [0, 1]$ on the input domain, defined by $P(x_0, \ldots, x_{n-1}) = P_0(x_0) \times \cdots \times P_{n-1}(x_{n-1})$.

*contact author: johan.fredriksson@mdh.se

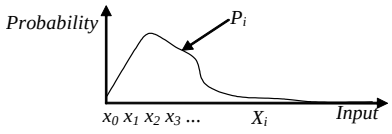


**Figure 1. Input variable I**

Furthermore we assume that $0 \leq pt < 1$ is a given probability threshold for ignoring low probability inputs (and consequently later their times). See also Figure 2 for an illustration of the concept.

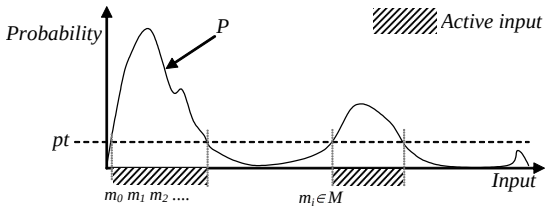


**Figure 2. Usage scenario**

## 4 Component WCET analysis

Components are reused in different products and different contexts. A different usage profile can substantially change the behavior of a component. To predict the execution time of a complex component with high accuracy, components must today be reanalyzed for every new usage profile – a very costly activity. Our method overcomes the problem by analyzing the execution times and their probability as a function of the input of the component.

We define an input domain $\mathbf{I}$ for a set of input variables $\{X_0, X_1, \ldots, X_{n-1}\}$ as $\mathbf{I} = X_0 \times X_1 \times \cdots \times X_{n-1}$. Each element $q$ in $\mathbf{I}$ is associated with an execution time $ET(q) \in \mathbf{W}$, where all execution times of the component are represented in the set $\mathbf{W}$. The longest execution time $\max(\mathbf{W}) = \text{WCET}^{abs}$ is the absolute WCET. We want to find the WCET for a specific usage. Because $\mathbf{I}$ often is very large, we can not perform WCET analysis for every element in $\mathbf{I}$ (every possible usage), instead we perform static WCET analysis with annotations on the input parameters, and perform a number of systematic runs with different bounds on the input parameters. When WCET analysis is performed with restrictions on the input parameters, not *all* input elements are considered, but rather a set of clusters $\{\mathbf{D}_l | \mathbf{D}_l \subseteq \mathbf{I}\}$, such that $\mathbf{D}_0 \oplus \mathbf{D}_1 \oplus \cdots \oplus \mathbf{D}_{n-1} = \mathbf{I}$. Thus, a cluster is a subset of all possible inputs, and a WCET tool can produce a WCET considering only that subset of inputs. Each cluster $\mathbf{D}_l$ is analyzed and associated with two execution times $et_l^{max} = \max(ET(d))_{d \in \mathbf{D}_l}$ and $et_l^{min} = \min(ET(d))_{d \in \mathbf{D}_l}$. The time $et_l^{max}$ is the result of running the WCET tool with the inputs represented in $\mathbf{D}_l$ with respect to WCET. The time $et_l^{min}$ is the result of running the WCET tool with the inputs represented in $\mathbf{D}_l$ with respect to best-case execution time (BCET).

### 4.1 Clustering WCETs

Theoretically, each single input combination has only one fixed execution-time. The difference between $et_l^{max}$ and $et_l^{min}$ of a cluster $\mathbf{D}_l$ shows the greatest difference between two execution times within the cluster. This in turn is an indicator of how similar the execution times are in the cluster. The sum of the difference between $et_l^{max}$ and $et_l^{min}$ should be minimized to get high accuracy. On one extreme, to get as high accuracy as possible, each cluster contains one element; however, a good solution is a trade-off between acceptable difference between $et_l^{max}$ and $et_l^{min}$ and max number of clusters.

Finding accurate clusters is a blind search problem since execution time data is built up during the search for clusters. There are many different possible approaches to search for clusters, rangin from random search to evolutionary algorithms. In previous work we have shown how binary search can be used effectively by recursively dividing the input space into two clusters until required accuracy has been achieved. Consider a simple with two input variables $x$ and $y$, where $x$ can take the values $[0..9]$ and $y$ can take the values $[0..4]$. In this small example there are only 50 possible input combinations, would be trivial to make an exhaustive search to find all combinations that give the same execution time. In a larger example, this is not possible. We have chosen such a simple example to simplify the visualization of the method. Consider an example where the usage scenario defines $x = \{3..6\}$ and $y = \{3..4\}$, the WCET will never occur. For two inputs the input domain is 2-dimensional, the WCET is visible in an execution time matrix as shown in Figure 3.

### 4.2 Component contracts

Each cluster can be transformed into a predicate with respect to the usage scenario $\mathbf{U}$. The predicate tests if the inputs $\mathbf{U}$ of the usage scenario is "inside" the clusters, i.e., $\mathbf{U} \subseteq \mathbf{D}_l$. For all clusters $\mathbf{W} = \forall l\{et_l | (\mathbf{D}_l \subseteq \mathbf{U})\}$ that have at least one element in $\mathbf{U}$, the WCET for the component $c_i$ with the usage scenario $\mathbf{U}$ is the longest execution time associated with any cluster, i.e., $\max(\mathbf{W})$. Thus the component contract is a function $f(\mathbf{U}) \rightarrow$WCET, where $f(\mathbf{U}) : \max(\mathbf{W})$. The probability distribution of the usage scenario $\mathbf{U}$ is used for calculating the probability of the occurrence of the inputs in a cluster $\mathbf{D}_l$, i.e., each cluster is associated with a probability for the specific usage scenario $\mathbf{U}$. If the probability of a cluster is lower than the

| x \ y | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 20 | 140 | 140 | 140 | 140 |
| 1 | 140 | 60 | 140 | 140 | 140 |
| 2 | 140 | 140 | 60 | 140 | 140 |
| 3 | 130 | 170 | 170 | 1 | 1 |
| 4 | 130 | 170 | 170 | 1 | 1 |
| 5 | 130 | 130 | 130 | 1 | 1 |
| 6 | 130 | 130 | 130 | 1 | 1 |
| 7 | 130 | 130 | 130 | 1 | 1 |
| 8 | 130 | 130 | 130 | 1 | 1 |
| 9 | 130 | 130 | 130 | 1 | 1 |

**Figure 3. Matrix of the inputs {x,y} with corresponding execution times with respect to the example code shown in Figure ??. The dotted line shows a cluster $\mathbf{D}_l : \{x = [3..9] \wedge y = [3..4]\}$**

probability threshold $pt$ the cluster can be ignored, and that execution time is disregarded in the contract.

## 5 Work in progress

We extend the contracts with the properties memory and energy consumption. Several properties implies interesting design trade-offs. Adding contracts with respect to both WCET, energy and memory consumption give the system designer possibility to decide if a component fulfils all requirements and what trade-offs are necessary at an early design phase. Similarly the designer may be able to restrict the usage scenario in such a way that the component will fulfill its given requirements. This gives new possibilities for reusable analysis and to consider extra-functional properties (EFPs) at an early design phase.

Memory can be statically analyzed in the same way as WCET; however there are currently no tools for static analysis of energy consumption. Hence, dynamic analysis is required to be used; therefore the confidence of an energy property is lower. The energy property is interesting because there is often a relation between WCET and energy.

Hence, we add a relation between the properties WCET and energy; simplified we state that there is a linear relationship between WCET and power consumption. By lowering the frequency of the processor the power consumption is decreased, but the WCET is increased. Thus, if a certain usage scenario fulfills the WCET but does not fulfill the required energy consumption it may be possible to lower the cpu-frequency such that both requirements are fulfilled. We do not consider memory consumption related to WCET or energy at this point.

Both properties memory and energy consumption are divided into clusters, similar to the WCET cluster as shown in Figure 3. We do not attempt to find joint clusters between the properties. Consider a usage $\mathbf{U}_1$ and three contracts $f_{WCET}$, $f_{Energy}$ and $f_{Memory}$; and a relationship $R^{WCET}_{Energy}$ for a specific processor.

$$f_{WCET}(\mathbf{U}_1) \rightarrow WCET$$

$$f_{Energy}(\mathbf{U}_1) \rightarrow Energy$$

$$f_{Memory}(\mathbf{U}_1) \rightarrow Memory$$

We extend the contracts $f_{WCET}$ and $f_{Energy}$ with *cpu frequency* ($frq$). We define $frq^0$ which is the cpu-frequency used in the reusable analysis and $frq^c$ which is the context frequency. Thus $frq$ is the ratio $\frac{frq^c}{frq^0}$. WCET and Energy are dependent on usage only, and the effect of the frequency is determined by the relation $R^frq$. Thus we define the contracts and the relationship as:

$$f_{WCET}(\mathbf{U}_1, frq) \rightarrow WCET$$

$$f_{Energy}(\mathbf{U}_1, frq) \rightarrow Energy$$

$$R^{frq} : Energy \propto WCET^{-1}$$

where the relation $R^{frq} : WCET \propto Energy^{-1}$ is an inverse linear relationship between WCET and Energy with respect to frequency frq. Thus, considering the usage-parameterized WCET and Energy, if the frequency is increased, the Energy is increased and the WCET is decreased.

Lets assume two requirements $WCET \leq 200$ and $Energy \leq 200$.

$$f_{WCET}(\mathbf{U}_1, 1.0) \rightarrow 130$$

$$f_{Energy}(\mathbf{U}_1, 1.0) \rightarrow 250$$

$$R^{frq} : 130 \propto 250^{-1}$$

WCET$= 130$ and Energy$= 250$ does not fulfill the requirements. Lets assume $frq^0 = 500$. We lower the frequency $frq^c = 400$, thus $frq = \frac{400}{500} = 0.8$.

$$f_{WCET}(\mathbf{U}_1, 0.8) \rightarrow 162.5$$

$$f_{Energy}(\mathbf{U}_1, 0.8) \rightarrow 200$$

$$R^{frq} : 163 \propto 200^{-1}$$

fulfills the stipulated requirements, $162.5 \leq 200$ and $200 \leq 200$. The relation $frq$ should be constant for the same usage. Thus $130 \propto 250^{-1} = 163 \propto 200^{-1}$

Additional hardware parameters can be added for better predicting the actual properties.

## 6 Future Work

Additional interesting properties that can be related to input are, e.g., *best case execution time* and *required resources*. To put several properties in a joint framework they require tool based property analysis with respect to usage. Today, there exist several WCET tools that support WCET analysis with respect to usage. Examples of such tools are aiT [11], RapiTime [12], Bound-t [13] and SWEET [14].

We plan on investigating and comparing different search algorithms for creating clusters and contracts. Previously we have used genetic algorithms in a simulation and a binary search method for a small case study. Further studies of both larger components and other search methods are planned.

Interesting future includes adding hardware context information to the contracts. To get a more accurate reusable prediction on WCET information such as compile and linker configurations as well as hardware properties need to be considered. However, the problem size grows exponentially with the number of parameters.

Other interesting work is combining the reusable analysis method with our previously developed component to task allocation algorithm [15]. Combining these two methods give possibilities for usage dependent real-time schedulability analysis. Other interesting future work includes evolutionary property prediction as discussed in [16].

## References


[1] Fredriksson, J., Nolte, T., Nolin, M., Schmidt, H.: Contract-based reusable worst-case execution time estimate. Technical report, Mälardalen Real-Time Centre (2007) (submitted to RTCSA'07).

[2] Fredriksson, J., Nolte, T., Ermedahl, A., Nolin, M.: Clustering worst-case execution times for software components. Technical report, Department of Computer Science and Electronics, Mälardalen University (2007) submitted to WCET'07.

[3] Beugnard, A., Jézéquel, J.M., Plouzeau, N., Watkins, D.: Making components contract aware. Computer **32** (1999) 38–45

[4] Reussner, R., Schmidt, H., Poernomo, I.: Reliability prediction for component-based software architectures. Journal of Systems and Software **66** (2003) 241–252

[5] Firus, V., Becker, S., Happe, J.: Parametric performance contracts for qml-specified software components. In: Formal Foundations of Embedded Software and Component-based Software Architectures (FESCA). Electronic Notes in Theoretical Computer Science, ETAPS 2005 (2005)

[6] Reussner, R.H., Firus, V., Becker, S.: Parametric performance contracts for software components and their compositionality. In Weck, W., Bosch, J., Szyperski, C., eds.: Proceedings of the 9. International Workshop on Component-Oriented Programming (WCOP 04). (2004)

[7] Happe, Jens; Koziolek, H.R.R.: Parametric performance contracts for software components with concurrent behaviour. In: 3rd Workshop on Formal Aspects of Component Software (FACS). Volume 167., Elsevier (2006) 15

[8] Lee, J.I., Park, S.H., Bang, H.J., Kim, T.H., Cha, S.D.: A hybrid framework of worst-case execution time analysis for real-time embedded system software. In: Aerospace, 2005 IEEE Conference, ieee (2005) 1–10

[9] Ji, M.L., Wang, J., Li, S., Qi, Z.C.: Automated wcet analysis based on program modes. In: AST'06, Shanghai, China, ACM (2006)

[10] David, L., Puaut, I.: Static determination of probabilistic execution times. In: ECRTS '04: Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS'04), Washington, DC, USA, IEEE Computer Society (2004) 223–230

[11] aiT: (ait execution time analyzer) Absint: http://www.absint.com/ait/.

[12] RapiTime: (Rapitime execution time analyzer) Rapita Systems: http://www.rapitasystems.com/.

[13] Bound-t: (Bound-t execution time analyzer) Tidorum Ltd: http://www.tidorum.fi/bound-t/.

[14] SWEET: (Swedish execution time tool) SWEET: http://www.mrtc.mdh.se/projects/wcet/.

[15] Fredriksson, J., Sandström, K., kerholm, M.A.: Optimizing Resource Usage in Component-Based Real-Time Systems. In: Proceedings of th 8th International Symposium on Component-Based Software Engineering (CBSE8). (2005)

[16] Fredriksson, J., Land, R.: Evolutionary context aware development of components for embedded real-time systems. In: To appear in Proceedings of the Internation Conference Information Technology Interfaces ITI'07. (2007)