Methods for Multi-Dimensional Robustness Optimization in Complex Embedded Systems

Arne Hamann, Razvan Racu, Rolf Ernst
Institute of Computer and Communication Network Engineering
Technical University of Braunschweig, Germany
{hamann|racu|ernst}@ida.ing.tu-bs.de

ABSTRACT

Design space exploration of embedded systems typically focuses on classical design goals such as cost, timing, buffer sizes, and power consumption. Robustness criteria, i.e. sensitivity of the system to variations of properties like execution and transmission delays, input data rates, CPU clock rates, etc., has found less attention despite its practical relevance.

In this paper we introduce multi-dimensional robustness metrics, expressing the static and dynamic design robustness of a given system, the former assuming a fixed parameter configuration, and the latter including parameter adaptations as response to property variations. Additionally, we propose a metric measuring the robustness gain that can be achieved through system reconfigurability.

Since determining multi-dimensional robustness is computationally expensive we introduce efficient exploration methods based on a stochastic sensitivity analysis technique capable of deriving upper and lower robustness bounds for a given system with low computational effort. We demonstrate the robustness optimization methods by means of a small but realistic case study.

Categories and Subject Descriptors

C.3 [**Special-Purpose and application-based systems**]: Real-time and embedded systems; C.4 [**Performance of systems**]: Modeling techniques; Performance attributes; Reliability, availability, and serviceability

General Terms

Algorithms, Design, Performance, Reliability, Verification

1. INTRODUCTION

In the embedded systems design flow, design robustness to property variations, such as execution and transmission delays, input data rates, CPU clock rates, etc., is playing an increasingly important role.

Generally, system robustness is desirable to account for estimation errors in early design phases, minor changes of specifications, bug fixes or later extensions and updates of the design to name just a few of the many situations where tolerance of hardware and run time system to modifications is expected. For instance, it is known that small task core execution time modifications in systems with complex performance dependencies can have drastic non-intuitive effects on the overall system behavior, and might lead to severe performance degradation effects [10].

In the current state of practice, designers reserve some slack for critical system parameters to ensure system robustness. A prominent example is the bus load model, where designers limit the average bus utilization to ensure system functionality and extensibility [11].

While such design guidelines used to work reasonably well in practice to ensure design robustness, they are gradually running out of steam. The main reason for this is the growing size and complex networked nature of modern embedded systems, making it extremely difficult to predict the effects of modifications on load and timing. The complexity of the problem is additionally increased by the large number of independently developed applications that are integrated on the same system leading to unknown coupling effects or limitations. Examples are cars or aircrafts. Results are an increasing design risk and non-extendable systems.

In this paper we address the application of formal models to robustness optimization. We first formulate the problem we address in this paper (Section 2) and give a brief survey of related work (Section 3). Afterwards, we introduce two formal robustness metrics for different design scenarios (Section 5), which are based on sensitivity analysis (Section 4), and propose efficient exploration methods considering them during design space exploration (Section 6). The static design robustness expresses the robustness of a given system with respect to property variation of a set of critical system properties for a fixed parameter configuration. The dynamic design robustness additionally includes counter actions (i.e. system parameter adaptations) as response to property variations. Based on the static and dynamic design robustness metrics we introduce a metric measuring the robustness gain that can be achieved through system reconfigurability. Finally, we demonstrate the proposed metrics and techniques by means of a small but realistic case study (Section 7).

2. PROBLEM STATEMENT

There are many notions of robustness in the context of embedded system design. Frequently, robustness is associated with fault tolerance, i.e. techniques that ensure that the system functions correctly in the presence of faults. Examples are task re-execution or replication mechanisms [8].

However, in this paper we define robustness differently. We call systems robust, if they can sustain property changes (e.g. WCETs, periods, CPU clock rates, etc.) without severe degradation of system functioning and performance. Consequently, the techniques presented in this paper are meant to support the designer in conceiving systems that are robust with respect to property variations.

Basically, we distinguish two different kinds of system property variations: variations influencing the system load, and variations influencing the system service capacity.

EMSOFT'07, September 30–October 3, 2007, Salzburg, Austria.

Reasons for system load variations are mainly changes of software execution path lengths, communication volumes, and input data rates. Scenarios under which such load variations can occur during design time or even in the field include late feature requests, product variants, software updates, and bug-fixes.

System service capacity variations are caused by modifications of the execution platform, e.g. processor or communication link performance changes. Such variations rarely occur in the field. However, they they are of particular interest during early design space exploration, where load requirements are still subject to changes, and where different alternative system components and architectures need to be evaluated.

Small but practical industrial case studies [14] have proven that the influences of unpredicted property variations on the system performance is an important issue during the design of complex embedded systems. In the automotive industry, for instance, the global system is decomposed into several components, which are independently developed by multiple suppliers based on requirement definitions. Once design is finished, the OEM integrates all components, which is mainly a network integration problem, i.e. the OEM must decide about bus topology, throughput, number of nodes, as well as assignment of priorities and time slots.

This massively parallel design style leads to a variety of severe problems. For instance, many system parameters, such as CAN message IDs (priorities), are fixed quite early in the design flow, where execution delays, periods, and transmitted data sizes are only rough estimations and still subject to changes.

In order to account for such uncertainties, it is essential that robustness criteria are taken into account as early as possible in the design flow, when system parameters can still be changed to optimize the design. By this means critical bottlenecks can be prevented, which reduces the overall design risk.

Therefore, we present in this paper methods helping the designer finding *system parameter configurations*, including the assignment of free system parameters like scheduling (priorities, time slots, etc.), leading to systems with high robustness (low sensitivity) to property variations. In other words, the system shall meet its timing constraints even under considerable changes of system properties, including worst-case execution and communication times, CPU clock rates, input data rates, etc. Note that we focus in this paper on hard real-time constraints.

3. RELATED WORK

One approach to achieve design robustness is parameter adaptation at run-time, such as in adaptive scheduling strategies [9]. Adaptation is very efficient but comes with run-time overhead and makes it more difficult to determine the resulting design robustness. Consequently, adaptation potentially improves the system but does not circumvent the need to determine system robustness.

If we want to determine robustness in a systematic way we need sensitivity analysis. Sensitivity analysis determines the boundaries between working and non-working systems for arbitrary properties. The one-dimensional sensitivity analyses approaches presented in [13, 15], for instance, can capture the effects of single system property variations taking into account complex global timing effects. They are capable of calculating the maximum (or minimum) permissible values for large variety of system properties, including worst-case execution times, communication volumes, input data rates, processor, communication link performance, etc.

Once we have a sensitivity analysis available, we can take the next step to include robustness as a regular design goal. For that purpose we need formal robustness metrics that allow to quantify robustness that are then used in design space exploration. The approach presented in [5] defines robustness metrics using one-dimensional sensitivity analysis techniques. However, since the underlying sensitivity analysis is one-dimensional these robustness metrics ignore dependencies between the considered system properties. For instance a small variation of one system property might have drastic effects on the robustness potential of another system property.

There are approaches that consider multi-dimensional robustness optimization problems [4, 1]. However, these approaches are based on very simple application models, which on the one hand facilitates the determination of the needed multi-dimensional robustness data, but on the other hand also relativizes the expressiveness of the obtained results.

In order to be significant, robustness optimization results must be based on state-of-the-art performance models [7, 2] that are able to tightly determine system performance properties such as timing, buffer sizes, and power consumption. Unfortunately, exact multi-dimensional sensitivity analysis approaches [12], which are applicable to such performance models, cannot be adapted in a straightforward manner to robustness optimization. The main reason for this is the high computational complexity of these methods, particularly for sensitivity analyses with dimensions greater than two, prohibiting the evaluation of a large number of system configurations for optimization.

In this paper we formally define expressive multi-dimensional robustness metrics for different design scenarios. We show how stochastic sensitivity analysis methods based on state-of-the-art performance models can be utilized to efficiently bound the multi-dimensional sensitivity properties of a given system with relatively little computational effort compared to exact approaches.

4. SENSITIVITY ANALYSIS

The robustness optimization methods presented in this paper are based on sensitivity analysis techniques.

Sensitivity analyses [13, 15] can capture the global effects of system property variations. They are capable of calculating the boundaries representing the transition between working and non-working systems with respect to a large variety of system properties, including worst-case execution times, communication volumes, input data rates, processor and communication link performance, etc.

Note that the techniques described in the remainder of this paper are applicable to system properties subject to maximization (e.g. worst-case execution times, processor and communication link performance, etc.). However, all definitions and algorithms can be easily adapted to also cover system properties subject to minimization (e.g. input data rates, etc.).

Modern sensitivity analysis approaches [13, 12] are independent of the underlying application and architectural model. They can thus be applied to arbitrary formal analysis engines. For the remainder of this paper we assume that we dispose of a system evaluation function (Definition 1) capable of checking whether or not a given system with specific system property values is working, i.e. the system fulfills all constraints (maximum end-to-end latencies, maximum jitters, maximum buffer sizes, etc.) and that no resource is overloaded.

Definition 1 (System Evaluation Function)
Let $\mathcal{S}_c$ denote the system $\mathcal{S}$ with parameter configuration c and $\mathcal{P} = \{p_1, \ldots, p_n\}$ a set of system properties. The system evaluation function $f_{\mathcal{S}_c}$ checks whether or not $\mathcal{S}_c$ is working *if the property values $\vec{v} = (v_1, \ldots, v_n)$ are applied to the properties $\mathcal{P}$.*

$$f_{\mathcal{S}_c}(\mathcal{P}, \vec{v}) = \begin{cases} \textit{true, if } \mathcal{S}_c \textit{ is working with values } \vec{v} \textit{ for the} \\ \qquad \textit{properties } \mathcal{P} \\ \textit{false, otherwise.} \end{cases}$$

Given this system evaluation function we can, for instance, use the approach in [13] to calculate the extreme values for arbitrary

system properties, which still lead to a working system. Calculating the extreme value for a single system property is called *one-dimensional sensitivity analysis*, and can be formalized as follows:

Definition 2 (Original and extreme property values)
Let $\mathcal{S}_c$ denote the system $\mathcal{S}$ with parameter configuration c. For a given system property p the original property value is denoted as $v(p)$. The extreme property value for p is denoted as $v^+_{\mathcal{S}_c}(p)$ and defined as follows:

$$v^+_{\mathcal{S}_c}(p) = \max\{x \in \mathbb{R}_+ \mid f_{\mathcal{S}_c}(p,x)\}$$

If we want to extend sensitivity analysis to the multi-dimensional case, i.e. taking into account dependencies between multiple system properties, we can use the results of one-dimensional sensitivity analysis to bound the space containing the sought-after front between working and non-working system property value combinations. We call this multi-dimensional bound the *bounding hypercube*.

Definition 3 (Bounding Hypercube)
Let $\mathcal{S}_c$ denote the system $\mathcal{S}$ with parameter configuration c. For a given set of system properties $\mathcal{P} = \{p_1, \ldots, p_n\}$ the n-dimensional bounding hypercube $\mathcal{H}_{\mathcal{S}_c}(\mathcal{P}) \subset \mathbb{R}^n_+$ is defined as follows:

$$\mathcal{H}_{\mathcal{S}_c}(\mathcal{P}) = \left[\min\left(v(p_1), v^+_{\mathcal{S}_c}(p_1)\right), \max\left(v(p_1), v^+_{\mathcal{S}_c}(p_1)\right)\right] \times \ldots \times \left[\min\left(v(p_n), v^+_{\mathcal{S}_c}(p_n)\right), \max\left(v(p_n), v^+_{\mathcal{S}_c}(p_n)\right)\right]$$

The exact boundary between working and non-working system property combination is called *sensitivity front*. It can be formally characterized using the notion of Pareto-optimality (Definition 4).

Definition 4 (Pareto-optimality)
Given a set V of n-dimensional vectors in $\mathbb{R}^n$, the vector $\vec{v} = (v_1, \ldots, v_n) \in V$ dominates the vector $\vec{w} = (w_1, \ldots, w_n) \in V$ iff for all elements $1 \leq i \leq n$ we have

1. *minimization problem: $v_i \leq w_i$ and for at least one element l we have $v_l < w_l$.*
2. *maximization problem: $v_i \geq w_i$ and for at least one element l we have $v_l > w_l$.*

A vector is called Pareto-optimal *iff it is not dominated by any other vector in V.*

Notations:

- $\vec{v} \triangleright_+ \vec{w}$ ($\vec{v} \triangleright_- \vec{w}$) denotes $\vec{v}$ Pareto-dominates $\vec{w}$ in the sense of a maximization (minimization) problem.
- $\Omega^+(V)$ ($\Omega^-(V)$) denotes the set of vectors in V which are Pareto-optimal in the sense of a maximization (minimization) problem.

The formal characterization of the sensitivity front is given in Definition 5, which generalizes Definition 2.

Definition 5 (Sensitivity Front)
Let $\mathcal{S}_c$ denote the system $\mathcal{S}$ with parameter configuration c. For a given set of system properties subject to maximization $\mathcal{P} = \{p_1, \ldots, p_n\}$ the sensitivity front is defined as the set of all working property value combinations which are not Pareto-dominated by any other working property value combination:

$$\mathcal{F}^{sens}_{\mathcal{S}_c}(\mathcal{P}) = \{\vec{x} \in \mathcal{H}_{\mathcal{S}_c}(\mathcal{P}) \mid f_{\mathcal{S}_c}(\mathcal{P},\vec{x}) \wedge \nexists \vec{y} \in \mathcal{H}_{\mathcal{S}_c}(\mathcal{P}) : (f_{\mathcal{S}_c}(\mathcal{P},\vec{y}) \wedge \vec{y} \triangleright_+ \vec{x})\}$$

The sensitivity front separates the space of working and non-working system property combinations. The robustness metrics presented in Section 5 are based on the sensitivity information it represents.

5. MULTI-DIMENSIONAL ROBUSTNESS METRICS

In this section we introduce multi-dimensional robustness metrics. We therefore first discuss the notion of hypervolume (Section 5.1), on which the proposed metrics are based. Afterwards, we define static and dynamic design robustness metrics, the former assuming a static system parameter configuration, and the latter including dynamic parameter adaptations as response to property variations (Sections 5.2 and 5.3). We explain several application scenarios ranging from platform optimization to critical component identification.

5.1 Hypervolume calculation

In this section we shortly discuss hypervolume calculation for an arbitrary number of dimensions. We first introduce the notion of absolute hypervolume (Section 5.1.1), which is for instance used as a metric in evolutionary optimization. Based on the absolute hypervolume we then define the weighted percentage hypervolume (Section 5.1.2), which we use in this paper to define expressive multi-dimensional robustness metrics.

5.1.1 Absolute hypervolume

According to Definition 6 we distinguish two different types of hypervolumes: the inner and the outer hypervolume.

Definition 6 (Absolute Inner and Outer Hypervolumes)
We consider a n-dimensional hypercube $\mathcal{H} = \left[\underline{b}_1, \overline{b}_1\right] \times \ldots \times \left[\underline{b}_n, \overline{b}_n\right] \subset \mathbb{R}^n_+$ and a set of n-dimensional vectors $\mathcal{V} = \{\vec{v}_1, \ldots, \vec{v}_m\}$ with $\forall_i \vec{v}_i \in \mathcal{H}$.

1. *The inner hypervolume of $\mathcal{V}$ in $\mathcal{H}$ is denoted as $\lambda^-_{\mathcal{H}}(\mathcal{V})$ and is defined as the volume of the space in $\mathcal{H}$ containing all vectors $\vec{w}$ which are Pareto-dominated by at least one vector $\vec{v} \in \mathcal{V}$:*

$$\lambda^-_{\mathcal{H}}(\mathcal{V}) = \text{vol}\left\{\vec{w} \in \mathcal{H} \mid \exists \vec{v} \in \mathcal{V} : \vec{v} \triangleright_+ \vec{w}\right\}$$

2. *The outer hypervolume of $\mathcal{V}$ in $\mathcal{H}$ is denoted as $\lambda^+_{\mathcal{H}}(\mathcal{V})$ and is defined as the difference between the volume of the hypercube $\mathcal{H}$ and the volume of the space in $\mathcal{H}$ containing all vectors w Pareto-dominating at least one vector $v \in \mathcal{V}$:*

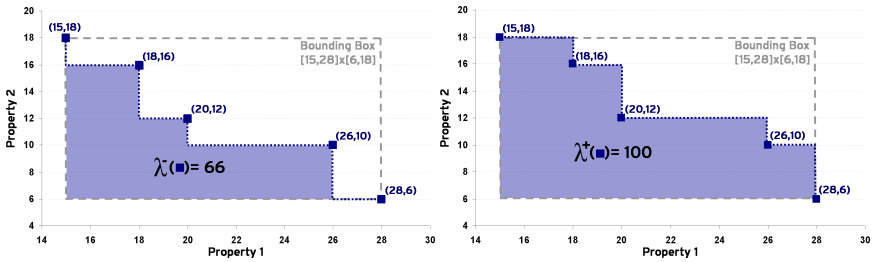
$$\lambda^+_{\mathcal{H}}(\mathcal{V}) = \text{vol}(\mathcal{H}) - \text{vol}\left\{\vec{w} \in \mathcal{H} \mid \exists \vec{v} \in \mathcal{V} : \vec{w} \triangleright_+ \vec{v}\right\}$$

Figures 1a and 1b visualize the difference between the inner and the outer hypervolumes in the two dimensional case: the inner hypervolume corresponds to the space covered by the lower step function, whereas the upper hypervolume corresponds to the space covered by the upper step function.

The inner hypervolume is usually used as a measure to compare efficiency and to ensure diversity in evolutionary multi-objective algorithms [18]. Especially for the second point it is required that the hypervolume can be calculated efficiently. Therefore, several efficient algorithms for hypervolume calculation were proposed in the last years [3, 16, 17].

Given the algorithm for calculating the *inner hypervolume* $\lambda^-(\mathcal{V})$, the *outer hypervolume* $\lambda^+(\mathcal{V})$ can be calculated according to Algorithm 1.

First, the hypercube bounding $\mathcal{V}$ is calculated (lines $1-6$). Afterwards, the origin of the vectors in $\mathcal{V}$ is translated to the extreme point of the bounding hypercube (lines $7-9$). Note that the inner



(a) Absolute inner hypervolume (b) Absolute outer hypervolume

1: Absolute inner and outer hypervolumes

Algorithm 1 $\lambda^+(\mathcal{V})$

Require: Set of n dimensional Pareto-optimal vectors $\mathcal{V}$
Ensure: Outer hypervolume λ^+ of $\mathcal{V}$
1: **for all** i such that $1 \leq i \leq n$ **do**
2: $min[i] = \infty$
3: $max[i] = -\infty$
4: **for all** $v \in \mathcal{V}$ **do**
5: $min[i] = \min(min[i], v[i])$
6: $max[i] = \max(max[i], v[i])$
7: **for all** $v \in \mathcal{V}$ **do**
8: **for all** i such that $1 \leq i \leq n$ **do**
9: $v[i] = max[i] - v[i]$
10: $\lambda^+_{tmp} = 1$
11: **for all** i such that $1 \leq i \leq n$ **do**
12: $\lambda^+_{tmp} = \lambda^+_{tmp} \times (max[i] - min[i])$
13: $\lambda^+(\mathcal{V}) = \lambda^+_{tmp} - \lambda^-(\mathcal{V})$

hypervolume of the translated vector set corresponds to the space containing all vectors dominating at least one vector of the initial set $\mathcal{V}$. Finally, $\lambda^+(\mathcal{V})$ is calculated by subtracting the inner hypervolume of the translated vector set from the hypervolume of the bounding hypercube (lines $10-13$).

5.1.2 Weighted percentage hypervolume

Based on the absolute hypervolume we define the so-called *weighted percentage hypervolume*.

For the percentage hypervolume the coordinates of the considered vectors are translated to the percentage increase with respect to the corresponding minimum values of the bounding hypercube. This is necessary for obtaining expressive and comparable results in case that the coordinate values in some dimensions exhibit significant differences in terms of absolute values (i.e. different orders of magnitude).

Additionally, the weighted percentage hypervolume allows to attach importance levels to each dimension, i.e. the space covered in one dimension might be considered more important than that covered in other dimensions.

Definition 7 (Weighted Percentage Hypervolumes)
We consider a n-dimensional hypercube $\mathcal{H} = [\underline{b}_1, \overline{b}_1] \times \ldots \times [\underline{b}_n, \overline{b}_n] \subset \mathbb{R}^n_+$ *and a set of n-dimensional vectors* $\mathcal{V} = \{\vec{v}_1, \ldots, \vec{v}_m\}$ *with* $\forall_i \vec{v}_i = (v_{i1}, \ldots, v_{in}) \in \mathcal{H}$. *Given a set of weights* $\mathcal{W} = \{w_1, \ldots, w_n\}$ *with* $\forall_i w_i \geq 1$ *the inner and outer weighted percentage hypervolumes of* $\mathcal{V}$ *are defined as follows:*

$$\tilde{\lambda}^-_{\mathcal{H}}(\mathcal{V}, \mathcal{W}) = \lambda^-_{\tilde{\mathcal{H}}_{\mathcal{W}}}(\tilde{\mathcal{V}}_{\mathcal{W}}) \quad \textit{and} \quad \tilde{\lambda}^+_{\mathcal{H}}(\mathcal{V}, \mathcal{W}) = \lambda^+_{\tilde{\mathcal{H}}_{\mathcal{W}}}(\tilde{\mathcal{V}}_{\mathcal{W}})$$

Thereby:

- $\tilde{\mathcal{H}}_{\mathcal{W}} = [0, f_1(\overline{b}_1)] \times \ldots \times [0, f_n(\overline{b}_n)]$, *and*
- $\tilde{\mathcal{V}}_{\mathcal{W}} = \{\vec{v}^*_1, \ldots, \vec{v}^*_m\}$, *with* $\vec{v}^*_i = (f_1(v_{i1}), \ldots, f_n(v_{in}))$,
- *where* $f_i(x) = \frac{x - \underline{b}_i}{\underline{b}_i} \times \left(\frac{x - \underline{b}_i}{\underline{b}_i} \times \frac{w_i - 1}{10} + 1 \right)$

Figure 2a visualizes the absolute inner hypervolumes for two example vectors sets. The corresponding inner percentage hypervolume without weighting is shown in Figure 2b. As we can observe, both vector sets cover exactly the same amount of space.

However, the vector set represented by the squares covers more space in x-dimension, whereas the vector set represented by the diamonds covers more space in y-dimension. This is reflected by the metric if we apply weighting. Figure 2c visualizes the percentage hypervolumes with weight 3 for the x-dimension. As expected we observe that the vector set represented by the squares has a higher weighted percentage hypervolume compared to the vector set represented by the diamonds. The other way around, if we assign the weight 3 to the y-dimension we observe, as shown in Figure 2d, that the vector set represented by the diamonds results in a higher weighted percentage hypervolume.

5.2 Multi-dimensional static design robustness

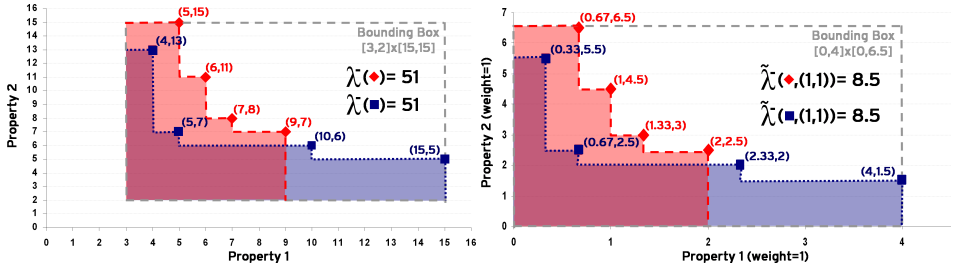
The *static design robustness (SDR)* metric expresses the robustness of a fixed parameter configuration for a given constrained system with respect to a set of critical system properties. The SDR metric is relevant for the design scenario where parameters are defined and fixed early at design time and cannot be modified later to reach compatibility for variants, bug-fixes, and updates.

Definition 8 (Multi-dim. static design robustness)
Let $\mathcal{S}_c$ *denote the system* $\mathcal{S}$ *with parameter configuration* c *and* $\mathcal{P} = \{p_1, \ldots, p_n\}$ *a set of system properties. Given a set of weights* $\mathcal{W} = \{w_1, \ldots, w_n\}$ *with* $\forall_i w_i \geq 1$, *the multi-dimensional static design robustness of* $\mathcal{S}_c$ *with respect to* $\mathcal{P}$ *is defined as follows:*

$$\text{SDR}_{\mathcal{S}_c}(\mathcal{P}, \mathcal{W}) = \tilde{\lambda}^-_{\mathcal{H}_{\mathcal{S}_c}(\mathcal{P})}(\mathcal{F}^{sens}_{\mathcal{S}_c}(\mathcal{P}), \mathcal{W})$$

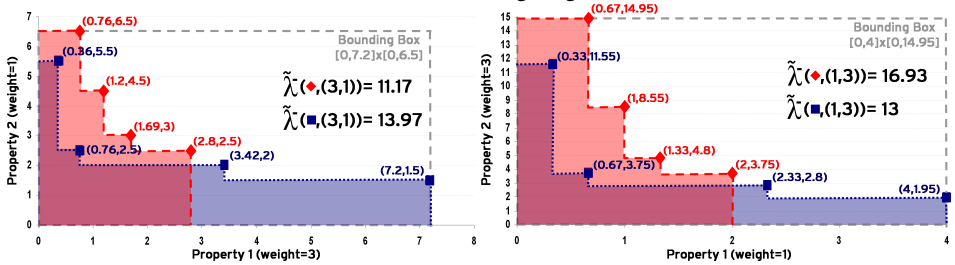
Since it cannot be known beforehand to what extent the included system properties are subject to change, it is desirable that a large number of different possibilities can be sustained by the system. Exactly this is expressed by the SDR metric. The more the system can sustain property value variations, the higher the hypervolume covered by the sensitivity front, and thus the higher the value of the SDR metric.

Note that the SDR metric allows to weight the influence of the included system properties by attaching weights to each dimension. This enables the designer to account for different levels of relevance linked to each of the considered system properties. Criteria for her to assign these weights include the estimated probability of future changes and the impact on the overall system performance.



(a) Absolute inner hypervolumes for two vector sets

(b) Percentage inner hypervolumes without weighting



(c) Percentage inner hypervolumes with weight 3 for x-dimension and no weighting for y-dimension

(d) Percentage inner hypervolumes with no weighting for x-dimension and weight 3 for y-dimension

2: Examples for weighted percentage hypervolume

5.3 Multi-dimensional dynamic design robustness

While static design robustness assumes a static system with fixed parameter configuration, *dynamic design robustness (DDR)* includes potential designer or system counteractions in reaction to system property variations. In other words, DDR describes the robustness potential of a system with respect to the variation of given properties, which can be achieved by system reconfiguration, i.e. for instance scheduling parameter adaptation. Consequently, the DDR metric is relevant for a design scenario where parameters can be modified during product life time or in the field.

Definition 9 (Multi-dim. dynamic design robustness)
We consider a system $\mathcal{S}$ and a set of system properties $\mathcal{P} = \{p_1, \ldots, p_n\}$. Given a set of possible parameter configurations $\mathcal{C}$ for $\mathcal{S}$ and a set of weights $\mathcal{W} = \{w_1, \ldots, w_n\}$ with $\forall_i w_i \geq 1$, the multi-dimensional dynamic design robustness of S with respect to $\mathcal{P}$ is defined as follows:

$$\mathrm{DDR}_{\mathcal{S},\mathcal{C}}(\mathcal{P},\mathcal{W}) = \tilde{\lambda}^{-}_{\mathcal{H}}\Big(\bigcup_{c\in\mathcal{C}} \mathcal{F}^{sens}_{\mathcal{S}_c}(\mathcal{P}),\mathcal{W}\Big) \text{ , where}$$

$$\mathcal{H} = \bigcup_{c\in\mathcal{C}} \mathcal{H}_{\mathcal{S}_c}(\mathcal{P})$$

The dynamic design robustness depends on the set of possible configurations $\mathcal{C}$. As an example, it can be possible to react to property changes by adaptation of scheduling parameters or by remapping parts of the application.

DDR can obviously be used to evaluate dynamic systems, but it can more generally be used for the evaluation of the design risk connected to specific components in a given system. More precisely, already early in the design flow the DDR metric allows the designer to determine boundaries for properties of specific components, allowing their integration into the system. This information effectively facilitates feasibility and requirements analysis and greatly assists the designer in pointing out critical system components requiring special focus during specification and implementation.

Another usage scenario for the DDR metric concerns reconfigurable systems. In such a scenario the designer can use the DDR metric to determine the theoretical robustness head room of crucial system components with respect to future changes of their properties. By early choosing a system architecture offering high DDR values for these crucial components the designer can significantly increase system stability and maintainability.

5.4 Robustness gain through reconfiguration

Given the formal definition of the static design robustness (Definition 8) and the dynamic design robustness (Definition 9), we can formally derive a metric (Definition 10) expressing the robustness increase, which can be achieved through dynamic system reconfiguration compared to the static case, where all parameters remain fixed during the systems' lifetime.

Definition 10 (Dynamic Robustness Gain)
We consider a system $\mathcal{S}$ and a set of system properties $\mathcal{P} = \{p_1, \ldots, p_n\}$. Given a set of possible parameter configurations $\mathcal{C}$ for $\mathcal{S}$ and a set of weights $\mathcal{W} = \{w_1, \ldots, w_n\}$ with $\forall_i w_i \geq 1$, the dynamic robustness gain which can be achieved through dynamic system reconfigurability compared to the static case is defined as follows:

$$\mathcal{G}_{\mathcal{S},\mathcal{C}}(\mathcal{P},\mathcal{W}) = \mathrm{DDR}_{\mathcal{S},\mathcal{C}}(\mathcal{P},\mathcal{W}) - \max_{c\in\mathcal{C}}\{\mathrm{SDR}_{\mathcal{S}_c}(\mathcal{P},\mathcal{W})\}$$

Given this metric the benefit in terms of system robustness of designing reconfigurable system components is explicitely measurable. Consequently, it can help system architects to decide whether or not investing engineering effort into creating reconfiguration mechanisms is worthwhile.

More generally, by adjusting the reconfiguration space $\mathcal{C}$ the metric can be used to identify critical components for which reconfigurability is particularly advantageous, i.e. leading to a significant robustness gain. By this means, engineering effort can be efficiently focused to conceive robust systems.

6. EXPLORING MULTI-DIMENSIONAL ROBUSTNESS

The robustness metrics introduced in the previous section are defined on the sensitivity fronts of one or several system configurations. However, the exact calculation of a multi-dimensional sensitivity front is computationally expensive. In this section we therefore introduce exploration techniques for efficiently deriving upper and lower robustness bounds for the static (Section 6.2) and the dynamic case (Section 6.3). The proposed methods are based on stochastic multi-dimensional sensitivity analysis (Section 6.1).

6.1 Stochastic sensitivity analysis

Performing an exact multi-dimensional sensitivity analysis for dimensions greater than 2 is computationally expensive.

The authors of [6] therefore proposed a stochastic sensitivity analysis approach. This stochastic approach formulates multi-dimensional sensitivity analysis as optimization problem, and is capable of bounding the sought-after sensitivity front from two sides, i.e. coming from the space of non-working system property combination and coming from the space of working system property combinations.

These two bounding fronts are called *bounding working Pareto-front* and *bounding non-working Pareto-front* and are defined as follows:

Definition 11 (Bounding Pareto-fronts)
Let $\mathcal{S}_c$ denote the system $\mathcal{S}$ with parameter configuration c. For a given set of system properties $\mathcal{P} = \{p_1, \ldots, p_n\}$ the bounding working and non-working Pareto-fronts are defined as follows:

1. *The bounding working Pareto-front $\mathcal{F}^{\mathrm{w}}_{\mathcal{S}_c}(\mathcal{P})$ is defined as a set of vectors $\vec{f}^{w}_1, \ldots, \vec{f}^{w}_n$ with the following properties: (I.) $\forall_i \vec{f}^{w}_i \in \mathcal{H}_{\mathcal{S}_c}(\mathcal{P})$, (II.) $\Omega^{+}(\mathcal{F}^{\mathrm{w}}_{\mathcal{S}_c}(\mathcal{P})) = \mathcal{F}^{\mathrm{w}}_{\mathcal{S}_c}(\mathcal{P})$, (III.) $\forall_i \exists \vec{x} \in \mathcal{F}^{sens}_{\mathcal{S}_c}(\mathcal{P}) : \vec{x} \triangleright_{+} \vec{f}^{w}_i$*

2. *The bounding non-working Pareto-front $\mathcal{F}^{\mathrm{nw}}_{\mathcal{S}_c}(\mathcal{P})$ is defined as a set of vectors $\vec{f}^{nw}_1, \ldots, \vec{f}^{nw}_n$ with the following properties: (I.) $\forall_i \vec{f}^{nw}_i \in \mathcal{H}_{\mathcal{S}_c}(\mathcal{P})$, (II.) $\Omega^{-}(\mathcal{F}^{\mathrm{nw}}_{\mathcal{S}_c}(\mathcal{P})) = \mathcal{F}^{\mathrm{nw}}_{\mathcal{S}_c}(\mathcal{P})$, (III.) $\forall_i \exists \vec{x} \in \mathcal{F}^{sens}_{\mathcal{S}_c}(\mathcal{P}) : \vec{x} \triangleright_{-} \vec{f}^{nw}_i$*

During stochastic sensitivity analysis the bounding Pareto-fronts are constantly refined using evolutionary exploration techniques [19]. Thereby, it is always assured that the exact sensitivity front $\mathcal{F}^{sens}_{\mathcal{S}_c}(\mathcal{P})$ lies between the bounding working and the bounding non-working Pareto-fronts $\mathcal{F}^{\mathrm{w}}_{\mathcal{S}_c}(\mathcal{P})$ and $\mathcal{F}^{\mathrm{nw}}_{\mathcal{S}_c}(\mathcal{P})$. If the stochastic sensitivity analysis runs long enough the bounding Pareto-fronts ultimately converge to the sought-after sensitivity front. Details about the multi-dimensional sensitivity analysis and the search space bounding strategy can be found in [6].

The main reason for applying the stochastic multi-dimensional sensitivity analysis to system robustness optimization is its capability to quickly derive upper and lower system sensitivity bounds with relatively little computational effort. This property will be used in in the following sections to efficiently approximate the static and dynamic multi-dimensional robustness metrics proposed in Section 5.

6.2 Static case

In the static case the optimization task consists in finding the system parameter configuration for a given system $\mathcal{S}$ that maximizes the multi-dimensional static design robustness with respect to a set of system properties $\mathcal{P}$ weighted according to $\mathcal{W}$.

In order to optimize a system for static design robustness we propose a nested design space exploration approach. The outer exploration loop variates free system parameters (e.g. scheduling parameters), whereas the inner exploration loop evaluates the robustness of each candidate system parameter configuration c using the stochastic multi-dimension sensitivity analysis presented in Section 6.1.

Based on the calculated bounding working Pareto-front $\mathcal{F}^{\mathrm{w}}_{\mathcal{S}_c}(\mathcal{P})$ and bounding non-working Pareto-front $\mathcal{F}^{\mathrm{nw}}_{\mathcal{S}_c}(\mathcal{P})$ we can derive upper and lower bounds for the sought-after static robustness.

Definition 12 (Static Robustness Bounds)
Let $\mathcal{S}_c$ denote the system $\mathcal{S}$ with parameter configuration c and $\mathcal{P} = \{p_1, \ldots, p_n\}$ a set of system properties. Given the bounding working and non-working Pareto-fronts $\mathcal{F}^{\mathrm{w}}_{\mathcal{S}_c}(\mathcal{P})$ and $\mathcal{F}^{\mathrm{nw}}_{\mathcal{S}_c}(\mathcal{P})$ as well as the set of weights $\mathcal{W} = \{w_1, \ldots, w_n\}$, the static design robustness $\mathrm{SDR}_{\mathcal{S}_c}(\mathcal{P}, \mathcal{W})$ is bounded by the minimum guaranteed robustness $\mathcal{R}^{-}_{\mathcal{S}_c}$ and the maximum possible robustness $\mathcal{R}^{+}_{\mathcal{S}_c}$:

$$\mathcal{R}^{-}_{\mathcal{S}_c}(\mathcal{P}, \mathcal{W}) \leq \mathrm{SDR}_{\mathcal{S}_c}(\mathcal{P}, \mathcal{W}) < \mathcal{R}^{+}_{\mathcal{S}_c}(\mathcal{P}, \mathcal{W}), \text{ where}$$

$$\mathcal{R}^{-}_{\mathcal{S}_c}(\mathcal{P}, \mathcal{W}) = \tilde{\lambda}^{-}_{\mathcal{H}_{\mathcal{S}_c}(\mathcal{P})}(\mathcal{F}^{\mathrm{w}}_{\mathcal{S}_c}(\mathcal{P}), \mathcal{W})$$

$$\mathcal{R}^{+}_{\mathcal{S}_c}(\mathcal{P}, \mathcal{W}) = \tilde{\lambda}^{-}_{\mathcal{H}_{\mathcal{S}_c}(\mathcal{P})}(\mathcal{F}^{\mathrm{nw}}_{\mathcal{S}_c}(\mathcal{P}), \mathcal{W})$$

It is always guaranteed that the real robustness of the analyzed system configuration is contained in the interval defined by $\mathcal{R}^{-}$ and $\mathcal{R}^{+}$.

Consequently, the precision of the approach can be safely scaled. This represents a huge advantage of the presented approach since even for low approximation qualities, i.e. little invested computational effort for stochastic sensitivity analysis, the minimum guaranteed and the maximum possible robustness metrics are very good indicators for identifying interesting system configurations worth to be analyzed in detail.

In order to find all system configurations with high robustness properties, despite variations in the evaluation results of the stochastic multi-dimensional sensitivity analysis, we propose to perform a two-dimensional Pareto-optimization of the minimum guaranteed robustness $\mathcal{R}^{-}$ and the maximum possible robustness $\mathcal{R}^{+}$ in the outer exploration loop. In so doing, the multi-dimensional robustness optimization yields on the one hand system configurations with large guaranteed robustness and on the other hand system configurations with possibly large robustness potential, which needs to be confirmed or disapproved by further analysis.

It is obvious that using this approach it is sufficient to invest comparatively little computational effort in the robustness evaluation during exploration to identify interesting system configurations, and to postpone their detailed robustness analysis after the exploration process.

6.3 Dynamic case

In order to determine the dynamic robustness of a given system we have to integrate the robustness potential of several system configurations. In order to do so we propose an iterative exploration approach.

During exploration we maintain and update a *dynamic bounding working Pareto-front* and a *dynamic bounding non-working Pareto-front*. The dynamic Pareto-fronts integrate the information contained in the bounding Pareto-fronts of each evaluated system parameter configuration. They therefore dynamically bound the space of working and non-working system property combinations over all evaluated system configurations at any time during exploration. Note that each successive exploration iteration refines the approximation of the dynamic bounding Pareto-fronts.

For a given set of system properties $\mathcal{P}$ and a set of system parameter configurations $\mathcal{C}$ the multi-dimensional dynamic design robustness of the system $\mathcal{S}$ can be approximated using the following iterative design space exploration approach:

Initialization. The initialization phase consists of the following operations:

1. Initialization of the dynamic bounding hypercube
$$\tilde{\mathcal{H}}_{\mathcal{S}}(\mathcal{P}) = 0$$
2. Initialization of the dynamic bounding working and non-working Pareto-fronts
$$\tilde{\mathcal{F}}^{\mathrm{w}}_{\mathcal{S},\mathcal{C}} = \emptyset \text{ and } \tilde{\mathcal{F}}^{\mathrm{nw}}_{\mathcal{S},\mathcal{C}} = \emptyset$$
3. Initialization of the sets containing the bounding working and non-working Pareto-fronts of all so far evaluated system parameter configurations
$$\Gamma^{\mathrm{w}} = \emptyset \text{ and } \Gamma^{\mathrm{nw}} = \emptyset$$

Exploration iteration. During an exploration iteration each considered configuration c is evaluated using the stochastic multi-dimensional sensitivity analysis (see Section 6.1). The resulting bounding Pareto-fronts $\mathcal{F}^{\mathrm{w}}_{\mathcal{S}_c}(\mathcal{P})$ and $\mathcal{F}^{\mathrm{nw}}_{\mathcal{S}_c}(\mathcal{P})$ are utilized to calculate the fitness value of c, and are stored in the sets Γ^{w} and Γ^{nw}, respectively, to update the dynamic bounding Pareto-fronts at the end of the current exploration iteration.

One possibility to assign fitness values to the evaluated system configurations is to simply take their robustness potential, i.e. the hypervolumes covered by their bounding Pareto-fronts. However, the robustness of a system configuration does not represent a very good metric to control the exploration in the dynamic case. The reason for this is that a high individual design robustness does not automatically mean that the configuration contributes to the dynamic design robustness, since another configuration might already cover the same property space.

Therefore, we propose to use the improvement of the dynamic design robustness, which is achieved by a configuration, as its fitness value. More precisely, the fitness value of an evaluated system configuration c is defined as the weighted percentage hypervolume of the space covered by its bounding working Pareto-front that is not already covered by the dynamic bounding working Pareto-front that was calculated after the previous exploration iteration:

$$\begin{aligned}\text{fitness}(c) =& \tilde{\lambda}^{-}_{\mathcal{H}}(\Omega^{+}(\tilde{\mathcal{F}}^{\mathrm{w}}_{\mathcal{S},\mathcal{C}} \cup \mathcal{F}^{\mathrm{w}}_{\mathcal{S}_c}(\mathcal{P})), \mathcal{W}) \\ & - \tilde{\lambda}^{-}_{\tilde{\mathcal{H}}_{\mathcal{S}}(\mathcal{P})}(\tilde{\mathcal{F}}^{\mathrm{w}}_{\mathcal{S},\mathcal{C}}, \mathcal{W})\end{aligned}$$

$$\text{, where } \mathcal{H} = \tilde{\mathcal{H}}_{\mathcal{S}}(\mathcal{P}) \cup \mathcal{H}_{\mathcal{S}_c}(\mathcal{P})$$

Note that during our iterative exploration approach the dynamic bounding working Pareto-front $\tilde{\mathcal{F}}^{\mathrm{w}}_{\mathcal{S},\mathcal{C}}$, which represents the baseline of the fitness assignment strategy, is updated after each iteration. By this means the exploration is adaptively controlled and re-focused after each iteration. This concept allows the exploration to concentrate in each iteration on areas of the search space with the most approximation improvement potential.

After each exploration iteration. After each exploration iterations the dynamic bounding hypercube $\tilde{\mathcal{H}}_{\mathcal{S}}(\mathcal{P})$ as well as dynamic bounding working and non-working Pareto-fronts $\tilde{\mathcal{F}}^{\mathrm{w}}_{\mathcal{S},\mathcal{C}}$ and $\tilde{\mathcal{F}}^{\mathrm{nw}}_{\mathcal{S},\mathcal{C}}$ are updated.

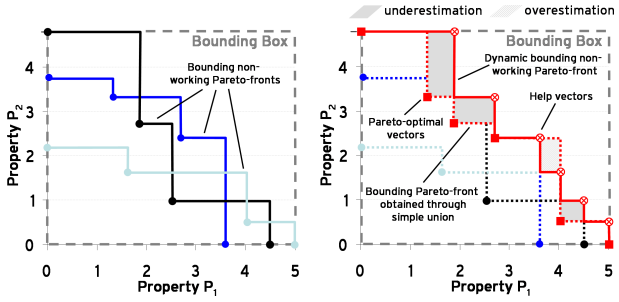
The dynamic bounding hypercube can simply be determined by calculating the union set of the bounding hypercubes corresponding to all so far evaluated system configurations:

$$\tilde{\mathcal{H}}_{\mathcal{S}}(\mathcal{P}) = \bigcup_{\text{evaluated } c} \mathcal{H}_{\mathcal{S}_c}(\mathcal{P})$$

The update of the dynamic bounding working Pareto-front is straightforward. It can be determined by simply calculating the Pareto-optimal vectors of the union set of the bounding working Pareto-fronts stored in Γ^{w}:

$$\tilde{\mathcal{F}}^{\mathrm{w}}_{\mathcal{S},\mathcal{C}} = \Omega^{+}(\bigcup \Gamma^{\mathrm{w}}) = \Omega^{+}(\{\vec{x} | \exists \mathcal{F} \in \Gamma^{\mathrm{w}} : \vec{x} \in \mathcal{F}\})$$

The calculation of the dynamic bounding non-working Pareto-front is more involved. Figure 3 shows how the bounding non-working Pareto-fronts of three considered system configurations are merged in order to obtain the integrated dynamic bounding non-working Pareto-front.



3: Dynamic bounding non-working Pareto-front update

In order to correctly integrate the information of the involved bounding non-working Pareto-fronts into the dynamic bounding non-working Pareto-front, the latter must cover all vectors which potentially represent a working system property combination. This is true for all vectors which are covered by at least one of the considered bounding non-working Pareto-fronts.

Therefore, we cannot simply proceed in the same way as for the dynamic bounding working Pareto-front, i.e. building the union set and then eliminating all Pareto-dominated vectors. This fact is visualized in Figure 3.

On the left side we see three bounding non-working Pareto-fronts, which we want to integrate into the dynamic bounding non-working Pareto-front. On the right side two different methods for deriving an integrated Pareto-front are visualized. The red dotted line connecting all Pareto-optimal vectors (marked with a square) with an upper step function represents the Pareto-front we obtain if we pursue the same integration strategy as for the dynamic bounding working Pareto-front. However, this strategy is not valid if we compare the obtained Pareto-front with the real bounding non-working Pareto-front visualized by the solid red line. In fact, the red dotted line incorrectly does not cover several regions colored in light gray, which corresponds to an underestimation of the real dynamic bounding non-working Pareto-front. Additionally, it covers the region hatched in light gray, which corresponds to an overestimation of the real dynamic bounding non-working Pareto-front.

In order to determine the correct dynamic bounding non-working Pareto-front, we need to calculate *help vectors*. The set of help vectors is equivalent to the set of all Pareto-optimal vectors that Pareto-dominate (in the sense of a maximization problem) no vector in at least one of the considered bounding Pareto-fronts. For the example given in Figure 3 the help vectors are marked with a cross. As we can see, they correctly define the limits of the integrated dynamic bounding non-working Pareto-front, and their inner hypervolume corresponds to the volume of the space it covers. Accordingly, the dynamic bounding non-working Pareto-front can be

calculated as follows:

$$\tilde{\mathcal{F}}^{\mathrm{nw}}_{\mathcal{S},\mathcal{C}} = \Omega^{+}\{\vec{f} \in \tilde{\mathcal{H}}_{\mathcal{S}}(\mathcal{P}) | \exists \mathcal{F} \in \Gamma^{\mathrm{nw}} : (\nexists \vec{x} \in \mathcal{F} : \vec{f} \triangleright_{+} \vec{x})\}$$

Dynamic robustness bounds. Like in the static case the dynamic bounding Pareto-fronts $\tilde{\mathcal{F}}^{\mathrm{w}}_{\mathcal{S},\mathcal{C}}$ and $\tilde{\mathcal{F}}^{\mathrm{nw}}_{\mathcal{S},\mathcal{C}}$ can be used to derive bounds for the dynamic design robustness of the analyzed system.

Definition 13 formally characterizes the dynamic minimum guaranteed robustness $\tilde{\mathcal{R}}^{-}_{\mathcal{S}}$, representing a lower robustness bound, and the dynamic maximum possible robustness $\tilde{\mathcal{R}}^{+}_{\mathcal{S}}$, representing a upper robustness bound.

Definition 13 (Dynamic Robustness Bounds)
We consider a system $\mathcal{S}$ and a set of system properties $\mathcal{P} = \{p_1, \ldots, p_n\}$. Given a set of evaluated parameter configurations $\mathcal{C}$ for $\mathcal{S}$, the dynamic bounding working and non-working Pareto-fronts $\tilde{\mathcal{F}}^{\mathrm{w}}_{\mathcal{S},\mathcal{C}}$ and $\tilde{\mathcal{F}}^{\mathrm{nw}}_{\mathcal{S},\mathcal{C}}$, as well as the set of weights $\mathcal{W} = \{w_1, \ldots, w_n\}$, the dynamic design robustness $\mathrm{DDR}_{\mathcal{S},\mathcal{C}}(\mathcal{P},\mathcal{W})$ is bounded by the dynamic minimum guaranteed robustness $\tilde{\mathcal{R}}^{-}_{\mathcal{S}}$ and the dynamic maximum possible robustness $\tilde{\mathcal{R}}^{+}_{\mathcal{S}}$:

$$\tilde{\mathcal{R}}^{-}_{\mathcal{S}}(\mathcal{P},\mathcal{W}) \leq \mathrm{DDR}_{\mathcal{S},\mathcal{C}}(\mathcal{P},\mathcal{W}) < \tilde{\mathcal{R}}^{+}_{\mathcal{S}}(\mathcal{P},\mathcal{W}), \text{ where}$$

$$\tilde{\mathcal{R}}^{-}_{\mathcal{S}}(\mathcal{P},\mathcal{W}) = \tilde{\lambda}^{-}_{\tilde{\mathcal{H}}_{\mathcal{S}}(\mathcal{P})}(\tilde{\mathcal{F}}^{\mathrm{w}}_{\mathcal{S},\mathcal{C}},\mathcal{W})$$

$$\tilde{\mathcal{R}}^{+}_{\mathcal{S}}(\mathcal{P},\mathcal{W}) = \tilde{\lambda}^{-}_{\tilde{\mathcal{H}}_{\mathcal{S}}(\mathcal{P})}(\tilde{\mathcal{F}}^{\mathrm{nw}}_{\mathcal{S},\mathcal{C}},\mathcal{W})$$

Note that $\tilde{\mathcal{R}}^{-}_{\mathcal{S}}$ represents a real lower dynamic robustness bound for the analyzed system, i.e. its real dynamic robustness potential is larger or equal to $\tilde{\mathcal{R}}^{-}_{\mathcal{S}}$. However, $\tilde{\mathcal{R}}^{+}_{\mathcal{S}}$ only represents an upper dynamic robustness bound for the set of system parameter configurations $\mathcal{C}$ evaluated during robustness approximation. Note that this is not a limitation of our approach, since generally a hard global upper bound cannot be determined without evaluating all possible system parameter configurations.

It is obvious that the approximation quality of the dynamic robustness bounds depends very much on the number of performed exploration iterations and the precision of the underlying multi-dimensional stochastic sensitivity analysis.

Illustrative example. Figure 5 gives an example for the multi-dimensional dynamic robustness approximation.

Figure 5a shows the initial approximations of the dynamic bounding working Pareto-fronts. Two new system parameter configurations are evaluated in the pending exploration iteration. The approximations of the bounding Pareto-fronts for these two configurations, obtained through stochastic multi-dimensional sensitivity analysis, are visualized in Figures 5b and 5d, respectively. The fitness values assigned to the newly evaluated configurations, i.e. the weighted percentage hypervolume of the space covered by their bounding working Pareto-fronts that is not already covered by the dynamic bounding working Pareto-front, are visualized in Figures 5c and 5e. Figure 5f shows the updated dynamic bounding Pareto-fronts at the end of the performed exploration iteration.

7. CASE STUDY

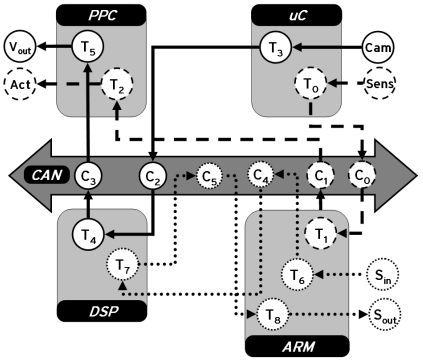
In this section we apply the robustness optimization techniques presented in this paper to a small but realistic example system. The system is described in Section 7.1 and in Section 7.2 we optimize its robustness properties.

7.1 Example system

We consider the example setup shown in Figure 4. It contains four different computational units connected by a bus. Three applications are mapped on the architecture. A video application (solid chain) gathers data from a camera controlled by the micro controller uC, performs preprocessing on the DSP, and post-processing on the PPC core. The second application (dashed chain) reads data from a sensor, which is first processed on the ARM core and then forwarded to the PPC core, which, in turn, controls an actor. The third application (dotted line) is a streaming application that runs on the ARM processor core and uses the DSP for data processing.

All three applications have constrained end-to-end latencies which need to be satisfied for the system to function correctly (Table 1c).

The computational resources (*PPC*, *uC*, *DSP*, and *ARM*) are all scheduled according to the static priority preemptive policy, and the interconnecting bus is arbitrated by the CAN protocol. Core execution and core communication times as well as priorities of all tasks and exchanged messages are given in Tables 1a and 1b, respectively. The periods of incoming data at the system inputs (*Cam*, *Sens*, and S_{in}) are specified in Table 1d.



4: Example system

Channel	*CCT*	*Priority*
C0	[10.4, 12.4]	5
C1	[12, 14.4]	3
C2	[20, 26.4]	2
C3	[15.2, 20]	6
C4	[10.4, 14.4]	1
C5	[15.2, 26.4]	4

(a) Core Communication Times

Task	*CET*	*Priority*
T5	[30.9, 43.1]	2
T2	[15.8, 27.4]	1
T3	[36.9, 46.3]	2
T0	[20.1, 48.5]	1
T4	[13, 86]	2
T7	[40.3, 44.5]	1
T1	[27.1, 160]	2
T6	[14.2, 63.6]	1
T8	[11.5, 291.2]	3

(b) Core Execution Times

Path	*Deadline*
$Sens \rightarrow Act$	850
$Cam \rightarrow V_{out}$	1100
$S_{in} \rightarrow S_{out}$	1000

(c) End-to-End Constraints

System Input	*Period $\mathcal{P}$*
Sens	500
Cam	100
S_{in}	1000

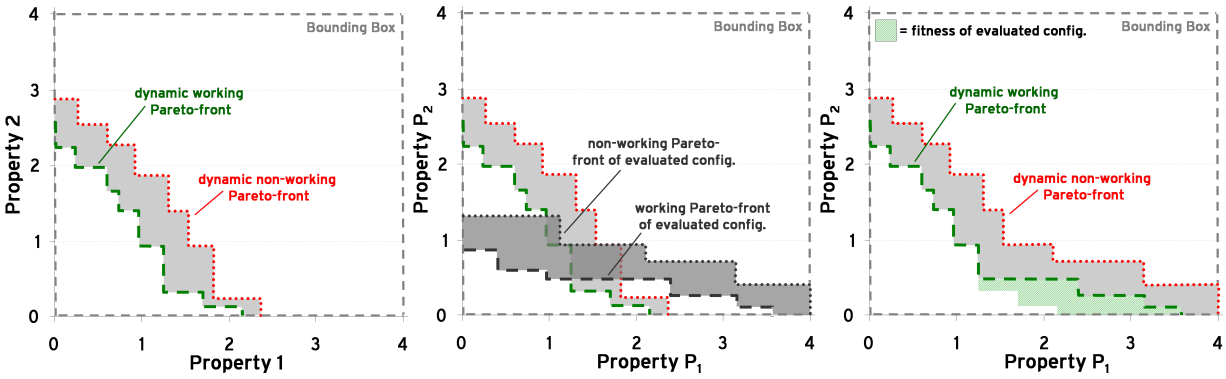
(d) Input Event Models

1: System Parameters

7.2 Robustness optimization

We optimize the two-dimensional static and dynamic robustness of the worst-case communication time of communication channel C3 and the worst-case execution time of task T1. Thereby, we attach a higher importance to the robustness of the task T1 (weight 2) than to the communication channel C3 (weight 1).

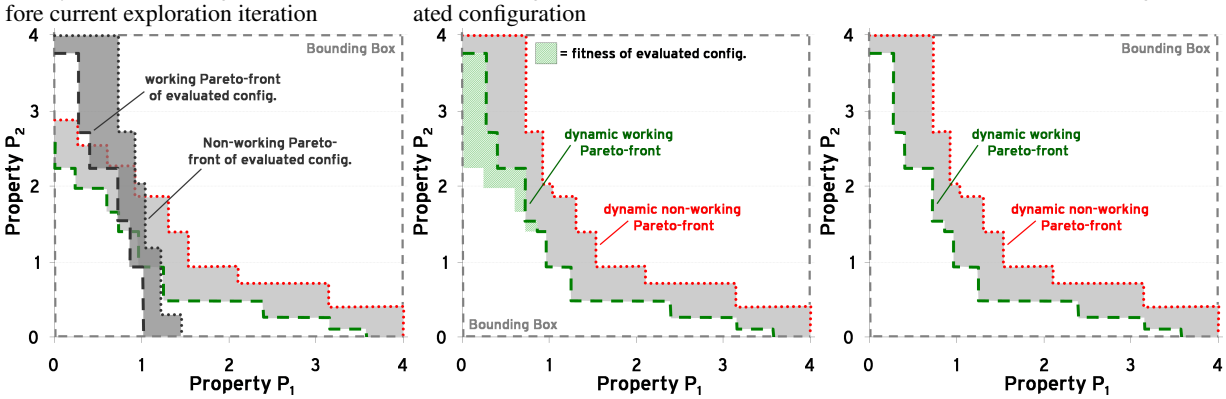
Table 2 shows the four relevant system configurations that we



(a) Dynamic bounding Pareto-fronts before current exploration iteration

(b) Bounding Pareto-fronts of first evaluated configuration

(c) Fitness of first evaluated configuration



(d) Bounding Pareto-fronts of second evaluated configuration

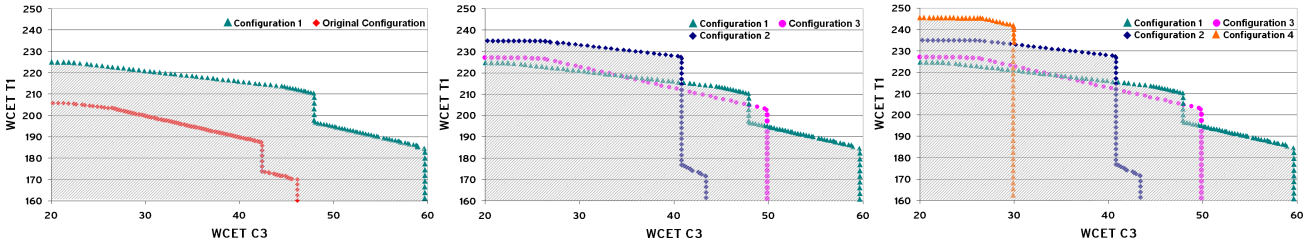
(e) Fitness of second evaluated configuration

(f) Dynamic bounding Pareto-fronts after current exploration iteration

5: Multi-dimensional dynamic design robustness approximation example

Config. #	*CAN*	*DSP*	*PPC*	*ARM*	*uC*	Robustness (C3 weight=1, T1 weight=2)
org	$C4 > C2 > C1 > C5 > C0 > C3$	$T7 > T4$	$T2 > T5$	$T6 > T1 > T8$	$T0 > T3$	0.2864
1	$C4 > C2 > C5 > C0 > C1 > C3$	$T7 > T4$	$T2 > T5$	$T6 > T1 > T8$	$T3 > T0$	**0.6452**
2	$C4 > C5 > C1 > C0 > C2 > C3$	$T7 > T4$	$T2 > T5$	$T6 > T1 > T8$	$T3 > T0$	0.5014
3	$C0 > C5 > C4 > C2 > C1 > C3$	$T7 > T4$	$T2 > T5$	$T6 > T1 > T8$	$T3 > T0$	0.5492
4	$C4 > C5 > C0 > C2 > C1 > C3$	$T7 > T4$	$T2 > T5$	$T6 > T1 > T8$	$T0 > T3$	0.2772

2: System configurations obtained during robustness optimization



(a) Static design robustness - original and optimized configurations

(b) Dynamic design robustness - including scheduling on CAN bus

(c) Dynamic design robustness - including scheduling on CAN bus and uC

6: Robustness optimization results: static and dynamic case

found during robustness optimization annotated with their individual robustness properties.

The highest static design robustness (0.6452) is achieved by configuration 1. Compared to the original configuration, which has a static robustness of 0.2864, this represents an increase of approximately 125% (Figure 6a).

If we add dynamic parameter reconfiguration we can further increase system robustness.

If we assume that the CAN message Ids (priorities) on the bus can be adapted as a reaction to property variations we can increase system robustness by more than 13% to 0.7317 (Figure 6b). Adding the priorities on *uC* to the reconfiguration space leads to a further dynamic design robustness increase to 0.7672, which corresponds to a total robustness increase of approximately 19% compared to the static case (Figure 6c).

If we apply the robustness gain metric $\mathcal{G}$ we obtain the following values for our two different reconfiguration spaces:

$$\mathcal{G}_{\mathcal{S},\{CAN\}}(\{C3,T1\},\{1,2\}) = 0.0865$$

$$\mathcal{G}_{\mathcal{S},\{CAN,uC\}}(\{C3,T1\},\{1,2\}) = 0.122$$

From these numbers we can conclude that bus reconfigurability is important to achieve high robustness for $C3$ and $T1$. Also, more surprisingly, a further robustness increase can be achieved if *uC* is reconfigurable. The robustness gain is smaller but still relevant.

8. CONCLUSION

In this paper we addressed the problem of system property variations during the design and the maintainance of complex embedded systems.

We presented expressive multi-dimensional robustness metrics for different design scenarios. The static design robustness is adapted to the scenario where parameters are defined and fixed early at design time and cannot be modified later as a reaction to system property variations, whereas the dynamic design robustness includes potential counteractions in reaction to system property variations. Since robustness optimization is a computationally intensive problem, we proposed efficient exploration methods allowing to derive upper and lower robustness bounds with little computational effort.

The consideration of the proposed multi-dimensional robustness criteria during early phases of the embedded system design flow can significantly reduce design risk and increase system stability and maintainability.

9. REFERENCES

[1] S. Ali, A. Maciejewski, H. Siegel, and J. Kim. Measuring the robustness of a resource allocation. *IEEE Transactions on Parallel and Distributed Systems*, 15(7):630–641, July 2004.

[2] S. Chakraborty, S. Künzli, and L. Thiele. A general framework for analysing system properties in platform-based embedded system designs. In *Proc. of the IEEE/ACM Design, Automation and Test in Europe Conference (DATE)*, Munich, Germany, 2003.

[3] M. Fleischer. The measure of pareto optima: Applications to multi-objective metaheuristics. *Lecture Notes in Computer Science*, 2632:519–533, 2003.

[4] D. Gu, F. Drews, and L. Welch. Robust task allocation for dynamic distributed real-time systems subject to multiple environmental parameters. In *Proc. of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS)*, Columbus, Ohio, USA, June 2005.

[5] A. Hamann, R. Racu, and R. Ernst. A formal approach to robustness maximization of complex heterogeneous embedded systems. In *Proc. of the IEEE/ACM/IFIP International Conference on HW/SW Codesign and System Synthesis (CODES-ISSS)*, Seoul, South Korea, October 2006.

[6] A. Hamann, R. Racu, and R. Ernst. Multi-dimensional robustness optimization in heterogeneous distributed embedded systems. In *Proc. of the 13th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Bellevue, WA, USA, April 2007.

[7] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst. System level performance analysis - the SymTA/S approach. *IEE Proceedings Computers and Digital Techniques*, 152(2):148–166, March 2005.

[8] V. Izosimov, P. Pop, P. Eles, and Z. Peng. Design optimization of time- and cost-constrained fault-tolerant distributed embedded systems. In *Proc. of the Design Automation and Test in Europe Conference (DATE)*, Munich, Germany, March 2005.

[9] C. Lu, J. Stankovic, S. Son, and G. Tao. Feedback control real-time scheduling: framework, modeling, and algorithms. *Real-Time Systems Journal*, 23(1-2):85–126, 2002.

[10] R. Racu and R. Ernst. Scheduling anomaly detection and optimization for distributed systems with preemptive task-sets. In *12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, San Jose, USA, April 2006.

[11] R. Racu, R. Ernst, M. Jersak, and K. Richter. A virtual platform for architecture integration and optimization in automotive communication networks. In *Proc. of the SAE World Congress*, Detroit, USA, April 2007.

[12] R. Racu, A. Hamann, and R. Ernst. A formal approach to multi-dimensional sensitivity analysis of embedded real-time systems. In *Proc. of the Euromicro Conference on Real-Time Systems (ECRTS)*, Dresden, Germany, July 2006.

[13] R. Racu, M. Jersak, and R. Ernst. Applying sensitivity analysis in real-time distributed systems. In *Proc. of the 11th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, San Francisco, California, March 2005.

[14] M. Verhoef, E. Wandeler, L. Thiele, and P. Lieverse. System architecture evaluation using modular performance analysis - a case study. In *Proc. of the 1st IEEE/ACM International Symposium on Leveraging Applications of Formal Methods (ISOLA)*, Pafos, Cyprus, Oct 2004.

[15] S. Vestal. Fixed-priority sensitivity analysis for linear compute time models. *IEEE Transactions on Software Engineering*, 20(4), april 1994.

[16] L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, February 2006.

[17] E. Zitzler. Hypervolume metric calculation:. *ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/hypervol.c*, 2001.

[18] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, Heidelberg, Germany, September 2004. Springer.

[19] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for multiobjective optimization. In *Proc. Evolutionary Methods for Design, Optimisation, and Control*, pages 95–100, Barcelona, Spain, 2002.