Ontological reasoning in the design space exploration of advanced cyber–physical systems

Ontological Reasoning in the Design Space Exploration of Advanced Cyber-Physical Systems

Yon Vanommeslaeghe*‡, Joachim Denil*‡, Jasper De Viaene*†§, David Ceulemans*‡,
Stijn Derammelaere*†§, Paul De Meulenaere*‡

*CoSys-Lab (FTI), University of Antwerp, Belgium
†Department of Electrical Energy, Metals, Mechanical Constructions and Systems, Ghent University, Belgium
‡AnSyMo/CoSys, Flanders Make, Belgium
§EEDT-DC, Flanders Make, Belgium

***Abstract*—Cyber-physical systems are becoming increasingly complex. In these advanced systems, the different engineering domains involved in the design process become more and more intertwined. Therefore, a traditional (sequential) design process becomes inefficient in finding good design options. Instead, an integrated approach is needed where parameters in multiple different engineering domains can be chosen, evaluated, and optimized to achieve a good overall solution. However, in such an approach, the combined design space becomes vast. As such, methods are needed to mitigate this problem.**

In this paper, we show a method for systematically capturing and updating domain knowledge in the context of a co-design process involving different engineering domains, i.e. control and embedded. We rely on ontologies to reason about the relationships between parameters in the different domains. This allows us to derive a stepwise design space exploration workflow where this domain knowledge is used to quickly reduce the design space to a subset of likely good candidates. We illustrate our approach by applying it to the design space exploration process for an advanced electric motor control system and its deployment on embedded hardware.

***Index Terms*—Embedded Systems, Cyber-Physical Systems, Co-Design, Design Space Exploration, Ontological Reasoning**

I. Introduction

The design of Cyber-Physical Systems (CPS) is inherently a co-design process between different engineering domains. Indeed, even in its simplest form, a CPS consists of a control or monitoring algorithm, designed by a control engineer, it runs on an embedded platform, which is designed by an embedded engineer, and which controls or monitors a physical system, designed in turn by a mechanical engineer. However, over the years, the complexity of CPS and the control and monitoring algorithms used therein have increased significantly. For example, in control, there has been an evolution from simple PID-controllers to more complex model-predictive control (MPC). This trend is expected to continue with more advanced algorithms that contain models of the physics of the systems to be monitored and controled. However, before these algorithms can be used in real-world applications, they need to be deployed on an embedded platform. As these models are computationally expensive, embedded deployment is challenging, especially when real-time performance is required. For example, model predictive path integral control (MPPI) [1] can be used in autonomous driving for trajectory optimization. However, this requires parts of the MPPI algorithm to be run on a graphics processing unit (GPU) to allow real-time operation [2].

In these situations, the different domains (i.e. control and embedded) become intertwined, with decisions made in one domain imposing additional restriction and requirements on the other. For this reason, a traditional, sequential design process, i.e. first design of the control algorithm followed by deployment on an embedded platform, becomes insufficient to ensure efficient deployment and integration. Indeed, it becomes necessary to view this as a multi-domain co-design process, where parameters in both the control and embedded domain must be considered and evaluated to arrive at a good combination of software and hardware configuration.

However, in such an integrated design process, the design space expands dramatically due to the increased number of degrees of freedom. Therefore, the design space exploration (DSE) needs to be performed efficiently. By means of an example, the current paper demonstrates how domain knowledge captured in an ontology can be used to guide the DSE process. We demonstrate this for the embedded deployment of an advanced motor control algorithm [3], [4].

The paper is organized as follows. First, Section II discusses related work. Next, in Section III, we describe our method and the different steps therein. After this, in Section IV, we present our use case and its associated design space. In Section V, we demonstrate our proposed method by applying it to this use case. The results of which are presented in Section VI and compared to those achieved with traditional design space exploration techniques. Lastly, in Section VII, we present our conclusions and future work.

II. Related Work

A lot of research has already been done regarding hardware/software co-design. Earlier hardware/software co-design frameworks, such as Metropolis [5] and later Metro II [6], generally follow the principle of the platform-based design [7] methodology: the “orthogonalization of concerns”, i.e. separation of different aspects of the design. These frameworks generally only consider “embedded” parameters, such

as timing and resource utilization, instead of system-level performance, for example, the impact of execution time on control performance. As such, the embedded system is often designed to support the most demanding algorithm configuration, regardless of the configuration that is eventually used. For complex cyber-physical systems, this is no longer realistic. As the different domains become more intertwined, the abstractions taken in these frameworks become inefficient in providing effective and efficient solutions. More recently, efforts have been made regarding the co-optimization of CPS designs in multiple domains and on resource-aware embedded control system design [8]. This also motivated the development of Metronomy [9], which allows for the co-simulation of architectural models from Metro II [6] with functional models from Ptolemy II [10].

More recently, effort has been made to optimize control and monitoring algorithms together with their embedded deployment strategy. Aminifar et al. [11] address the controller-server co-design problem for multiple control tasks running on a shared platform. Here, virtual servers are used to divide the computational resources among the different control tasks. However, the configuration of these servers has a large impact on the delay and jitter experienced by each controller, and thus their stability. Their goal is to optimize the server configurations, together with the sample period of the corresponding controllers. To do this, a stability curve is first computed to determine the maximum tolerable response-time jitter for a given nominal delay that the control application experiences. This information is then used to determine the optimal configuration of server parameters and sample period that minimizes the overall resource utilization while guaranteeing system stability. This approach is compared to a previously presented server design approach [12] where the server configurations are optimized, but the sample period is given as would likely be the case in a more traditional, sequential design process. Their results show that the co-design approach achieves, on average, a 44% lower resource utilization than the traditional approach. Similarly, Roy et al. [13] consider the design of a (distributed) FlexRay-based embedded control system. Here, they co-optimize controller design together with the FlexRay schedule, resulting in a Pareto front depicting the trade-off between control performance and bus-utilization.

While Roy et al. [13] and Aminifar et al. [11], [12] consider offline optimization, Cervin et al. [14] have previously studied online optimal sampling period assignment for multiple controllers running on a shared embedded platform. In their work, a cost associated with each controller's performance given a certain sampling period is precomputed and stored in memory on the embedded platform. A feedback scheduler is then used to periodically reassign the sampling periods for the controllers. This is done based on, among other things, the current plant noise as estimated by the controller. The feedback scheduler optimizes the sampling periods to obtain the best overall control performance while ensuring schedulability. Their results show that the optimal sampling period assignment achieves much better overall control performance than a traditional design with fixed periods.

As previously mentioned, the design of cyber-physical systems is a co-design process. Engineers from different disciplines work concurrently on different aspects of the systems, after which the different parts are combined. However, inconsistencies during the concurrent phase cause problems during the integration phase. This led to the concept of contract-based design (CBD) [15], which focuses on establishing a consistent design during concurrent design phases. Contracts specify certain assumptions and guarantees between two parts of the system based on negotiations between different engineers. Such contracts have also been used in DSE for CPS. For example, Finn et al. [16] propose a mixed discrete-continuous optimization methodology for CPS architecture exploration where they make use of contracts to ensure consistent architecture candidates.

From the state of the art, it is clear that there is a benefit to co-optimization between the control- and embedded domains and that it is possible to come up with efficient strategies for design-space exploration and optimization. However, it is apparent that determining these strategies is not straightforward. It generally requires a good grasp of both the control- and embedded domain and their interdependencies (domain knowledge). As such, methods that allow engineers from different domains to reason about these cross-domain relationships are needed for them to derive an efficient design-space exploration strategy. It is here that ontologies can be used to explicitly capture the combined domain knowledge and leverage it to guide the DSE process.

Previously, Vanherpen et al. [17] worked on combining contract-based design with ontological reasoning to reason about the content of such contracts. Similarly, Dávid et al. [18] use ontologies in the context of (in)consistency management, where they defined different levels of precision for the relationships captured in an ontology. However, neither explored the use of ontologies in determining an efficient DSE process.

In previous work, we proposed the use of ontologies to capture domain knowledge in a co-design process, including engineers from different domains, and subsequently leveraging this ontology to derive an efficient design space exploration strategy for a system under design [19]. In the current paper, we extend this method by introducing characterization and ontology updating steps. These steps allow the systematic updating of the ontology as new information or insights become available during the design process. We demonstrate this by performing sensitivity analyses on different parts of the system under the design. The results of which are subsequently used to update the ontology. Adding this sensitivity information allows us to prioritize certain (more sensitive) parameters over others during the DSE process, further increasing its efficiency. Similarly, we show how the results of the characterization steps can be reused in certain situations to provide an initial best guess for certain parameters, further reducing the design space. Additionally, we compare our approach to traditional design space exploration techniques regarding the quality of the solutions as well as the time required to perform the

exploration.

III. Method

We propose the use of ontologies to facilitate the optimization of systems involving multiple engineering domains. Ontologies enable the representation of shared knowledge in a domain of discourse using a common language [20], including the explicit definition of different concepts with their corresponding properties. In the presented method, these ontologies are used to capture the different properties in different domains that are important to the design of the system, as well as their relationships to each other. This allows engineers from different domains to reason about the design as a whole. Consequently, these ontologies are useful in discovering potential trade-offs and points of attention, and ultimately to derive an efficient design space exploration strategy by leveraging this information. However, these ontologies are not static. They need to be updated and refined when new information or insights become available. As such, we discern a number of different steps to be taken during the design process to define and update these ontologies:

1) **Ontology definition (at the system-level)**
   This step entails the definition of a first (system-level) ontology by engineers from the different domains involved in the system under design. This step takes place early on in the design process and happens in conjunction with requirements specification and the definition of the overall system architecture as these actions provide information about (i) different concerns and (ii) connections between different components of the system. This system-level ontology is defined by the engineers based mainly on previous experience. It serves as a first high-level overview of the different concerns in the different domains and how they relate to each other. However, it can already be used to identify potential trade-offs and other points of attention.

2) **Ontology updating**
   During the development process, the engineers gain more in-depth knowledge about the different parameters and properties they need to consider, as well as relationships between these parameters and components. New information may originate from deductive reasoning about the design or from observations (inductive reasoning), e.g. based on simulation results or other analyses. This information is instrumental in further refining the system-level ontology. However, as the design of cyber-physical systems is generally a co-design process, different components of the overall system are developed concurrently. As such, we further discern different substeps at the component level that serve to update the system-level ontology:

   a) *Ontology definition (on a component-level)*
      Similar to the system-level ontology, this first step entails the definition of a more low-level, possibly domain-specific, ontology for each specific component. This ontology is defined based on things like requirements, but also in/outputs of the component and associated properties, intermediate properties considered during the design process, etc. At this substep, the ontology is again mostly based on experience, by reasoning about the design. As a result, information about the nature of the relationships between different properties is limited. However, this provides insight into which experiments need to be performed, and can be performed, to further characterize these relationships in the next substep.

   b) *Characterization*
      This step serves to quantify the relationships captured in the (component-level) ontology. Relationships can be characterized exactly, using mathematical equations, or based on results of experiments. For example, sensitivity analyses can be performed to determine how sensitive different properties are to changes in considered design parameters. The information gathered in this substep is used in the following substep to prune and simplify the component-level ontology in preparation for the last step (lifting).

   c) *Pruning*
      In this substep, results from the characterization substep are used to prune and simplify the ontology by pruning unimportant relationships and intermediate properties where possible, taking into account relationships to system-level properties. This results in a more abstract ontology, which ideally includes only parameters and properties linked to the system-level ontology. As such, it allows engineers from different domains to reason about these relationships at a higher level of abstraction, without having to consider unnecessary details. This is done in preparation for the final substep.

   d) *Lifting*
      Lastly, the information contained in the pruned component-level ontology is lifted to the system-level, where it is linked to system-level properties and combined with the information about other components. As such, the system-level ontology is refined and updated based on the new information learned about each component.

   These substeps can also be applied recursively. For example, when one component can be further decomposed into smaller subcomponents, which can themselves be further decomposed, etc. In these cases, these substeps can be applied to each subcomponent, after which the information about each subcomponent can be combined and lifted to the component level,

where it is combined with information about other components and again lifted, etc.

3) **Determining a design space exploration strategy** Engineers can use the ontology and the information contained therein to determine an efficient design space exploration strategy. Indeed, reasoning about the relationships between different properties reveals where trade-offs can be made and where evaluations in one domain can be used to constrain the design space in another domain. Additionally, information about the nature of these relationships, as captured during the *characterization* substep, can be used to prioritize certain parameters, e.g. by fixing the most sensitive parameters first. Similarly, information about the time it takes to evaluate different objectives can be used to prioritize different objectives, as fast evaluations may be used to quickly constrain the design space early on in the design space exploration process.

These steps are demonstrated in Section V, where we show an example workflow. In this workflow we apply the different steps to an example use case. The use case itself is introduced in the following section.

IV. Example Use Case

In this paper, we consider the deployment of an advanced, energy-efficient load angle control system for a brushless DC (BLDC) motor, based on the algorithm proposed by De Viaene et al. [3]. A general overview of this system is given in Subsection IV-A, while the design space considered in this paper is defined in Subsection IV-B. Lastly, in Subsection IV-C, we describe how the system-level performance of potential design candidates is evaluated for this system. After this, in Section V, we apply the different steps of the presented method to analyze the problem and derive a corresponding DSE-strategy.

A. Background Information

Electric motors generally consist of two main parts: a stationary part (stator) and a rotating part (rotor). In brushless DC motors, the rotor contains permanent magnets. By injecting alternating three-phase currents in the stator windings, a rotating magnetic stator field is created which attracts the magnetic rotor field created by the permanent magnets (Figure 1) [21]. The generated electromagnetic motor torque $\mathbf{T}_{\mathrm{em}}$ can be expressed as the cross product of the stator flux linkage space vector $\mathbf{\Psi}_{\mathrm{s}}$ and the stator current vector $\mathbf{i}_{\mathrm{s}}$ [22].

$$\mathbf{T}_{\mathrm{em}} = \mathbf{\Psi}_{\mathrm{s}} \times \mathbf{i}_{\mathrm{s}} \tag{1}$$

By neglecting saturation [23], [24], the stator flux linkage space vector $\mathbf{\Psi}_{\mathrm{s}}$ can be written as the sum of the stator flux linkages, established by the two stator currents in the dq-reference frame and the permanent magnet rotor flux vector $\mathbf{\Psi}_{\mathrm{r}}$ (Figure 1). The electromagnetic motor torque $\mathbf{T}_{\mathrm{em}}$ can be rewritten as:

$$\mathbf{T}_{\mathrm{em}} = (\mathbf{\Psi}_{\mathrm{r}} + L_{\mathrm{d}}\mathbf{i}_{\mathrm{d}} + L_{\mathrm{q}}\mathbf{i}_{\mathrm{q}}) \times \mathbf{i}_{\mathrm{s}} \tag{2}$$

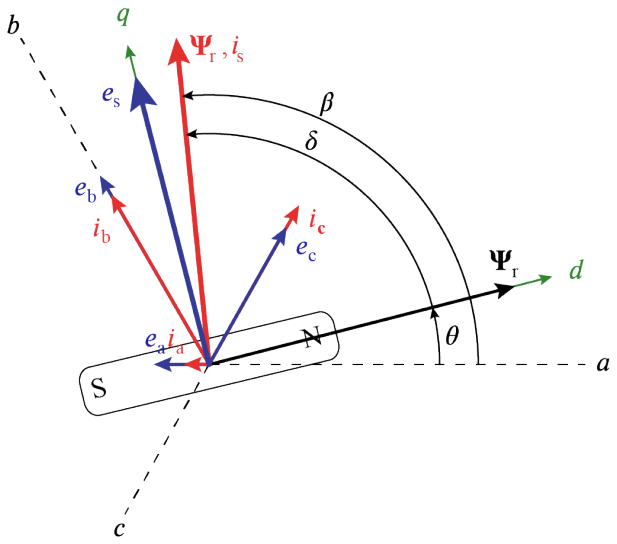

Fig. 1. Three-phase vector diagram with stator current $\mathbf{i}_{\mathrm{s}}$ and stator flux $\mathbf{\Psi}_{\mathrm{s}}$, rotor flux $\mathbf{\Psi}_{\mathrm{r}}$ and load angle δ represented in dq-reference frame fixed to the rotor flux.

where L_{d} and L_{q} respectively denote the d-axis and q-axis inductance in the dq-reference frame. Elaboration of the vector products leads to an equation describing the electromagnetic torque as a function of stator current i_{s}, the rotor flux Ψ_{r} and the load angle δ, defined as the angle between $\mathbf{i}_{\mathrm{s}}$ and the rotor flux $\mathbf{\Psi}_{\mathrm{r}}$ (Figure 1):

$$T_{\mathrm{em}} = |\mathbf{\Psi}_{\mathrm{r}}|.|\mathbf{i}_{\mathrm{s}}|.\sin(\delta) + \frac{L_{\mathrm{d}} - L_{\mathrm{q}}}{2}|\mathbf{i}_{\mathrm{s}}|^2.\sin(2\delta) \tag{3}$$

In this equation, the first term represents the torque generated by the interaction between the permanent magnet rotor flux $\mathbf{\Psi}_{\mathrm{r}}$ and the stator current i_{s}. This term depends on the sine of the load angle δ. The second term represents the reluctance effect, which is solely due to the saliency and disappears for a smooth-air-gap machine where $L_{\mathrm{d}} = L_{\mathrm{q}}$ [23], [24]. By neglecting the reluctance effect, the electromagnetic torque with the torque constant k_{t} can be written as:

$$\mathbf{T}_{\mathrm{em}} = \mathbf{\Psi}_{\mathrm{r}} \times \mathbf{i}_{\mathrm{s}} \tag{4}$$

$$T_{\mathrm{em}} = |\mathbf{\Psi}_{\mathrm{r}}|.|\mathbf{i}_{\mathrm{s}}|.\sin(\delta) \tag{5}$$

$$= k_{\mathrm{t}}.|\mathbf{i}_{\mathrm{s}}|.\sin(\delta) \tag{6}$$

From (6), it can be concluded that maximum torque generation is achieved when the load angle δ is equal to 90°. The load angle can thus be seen as a quality factor of the torque generation because it contains information about the torque/current ratio:

$$\sin(\delta) \sim \frac{T_{\mathrm{em}}}{|\mathbf{i}_{\mathrm{s}}|} \tag{7}$$

Standard control methods make use of feedback mechanisms such as Hall sensors or absolute/incremental encoders to measure the rotor position [25]–[27]. This is interesting as optimal torque generation can be achieved by injecting the phase currents in such a way that the resulting stator current

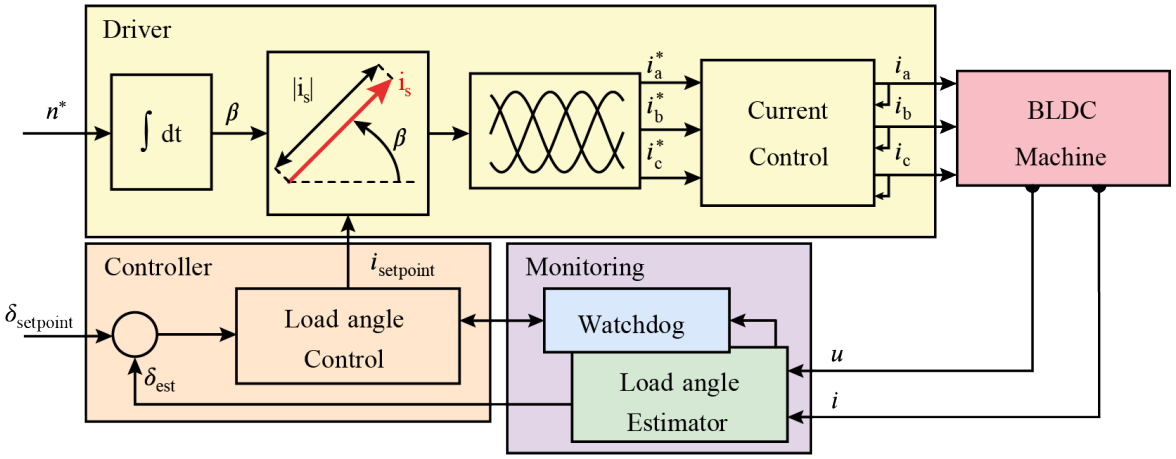

Fig. 2. Sensorless load angle control scheme for BLDC motors, the load angle controller adjusts the amplitude $|\mathbf{i}_s|$ of the stator current vector $\mathbf{i}_s$ while the position β of this vector is imposed by integration of the speed setpoint n^*.

vector $\mathbf{i}_s$ leads the rotor flux $\mathbf{\Psi}_r$ by $\delta = 90°$. By adapting the current amplitude $|\mathbf{i}_s|$, the motor speed can be controlled.

The estimation and load angle control method proposed by De Viaene et al. [3] totally differs from the conventional method. The speed is purely imposed in open-loop by the rotating current vector while the closed-loop load angle controller handles torque quality by adapting the current amplitude based on feedback of the estimated load angle (δ_{est}) (Figure 2). The advantage of this method compared to conventional control is that no position feedback or control is necessary. However, the added load angle controller is used to ensure the speed setpoint is correctly followed.

A simplified overview of this system is shown in Figure 3. It consists of two main parts: a monitoring part, containing an estimator which estimates the actual load angle of the motor (δ_{est}) from measured line voltages and currents (u and i respectively), and a controller, which drives the load angle of the motor to a setpoint ($\delta_{setpoint}$) by changing the current setpoint of the motor driver ($I_{setpoint}$). The load angle can be increased by reducing the current (Figure 4), leading to better energy efficiency. However, this affects the robustness of the system. For example, an increase in load torque can cause a sudden increase in the load angle. If the load angle increases past the optimal value of 90°, the motor torque decreases (Figure 4 & Equation 6), which results in a loss of synchronism between the stator and rotor field. For this reason, the monitoring part contains a watchdog that monitors the estimated load angle. If the estimated load angle passes a certain threshold (δ_{max}), the watchdog will overrule the controller by momentarily increasing the current setpoint to its maximum (I_{max}) and will reset the controller. This is done to avoid the system becoming unstable and causing the motor to lose synchronism.

There are a number of parameters that can be changed that impact the system's overall performance, as well as the embedded deployment. These include the monitoring period, controller period, and load-angle setpoint and threshold in

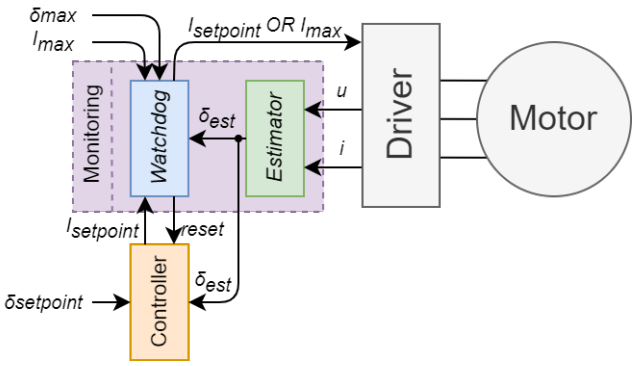

Fig. 3. Schematic overview of the load-angle controller [19].

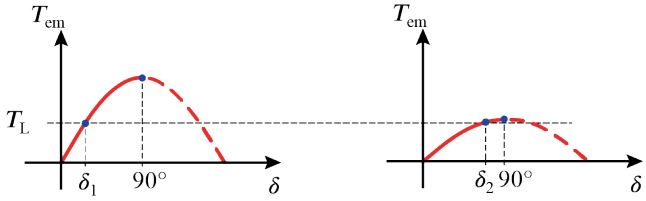

Fig. 4. The impact on the load angle δ and the motor torque T_{em} when the current amplitude $|\mathbf{i}_s|$ is halved and the load torque T_l is constant.

the control domain, but also the task assignment of these different parts in the embedded domain. These parameters are further defined in the following subsection. Additionally, the monitoring part can be split into multiple sections, as shown in Figure 5. This increases the flexibility regarding the embedded deployment as we can make use of parallelization on multiple cores. The load angle estimator can be divided into four sections: some preprocessing (*Pre.*), two sliding discrete Fourier transforms [28] (*SDFT*), and the actual load angle calculation (*Angle*), the last section of the monitoring part is then simply the watchdog (*WD*).

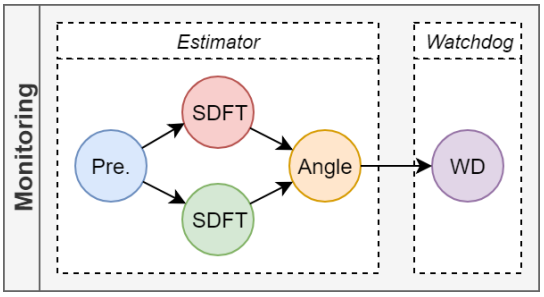

Fig. 5. Different code sections of the monitoring part [19].

B. Design Space

From the description given in the previous section, the decision variables (DV) and objective functions (OF) can be determined. This subsection gives an overview of these decision variables and objective functions in both the control and embedded domain. The ranges of values for these decision variables, as considered in this paper, are also listed where appropriate. For the objective functions, an indication of the time needed to evaluate them is given.

- Control domain
  - Decision variables
    - Monitor period: *[5 .. 40] μs*
    - Control period: *[5 .. 40] μs*
    - Load angle setpoint: *[45 .. 90] degrees*
    - Load angle threshold: *[90 .. 120] degrees*
  - Objective functions
    - Energy consumption: *slow evaluation (minutes)*
    - System stability: *slow evaluation (minutes)*
- Embedded domain
  - Decision variables
    - Task assignment/partitioning: *on [1 .. 2] cores*
    - Clock speed: *200 MHz*
  - Objective functions
    - Schedulability: *fast evaluation (seconds)*

These parameters need to be balanced to achieve good system-level performance. As such, a multi-domain co-design approach (in this case, a combined control and embedded engineering approach) may result in a non-intuitive deployment option. Here, we use design-space exploration to find viable (i.e. deployable and stable) and good (i.e. high energy efficiency) configurations.

C. System-level Performance Evaluation

System-level performance in the context of this use-case is defined as the average energy consumption of the system. As such, we want to find algorithm configurations and corresponding deployment options that minimize the power consumption while maintaining system stability. To evaluate the energy consumption and stability of a given design candidate we make use of simulations. Here, a Simulink model of the load angle control algorithm, together with a plant model of a BLDC motor, is used to determine the performance of the system during a predefined test sequence.

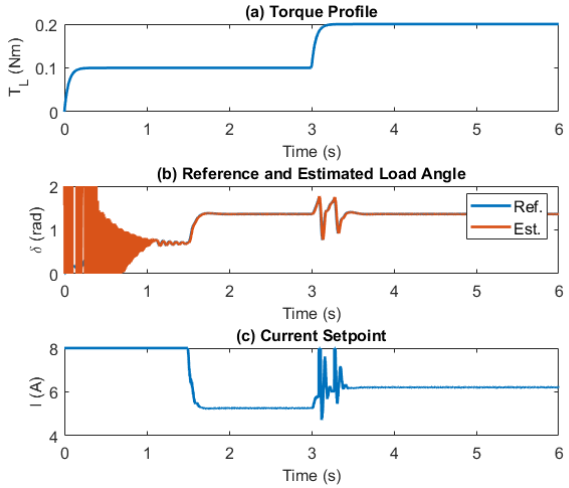

Fig. 6. Example traces obtained during a test run [19].

Example traces collected during such a test sequence are shown in Figure 6. Here, the motor first starts in open loop, with maximal current. When the motor reaches its operating speed, the system switches to the load angle control system ($t \approx 1.5s$). The motor current decreases as the controller drives the load angle to its setpoint, decreasing power consumption. After three seconds ($t = 3s$) the load on the motor increases. This tests the watchdog response as it causes a sudden increase in load angle. When the watchdog responds, this causes a momentary increase in current consumption. Once the system has stabilized again ($t \approx 3.5s$), it switches back to the load angle control. During this test sequence, the power consumption as well as the motor speed are monitored. If the motor stalls during the test sequence, the configuration is considered unstable. This allows us to determine (i) the average power consumption and (ii) the stability of the system.

V. Example Workflow

In this section, we show an example workflow, demonstrating how our approach can be used to analyze and optimize the design of an advanced control algorithm for a CPS. We do this by applying the different steps presented in Section III to the use case presented in the previous section. In doing so, we gradually build the ontology for this use case. After this we use this ontology to derive an efficient design space exploration strategy, allowing us to quickly find good design candidates for the optimal deployment of this advanced algorithm.

A. Ontology Definition

We start by building an ontology to capture different concepts important to this problem, as well as their relationship to each other. We do this for both the control and embedded domain. The full ontology is shown in Figure 7. Here, decision variables and objective functions, as listed in Section IV-B, are indicated in blue and green, respectively. The arrows indicate

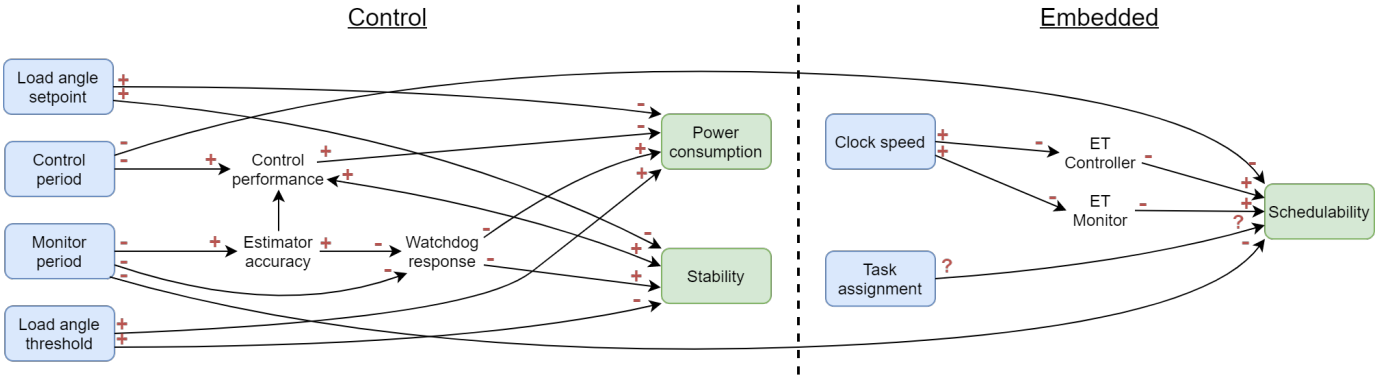

Fig. 7. Ontology relating to the example case.

which parameter affects which directly. For example, in the control domain, we know that the monitor period affects the accuracy of the load angle estimate (estimator accuracy). This in turn affects the watchdog's ability to respond quickly to situations where the load angle becomes too high, which can affect the stability of the system and ultimately the energy efficiency. In the embedded domain, we know that the clock speed of the CPU affects the execution time of the two tasks (i.e. monitor and controller). This, together with the task mapping, affects the schedulability of the system. Note that the clock speed is kept constant in this paper. In future work, the clock speed could be linked to the energy consumption of the embedded platform itself, adding an additional trade-off that can be taken into account. Additionally, some information about the nature of the relationships between the parameters has been added. The plus and minus signs indicate how these parameters affect each other. For example, decreasing the monitor period (-) generally leads to increased accuracy of the estimator (+). Similarly, increasing the CPU clock (+) generally decreases the execution times (-). Lastly, while we know there is some connection between task assignment and schedulability, the relationship is ambiguous. This is indicated using question marks.

The ontology also shows some relationships between parameters in the different domains. The chosen monitor and control periods in the control domain affect the schedulability in the embedded domain, as they directly determine the task periods on the embedded platform. Conversely, this means that the schedulability in the embedded domain constrains the design space in the control domain.

This ontology is further updated and refined over the course of the development process. This is shown in the following subsection. However, by reasoning about the ontology, we can already draw some conclusions:

1) As previously mentioned, the schedulability in the embedded domain constrains the design space in the control domain. As the schedulability can be evaluated quickly, this means we can quickly evaluate and discard infeasible combinations of monitor and controller periods.
2) In general, "faster is better". From the ontology, we can see that, in general, a shorter monitor or controller period should lead to better energy efficiency. However, shorter periods do decrease the schedulability of the system.

Additionally, by building the ontology, a third conclusion, which is not directly derived from the ontology, was brought forward by the control engineers:

3) It is not useful to have a controller period shorter than the monitor period because the controller uses the load angle estimated by the monitor as feedback. However, having a shorter monitor than controller period could be beneficial as this would lead to a more accurate estimate and thus, potentially better control performance. Additionally, a shorter monitor period means the watchdog can respond more quickly, potentially increasing system stability.

B. Ontology updating

In this subsection, we consider each component of the overall architecture (i.e. estimator, watchdog, and controller) separately. For each component, we first construct a smaller but more detailed ontology. After this, we use a parameter sweep (for the estimator) and sensitivity analysis techniques (for the controller and watchdog) to characterize the different relationships to determine the most important ones. Lastly, in Subsection V-C, the information gained is used to update the system-level ontology.

1) Estimator:

a) Ontology definition: The ontology of the estimator is shown in Figure 8. Design parameters are shown on the left, and objectives on the right. Additionally, some intermediate properties are shown. **Design parameters** include the monitor period (P_M) (from the system level ontology), as well as some application-specific parameters such as the number of motor poles (p) and required (mechanical) rotational speed of the motor (ω_M). In the current paper, we consider these last two parameters to be fixed. However, in a real-world scenario, these could be linked to operating range requirements. **Intermediate properties** include the electrical rotational speed (ω_E) and the number of samples per signal period (N). **Objectives** include estimator accuracy in the control domain, and memory footprint and execution time in the embedded domain.

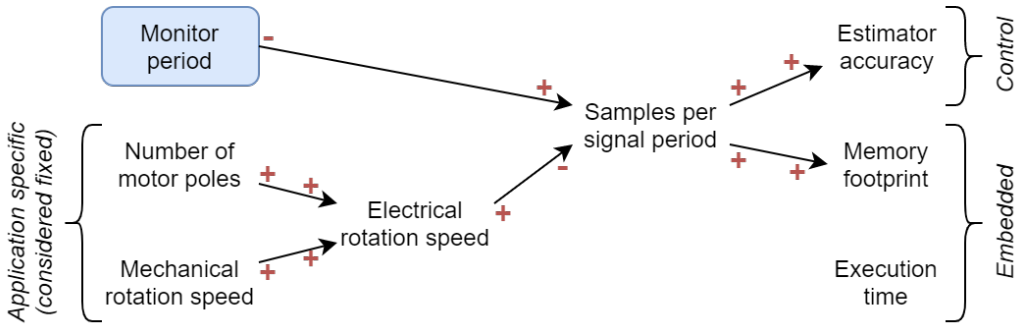

Fig. 8. Ontology of the estimator.

b) Characterization: A number of the relationships shown can be expressed using exact mathematical relationships. For example, the number of samples (N) can be calculated from the monitor period (P_M) and electrical rotational speed (ω_E), as follows: $N = 2\pi/(\omega_E P_M)$. The electrical rotational speed (ω_E) can in turn be calculated from the mechanical rotational speed (ω_M) and the number of poles (p). Other relationships are harder to define, such as the relationship between the number of samples and the estimator accuracy or memory footprint. However, these may be determined empirically, as discussed further below. Lastly, there is no relationship between the execution time and other properties for this specific estimator implementation. However, this may be the case in other situations. For example, if a moving horizon estimator (MHE) were to be used, the number of samples would impact the execution time.

As previously mentioned, the relationship between estimator accuracy and the number of samples can be determined empirically. As we consider other parameters (i.e. number of poles and mechanical rotation speed) to be fixed, the number of samples is entirely dependent on the chosen monitor period. As such, we perform a parameter sweep of the monitor period and evaluate the resulting estimator accuracy in a test scenario in simulation. The resulting estimator accuracy is characterized using the mean (μ) and standard deviation (σ) of the error between the estimated load angle and the ground truth. Results are shown in Figure 9, which shows the mean error (μ) in blue and corresponding confidence band ($\mu \pm \sigma$) in red. This shows that the chosen monitor period significantly impacts the standard deviation, but not the mean error. In this paper, we do not explicitly consider the memory footprint; consequently it is not characterized.

c) Pruning: The ontology of the estimator can be pruned quite easily. First, as we consider the number of motor poles and mechanical rotation speed to be fixed, we can prune these parameters as well as the electrical rotational speed from the ontology. Second, as we characterized the path from *monitor period* to *estimator accuracy* at once, we replace it with a single relationship, characterized by the results shown in Figure 9. The resulting (pruned) ontology is shown in Figure 10.

2) Controller:

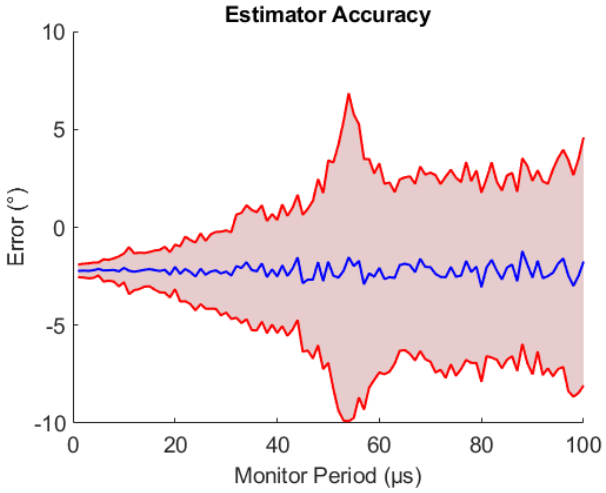

Fig. 9. Load angle estimator accuracy in function of chosen monitor period.

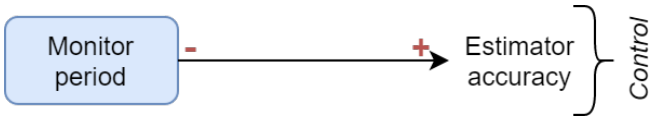

Fig. 10. Simplified ontology of the estimator.

a) Ontology definition: The ontology of the controller is shown in Figure 11. Design parameters are shown on the left, intermediate properties in the middle, and objectives on the right. The controller's **design parameters** include the control period and load angle setpoint, which are explicitly chosen by the control engineer, but also the estimator accuracy, which depends on the chosen estimator configuration, as discussed in the previous subsection. The only **intermediate property**, in this case, is the control performance. **Objectives** include power consumption (from the system level ontology) and stability in the control domain and execution time in the embedded domain.

b) Characterization: In this case, none of the relationships can be readily described as exact mathematical relationships. While we expect the power consumption to be mainly proportional to $1/sin(\delta)$ (Equation 7), this will also be affected by the actual control performance. However, we

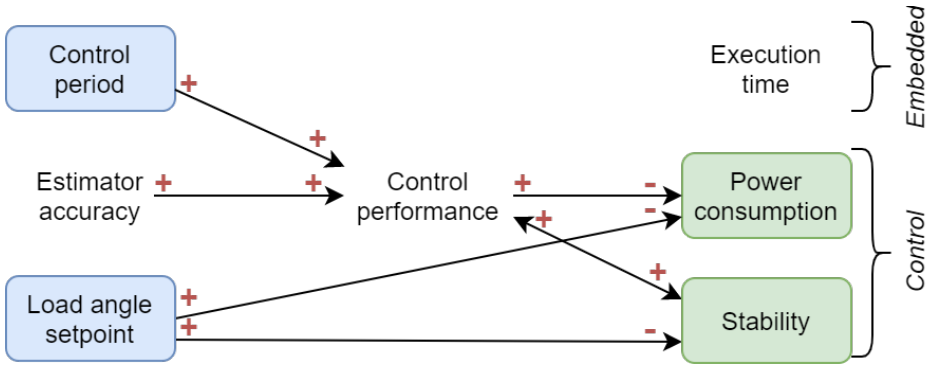

Fig. 11. Ontology of the controller.

		Control performance		Power consumption	Stability
		Mean	*Std.*		
Control period		0.002	0.001	0.017	0.000
Estimator accuracy	***Mean***	0.000	0.001	0.005	0.000
	Std.	**0.111**	**0.133**	**0.043**	**0.131**
Load angle setpoint		**0.451**	**0.431**	**0.304**	**0.422**

TABLE I
SENSITIVITY ANALYSIS RESULTS OF THE CONTROLLER.

can add sensitivity information to the relationships by performing sensitivity analysis. This is described in the following paragraph. Lastly, as with the estimator, none of the design parameters affect the execution time.

We perform sensitivity analysis on the controller to add sensitivity information to the relationships between design parameters, and intermediate properties and objectives. Here, we vary the different parameters and evaluate each parameter's contribution to observed variations in intermediate properties and objectives in simulation. The simulation setup includes the controller, a plant model, and an approximation of the estimator. The estimator is approximated by adding noise to the ground truth load angle obtained from the plant model. As such, we can freely vary the mean and standard deviation of the noise to evaluate their impact. We characterize control performance using the mean error and corresponding standard deviation of the actual load angle compared to the setpoint. Power consumption is defined as the average power consumption of the motor during a simulation run, while stability is defined as *pass/fail*, which indicates if the motor stalled during simulation. Results are summarized in Table I, which shows the first-order sensitivity index of each parameter for each intermediate property or objective. This first-order sensitivity index indicates to what extent introduced variations in each parameter contribute to variations observed in each property/objective. As such, it indicates how sensitive a property/objective is to changes in each parameter, relative to every other parameter, with a higher index indicating a higher sensitivity. The most significant sensitivity indices are shown in bold.

The sensitivity analysis results show that both the control performance and stability are most sensitive to changes in the load angle setpoint, but also to the standard deviation of the estimated load angle. Similarly, the power consumption is most sensitive to changes in the load angle setpoint, but less to changes in estimator accuracy. Interestingly, variations in the control period have little to no effect on the considered properties and objectives, at least not for the considered range of this parameter. This, coupled with the fact that estimator accuracy depends entirely on the chosen monitor period, indicates that engineers need to prioritize a lower monitor period over a lower control period during deployment to achieve the best system-level performance.

Additionally, as we now know that the stability is mostly dependent on the load angle setpoint and estimator accuracy (standard deviation), we reuse the simulation results from the sensitivity analysis to analyze the stability in function of these two parameters. This is shown in Figure 12, which shows valid (stable) and invalid (unstable) configurations in function of these two parameters. This shows that overall there is a clear boundary between stable and unstable configurations. This is valuable information for the engineers as it can be used to determine the maximum allowed standard deviation on the load angle estimate, and subsequently, the maximum allowed monitor period (Figure 9), based on the desired load angle setpoint. Or, conversely, the maximum setpoint that can be chosen, based on the chosen monitor period. However, we do see some unstable configurations surrounded by stable configurations near the boundary for higher setpoints. As such, a safety margin will likely need to be taken into account when making the trade-off between estimator accuracy and load angle setpoint.

Lastly, as we see that the average power consumption is mostly dependent on the chosen load angle setpoint, we reuse the simulation results to investigate this relationship further. Figure 13 shows the average power consumption for different

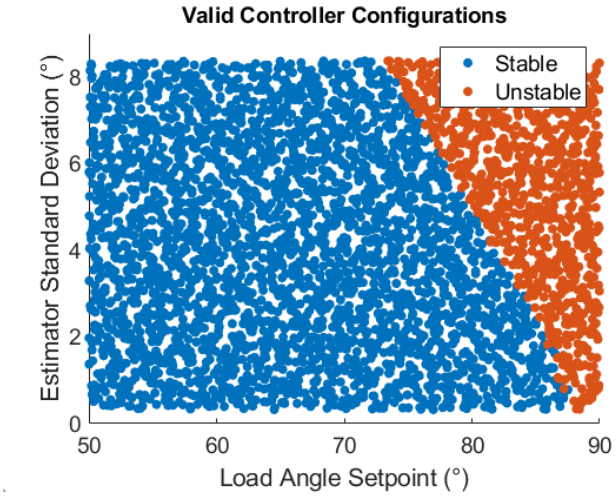

Fig. 12. Stable and unstable controller configurations in function of estimator accuracy and load angle setpoint.

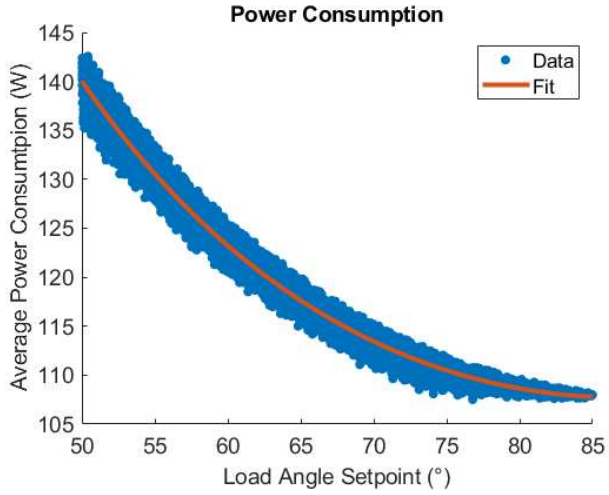

Fig. 13. Average power consumption for different controller configurations.

controller configurations in function of the load angle setpoint. By fitting a curve to the simulation data, we do see that the power consumption is mainly proportional to $1/sin(\delta_{setpoint})$ as we would expect. Deviations from this curve are likely due to variations in control performance or other effects we may have missed.

c) Pruning: The different results from the characterization process are used to simplify the controller ontology. First, as we see no properties or objectives are sensitive to changes in *control period*, all relationships originating from this parameter are pruned. Second, all paths between design parameters and objectives have been characterized at once during the sensitivity analysis. As such, they are simplified using single relationships with their corresponding sensitivity indices. This is shown in Figure 14. Additionally, the stability analysis results can be used to constrain which combinations of *estimator accuracy* and *load angle setpoint* are valid. Similarly, the fitted curve shown in Figure 13 can

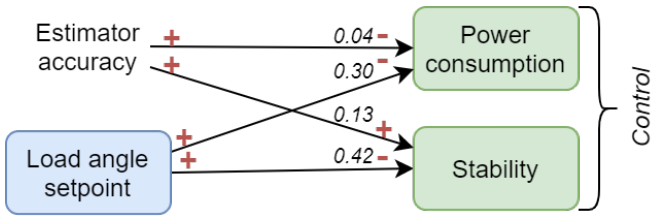

Fig. 14. Simplified ontology of the controller.

be used to provide an approximate mathematical description of the relationship between load angle setpoint and power consumption.

3) Watchdog:

a) Ontology definition: Lastly, the ontology for the watchdog is shown in Figure 15. Again, design parameters are shown on the left, intermediate properties in the middle, and objectives on the right. The watchdog's **design parameters** include the monitor period and load angle threshold, which are explicitly chosen by the control engineer, but also the estimator accuracy, which depends on the chosen estimator configuration, as discussed previously. The only **intermediate property**, in this case, is the watchdog response. **Objectives** include power consumption (from the system level ontology) and stability in the control domain and execution time in the embedded domain.

b) Characterization: None of the relationships can be readily described exactly. However, we can again add sensitivity information to the relationships by performing sensitivity analysis. This is described in the following paragraph. Lastly, as with the estimator and controller, none of the design parameters affect the execution time.

Similar to the controller, we add sensitivity information to the relationships between the different parameters, properties, and objectives by performing a sensitivity analysis on the watchdog. We do this by varying the different design parameters within a certain range and subsequently evaluating their contribution to observed variations in intermediate properties and objectives in simulation. The simulation setup consists of the watchdog, a plant model, and an approximation of the estimator. In this case, the estimator is approximated as $\delta_{est} = \delta_{GT} + \mu - n\sigma$, where δ_{est} is the (approximated) estimated load angle, δ_{GT} is the ground truth load angle obtained from the plant model, and μ and σ are the mean estimator error and standard deviation respectively. As such, the approximation of the load angle represents a worst-case scenario for the watchdog. The factor n is chosen by the designer based on the requirements. In this paper, we consider $n = 1$. Watchdog response is defined as the time between the ground truth load angle passing the threshold and the watchdog responding based on the estimated load angle. The results of the sensitivity analysis are summarized in Table II, which shows the first-order sensitivity index of each parameter for each intermediate property or objective. The most significant sensitivity indices are shown in bold.

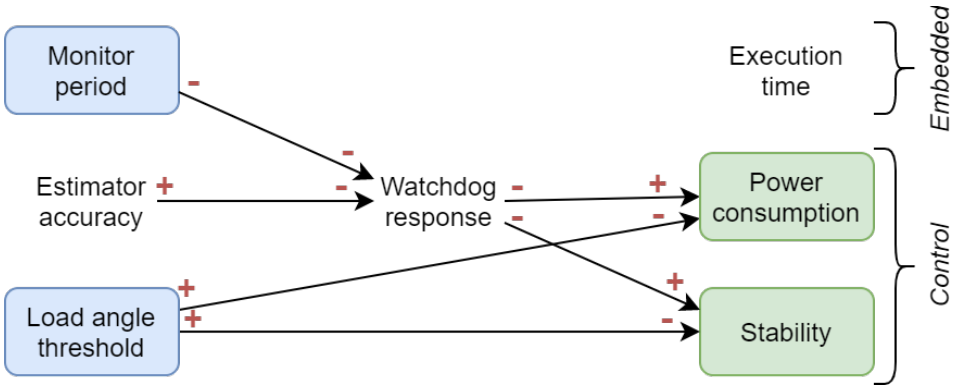

Fig. 15. Ontology of the watchdog.

		Watchdog response	Power consumption	Stability
Monitor period		0.000	0.000	0.000
Estimator accuracy	***Mean***	0.032	0.003	0.004
	Std.	**0.666**	**0.028**	**0.032**
Load angle threshold		**0.269**	**0.816**	**0.836**

TABLE II
SENSITIVITY ANALYSIS RESULTS OF THE WATCHDOG.

The results of the sensitivity analysis show that the watchdog response is very sensitive to changes in estimator accuracy (specifically the standard deviation), but also sensitive to changes in the load angle threshold. The results also show that both the power consumption and stability are very sensitive to changes in the load angle threshold, but much less to changes in estimator accuracy (standard deviation). From these results, we conclude that, when considering the watchdog, choosing a good threshold is most important to guarantee stability, while estimator accuracy is less important. As such, we further investigate the stability of the system in function of these two parameters. This is shown in Figure 16, which shows the stable and unstable configurations of the watchdog. Similarly to the controller, we see a clear boundary between the stable and unstable configurations. As such, this information can be used to make informed decisions when setting the load angle threshold.

c) Pruning: The results from the characterization process are used to simplify the watchdog ontology. First, as we see no properties or objectives are sensitive to changes in monitor period, all relationships originating from this parameter are pruned. Second, all paths between design parameters and objectives have been characterized at once during the sensitivity analysis. As such, they are simplified using single relationships with their corresponding sensitivity indices. This is shown in Figure 17. Additionally, the stability analysis results can be used to determine which combinations of estimator accuracy and load angle threshold are valid.

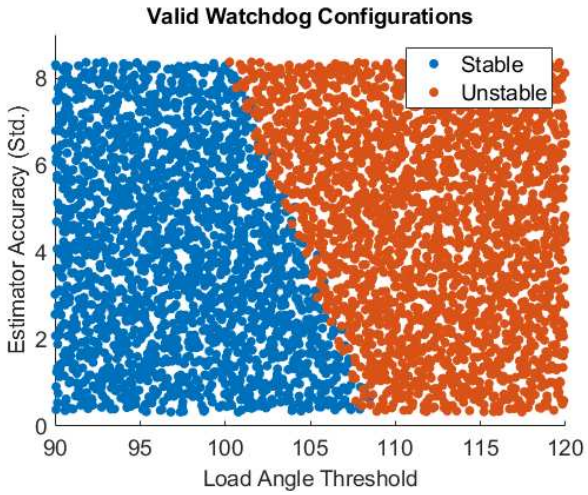

Fig. 16. Stable and unstable watchdog configurations.

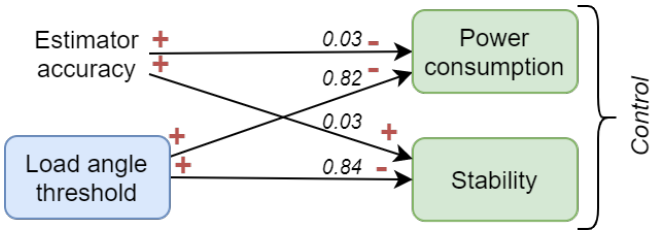

Fig. 17. Simplified ontology of the watchdog.

C. Updated Ontology (Lifting)

The information contained in the component-level ontologies is *lifted* to the system-level, where it is used to update the system-level ontology. The resulting updated ontology is

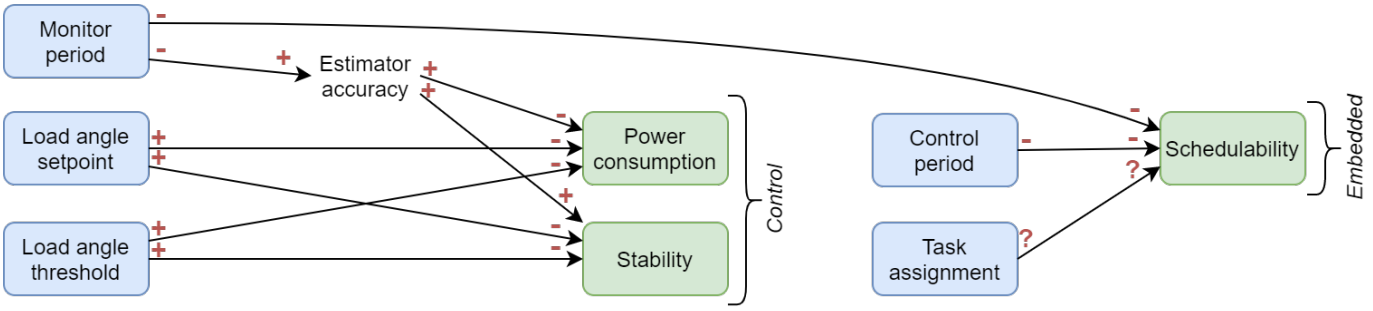

Fig. 18. Updated system-level ontology.

shown in Figure 18. By reasoning about this updated ontology, we can refine and extend the conclusions from Subsection V-A regarding design space exploration as follows:

1) The schedulability in the embedded domain constrains the design space in the control domain. As the schedulability can be evaluated quickly, this means we can quickly evaluate and discard infeasible combinations of monitor and controller periods. *As such, schedulability analysis should be performed as early as possible in the design space exploration process.*
2) In general, "faster is better". From the ontology, we see that, in general, a shorter monitor period should lead to better energy efficiency and stability. However, this does decrease schedulability. Additionally, having a shorter control period is less important for overall system performance. *As such, we should prioritize a shorter monitor period over a shorter control period.*
3) It is not useful to have a controller period shorter than a monitor period because the controller uses the load angle estimated by the monitor as feedback. However, having a shorter monitor than controller period could be beneficial. *As such, the monitor period should be shorter than or equal to the control period.*

Additionally, results from the different characterization processes can be used to aid the design space exploration process:

4) The results of the stability analysis of the controller can be used to constrain which combinations of estimator accuracy and load angle setpoint are valid. *As such, once a monitor period has been chosen, the maximum allowed load angle setpoint for the controller can be determined.*
5) The results of the stability analysis of the watchdog can be used to constrain which combinations of estimator accuracy and load angle threshold are valid. *As such, once a monitor period has been chosen, the maximum allowed load angle threshold for the watchdog can be determined.*

These conclusions are used in the following subsection to guide the design space exploration process and to constrain the design space.

D. Design-Space Exploration Workflow

If we consider the design space in the control domain as presented in Subsection IV-B, assuming a granularity of 1μs for the periods and 1° for the load angle setpoint, this leads to 36 possible values for each period, 46 for the load angle setpoint, and 31 for the load angle threshold, or $36 \cdot 36 \cdot 46 \cdot 31 = 1,848,096$ possible design candidates in total. A single Simulink simulation to evaluate a design candidate's power consumption and stability takes between 2.5 and 3 minutes on a 2.7GHz Xeon server.However, the evaluation can easily be parallelized. As such, an exhaustive search of this design space would take on average $((36 \cdot 36 \cdot 46 \cdot 31)/48) \cdot 2.75$ minutes $=$ 105,881 minutes, or just over 73.5 days, on a 48-core server. As this would make a comparison between our results and an exhaustive search impractical, we consider the load angle threshold to be fixed at 100°. This reduces the design space to $36 \cdot 36 \cdot 46 = 59{,}616$ design candidates. Consequently, an exhaustive search would take on average 3,416 minutes, or just under 57 hours on a 48-core server.

Using the previous section's conclusions, we can significantly reduce the required computation time in a couple of steps.

1) Schedulability Analysis: The *first conclusion* from Subsection V-C states that schedulability analysis (embedded domain) can be used to constrain the design space in the control domain. As none of the design parameters impact the actual execution time, timing analysis is performed to characterize the execution time of the different algorithm sections on the considered embedded platform. This information is then used to determine the schedulability of the different configurations (periods).

In the example case, we consider both a single- and dual-core version of the embedded platform. As there are six algorithm sections, this results in $2^6 = 64$ possible task mapping/partitioning schemes. These are all evaluated regarding achievable monitor and controller period. We consider a variation on cyclic executive scheduling [29], where we take into account possible synchronization between the two cores to allow for the parallelization of algorithm sections. The schedulability analysis is performed by automatically constructing an execution timeline based on a directed graph that captures the dependencies between algorithm sections, corresponding execution time measurements, and a candidate task mapping/partitioning scheme. This is then used to evaluate, among other things, resource utilization and to determine

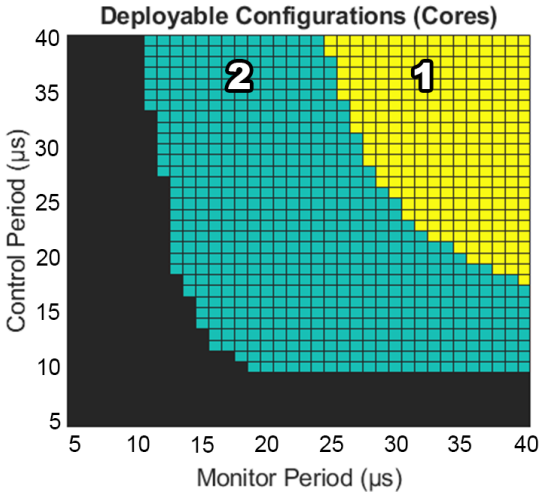

Fig. 19. Deployable configuration on both single-core (1) and dual-core (2) platforms.

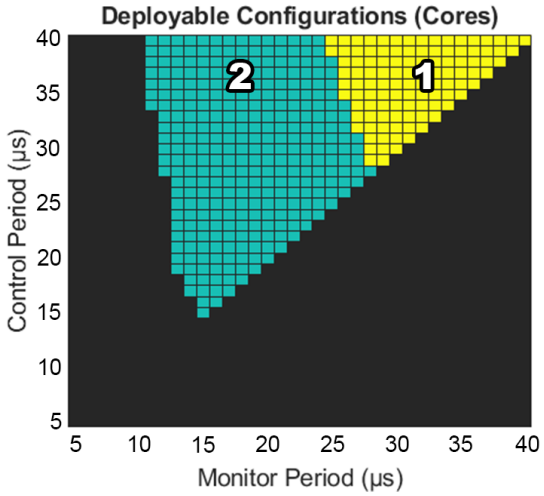

Fig. 20. Deployable configuration on both single-core (1) and dual-core (2) platforms (constrained).

the feasibility of the candidate scheme.

As this evaluation can be performed quickly, all combinations are evaluated in less than 60 seconds of computation time. The results of this schedulability analysis are shown in Figure 19, where the colours indicate a feasible dual- or single-core configuration. Unfeasible configurations are shown in black. Note that this only shows the different periods. As such, each square still represents 46 possible load angle setpoints.

This also shows that by performing the schedulability analysis first, the design space is quickly reduced from 59,616 to 39,698 candidates (around two thirds of the initial design space). Subsequently, the time needed to further validate the remaining combinations regarding energy efficiency is reduced to just under 38 hours on the same server.

2) $T_{monitor} \leq T_{control}$: The *third conclusion* states that, from a control standpoint, having a control period shorter than the monitor period is not useful due to the way the system works. This adds an additional constraint that is used to further reduce the design space. By discarding all configurations that violate this constraint, the design space is further reduced by about half, resulting in a design space with a size of 19,044 candidates, or under one third of the original. This is illustrated in Figure 20. Further evaluation of the remaining configurations would take just over 18 hours. However, this still leaves a lot of possible configurations. We introduce more domain knowledge to further limit the design space in the next step.

3) "Faster is Better": The *original second conclusion*, as presented in Subsection V-A, states that, in general, a smaller monitor and control period should lead to better energy efficiency. However, it does not tell us anything about the optimal ratio between these periods. For these reasons, the design space is further reduced to two Pareto fronts of options at the edge of the viable regions. This reduces the design space to a fraction of the original. The 1,748 remaining configurations can be seen in Figure 21. Further evaluation

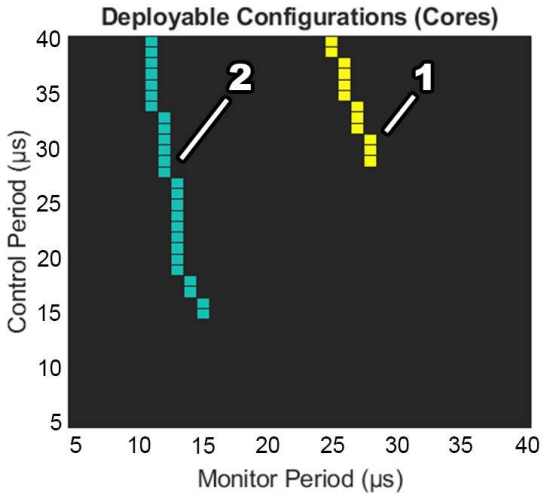

Fig. 21. Deployable configuration on both single-core (1) and dual-core (2) platforms (fronts).

of these candidates would take under 2 hours.

By updating the ontology with new information obtained during the development process, we were able to further refine this conclusion. The *updated second conclusion*, as presented in Subsection V-C, states that not only a shorter period is better, we should prioritize monitor period over control period. As such, the design space is further reduced to the subset of candidates shown in Figure 22. This remaining subset contains 414 candidates or around 0.69% of the original design space. Further evaluation of the remaining configurations takes about 24 minutes of computation time on a 48-core Xeon server.

4) Maximum load angle setpoint and threshold: For each remaining design candidate, a parameter sweep of the load angle setpoint is performed to determine the best configuration, i.e. lowest power consumption while maintaining system stability, in simulation. However, the results from the characterization processes shown in Subsection V-B can be

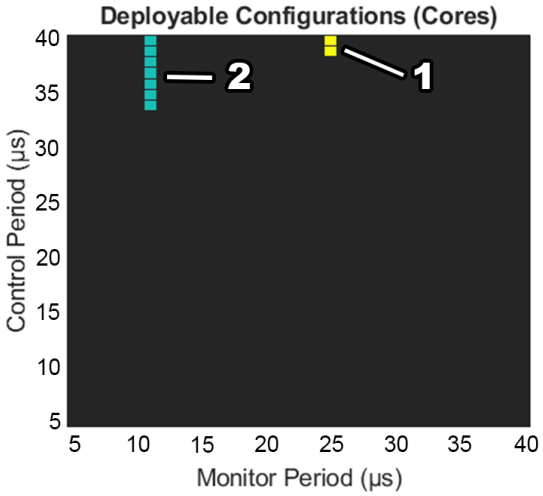

Fig. 22. Deployable configuration on both single-core (1) and dual-core (2) platforms, with monitor period prioritized.

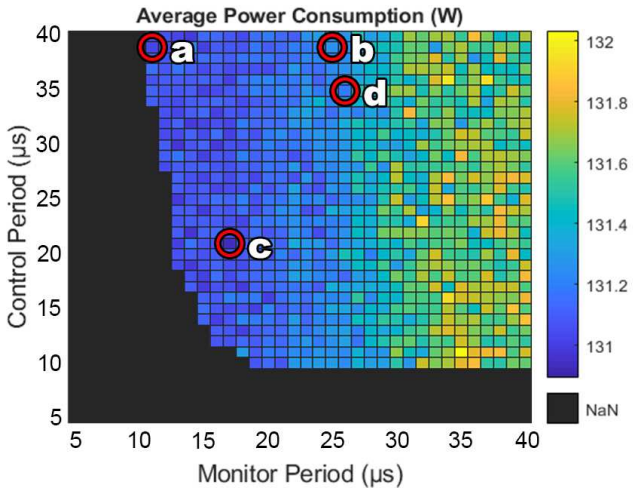

Fig. 23. Average power consumption for all deployable configurations.

used to further reduce the number of required evaluations (*conclusion 4*). Indeed, the results shown in Figure 9 can be used to determine the expected estimator accuracy based on the chosen monitor period. For example, a monitor period of $10\mu s$ would result in an expected estimator standard deviation of $0.737°$. After this, the stability analysis results shown in Figure 12 can be used to determine the maximum allowed load angle setpoint. For an expected estimator standard deviation of $0.737°$, this would be $87°$. As the ontology shows that a higher load angle setpoint should lead to lower power consumption, this maximum setpoint can be used as an initial best guess, evaluated in simulation, and if needed, lowered until a good, stable configuration is found. As such, each combination of monitor and control period would require only a small number of evaluations to determine the best possible configuration.

Similarly, the stability analysis results shown in Figure 16 can be used to determine the maximum allowed load angle threshold for the watchdog (*conclusion 5*).

VI. Results

In the previous section, we showed how domain knowledge can be used to steer the design-space exploration process. This allows us to quickly reduce the design space to a subset of likely good candidates. In doing so, the design space is reduced from 59,616 possible configurations to just 414. However, for this method to be useful, it needs to (i) provide good solutions and (ii) it needs provide benefits over traditional design space exploration techniques. To evaluate these aspects, we first compare the results obtained using our approach to the global optima found using an exhaustive search in Subsection VI-A. After this, in Subsection VI-B we compare our approach to traditional design space exploration techniques regarding the quality of the solutions and the computation time. Lastly, in Subsection VI-C, we list potential threats to the validity of our results.

A. Quality of the Solutions

The entire subset of deployable configurations was further evaluated using the method described in Subsection IV-C. The results of this exhaustive search are shown in Figure 23. This figure shows the average energy consumption at the optimal load angle setpoint for each combination of control and monitor period. Unfeasible (i.e. not deployable) configurations are shown in black. (Note that the range here differs from that in Figure 13 as a more complex test sequence is considered here, where the average load of the motor is higher). In this figure, we discern four different configurations:

- a The best performing dual-core solution determined using our ontology-based method, with an average power consumption of 130.88W
- b The best performing single-core solution determined using our ontology-based method, with an average power consumption of 131.16W
- c The best performing dual-core solution determined using the exhaustive search, with an average power consumption of 130.79W
- d The best performing single-core solution determined using the exhaustive search, with an average power consumption of 131.07W

By comparing these different configurations, we see that while our method does not find the top-performing (deployable) candidates, it does find candidates with very similar performance much more quickly. However, we also see that the best performing candidates are local minima, surrounded by slightly worse performing ones, indicating that these solutions might not be entirely stable. This is not the case for the candidates determined using our method.

These results also show that indeed, a shorter monitor period leads to better overall performance, while the control period has little to no effect, as we see a clear gradient from left to right. This indicates that *conclusion 2*, as listed in Subsection V-C, is likely correct. Additionally, this shows

		Single-Core				Dual-Core			
		Evaluations		Power (W)		Evaluations		Power (W)	
		Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
Exhaustive Search		11960		131.07		39698		130.79	
Particle Swarm Optimization		945	214	131.20	0.0982	1019	230	130.83	0.0352
Simulated Annealing		1498	610	131.19	0.0789	2378	833	130.85	0.0347
Genetic Algorithm		2582	184	131.38	0.3585	2610	145	130.95	0.1535
Ontology-Based	**Before SA**	552		131.16		1196		130.88	
	After SA	92		131.16		322		130.88	
	Best guess	4		131.16		14		130.88	

TABLE III
NUMBER OF EVALUATIONS AND SOLUTION PERFORMANCE FOR DIFFERENT DSE APPROACHES.

that configurations where the control period is shorter than the monitor period don't necessarily perform poorly. However, they do not necessarily perform better either (*conclusion 3*). This is in contrast with results in a previous publication [19], which showed that configurations with a shorter control period than monitor period generally performed poorly or were unstable. This is because a newer iteration of the controller was used in this paper, which is overall more stable.

In Subsection V-D4, we discussed how the results from the characterization steps can be used to obtain an initial best guess for the load angle setpoint. This was validated by comparing the initial best guess for each candidate with the corresponding optimal load angle setpoint found during the exhaustive search. The results of this analysis show that, on average, the best guess was less than 1° higher than the actual optimal setpoint (optimistic), with differences ranging from -1° (pessimistic) to 3° (optimistic). As such, if we start with the initial best guess, only a small number of evaluations are required to find the optimal load angle setpoint, further significantly reducing the design space.

B. Computation Time

To further evaluate the practicality of our approach, we compare it to traditional design space exploration techniques regarding the required computation time, but also the performance of the solutions. Here, we consider particle swarm optimization (PSO), simulated annealing (SA) and genetic algorithms (SA) to perform the optimization. The design variables considered here are the same as in Subsection V-D, i.e. monitor period, control period and load angle setpoint.

The evaluation function used by each algorithm first evaluates the schedulability of a candidate. Only if the candidate is schedulable, the system-level performance of the candidate is evaluated. As a metric for the required computation time for each algorithm, we count the number of (system-level) performance evaluations during the exploration as these take the most significant amount of time. As such, unschedulable candidates are not counted. Additionally, as the considered DSE algorithms introduce randomness in their search processes, we run each algorithm 1000 times and calculate the average number of performance evaluations and average power consumption of the solutions. To keep this evaluation practical, we use the results of the exhaustive search from the previous subsection to construct a look-up-table of the performance of each design option. This is then used instead of running the simulation for each evaluation. The mean number of model evaluations and achieved power consumption, as well as the corresponding standard deviations, for each technique are shown in Table III. In this table, we explicitly consider three different situations for our approach. First, using only the conclusions presented in Subsection V-A, i.e. before performing the sensitivity analyses ("Before SA"). Then, using the updated conclusions as presented in Subsection V-C, i.e. after performing the sensitivity analyses ("After SA"). Lastly, using the initial best guess, as discussed in Subsection V-D4 ("Best guess"). We make this distinction as these different situations require different amounts of effort and computation time upfront, i.e. before the design space exploration can be performed. Note that our method does not introduce randomness in its search strategy. As such, no standard deviation is listed. Additionally, the results of the exhaustive search are listed for reference.

The results show that our ontology-based approach finds, on average, similarly performing solutions as the traditional approaches. It does so for both single- and dual-core systems. When comparing the number of evaluations, we see that, even without using the sensitivity analysis results, our approach outperforms the other techniques for the single-core system. When making use of the sensitivity information, it requires an order of magnitude fewer evaluations to find a good solution. This is again reduced by an order of magnitude when making use of the initial best guess for the load angle setpoint. For the dual-core system we see similar results. However, our approach, without the sensitivity information does perform slightly worse than the particle swarm optimization for this case, requiring $\sim$10% more evaluations to find a good solution. However, after adding the sensitivity information, it again requires significantly fewer evaluations. This is further reduced by using the initial best guess. Additionally, by examining the solutions found using the traditional approaches, we found that they often provided solutions that were local minima, i.e. well performing solutions surrounded by worse performing ones. This indicates that these solutions might not be entirely stable. This is again not the case for solutions found using our approach.

C. Threats to Validity

As the work presented here is essentially design research, it is important to consider possible threats to the validity of

the obtained results [30]. Possible threats are listed below. We intend to address these in future work.

a) Internal Validity: asks the question how sure we can be that the presented approach caused the observed outcome. In this regard, there are two main factors we need to take into account. Firstly, in Subsection VI-B, we used the number of performance evaluations as a metric to compare the expected computation time for the different algorithms. However, the extent to which these evaluations can be performed in parallel for the different approaches can affect these results. For example, for the particle swarm optimization, each particle can be evaluated in parallel, as such the number of parallel evaluations depends on the number of particles. Similarly, for the genetic algorithm, the number of parallel evaluations depends on the population size. Simulated annealing however cannot easily be parallelized. For our ontology-based approach, as presented here, there is theoretically no limit to the number of evaluations that can be performed in parallel, as they are completely independent. As such, our approach should still outperform the other techniques. Secondly, in the current paper, we rely on simulation results to evaluate the performance of the design candidates. How well they actually perform in a real-world setup remains to be seen.

b) External Validity: is concerned with whether we can generalize the results outside the scope of the study. In this regard, the scope of the presented use case is still quite limited. As such, it remains to be seen how well the approach scales to more complex use cases, and to industrial settings, where there may be large teams of engineers from multiple domains working on the same project.

VII. Discussion and Future Work

This paper shows how cross-domain knowledge can be used to solve a multi-domain optimization problem. By building an ontology of important design parameters and their interdependencies, it is possible to reason about the design space exploration process. By reasoning about this process, using this ontology, it becomes clear how evaluation or information in one domain can be used to constrain parameters in another domain. As such, the design space can quickly be narrowed down to a subset of likely candidates using the relationships between parameters in both domains. This is shown using an example case where the design space is quickly limited to $\sim$0.69% of its original size, while still finding solutions that perform similarly to the best possible candidates. Additionally, the presented approach requires significantly less computation time than traditional DSE techniques to find similarly performing solutions.

The results shown are promising, especially when keeping in mind the development of new, advanced cyber-physical systems. However, the design space for the case study presented in this paper is still quite limited. As such, it remains to be seen how well this approach scales to more complex cases. In future work, we intend to extend the work presented here in three main ways:

a) Modelling of Domain Knowledge: In the current paper, the ontology is built up manually and only exists as a drawing. As previously mentioned, Vanherpen et al. [17] worked to combine contract-based design (CBD) [31] with ontological reasoning. This resulted in, among other things, an ontology specific to the co-design of cyber-physical systems, modelled in tools such as Protégé [32]. In our future work, we will investigate how this existing ontology can be extended specifically to support the approach presented in this paper. We will investigate which parts of the domain knowledge can be captured in a generic way and which parts are application-specific. Additionally, in Subsection V-A, we introduced the concept of an ambiguous relationship. How to deal with such relationships is still an open question.

The third conclusion from Subsection V-A ($T_{monitor} \leq T_{control}$) is interesting as it is not directly apparent from the ontology itself. With this in mind, we will investigate how this kind of information might be added to the ontology or otherwise modelled. For example, this might be done by introducing additional relationships that explicitly capture such constraints.

b) Automation: The method, as presented in this paper, requires manual work to build and update the ontology and to derive the design-space exploration process. For this reason, we will investigate to what extent this process can be derived automatically from the captured domain knowledge, for example, by detecting specific patterns in the ontology and providing hints to the engineer which parameters should be fixed first or where possible trade-offs can be made. We also intend to investigate if this can be extended to the characterization process, for example, by using the ontology to determine which relationships need to be characterized and using this information to run a sensitivity analysis or other experiments.

c) Further Validation: We intend to further validate the presented approach to mitigate the threats to validity presented in Subsection VI-C. Firstly, while the load angle control algorithm itself has been validated in a real-world test setup, the optimized design candidates presented in this paper have only been evaluated in simulation. It remains to be seen how well these candidates perform in real-world situations. As such, we intend to further evaluate the performance of these candidates in a physical test setup. Secondly, while the presented approach shows promising results for the presented use case, the use case is still quite limited in scope. It remains to be seen how well the presented approach scales to more complex cases. As such, we will further develop and validate our approach using more complex use cases. For example, cases where there are multiple different algorithms that need to run on a shared embedded platform and where trade-offs need to be made regarding the performance of the different algorithms, which is a common problem in the design of cyber-physical systems.

ACKNOWLEDGMENTS

This research was supported by Flanders Make, the strategic research centre for the manufacturing industry in Belgium, with the Model-based Force Measurements (MoForM) project.

REFERENCES

[1] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1433–1440.

[2] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[3] J. De Viaene, F. Verbelen, S. Derammelaere, and K. Stockman, "Energy-efficient sensorless load angle control of a bldc motor using sinusoidal currents," *IET Electric Power Applications*, vol. 12, no. 9, pp. 1378–1389, 2018.

[4] S. Derammelaere, B. Vervisch, J. De Viaene, and K. Stockman, "Sensorless load angle control for two-phase hybrid stepper motors," *Mechatronics*, vol. 43, pp. 6–17, 2017.

[5] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli, "Metropolis: An integrated electronic system design environment," *Computer*, vol. 36, no. 4, pp. 45–52, 2003.

[6] A. Davare, D. Densmore, L. Guo, R. Passerone, A. L. Sangiovanni-Vincentelli, A. Simalatsar, and Q. Zhu, "Metro ii: A design environment for cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 1s, p. 49, 2013.

[7] A. Sangiovanni-Vincentelli and G. Martin, "Platform-based design and software design methodology for embedded systems," *IEEE Design & Test of Computers*, vol. 18, no. 6, pp. 23–33, 2001.

[8] W. Chang, S. Chakraborty *et al.*, "Resource-aware automotive control systems design: A cyber-physical systems approach," *Foundations and Trends® in Electronic Design Automation*, vol. 10, no. 4, pp. 249–369, 2016.

[9] L. Guo, Q. Zhu, P. Nuzzo, R. Passerone, A. Sangiovanni-Vincentelli, and E. A. Lee, "Metronomy: a function-architecture co-simulation framework for timing verification of cyber-physical systems," in *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*. ACM, 2014, p. 24.

[10] C. Ptolemaeus, *System design, modeling, and simulation: using Ptolemy II*. Ptolemy. org Berkeley, 2014, vol. 1.

[11] A. Aminifar, E. Bini, P. Eles, and Z. Peng, "Bandwidth-efficient controller: server co-design with stability guarantees," in *Proceedings of the conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2014, p. 55.

[12] ——, "Designing bandwidth-efficient stabilizing control servers," in *2013 IEEE 34th Real-Time Systems Symposium*. IEEE, 2013, pp. 298–307.

[13] D. Roy, L. Zhang, W. Chang, and S. Chakraborty, "Automated synthesis of cyber-physical systems from joint controller/architecture specifications," in *2016 Forum on Specification and Design Languages (FDL)*. IEEE, 2016, pp. 1–8.

[14] A. Cervin, M. Velasco, P. Marti, and A. Camacho, "Optimal online sampling period assignment: Theory and experiments," *IEEE transactions on control systems technology*, vol. 19, no. 4, pp. 902–910, 2010.

[15] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. A. Henzinger, K. G. Larsen *et al.*, "Contracts for system design," *Foundations and Trends® in Electronic Design Automation*, vol. 12, no. 2-3, pp. 124–400, 2018.

[16] J. Finn, P. Nuzzo, and A. Sangiovanni-Vincentelli, "A mixed discrete-continuous optimization scheme for cyber-physical system architecture exploration," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015, pp. 216–223.

[17] K. Vanherpen, J. Denil, I. Dávid, P. De Meulenaere, P. J. Mosterman, M. Torngren, A. Qamar, and H. Vangheluwe, "Ontological reasoning for consistency in the design of cyber-physical systems," in *2016 1st International Workshop on Cyber-Physical Production Systems (CPPS)*. IEEE, 2016, pp. 1–8.

[18] I. Dávid, J. Denil, K. Gadeyne, and H. Vangheluwe, "Engineering process transformation to manage (in) consistency," in *Proceedings of the 1st International Workshop on Collaborative Modelling in MDE (COMMitMDE 2016) co-located with ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2016) St. Malo, France, October 4, 2016/Muccini, Henry [edit.]; et al.*, 2016, pp. 7–16.

[19] Y. Vanommeslaeghe, J. Denil, J. De Viaene, D. Ceulemans, S. Derammelaere, and P. De Meulenaere, "Leveraging domain knowledge for the efficient design-space exploration of advanced cyber-physical systems," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*. IEEE, 2019, pp. 351–358.

[20] T. R. Gruber *et al.*, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, pp. 199–221, 1993.

[21] K.-y. Hwang, S.-b. Rhee, B.-y. Yang, and B.-i. Kwon, "Rotor Pole Design in Spoke-Type Brushless DC Motor by Response Surface Method," vol. 43, no. 4, pp. 1833–1836, 2007.

[22] S. Derammelaere, C. Debruyne, F. De Belie, K. Stockman, and L. Vandevelde, "Load angle estimation for two-phase hybrid stepping motors," *IET Electric Power Applications*, vol. 8, no. 7, pp. 257–266, 2014.

[23] K. Premkumar and B. V. Manikandan, "Adaptive Neuro-Fuzzy Inference System based speed controller for brushless DC motor," *Neurocomputing*, vol. 138, pp. 260–270, 2014.

[24] P. Vas, "Sensorless Vector and Direct Torque Control," *Monographs in Electrical and Electronic Engineering, Oxford University Press, USA*, vol. 1., p. 768, 1998.

[25] M. Bendjedia, Y. Ait-Amirat, B. Walther, and A. Berthon, "Position control of a sensorless stepper motor," *IEEE Transactions on Power Electronics*, vol. 27, no. 2, pp. 578–587, 2012.

[26] Y. Zhao, W. Huang, and J. Yang, "Fault diagnosis of low-cost Hall-effect sensors used in controlling permanent magnet synchronous motor," *19th International Conference on Electrical Machines and Systems (ICEMS)*, no. 1, pp. 2–6.

[27] J. C. Gamazo-Real, E. Vázquez-Sánchez, and J. Gómez-Gil, "Position and speed control of brushless dc motors using sensorless techniques and application trends," *Sensors*, vol. 10, no. 7, pp. 6901–6947, 2010.

[28] E. Jacobsen and R. Lyons, "The sliding dft," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 74–80, 2003.

[29] T. P. Baker and A. Shaw, "The cyclic executive model and ada," *Real-Time Systems*, vol. 1, no. 1, pp. 7–25, 1989.

[30] R. Feldt and A. Magazinius, "Validity threats in empirical software engineering research-an initial survey." in *Seke*, 2010, pp. 374–379.

[31] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. Henzinger, and K. G. Larsen, "Contracts for system design," Ph.D. dissertation, Inria, 2012.

[32] M. A. Musen, "The protégé project: a look back and a look forward," *AI Matters*, vol. 1, no. 4, pp. 4–12, 2015. [Online]. Available: https://doi.org/10.1145/2757001.2757003