Optimising Vehicular System Architectures with Real-time Requirements: An Industrial Case Study

Arman Hasanbegović*, Marcus Ventovaara†, Jimmie Wiklander†, Saad Mubeen*
*Mälardalen University, Sweden
†Volvo Construction Equipment, Sweden
*{ahc16002, saad.mubeen}@mdh.se
†{marcus.ventovaara, jimmie.wiklander}@volvo.com

***Abstract*—In time-constrained vehicular systems, optimisation of the system architectures with respect to the system resources has a significant impact on the development cost, performance, and reliability of the systems. This paper adapts the state-of-the-art methods to derive optimised system architectures in the vehicular industrial settings, while taking real-time constraints, resource requirements, communication dependencies, and design choices into account. In this regard, the paper describes and evaluates an industrial use case wherein the proposed method is applied to derive an optimised allocation of the system architecture. The paper also presents evidence and validations that suggest the viability of the method and its applicability in the vehicle industry.**

I. Introduction

Recently there has been a noticeable strive towards automation within the vehicle industry to develop advanced and efficient vehicles [1]. The push for automation does not only consider highly automated vehicles capable of manoeuvring in complex urban environments, but also to develop various driver assistance functionalities that can assess and alleviate risks and increase driver comfort. In the past, such functionalities would each have been allocated onto their own computer, in what is called a *federated architecture* [2]. However, with the availability of increasingly powerful processors, and the growing demand on communication induced partly by additional and more precise sensors, such schemes can no longer be justified. Therefore, in recent years, many vehicle developers have started moving to what is commonly denoted *integrated architectures* [2], [3], where one computer can accommodate multiple functionalities at any given time, as opposed to one functionality per hardware platform. Integrated architectures have many advantages over their federated counterparts, such as reduced system complexity and cost, lower demands on communication, and increased dependability [4].

Integrated architectures may be centralised or decentralised, in terms of the topology of the network that connects the system components. In the centralised topology, a master component communicates with the other components and deals with the distribution of functionality. A decentralised topology lacks this central component, and delegates the distribution of functionality between components to all nodes connected to the network. Some important aspects of the system architecture that can benefit from decentralisation are the real-time characteristics of the system. Parameters such as task response times and processor utilisation can be improved by optimised division of responsibility among the processors. Another major benefit of decentralised systems is the induced executional independence and ease of continued development. A new or updated subsystem can be verified and tested independently from the remainder of the system, reducing the complexity and scope. The integrated and decentralised vehicular system architecture is the main focus of this paper.

A. Problem Statement

There are several challenges in developing and supporting decentralised system architectures in vehicular computing systems. The architecture should remain integrated in terms of not isolating the hardware platforms from one another [2], and, how to identify the implications of decentralisation are two such challenges that must be dealt with. Knowing such information in an automation system can be crucial, especially in the segment of construction vehicles, where there are specific requirements on standardisation [5], [6] and productiveness from the manufacturer [7]. The system architecture and resource allocation requirements can be more constrained in this segment compared to the segment of high-end cars. For example, the number of Electronics Control Units (ECUs) in construction vehicles is around 10 depending on the size of the vehicle, whereas this number can be up to 100 in the segment of high-end cars [8]. Furthermore, certifying safety-critical parts of a system is a costly process and thus the implementational independence of safety-critical functionality is also important in this regard, especially considering cost. Making these design decisions and later providing the evidence needed for certification can be partly solved by using an automated optimisation method. If the high-level design requirements of a system architecture can be accurately defined and fed to one such method, the method itself could be used as evidence of the system architecture meeting such requirements.

As such, a robust method for architectural design, from which an adequate amount of computational platforms can be decided upon, and an optimal functional distribution across the platform, is needed. The method must be able to sufficiently address the real-time, safety and fault tolerance requirements set by the vehicle autonomy. There is no industry standard for solutions within this practice, and many solutions for automation are not benefited by pre-autonomous experiences. There are, however, various recently developed approaches that can be considered as guidelines when deriving optimised allocation of vehicular system architectures [9], [10], [11].

B. Paper Contribution

This paper adapts the state-of-the-art allocation methods [9], [10], [11] for vehicular system architectures with real-time

constraints, exerting special attention to the construction vehicle domain. The paper incorporates an industrial use case in cooperation with an established Original Equipment Manufacturer (OEM) of construction vehicles. The use case not only serves to compile some of the state-of-practice requirements related to the architectural design, but also provides a proof of concept for the method.

C. Paper Layout

The rest of this paper is organised as follows. Section II discusses the background and related work. Section III presents the method for the allocation of the system architectures. Section IV discusses the implementation of the method. Section V provides a proof-of-concept for the method on an industrial use case. Section VI presents the experimental evaluation. Section VII concludes the paper and discusses the future work.

II. Background and Related Work

The vehicle industry is striving towards automating vehicle operations. To achieve this, research is focused on challenges such as safety, predictability, and improving the overall operational performance [8]. The end goal is to achieve automated vehicle operation that outperforms human operation in every aspect. Current research is focused on moving the system architecture from a federated to an integrated design, reducing the number of ECUs and allowing one ECU to have more than one role in the overall system [2]. This trend was set by advances in avionics [3] and is now establishing itself in the vehicle domain as well. An integrated architecture places all the functionality onto a distributed computing system. The system can have one or multiple hardware platforms, as long as their connection allows uninterrupted computation while sharing roles and resources.

One way of systematically designing a system architecture is by formulating all the requirements as a mathematical model and then optimising the architecture based on some cost function. This type of systematic approach is known as the design space exploration [12]. Optimising the allocation of tasks on distributed systems has been heavily researched, although currently, the research has yet to produce a solution that is a viable candidate for an industry standard [9].

Currently, much of the research is focused on multi-objective evolutionary algorithms for deriving an optimised system allocation. Such research was performed in [13], where a Pareto Archived Evolutionary Strategy [14] was used for multi-objective optimisation in task allocation on a real-time system. The method described by Švogor and Carlson [15] is based on an analytic heuristics process to perform multi-objective optimisation of the distribution space. The mapping of the software components in the cited work is also performed using the evolutionary algorithms. The work in [16] proposes a framework for the allocation of control software functions to a distributed system with an objective of optimising the network load. In comparison, the method proposed in this paper optimises the system architectures with respect to communication as well as computation resources in a possibly distributed system while meeting the specified timing requirements.

In [17], a multi-objective genetic algorithm (Non-dominated Sorting Genetic Algorithm II) is used to solve the task of allocating software components onto ECUs within an automotive architecture. This work focuses on the load on communication bus, memory utilisation and the delivery times of signals in the communication bus as the main restrictions for the optimisation. The authors developed a framework that produces both hardware and software architectures which are optimised based on the input requirements. The results show that this method is very effective at optimising small-scale architectures although the method exhibits inconsistencies when the size of the architectures grow further. Conclusively, the authors state that input from the vehicle industry would be beneficial for any related future research.

A tool designed following the model-driven paradigm and platform-based design principles is presented in [18]. The tool is used to transform initial platform-independent software component specifications into a model which is platform-specific and is used to integrate software components of mixed criticality onto one common computing platform. The approach combines dependability and real-time requirements and optimises the architecture with restrictions regarding error propagation and schedulability analysis, among other criteria.

The work in [10] proposes a design-space exploration approach that optimises topologies, process bindings, and communication routing of a system. It uses multi-objective optimisation to find a solution and provides a use case of the approach for a Motion-JPEG decoder application on a multimedia hardware architecture. Design-space exploration plays an important role in the vehicle industry as optimised integrated architecture designs and allocations can significantly help to alleviate complexity and reduce cost [4].

The work in [11] proposes a two-step optimization approach for the placement of system architectures in distributed systems. It utilises MILP and genetic algorithms to achieve the optimized architectures. In comparison, this paper simplifies the method in [11] to a single step in deriving optimal allocations of system architectures in the construction equipment vehicles domain, while taking real-time constraints and resource requirements into account. Since many such constraints can be defined mathematically, the various works discussed in this section have relation to the paper in the sense that they explore different means of design-space exploration in order to optimise the system architectures.

III. Optimised Allocation of the System Architectures

This section presents the proposed method to derive optimised vehicular system architectures while taking the specified real-time constraints into account.

In order to sufficiently guarantee that no resource in the system ends up over or under-allocated, certain characteristics such as the Worst-case Execution Time (WCET) and periodicity of tasks or *software functions* must be assessed. Note that this paper considers a software function as a synonym for a software component, which is a well-defined term in component-based software engineering [19]. Moreover, the paper considers a one-to-one mapping between a software function (design-time entity) and a task (run-time entity), which is in-line with many component models and modelling languages for vehicular embedded systems [8], [20]. Hence, the terms “software function” and “task” are used interchangeably in this paper. The WCETs, periods, and deadlines of

tasks can further be used in a real-time schedulability analysis which, in turn, can be used to provide information on the schedulability of each subsystem, and the system in its entirety.

The proposed method consists of three steps, visualised in Figure 1, that all need to be passed in order to guarantee an optimised allocation of the system architecture. The first step is an assessment of the functionality, i.e., the WCET, periodicity, and communication dependencies for each software function in the system are assessed. These are the necessary preconditions that allow a set of tasks to be optimally allocated across a set of resources. Secondly, the variables defined previously are used to formulate a set of linear constraints and an objective to optimise towards in the step of design-space exploration. The third and final step utilises the initial variables and performs an early verification of the system resources by means of the schedulability analysis [21].

Extrapolating on the first step, the design-space exploration method proposed in this paper is indifferent to any methods used for WCET estimation. Any means with which WCETs can be estimated have relevance, such as statistical methods [22], or flow analysis as performed by the Swedish Execution Time tool (SWEET) described in [23]. A statistical method based on extreme value theory, such as the one described in [22], further guarantees that the WCET estimate is only exceeded by future invocations at the same rate as the exceedance probability. Therefore, this paper proposes the use of statistical extreme value theory methods since a comparison between such and SWEET suggest them to be sufficiently precise while gaining generality and applicability over their static counterparts [24]. Periodicity and communication dependencies should be assessed by manual analysis of the existing codebase. Continuing the extrapolation of each step, the second step utilises such metrics to optimise the allocation of a system architecture. Accounting for such metrics and systematically mapping processes onto multiple resources is done by applying methods for design-space exploration. The idea is to formulate the mapping problem as an Integer Linear Program (ILP), which can then be solved using an ILP solver. The problem is defined as:

$$\min_{\mathbf{x}} \; f^T \mathbf{x} \tag{1}$$

$$\begin{aligned} A\mathbf{x} &\leq b \\ A_{eq}\mathbf{x} &= b_{eq} \\ \mathbf{x} &\geq 0 \\ \mathbf{x} &\leq 1 \\ \mathbf{x} &\in \mathbb{Z}. \end{aligned} \tag{2}$$

Expression 1 consists of the objective function, f, and the vector, x, containing all the system variables. Equation set 2 consists of all the linear constraints that define the problem, i.e., equations, inequations, and constraints limiting the system variables to only assume binary values 0 and 1. A and A_{eq} are matrices of coefficients, b, b_{eq}, and f are vectors of coefficients, while $\mathbb{Z}$ is the set of all integers. These equations can be interpreted as a linear minimisation problem where the variables are binary and are subject to linear constraints. The linear constraints and objective function are then tailored to model the specifics of the architecture being analysed. The linear constraints are dictated not only by the conventions and logic of the implementation set but also by the real-time characteristics of the individual functions.

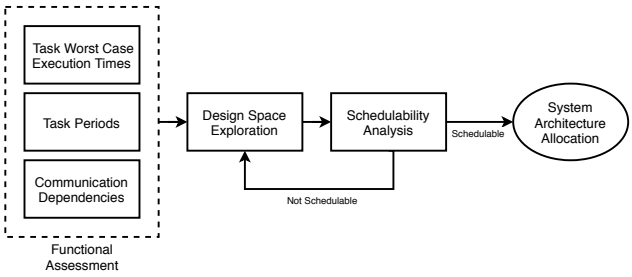

Fig. 1: Information flow in the proposed method, depicting the possibility of reiterating the design-space exploration as a result of the schedulability analysis.

As the constraints in question cannot ensure, nor enforce, the schedulability of a system, a schedulability analysis is necessary. This forms the third step of the proposed method. This paper proposes to use response-time analysis [21], [25], which is a well-defined schedulability analysis technique that can be applied directly after the design-space exploration. This way, infeasible allocations are discarded at an earlier stage. However, the schedulability test can only be performed after a system allocation has been generated as the results of the schedulability test determine the feasibility of the allocation. If even one task allocated to one resource is proven not to be schedulable, the whole allocation needs to be reconfigured by modifying the linear constraints. This reiteration of the design-space exploration step is depicted in Figure 1 and happens as a result of an unschedulable allocation.

It is assumed that all computing platforms are single-core processors, so a uniprocessor schedulability test can be used. Schedulability conditions can be represented or approximated as linear constraints [26]. However, the method presented in this paper does not aim at achieving an optimal schedulability of the task distribution, but rather at providing a schedulable task distribution which may be sub-optimal. Aiming for optimal schedulability would greatly increase the complexity of finding the system allocation, a cost easily avoided by performing a simple schedulability test after an iteration of design space exploration, slightly modifying the linear constraints in case no schedulable system allocation can be achieved, and then rerunning the design space exploration from the beginning. As it will be later shown, there are mechanisms which can enable custom constraints onto the design space exploration, such as manually grouping or separating tasks.

IV. Implementation of the Proposed Method

This section provides a proof-of-concept implementation the proposed method, detailing the design space and the set of linear constraints considered in the design-space exploration. Additionally, this section addresses the design objective and schedulability analysis.

A. Design Space

The linear constraints used in the ILP consider the set of resources R, where a resource $r \in R$, and the set of processes P, where a process $p \in P$. Each process p can

be implemented on a resource from R_p, where $R_p \subseteq R$. The subscript r denotes the allocation of a certain property on the resource r. For example, p_r indicates the allocation of process p onto resource r, and c_r indicates the allocation of the communication task c onto resource r. Let N_r be the number of resources and N_p be the number of processes, each process is then defined by N_r binary variables, stating if a process is allocated onto a specific resource or not. This, by extension, means that there are $N_r \times N_p$ variables describing the allocation of processes onto resources. Resources in this implementation are considered as either processors or cores of multi-core processors. Processes in this context are real-time tasks, which are executed on the processors. Activation of a resource is derived from having at least one process allocated onto it. The linear constraints, which are derived from [10], disregarding the constraints describing the communication of processes, consist of the following:

$\forall p \in P \ :$

$$\sum_{r \in R_p} p_r = 1 \tag{3}$$

$\forall p \in P, \ r \in R_p \ :$

$$p_r - r \leq 0 \tag{4}$$

$\forall r \in R \ :$

$$r - \sum_{p \in P \wedge r \in R_p} p_r \leq 0 \tag{5}$$

where Equation 3 assures that a process p is allocated onto one and only one resource in the system. The linear constraint considers the vector p_r consisting of binary values indicating if a process p is allocated onto a resource r. Modifying the right side of the equality enables a process to be simultaneously allocated onto multiple resources in the system, which can be used to host redundant copies of a process.

Equation 4 assures that a process p is, and can only be, allocated onto an activated resource r. In this equation, and the following, r represents a binary value indicating whether a resource is activated in the system or not. Equation 5 guarantees that a resource r is only activated if a process p is allocated onto it.

Let C be a set of communication tasks, where a communication task $c \in C$, and let N_c be the number of communication tasks. Communication tasks are then distributed among resources from $R_c \subseteq R$. For a certain process p and a communication task c that are allocated to a resource $r \in R_c \cap R_p$, where process p uses communication task c to communicate with another process on another resource connected to the communication bus. Using this logic, it can be said that p and c belong to a common edge, and this, in turn, can be written as $(p, c) \in E_T$, where E_T is a set containing all the edges in the system. One communication task is responsible for connecting exactly two processes. Each communication task, therefore, has two instances, one for each process participating in the communication. Similar to processes, there are $N_r \times N_c$ variables describing the allocation of communication tasks onto resources. The linear constraints describing communication are then defined such that:

$\forall c \in C \ :$

$$\sum_{r \in R_c} c_r \leq 2 \tag{6}$$

$\forall c \in C, \ r \in R_c \ :$

$$c_r - r \leq 0 \tag{7}$$

$\forall c \in C, \ r \in R_c \ :$

$$c_r - \sum_{(p,c) \in E_T} p_r \leq 0 \tag{8}$$

$\forall c \in C, \ r \in R_c \ :$

$$c_r + \sum_{(p,c) \in E_T} p_r \leq 2 \tag{9}$$

$\forall c \in C, \ (p^m, p^n) \in \{(\widetilde{p^m}, \widetilde{p^n}) | ((\widetilde{p^m}, c) \in E_T \wedge (\widetilde{p^n}, c) \in E_T)\}, \ r \in R_c \cap R_p \ :$

$$\begin{aligned} p_r^m - p_r^n - c_r \leq 0 \\ p_r^n - p_r^m - c_r \leq 0. \end{aligned} \tag{10}$$

The idea behind this system of equations is to have one communication task allocated next to every process that needs to communicate with a process on another resource. This is done to model the communication latency of interacting with the communication buffer. Each connection is described by two communication tasks, one for each process participating in the communication. If two processes that need to communicate with each other end up allocated on the same resource (intentionally or unintentionally), the communication task should not be allocated at all, since there is no interaction with the communication bus.

Equation 6 limits the number of allocated communication tasks to two. Equation 7 assures that communication tasks are allocated only to activated resources. Equation 8 guarantees that no communication tasks are allocated to a resource on which none of the participating processes are allocated. Similarly, Equation 9 removes the communication task from the resource where both of the processes participating in the communication are allocated. Finally, the dual constraint described by Equation 10 allocates the communication task to the resource where exactly one of the participating processes is allocated. Variables $\widetilde{p^m}$ and $\widetilde{p^n}$ describe all process pairs which require to communicate with each other.

$\forall r \in R \ :$

$$r - \sum_{c \in C \wedge r \in R_c} c_r - \sum_{p \in P \wedge r \in R_p} p_r \leq 0 \tag{11}$$

To expand the initial model with the communication constraints, it is also important to modify Equation 5 to consider communication tasks, the modification of which is given by Equation 11.

To add a constraint describing the capacity and utilisation of each resource, the following equation is used:

$\forall r \in R \ :$

$$-\zeta(r) r + \sum_{p \in P} \zeta(p_r) p_r + \sum_{c \in C} \zeta(c_r) c_r \leq 0 \tag{12}$$

where $\zeta(r)$ is the capacity of resource r and $\zeta(p_r)$ is the amount of that capacity used up by process p if it ends up allocated to resource r.

To account for real-time characteristics in the sense of resource capacities and requirements of each process, the

period and WCET of each process — or, in the real-time terminology, task — are considered.

The capacity of all resources should be equal and is defined as the hyperperiod of all tasks in the task set, therefore $\zeta(r) = H$. The hyperperiod, H, of a task set is defined as the least common multiple of the periods of all the tasks in the task set. The resource requirement for each task is defined by the formula:

$\forall p \in P \ :$

$$\zeta(p) = \frac{H}{T_p} C_p \tag{13}$$

where T_p is the period of task p, and C_p is its WCET. As can be seen in Equation 13, the resource requirement of each task based on its WCET and period is actually the sum of the WCETs of all instances of the task in the hyperperiod. Additionally, it is assumed that all communication tasks have an equal resource requirement, $\zeta(c_r)$, which is defined by the bandwidth of the communication between two hardware platforms and all other delays the communication may be subject to. The communication requirements are independent of the real-time characteristics of the processes participating in the communication since the assumption is that all messages are available on the bus at all times. The absence of a message from the communication bus indicates that a fault is present in the system, and should be dealt with in an earlier step.

Aperiodic tasks can be included in the analysis using servers [27] — periodic tasks that reserve execution time for aperiodic tasks that may or may not need execution in each cycle. The period of the server should be determined manually, according to the WCET of the corresponding aperiodic task and its importance to the functionality of the overall system.

In order to manually group a set of processes together, the following constraint can be enforced:

$\forall r \in R \ :$

$$p_r^m - p_r^n = 0 \tag{14}$$

where p^m and p^n are two arbitrary processes where $m \neq n$. Equation 14 guarantees that processes p^m and p^n will be allocated to the same resource, and the optimal solution will be found by taking this constraint into consideration. One scenario in which this requirement can be considered useful is when two processes require fast or constant communication, which would greatly increase the load on communication bus.

Similarly, a constraint can be added to manually separate two processes from each other:

$\forall r \in R \ :$

$$p_r^m + p_r^n \leq 1 \tag{15}$$

Equation 15 enforces this constraint, by allowing only one of the two selected processes to be allocated onto one resource.

Manually setting constraints using Equations 14 and 15 relying on prior knowledge of the specific system for which the allocation is being optimised is particularly useful for following functional safety guidelines set by standards often used in the construction vehicle domain, such as IEC 61508 [5] and ISO 26262 [6]. These standards, along with system-specific domain knowledge can help formulate constraints in order to achieve properties such as redundancy, fault detection or even graceful degradation.

B. Design Objective

In the problem at hand, the objective function is minimised and is defined as:

$$f = \sum_{r \in R_p} r + \sum_{c \in C \wedge r \in R_c} c_r \tag{16}$$

wherein the first sum is used to minimise the number of activated resources in the system, and the second sum is used to minimise the number of communication tasks in the system, which is done by grouping tasks that need to communicate on the same resource.

C. Schedulability Analysis

The schedulability analysis is performed for each resource independently, as the influence of tasks on other resources is embedded into the communication tasks resource requirements. As it was stated, all computation platforms are assumed to be single-core processors.

To perform a schedulability analysis, Response Time Analysis [25], [21] is used by applying the following iterative formula:

$$R_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j \tag{17}$$

where R_i^n is the response time of task i for iteration n and T_j is the period of Task j. The subset of the task set denoted by $hp(i)$ contains all the tasks with an equal or higher priority than task i, excluding task i. In this context, tasks are considered independent and no resource sharing is considered. Therefore, a lower priority task cannot block a higher priority task. Note that the scheduling of communication tasks is dependent upon the type of communication protocol. For instance, Controller Area Network (CAN) [28] uses the fixed-priority non-preemptive scheduling [29]. In that case, the corresponding response-time analysis (as discussed in [30]) is used instead of Equation 17. For the first iteration in Equation 17, it is assumed that $R_i^0 = C_i$. The calculation given by the iterative Equation 17 is repeated until the value of R_i remains unchanged through iterations, i.e., until $R_i^{n+1} = R_i^n$ or the specified timing requirement (e.g., deadline) is exceeded.

V. Industrial Use Case: Autonomous Hauler

This paper considers an industrial use case from the segment of construction vehicles, which was gracefully provided by Volvo Construction Equipment (Volvo CE)[1]. The use case serves to provide a proof of concept and show usability of the proposed method. The use case consists of an autonomous hauler that currently has a set of functionalities implemented on a single, centralised, hardware platform. The hauler is a part of an autonomous quarry, where it autonomously transports crushed stones and gravel from one location to another. The specific requirement in this context is an assessment of the effects of decentralisation, and, to find an appropriately decentralised architecture that can retain a similar level of productivity compared to the existing architecture. This is a complex task since there is a great number of considerations

[1]https://www.volvoce.com/

that must be taken into account, many regarding safety, redundancy, efficiency and cost as well as extensibility.

A. System Overview

The system of interest in this use case is an on-board computer architecture of the autonomous hauler. The computer has access to information from various sensors in the vehicle, and, has access to different communication channels and vehicle controls. The system is identified as a machine automation system, depicted in Figure 2. This system is integrated into the machine[2] and it operates the machine via an interface to the machine platform — based on missions that are received from a remote fleet management system. Moreover, any sensor needed for sensing the environment is also considered part of the machine automation system while the position of the machine is part of the machine platform.

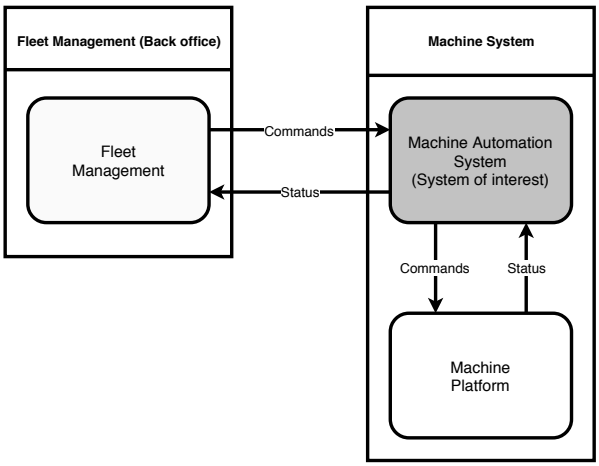

Fig. 2: System-level overview of the system in an autonomous construction vehicle use case.

The functions considered in this use case are generally essential to any vehicular automation system, though in this case, specifically developed for the autonomous construction vehicles domain. These include path execution, object detection, collision avoidance, remote control and operation, teleoperation, and safety-critical functions. The functions in the use case are currently implemented on one single computer. The aim is to decentralise these functions by optimised allocation of the system architecture with respect to the available system resources, while meeting the specified constraints.

B. Existing System Architecture

The existing system that was available for the demonstration of the use case consisted of a centralised approach where all the functionality was executed on one computer. The functional architecture was an atomic implementation consisting of several functionalities such as object detection and machine control. In Figure 3, an abstracted[3] overview of the functional architecture of the machine system in focus is depicted, where, the communication between tasks is represented by the edges in the graph. Task A is a periodic task with a period of 10 ms while tasks B–F are aperiodic and have additional communication dependencies derived from the sensors. These dependencies are addressed by grouping several functionally related sub-tasks into one task, for sub-tasks which should not be separated onto different hardware resources for logical reasons.

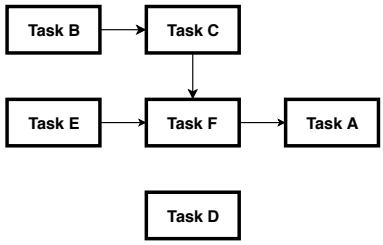

Fig. 3: Functional architecture of the machine system together with the communication dependencies among the tasks.

VI. Experimental Evaluation and Validation

In this section, the results gathered from applying the method to the proposed use case are discussed. First, the worst-case execution times of each task in the evaluated system are estimated. Thereafter, the system allocation is generated by the implemented method. Note that the proposed method is implemented as an in-house tool.

A. Worst-case Execution Time Estimation

This paper applies a statistical method based on the extreme value theory [22] to estimate the WCETs of the tasks A–F in the use case. Note that any other method, such as the flow analysis performed by the SWEET tool and benchmark [23], can also be used for the WCET estimation — refer to [24] for an extensive comparison of the WCET estimation method used in this paper and SWEET. Since the WCET estimation method used in this work is statistical, it is subject to a probability of exceedance. This means, the estimated WCET is guaranteed to only be exceeded by future invocations at the same rate as the exceedance probability. Overall, the number of measured executions was approximately 2.5 million for each task in the use case. Using an exceedance probability of $p_e = 10^{-5}$, the WCET estimates for each task, and their respective periods, are given in Table I. In this table, the tasks with periods denoted by N/A do not have a defined period. Such tasks are considered aperiodic and the time between two invocations is mostly dependent on the length of their execution, therefore not constrained to a fixed period.

B. System Architecture Allocation

The measurements attained from the WCET analysis are used as the basis for the design-space exploration. Tasks A–F are mapped onto Processes 1–6 and given WCETs and periods as parameters. Based on the connection requirements depicted in Figure 3, a partial adjacency matrix that describes the communication relations of the processes is defined as follows:

[2]This paper uses the terms "machine" and "vehicle" interchangeably as construction vehicles are often referred to as construction machines in the industry.

[3]For the sake of intellectual property protection, the names of the tasks and other related information are anonymised.

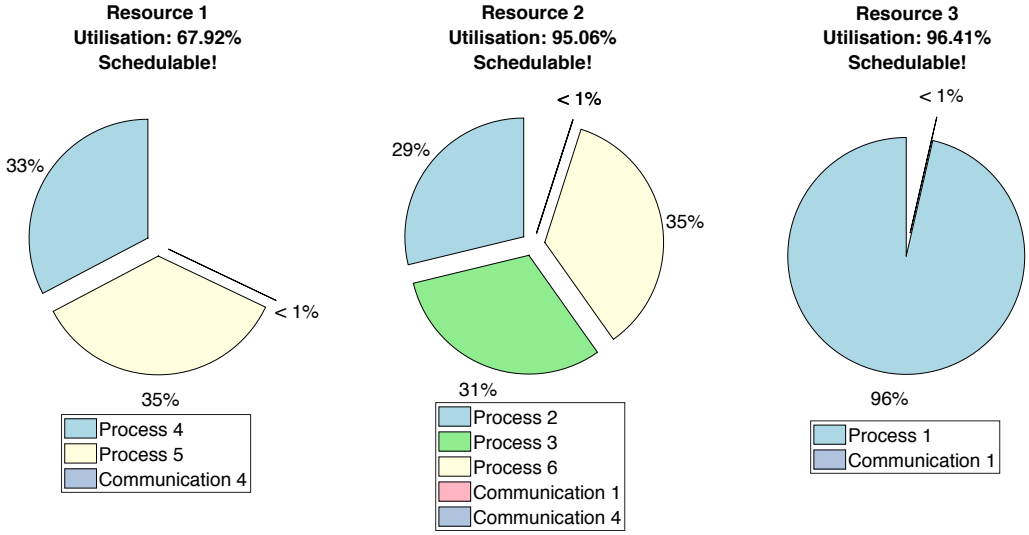

Fig. 4: Resulting system architecture after optimised allocation to the system resources.

TABLE I: Table listing the periodicity and WCET estimate for each task in the use case.

Task	Period (ms)	WCET (ms)
A	10	9.6401
B	N/A	23.0226
C	N/A	24.8154
D	N/A	26.1708
E	N/A	28.1620
F	N/A	28.2108

$$M_c = \begin{bmatrix} 0 & - & - & - & - & - \\ 0 & 0 & - & - & - & - \\ 0 & 1 & 0 & - & - & - \\ 0 & 0 & 0 & 0 & - & - \\ 0 & 0 & 0 & 0 & 0 & - \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

where rows and columns both represent processes, and the value of each element shows if the two processes are communicating or not. For example, a value of 1 at position (3, 2) indicates that Process 3 communicates with Process 2.

Each aperiodic task (Tasks B–F) requires a designated aperiodic server. Hence, each aperiodic task is assigned a polling server with a period of 80 milliseconds and capacity of 30 milliseconds, which is sufficient from the WCET standpoint of the aperiodic tasks. Figure 4 shows the system architecture allocation as a result of the design-space exploration. There are three hardware resources that are activated in the system, namely Resource 1, 2 and 3. The method allocates two processes (4 and 5) and one communication task (4) to Resource 1. Furthermore, three processes (2, 3 and 6) and two communication tasks (1 and 4) are allocated to Resource 2. Finally, process 1 and communication task 1 are allocated to Resource 3. Communication task 1 models the communication between processes 1 and 6, while communication task 4 models the communication between processes 5 and 6. Although three resources are activated in this experiment, the method can derive optimised system architecture for any number of resources.

Each pie chart in Figure 4 represents the capacity utilisation for one hardware resource. Each segment of a pie chart represents one process (task) or a communication task. The percentage next to each segment shows the capacity utilisation of that process/communication task on that hardware platform. The individual utilisation of Resource 1, 2 and 3 is 67.92%, 95.06% and 96.41% respectively. Whereas, the average utilisation achieved for the derived system architecture with respect to the three resources is 86.46%. Note that the optimisation was executed on a computer with an i7-6500U 2.5 GHz CPU and 8GB of RAM. The optimisation was completed in 0.7693 seconds.

In this experiment, it is considered that the WCET measurements contain the communication loads, so the communication tasks are given a small load of 1 microsecond, only for the purpose of visualisation, which is also identified in Figure 4. The system has no requirements on process redundancy, as no processes are identified as safety-critical. A schedulability analysis using the rate-monotonic priority assignment and response-time analysis is performed on all the task sets individually, and it is concluded that the system architecture allocation is feasible since all the task sets are schedulable.

VII. Conclusion

This paper has adapted the state-of-the-art techniques to provide an optimised allocation method for vehicular system architectures with real-time constraints. The adapted method is simplified and easier to apply to the industrial settings. The method consists of three general steps that include: worst-case execution time estimation, design-space exploration, and schedulability analysis. The proof of concept and usability of the method is demonstrated on a use case from the construction vehicles industry. The evaluation results indicate

that the method can successfully derive optimised system architectures according to the specified real-time constraints, while permitting specific design choices made by the designer. This further allows the method to accommodate, in addition to revolutionary architecting, the evolutionary architecting paradigm in cases where pre-defining the existing architecture and allocation is possible.

The work in this paper is done keeping functional safety requirements implicitly in mind. OEMs rely on standards to give guarantees that the products developed are safe to use in real-world situations where functional safety is a high priority. It has already been shown that the optimisation constraints can be used to enforce functional safety guidelines suggested by standards such as IEC 61508 and ISO 26262. One future work entails the accommodation of functional safety standards by the method, as well as what guarantees regarding functional safety can be inherently given by its application. Furthermore, adapting the design-space exploration method to account for heterogeneous systems would be of benefit to optimise for, e.g., system cost. Such work would necessitate either generalising the metric of allocation, which in this paper was defined as the system utilisation, calculated from worst-case execution time estimates and periods, or deriving additional constraints in order to consider resource-dependent worst-case execution times of processes. A more comprehensive scheme to account for communication dependencies would be a beneficial extension to this work. The industrial use case considers the system to communicate through one communication bus, connecting each resource with the shortest possible distance between each other, i.e., one hop. Extending such constraints would allow the method to derive optimised system architectures in which the resources are not at a uniform distance apart.

ACKNOWLEDGEMENT

The work in this paper is supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA) through the DESTINE project and the Swedish Knowledge Foundation (KKS) through the projects DPAC and HERO. The authors would like to thank all industrial partners, especially Volvo Construction Equipment, Sweden.

REFERENCES

[1] Technavio, Global Automation Market in Automotive Industry–2018-2022, technical report SKU: IRTNTR20198, pp. 1-130, Jan., 2018, avialbale at https://www.technavio.com.

[2] M. D. Natale and A. L. Sangiovanni-Vincentelli, "Moving From Federated to Integrated Architectures in Automotive: The Role of Standards, Methods and Tools," *Proceedings of the IEEE*, vol. 98, no. 4, pp. 603–620, Apr. 2010.

[3] C. B. Watkins and R. Walter, "Transitioning from Federated Avionics Architectures to Integrated Modular Avionics," in *26th IEEE/AIAA Digital Avionics Systems Conference*, Oct. 2007.

[4] R. Obermaisser, C. E. Salloum, B. Huber, and H. Kopetz, "From a Federated to an Integrated Automotive Architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 7, pp. 956–965, Jul. 2009.

[5] International Electrotechnical Commission, IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems, Part 3: Software requirements, IEC SC 65A, 1998.

[6] International Organization for Standardization (ISO), ISO 26262-1:2011: Road vehicles – Functional safety. http://www.iso.org/.

[7] S. Baumgart, J. Fröberg, and S. Punnekkat, "Towards Efficient Functional Safety Certification of Construction Machinery Using a Component-based Approach," in *3rd International Workshop on Product LinE Approaches in Software Engineering*, Jun. 2012, pp. 1–4.

[8] L. Lo Bello, R. Mariani, S. Mubeen, and S. Saponara, "Recent advances and trends in on-board embedded and networked automotive systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1038–1051, Feb 2019.

[9] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya, "Software Architecture Optimization Methods: A Systematic Literature Review," *IEEE Transactions on Software Engineering*, vol. 39, no. 5, pp. 658–683, May 2013.

[10] M. Lukasiewycz, M. Streubuhr, M. Glass, C. Haubelt, and J. Teich, "Combined System Synthesis and Communication Architecture Exploration for MPSoCs," in *2009 Design, Automation Test in Europe Conference Exhibition*, Apr. 2009, pp. 472–477.

[11] A. Mehiaoui, E. Wozniak, S. Tucci-Piergiovanni, C. Mraidha, M. Di Natale, H. Zeng, J.-P. Babau, L. Lemarchand, and S. Gerard, "A two-step optimization technique for functions placement, partitioning, and priority assignment in distributed systems," in *Proceedings of the 14th ACM SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems*, 2013, pp. 121–132.

[12] E. Kang, E. Jackson, and W. Schulte, "An Approach for Effective Design Space Exploration," in *16th Monterey Conference on Foundations of Computer Software: Modeling, Development, and Verification of Adaptive Systems*, ser. FOCS'10. Springer-Verlag, 2011, pp. 33–54.

[13] R. Bouaziz, L. Lemarchand, F. Singhoff, B. Zalila, and M. Jmaiel, "Efficient Parallel Multi-objective Optimization for Real-time Systems Software Design Exploration," in *2016 International Symposium on Rapid System Prototyping (RSP)*, Oct. 2016, pp. 1–7.

[14] J. Knowles and D. Corne, "The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimisation," in *Proceedings of the Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 1, 1999, p. 105 Vol. 1.

[15] I. Švogor and J. Carlson, "SCALL: Software Component Allocator for Heterogeneous Embedded Systems," in *Proceedings of the European Conference on Software Architecture Workshops*, ser. ECSAW '15. New York, NY, USA: ACM, 2015, pp. 66:1–66:5.

[16] T. Rambow, U. Kiencke, R. Schlör, R. Busch, A. Seibertz, "A Framework for Optimized Allocation of Control Functions to a Distributed Architecture", *SAE Transactions*, vol. 114, pp. 251-258, 2005.

[17] A. Lorentzon and V. Tengnäs, "Optimization of intra-vehicle architecture using amulti-objective genetic algorithm," Master's thesis, Institutionen för tillämpad mekanik, Fordonsteknik och autonoma system, Chalmers tekniska högskola, 2017.

[18] S. Islam, N. Suri, A. Balogh, G. Csertán, and A. Pataricza, "An optimization based design for integrated dependable real-time embedded systems," *Design Automation for Embedded Systems*, vol. 13, no. 4, p. 245, Jun 2009.

[19] I. Crnkovic and M. Larsson, *Building Reliable Component-Based Software Systems*. Norwood, MA, USA: Artech House, Inc., 2002.

[20] T. Vale, I. Crnkovic, E. S. de Almeida, P. A. da Mota Silveira Neto, Y. C. Cavalcanti, and S. R. de Lemos Meira, "Twenty-eight years of component-based software engineering," *Journal of Systems and Software*, vol. 111, pp. 128 – 148, 2016.

[21] L. Sha, T. Abdelzaher, K.-E. Arzen, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real Time Scheduling Theory: A Historical Perspective," *Real-Time Systems Journal*, vol. 28, no. 2/3, pp. 101–155, 2004.

[22] J. Hansen, S. A. Hissam, and G. A. Moreno, "Statistical-Based WCET Estimation and Validation," *Proceedings of the 9th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis.*, Jan. 2009.

[23] B. Lisper, "SWEET – a Tool for WCET Flow Analysis," in *6th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*. Springer-Verlag, Oct. 2014, pp. 482–485.

[24] V. Marcus and H. Arman, "A method for optimised allocation of system architectures with real-time constraints," Master's thesis, Mälardalen University, School of Innovation, Design and Engineering, 2018.

[25] M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.

[26] H. Zeng and M. Di Natale, "Improving real-time feasibility analysis for use in linear optimization methods," in *2010 22nd Euromicro Conference on Real-Time Systems*, July 2010, pp. 279–290.

[27] B. Sprunt, L. Sha, and J. Lehoczky, "Aperiodic task scheduling for hard-real-time systems," *Real-Time Systems*, vol. 1, no. 1, pp. 27–60, 1989.

[28] ISO 11898-1, "Road Vehicles – Interchange of Digital Onformation – Controller Area Network (CAN) for High-speed Communication, ISO Standard-11898, Nov. 1993."

[29] L. Sha, T. Abdelzaher, K.-E. årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real time scheduling theory: A historical perspective," *Real-Time Systems*, vol. 28, no. 2, pp. 101–155, Nov. 2004.

[30] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (can) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, Apr 2007.