**Proceedings Paper:**
Mckee, DW, Webster, D, Xu, J et al. (1 more author) (2015) DIVIDER: Modelling and Evaluating Real-Time Service-Oriented Cyberphysical Co-Simulations. In: Proceedings of 2015 IEEE 18th International Symposium on Real-Time Distributed Computing (ISORC). Real-Time Distributed Computing (ISORC), 2015 IEEE 18th International Symposium on, 13-17 Apr 2015, Auckland, New Zealand. IEEE , 272 - 275.

https://doi.org/10.1109/ISORC.2015.30

# DIVIDER: Modelling and Evaluating Real-Time Service-Oriented Cyberphysical Co-Simulations

David McKee, David Webster, Jie Xu
School of Computing,
University of Leeds
Leeds, UK
{scdwm , D.E.Webster, J.Xu}@leeds.ac.uk

David Battersby
Jaguar Land Rover
Gaydon, UK
dbatters@jaguarlandrover.com

***Abstract*—The ability to reliably distribute simulations across a distributed system and seamlessly integrate them as a workflow regardless of their level of abstraction is critical to improving the quality of product manufacturing. This paper presents the DIVIDER architecture for managing and maintaining real-time performance simulations integrated through SOAs. The described approach captures features present in complex workflow patterns such as asynchronous arbitrary cycles and estimates the worst case execution time in the context of the interfering execution environment.**

***Keywords-real-time;SOA;workflow,automotive;integration; performance interference***

## I. INTRODUCTION

The capability to perform reduced physical prototyping and automation in manufacturing industries is critical as industries seek to reduce the time moving through each engineering phase [1]. An example of prototyping is the integration of electrical control units of automotive sub-systems with hardware-in-the-loop simulation [2]. A second example is human (or driver)-in-the-loop (DIL) systems that integrate the control unit models with driving profiles [3]. The rapid integration of simulation models with DIL systems or HIL test-benches remains a challenge.

Service-Oriented Architectures (SOA) have been demonstrated to support system dependability characteristics [4], [5] when integrating distributed systems through the likes of *ultra-late binding* whereby a service is bound *'just in time'*. Discrete event simulation (DES) technologies such as DDS and HLA use federated approaches which lack the ability to assure the delivery of real-time capability for a given service.

In response to this need for rapid automotive simulation integration we propose DIVIDER, a novel Service-Oriented integration architecture enabling the dynamic integration of heterogeneous tools and models, with real-time performance. Adherence to strict real-time deadlines is achieved by formally modelling the simulation workflows, the data exchange patterns, and the underlying resource characteristics of the execution platforms.

We present an automotive use case in section II. In section III we demonstrate this as a service oriented problem and modelled the performance with respect to the underlying network infrastructure. We demonstrate our implementation using the automotive scenario in section IV. Finally we present our conclusions and future work in section V.

## II. MOTIVATION FOR INTEGRATED SIMULATION

In the automotive industry the process of designing, testing, and manufacturing vehicles is known to take up to 7 years [6] in part due to limited capability for traversing the levels of abstraction seen in engineering lifecycle phases. In this section we discuss a motivating scenario, consider the challenges of co-simulation, and the related work in real-time (RT)-SOA techniques with emphasis on real-time workflows for system integration.

### A. Motivating Scenario

Figure 1 depicts a transmission system of a vehicle consisting of several components at different abstractions:

1. Engine and transmission modelled in Simulink.
2. Transmission control unit using HIL technologies.
3. Vehicle dynamics as DIL behavior using TORCS [7].

The transmission control unit (TCU) and the vehicle dynamics run asynchronously from the remainder of the system with their own control loops. We control the clock

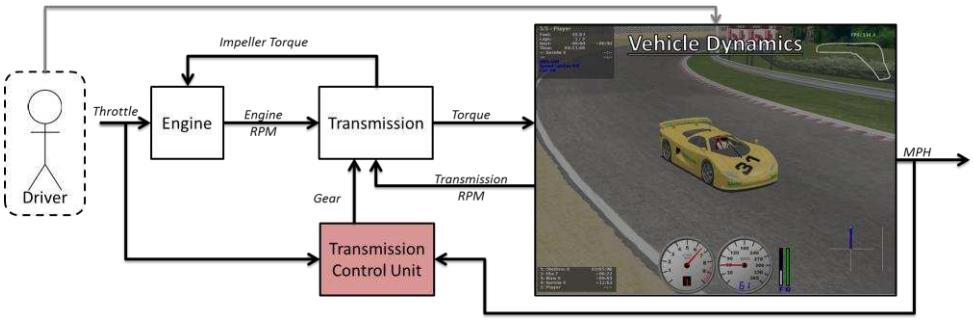


**Figure 1: Motivating Scenario Simulating a Transmission Control System**

rate of the later, but the TCU runs relative to *'wall'* clock.

If we were to assume that each system component takes the same amount of time to execute, the following simple workflow of simulation communications would be observed:

- Vehicle dynamics →Transmission and TCU
- Engine →Transmission
- Transmission → Vehicle dynamics

If the constraint of discrete events is removed, as is the case with HIL and DIL components, each of the above communications may occur at differing times.

### B. The Challenge of Real-time Co-Simulation with SOA

Current work in RT-SOA has demonstrated that SOA cannot guarantee real-time deadlines subject to following key limitations [4], [8]:

- A lack of complete modelling of the execution environment results in delayed executions and longtails [9]. A preconfigured environment has been modeled, but not its runtime changes [10].
- Current techniques react to performance degradations without knowledge of the environment [11].
- Approaches such as the iLand project [12] address aspects related to composition the approach is limited to dealing with soft real-time deadlines.

These and other RT-SOA techniques demonstrate that there is still an unresolved research challenge in addressing hard real-time deadlines with SOA lacking sufficient support for the domain of co-simulation.

When using DES for controlling communication between real-time simulations the responsibility of managing execution time with respect to wall clock is the responsibility of the federate rather than the infrastructure [13]. DES approaches describe real-time QoS requirements but are unable to manage the delivery of functionality under strict real-time conditions. Therefore a solution must capture the concepts of control, data, and resource flow relevant to performance management [8]. In our approach system performance is managed through CPU and memory utilization in addition to task interference between multi-tenant tasks [8], [14].

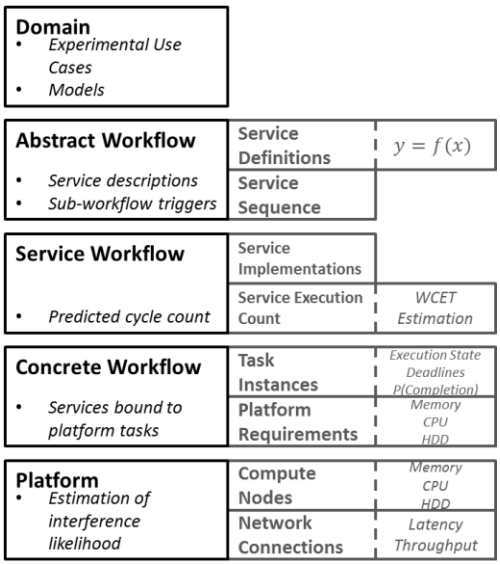


**Figure 2: The SOA system stack**

### C. Understanding Real-time Workflows

The control flow patterns that are present in simulation systems possess limited support by current workflow technologies. In traditional workflows tasks can be orchestrated into logical executable sequences with distinct patterns of: control flow, data flow and routing, exception handling, interactions, and resource utilization. The presence of arbitrary cycles and transient triggers from simulators, however, introduce challenges of non-determinism that current approaches such as BPEL do not handle [15]. Arbitrary cycles represent cycles in a process that have multiple entry or exit points whilst transient triggers are caused by the environment and require immediate handling.

In our approach we aim to mitigate these by using data driven workflows [16], however, this creates a dependency on timely delivery of data by the services.

## III. METHODOLOGY

This section outlines the approach taken to translate the control flow described in Figure 1 into a time dependent data driven SOA workflow. The system stack is discussed followed by a system architecture. This is followed by the underlying mathematical foundations and formalisms.

### A. System Stack: Domain through SOA to Deployment

Figure 2 demonstrates the SOA levels of abstraction, the mapping to the underlying platforms on which service instances execute, as well as the domain models which include engineering test cases. The platform layer allows the concrete workflow services to be mapped onto specific tasks that execute within the computing environment, monitoring execution delays due to performance interference.

Using the schema proposed in our previous work [8], we annotate services with their execution state, deadlines (technical, tolerated, and critical [17]), and probability of conformance to those deadlines. Platform requirements are expressed as prerequisites for delivering a specified QoS, including CPU and memory requirements. We use the network connection profiles to predict communication lags between the workflow engine, services, and tasks in the environment. This activity requires:

1. Methods for using workflows to predict messaging passing between services and the workflow engine.
2. Techniques for identifying communication between a given service and other environmental tasks in order to more accurately predict performance degradations.

### B. Proposed System Architecture

The high-level system architecture for our approach - which we name the **Di**stributed **V**irtual **I**ntegration **D**evelopment **E**nvi**R**onment - is illustrated in Figure 3 with the following key processes:

1. Publishing a service, capturing the state attributes and the effect that the service has on those attributes. This

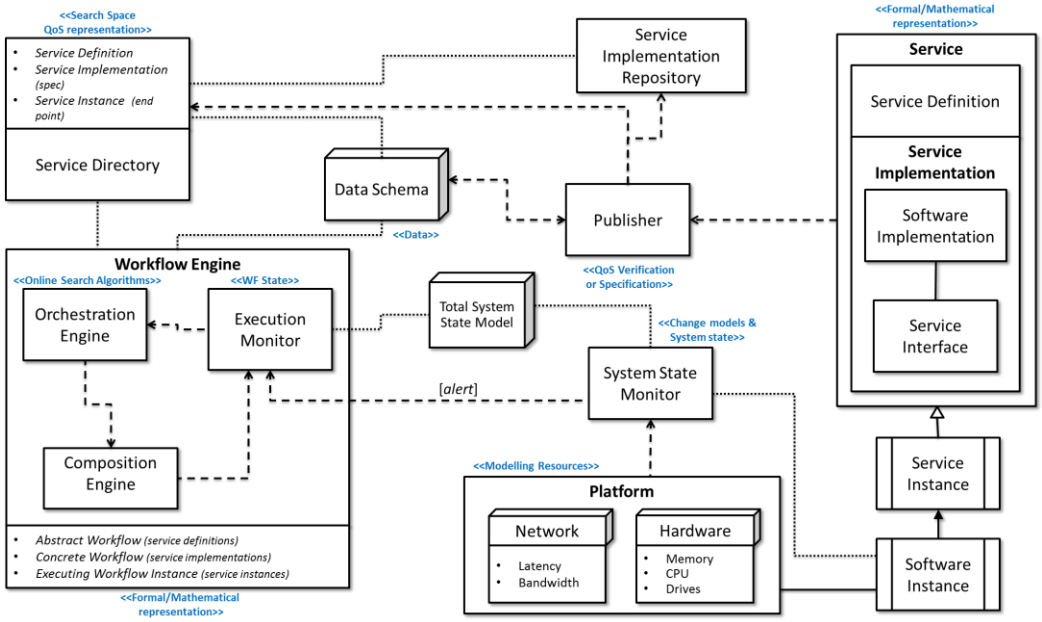


**Figure 3: DIVIDER – DIstributed Virtual Integration Development EnviRonment**

captures the underlying platform requirements as well as the WCET under various configurations.

2. Live monitoring of the system state. The workflow engine must be alerted by and react to changes in the execution environment leading to execution delays.
3. The workflow engine encapsulates the following: orchestration, composition, and execution monitoring.

### C. *Mathematical Foundations*

To support our approach Petri nets are used as the underlying formalism. We introduce certain fundamental concepts into our system: singleton service instances, and states consisting of multiple state attributes in order to deal with lack of instance information in Petri nets [18]. The number of instances for a given service are restricted to allow for more accurate measurement of service state. Subsequent requests of the service must be queued as can be seen in Figure 4. Additionally we capture the state attributes

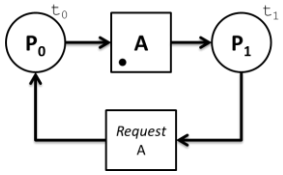


**Figure 4a: Singleton instances of a service. The request cannot be processed until state '$P_1$' is reached**

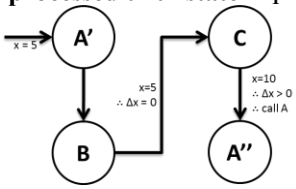


**Figure 4b: Sub-workflow {A→B→C} call upon change of state attribute 'x's value**

upon which each service operates. Given a known value for each attribute we detect a change of value and generate a transient trigger to a specified sub-workflow. A sub-workflow reflects the service calls that can be executed without interference from any additional triggers (figure 5).

### D. *Predicting Execution Performance*

Using the set of transient triggers that cause sub-workflows to execute (and the ordered set of service calls) we predict the worst case execution time (WCET) of the workflow. The service functionality is approximated as the effected change of '$\delta$' on 'y' due to attribute 'x':

$$\delta y = f(\delta x)$$

Given a target 'y' and a starting '$y_0$' the change in 'x' is predicted in addition to the worst case number of iterations of $f(x)$. Using Djikstra's algorithm the WCET of the overall workflow is then estimated.

At the concrete workflow layer the environmental interference is also considered allowing the workflow engine to react and improve the ability to adhere to deadlines. Table 1 and figure 5 show the services from the automotive scenario and the changes which state attributes may invoke.

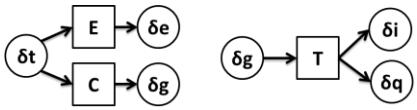


**Figure 5: Sample sub-workflows.**

| Services | | State Attributes | | Next |
|---|---|---|---|---|
| **E** | Engine | **e** | Engine RPM | { T } |
| **T** | Transmission | **i** | Impeller Torque | { E } |
| | | **q** | Torque | { V } |
| **C** | Transmission Control | **g** | Gear | { T } |
| **V** | Vehicle Dynamics | **r** | Transmission RPM | { T } |
| | | **m** | MPH | { C } |
| **Throttle Trigger** | | **t** | Throttle | { E,C } |

**Table 1: Transmission System, state-transition table**

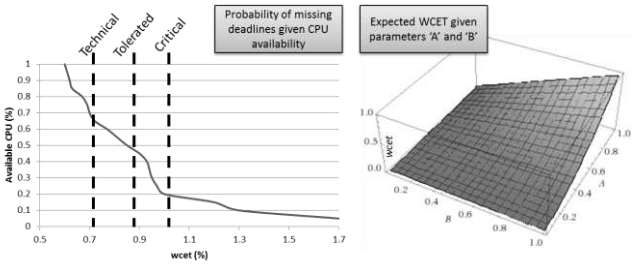


**Figure 6: WCET with respect to CPU availability and required parameter changes**

## IV. EVALUATION

Through holistic monitoring of the execution environment DIVIDER aims to provide accurate real-time performance guarantees of service QoS. The SOA approach supports the application of fault tolerance and dependability mechanisms to increase the systems robustness. Our approach also provides the capability of dynamically changing the level of abstraction that is used for a specific service instance. A further advantage of this dynamism is the ability to adapt workflows and models used in engineering projects for use in concurrent and future projects. The data driven approach increases the scalability of the system through limiting service calls duplication.

The WCET (figure 6) is estimated by analyzing service execution times with respect to input and output parameters. The DIVIDER's workflow engine, therefore, preemptively reconfigures the workflow and execution environment, improving the likelihood of a service or workflow meeting its critical deadline [17], observing and responding to the likelihood of completion.

## V. CONCLUSIONS

This paper has presented a novel approach enabling the transformation of automotive engineering control flow models into data-driven service oriented workflows. In the context of strict real-time constraints in co-simulation of HIL and DIL domains, we discuss each layer of the SOA system stack and propose a novel architecture (DIVIDER) focused on managing the processes of: service publication, orchestration, composition, and fault tolerance within a simulation integration environment. We presented mathematical formalisms allowing the description and transformation of control and data flows into executable SOA workflows and highlighted the limitations of existing technologies and the requirement for further research utilizing SOA workflow in manufacturing domains.

### A. Future Work

The next phase of the work presented here will focus on experimental evaluation. It will be necessary to explore the significance of complex asynchronous workflow patterns from external domain simulators. In this work we must formally define the semantics for expressing complex workflows to fully capture the concepts of: engineering abstraction; inter-service communication; and real-time execution constraints. It is clear that significant advances are required to enable dependable handling of real-time constraints in light of unpredictable and varying environmental conditions.

## ACKNOWLEDGMENT

This work was supported by Jaguar Land Rover and the UK-EPSRC grant EP/K014226/1 as part of the jointly funded Programme for Simulation Innovation.

## REFERENCES

[1] I. Scheeren and C. E. Pereira, "Combining Model-Based Systems Engineering, Simulation and Domain Engineering in the Development of Industrial Automation Systems: Industrial Case Study," *2014 IEEE 17th Int. Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput.*, pp. 40–47, Jun. 2014.

[2] M. Dooner, J. Wang, and A. Mouzakitis, "Development of a Simulation Model of a Windshield Wiper System for Hardware in the Loop Simulation," in *Automation and Computing (ICAC), 2013 19th International Conference on*, 2013.

[3] M. Dempsey, G. Fish, and A. Picarelli, "Using Modelica models for Driver-in-the-loop simulators," pp. 571–578, Nov. 2012.

[4] D. McKee, D. Webster, P. Townend, J. Xu, and D. Battersby, "Towards a Virtual Integration Design and Analysis Enviroment for Automotive Engineering," in *2014 IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, 2014, pp. 413–419.

[5] M. P. Papazoglou and W.-J. van den Heuvel, "Service-Oriented Computing: State of the Art and Open Research Challenges," *Computer (Long. Beach. Calif).*, vol. 40, no. 11, Nov. 2007.

[6] M. Broy, "Challenges in automotive software engineering," *Proceeding 28th Int. Conf. Softw. Eng. - ICSE '06*, p. 33, 2006.

[7] "Torcs." [Online]. Available: http://torcs.sourceforge.net/.

[8] D. W. Mckee, D. Webster, and J. Xu, "Enabling Decision Support for the Delivery of Real-Time Services," in *2015 IEEE 15th International Symposium on High-Assurance Systems Engineering*, 2015.

[9] J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, p. 74, Feb. 2013.

[10] W.-T. Tsai, Q. Shao, X. Sun, and J. Elston, "Real-Time Service-Oriented Cloud Computing," in *2010 6th World Congress on Services*, 2010, pp. 473–478.

[11] S. D. G. Avila and K. Djemame, "Fuzzy Logic Based QoS Optimization Mechanism for Service Composition," in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, 2013, pp. 182–191.

[12] M. Garcia Valls, I. R. Lopez, and L. F. Villar, "iLAND: An Enhanced Middleware for Real-Time Reconfiguration of Service Oriented Distributed Real-Time Systems," *IEEE Trans. Ind. Informatics*, vol. 9, no. 1, pp. 228–236, Feb. 2013.

[13] T. Schierz, M. Arnold, and C. Clauß, "Co-simulation with communication step size control in an FMI compatible master algorithm," no. 2, pp. 205–214, Nov. 2012.

[14] I. S. Moreno, P. Garraghan, P. Townend, and J. Xu, "An Approach for Characterizing Workloads in Google Cloud to Derive Realistic Resource Utilization Models," *2013 IEEE Seventh Int. Symp. Serv. Syst. Eng.*, pp. 49–60, Mar. 2013.

[15] N.A. Mulyar, "Patterns for process-aware information systems : an approach based on colored Petri nets," 2009.

[16] N. Russell, A. H. M. Hofstede, D. Edmond, and W. M. P. Van Der Aalst, "Workflow Data Patterns : Identification , Representation and Tool Support," pp. 353–368, 2005.

[17] R. Kirner, "A Uniform Model for Tolerance-Based Real-Time Computing," *2014 IEEE 17th Int. Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput.*

[18] W. Van Der Aalst, "Pi Calculus Versus Petri Nets," 2005.