Cyber-Physical System Ensembles: Unlocking opportunities when machines collaborate

Pieter J. Mosterman

Chief Research Scientist, Director
MathWorks

Adjunct Professor
School of Computer Science, McGill University

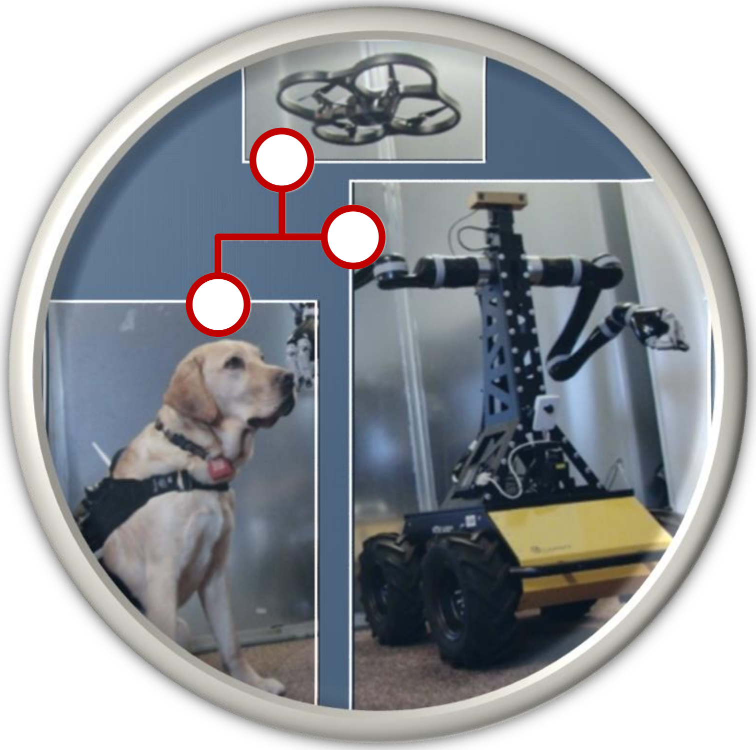
Justyna Zander

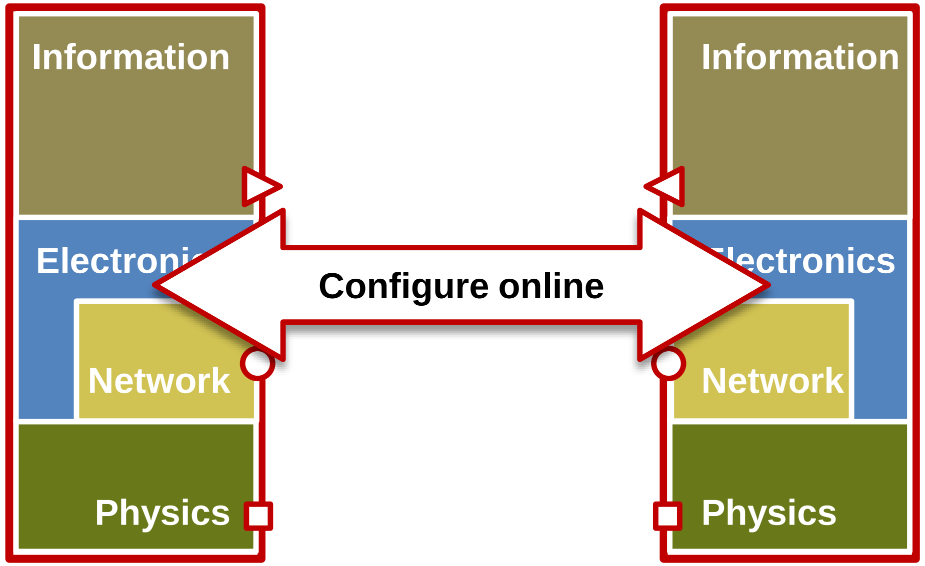
MathWorks Research Fellow
Worcester Polytechnic Institute

Configure online

Confidently design systems as part of a reliable system ensemble

Exploit exogeneous functionality for efficient, economical, and resilient operation





Configure online

Confidently design systems as part of a reliable system ensemble

Exploit exogeneous functionality for efficient, economical, and resilient operation

Impact

Need

Challenge

Technology

Virtual system integration

Proper models in design

Generation of models with necessary detail based on property selection

Linearization, implementation model generation, etc.

Configure online

Confidently design systems as part of a reliable system ensemble

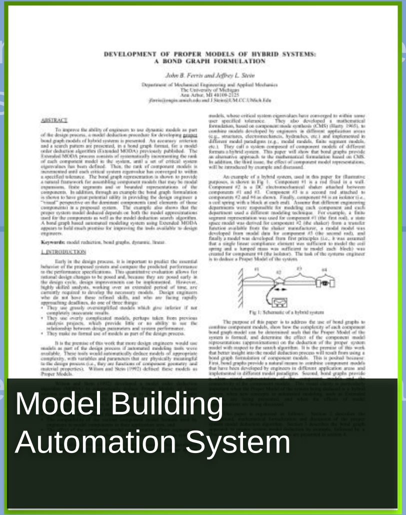
Exploit exogeneous functionality for efficient, economical, and resilient operation

Virtual system integration

Proper models in design ▶

Generation of models with r

Linearization, impleme



Configure online

Confidently design systems as part of a reliable system ensemble

Exploit exogeneous functionality for efficient, economical, and resilient operation

Impact

Need

Challenge

Technology

Virtual system integration

System-level design and analysis by using models ▶

Connecting, combining, and integrating models represented in different formalisms	Efficient simulation models to be used across dynamic and execution semantics
Link for cosimulation, simulation API, code generation	Solver configurations for continuous time, discrete time, discrete event

Configure online

Confidently design systems as part of a reliable system ensemble

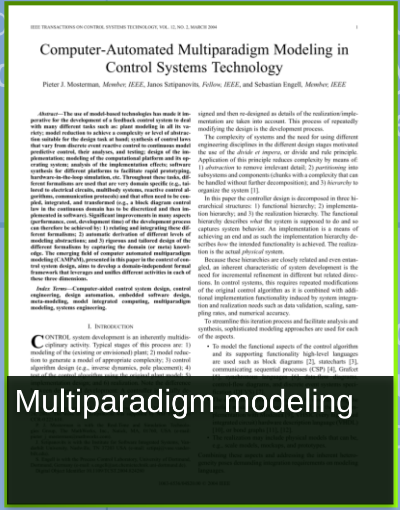
Exploit exogeneous functionality for efficient, economical, and resilient operation

Virtual system integration

System-level design and analysis by using models ▶

Connecting, combining, and integrating models represented in different formalisms

Link for cosimulation, simulation API, code generation



Computer-Automated Multiparadigm Modeling in Control Systems Technology

Pieter J. Mosterman, Member, IEEE, Janos Sztipanovits, Fellow, IEEE, and Sebastian Engell, Member, IEEE

Multiparadigm modeling

An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages

Pieter J. Mosterman

Hybrid dynamic systems

Configure online

Confidently design systems as part of a reliable system ensemble

Exploit exogeneous functionality for efficient, economical, and resilient operation

Impact

Need

Challenge

Technology

Virtual system integration

Connectivity among models, software, and hardware

Open tool platforms with trusted interfaces for communication across synchronized and coordinated models, software, and hardware devices

Data streaming, target connectivity support, standardized communication protocols (TCP, UDP), real-time simulation

Configure online

Confidently design systems as part of a reliable system ensemble

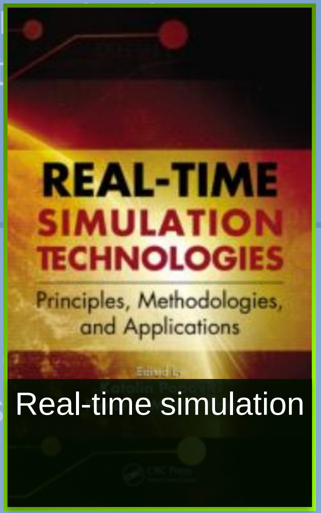
Exploit exogeneous functionality for efficient, economical, and resilient operation

Virtual system integration

Connectivity among models, software, and hardware

Open tool platforms with trusted interfaces for comm
across synchronized and coordinated models, soft
hardware devices

Data streaming, target connectivity support, stan
communication protocols (TCP, UDP), real-time s



Real-time simulation

Configure online

Confidently design systems as part of a reliable system ensemble

Exploit exogeneous functionality for efficient, economical, and resilient operation

Impact

Need

Challenge

Technology

Runtime system adaptation

Reasoning and planning adaptation of an ensemble of systems ▶

Introspection of the system state, configuration, and available services	Handling of ensemble (in)consistency with sufficient runtime fidelity	Online model calibration
Runtime variants, middleware service description specification (.srv)	Traceability (direct and across transformations)	Runtime curve fitting and design optimization

Configure online

Confidently design systems as part of a reliable system ensemble

Exploit exogeneous functionality for efficient, economical, and resilient operation

Runtime system adaptation

Reasoning and planning adaptation of an ensemble of systems ▶

Introspection of the system state, configuration, and available services

Handling of ensemble (in)con... suffici...

Runtime variants, middleware service description specification (.srv)

Traceabi... trans...

R

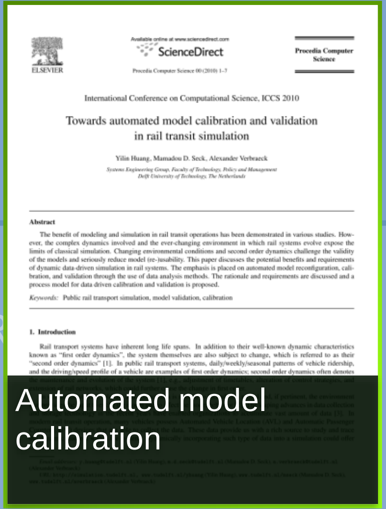
Mechanisms for Leveraging Models at Runtime in Self-adaptive Software

Amel Bennaceur[1], Robert France[2], Giordano Tamburrelli[3], Thomas Vogel[4], Pieter J. Mosterman[5], Walter Cazzola[6], Fabio M. Costa[7], Alfonso Pierantonio[8], Matthias Tichy[9], Mehmet Akşit[10], Pär Emmanuelson[11], Huang Gang[12], Nikolaos Georgantas[1], and David Redlich[13]

[1] Inria, France
[2] Colorado State University, US
[3] Università della Svizzera Italiana, Switzerland
[4] Hasso Plattner Institute at the University of Potsdam, Germany
[5] MathWorks, US
[6] Università degli Studi di Milano, Italy
[7] Universidade Federal de Goiás, Brazil
[8] Univ. degli Studi di L'Aquila, Italy
[9] Chalmers, University of Gothenburg, Sweden
[10] University of Twente, Netherlands
[11] Ericsson AB, Sweden
[12] Peking University, China
[13] Lancaster University, UK

Abstract. Modern software systems are often required to adapt their behavior at runtime in order to maintain or enhance their utility in dynamic environments. Models at runtime research aims to provide suitable abstractions, techniques, and tools to manage the complexity of adapting software systems at runtime. In this chapter, we discuss challenges associated with developing mechanisms that leverage models at runtime to support runtime software adaptation. Specifically, we discuss challenges associated with developing effective mechanisms for supervising running systems, reasoning about and planning adaptations, maintaining consistency among multiple runtime models, and maintaining fidelity of runtime models with respect to the running system and its environment. We discuss related problems and state-of-the-art mechanisms, and identify open research challenges.

Models @ runtime



ELSEVIER
Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 00 (2010) 1–7
Procedia Computer Science

International Conference on Computational Science, ICCS 2010

Towards automated model calibration and validation in rail transit simulation

Yilin Huang, Mamadou D. Seck, Alexander Verbraeck

Systems Engineering Group, Faculty of Technology, Policy and Management Delft University of Technology, The Netherlands

Abstract

The benefit of modeling and simulation in rail transit operations has been demonstrated in various studies. However, the complex dynamics involved and the ever-changing environment in which rail systems evolve expose the limits of classical simulation. Changing environmental conditions and second order dynamics challenge the validity of the models and seriously reduce model (re-)usability. This paper discusses the potential benefits and requirements of dynamic data-driven simulation in rail systems. The emphasis is placed on automated model reconfiguration, calibration, and validation through the use of data analysis methods. The rationale and requirements are discussed and a process model for data driven calibration and validation is proposed.

Keywords: Public rail transport simulation, model validation, calibration

1. Introduction

Rail transport systems have inherent long life spans. In addition to their well-known dynamic characteristics known as "first order dynamics", the system themselves are also subject to change, which is referred to as their "second order dynamics" [1]. In public rail transport systems, daily/weekly/seasonal patterns of vehicle ridership, and the driving/speed profile of a vehicle are examples of first order dynamics; second order dynamics often denotes

Automated model calibration

Configure online

Confidently design systems as part of a reliable system ensemble

Exploit exogeneous functionality for efficient, economical, and resilient operation

Impact

Need

Challenge

Technology

Runtime system adaptation

Testing with functionality on deployed systems

Environment models to enable surrogate interactions

Regression modeling, model selection (artificial neural network, support vector machine, rational model)

Configure online

Confidently design systems as part of a reliable system ensemble

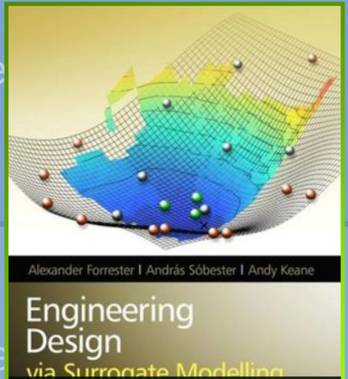
Exploit exogeneous functionality for efficient, economical, and resilient operation

Runtime system adaptation

Testing with functionality on deployed systems

Environment models to enable surrogate inte

Regression modeling, model selection (artificial ne
support vector machine, rational mode



Testing with surrogate models

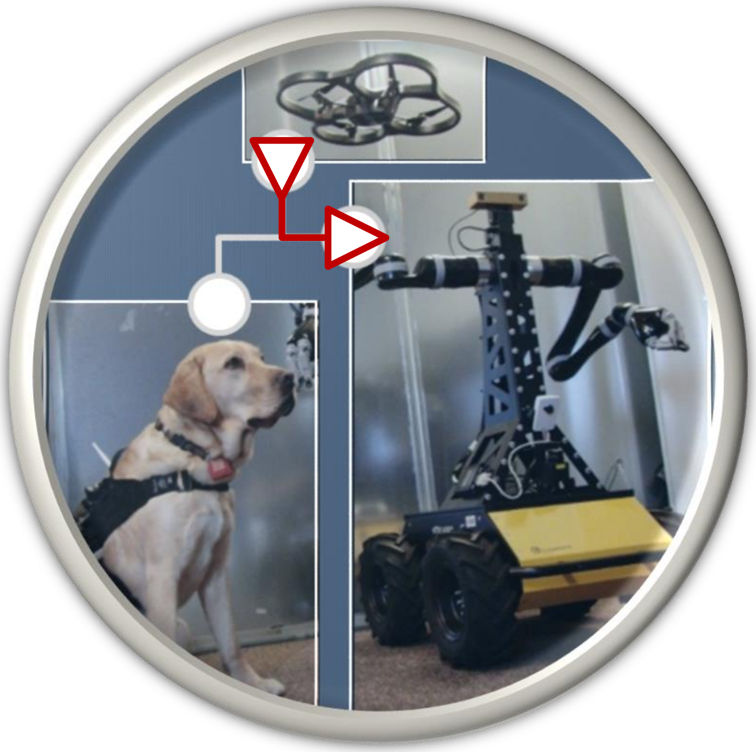
Collaborate

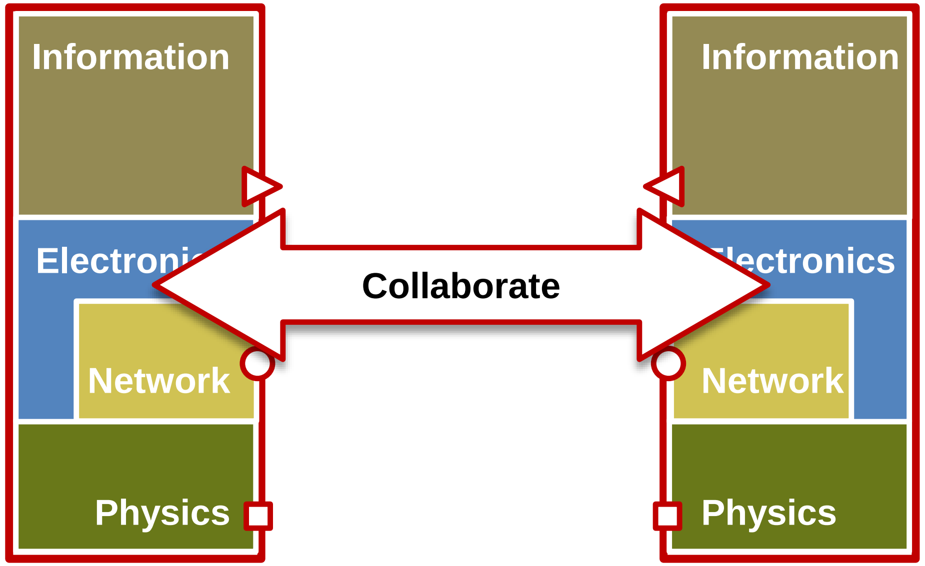
Systematically design systems that are part of a system ensemble to optimally realize desired ensemble behavior

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.





Collaborate

Systematically design systems that are part of a system ensemble to optimally realize desired ensemble behavior

Effectively exploit distributed information resources for ... es

Create novel system features post deployment

Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

Impact

Need

Challenge

Technology

Emerging behavior design

Reasoning and planning adaptation of an ensemble of systems

Analysis methods across loosely coupled architectures	Planning and synthesis of distributed control functionality on concurrent resources	Accessible formal methods that apply to collaborative problems
Event-driven control, discrete event modeling and analysis, uncertainty modeling	Concurrency and platform modeling, functionality decomposition, service composition	Concurrency semantics, property proving with performance models

Collaborate

Systematically design systems that are part of a system ensemble to optimally realize desired ensemble behavior

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

Emerging behavior design

Reasoning and planning adaptation of an ensemble of systems

Analysis methods across loosely coupled architectures

Planning and synthesis

Event-driven control, discrete event modeling and analysis, uncertainty modeling

Service oriented sensor programming

Service orchestration

Collaborate

Systematically design systems that are part of a sys…
rea…
beh…

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

Impact

Need

Challenge

Technology

Data sharing

Multirate architectures ▶

Synchronization of data from incongruent sources

Communication modeling, double buffering analysis, timing properties of software, clock recovery

Collaborate

Systematically design systems that are part of a system ensemble to optimally realize desired ensemble behavior

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

Data sharing

Multirate architectures ▶

Synchronization of data from incong
sources

Communication modeling, double buffering a
properties of software, clock recov

Technology challenges in CPS

Collaborate

Systematically design systems that are part of a sys… rea… behavior

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

Impact

Need

Challenge

Technology

Data sharing

Extracting and deriving specific value from general information

Information represented as high-level models with well-defined metamodels and ontologies

Version based model import/export, metamodel generation, model concepts sharing and comparing

Collaborate

Systematically design systems that are part of a system ensemble to optimally realize desired ensemble behavior

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

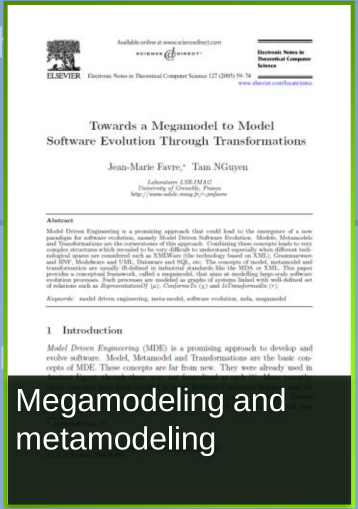
Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

Data sharing

Extracting and deriving specific value from general information

Information represented as high-level models wit
metamodels and ontologies

Version based model import/export, metamodel ge
concepts sharing and comparing


ELSEVIER

Towards a Megamodel to Model Software Evolution Through Transformations

Jean-Marie Favre, Tam NGuyen

Abstract

1 Introduction

Collaborate

Systematically design systems that are part of a system ensemble to optimally realize desired ensemble behavior

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

Functionality sharing

Multi-use functionality post-deployment ▶

Generation of models for a task by property identification and model behavior selection	Performance characterization via performance models and measures	Online calibration based on objective and performance criteria
Property based model slicing, behavioral analysis, functionality mining	Critical path analysis, code performance report and advisor	Adaptive filtering, distortion modeling, groundtruthing (baselining)

Collaborate

Systematically design systems that are part of a system ensemble to optimally realize desired ensemble behavior

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

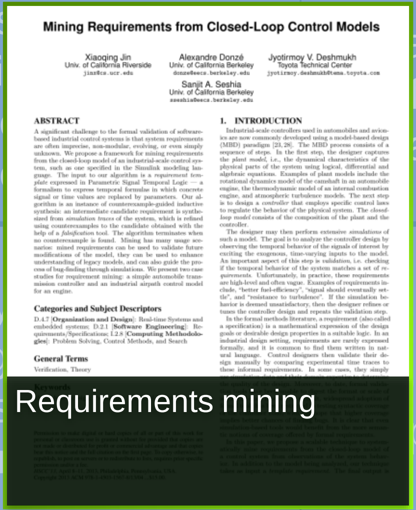
Functionality sharing

Multi-use functionality post-deployment ▶

Generation of models for a tas…

Performance…

Property slicing analysis…



Requirements mining

Data distribution service (DDS)

Online calibration

Collaborate

Systematically design systems that are part of a system ensemble to optimally realize desired ensemble behavior

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

Functionality sharing	Feature interaction
	Assumption formalization and dependency effect analysis Property and assumption based model slicing, trace to source and destination, assumptions in functionality to behavior mapping

Collaborate

Systematically design systems that are part of a system ensemble to optimally realize desired ensemble behavior

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

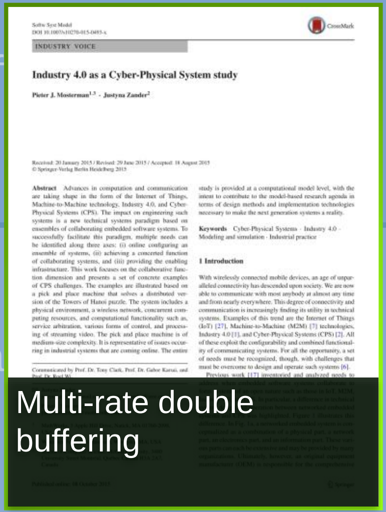
Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

Functionality sharing

Feature interaction

Assumption formalization and dependency ef

Property and assumption based model slicing, tra
destination, assumptions in functionality to beh



Collaborate

Systematically design systems that are part of a system ensemble to optimally realize desired ensemble behavior

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

Collaborative functionality testing

Systematic test suite generation and automated test evaluation ▶

Model-based test generation from requirements while preserving the context of dynamic configuration

Coverage based automatic test generation, variants-based testing, closed-loop testing

Collaborate

Systematically design systems that are part of a system ensemble to optimally realize desired ensemble behavior

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

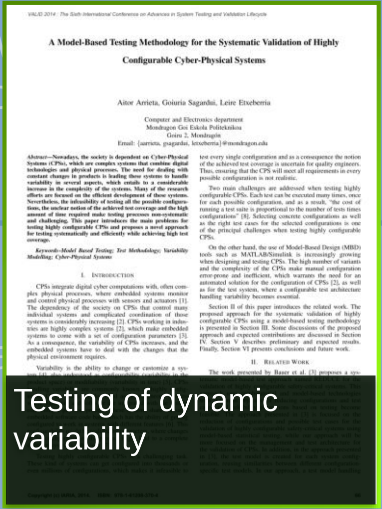
Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

Collaborative functionality testing

Systematic test suite generation and automated test evaluation ▶

Model-based test generation from requirements v
the context of dynamic configuratio

Coverage based automatic test generation, va
testing, closed-loop testing


A Model-Based Testing Methodology for the Systematic Validation of Highly Configurable Cyber-Physical Systems

Aitor Arrieta, Goiuria Sagardui, Leire Etxeberria

Testing of dynamic variability

Collaborate

Systematically design systems that are part of a system ensemble to optimally realize desired ensemble behavior

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

Collaborative functionality testing

Reproducible test results under minimum uncertainty

Setting of initial conditions and injecting fault data	Temporal and spatial partitioning to isolate functionality for a specific system architecture under investigation
System state restoration, stateless services, test fixture generation	Time partition testing, functionality extraction

Collaborate

Systematically design systems that are part of a system ensemble to optimally realize desired ensemble behavior

Effectively exploit distributed information resources for exclusive system features

Create novel system features post deployment

Assure the collaboration quality on shared resources. Identify and automatically mitigate root causes of failure in a distributed environment.

Collaborative functionality testing

Reproducible test results under minimum uncertainty

Setting of initial conditions and injecting fault data	Temporal and spatial partitioning to isolate functi… system ar… inv…
System state restoration, stateless services, test fixture generation	Time partition… e…


The Dark Side of SOA Testing: Towards Testing Contemporary SOAs Based on Criticality Metrics

Philipp Leitner, Stefan Schulte, Schahram Dustdar
Ingo Pill, Marco Schulz, Franz Wotawa

Service oriented architecture testing

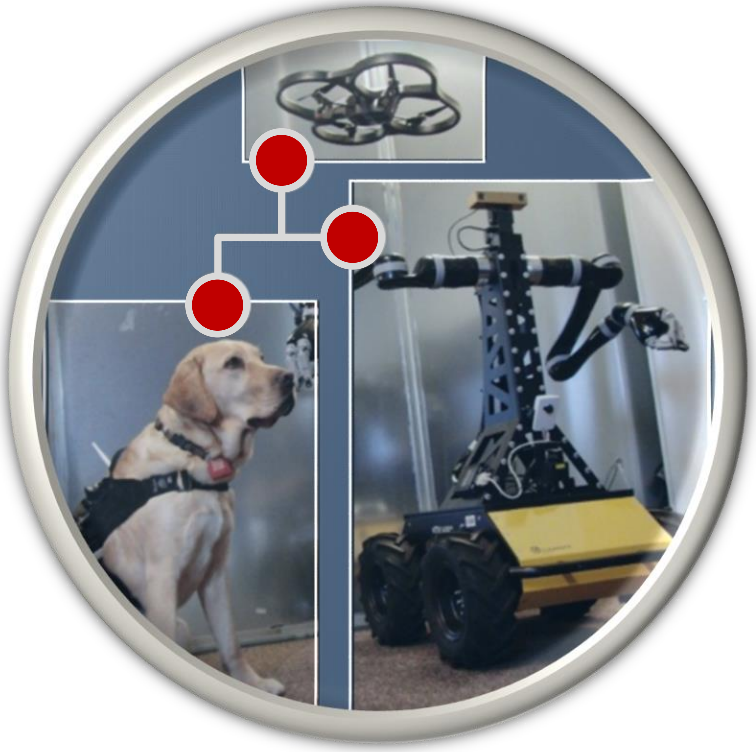
Infrastructure

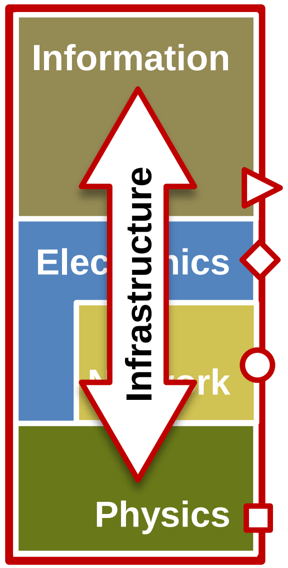
Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

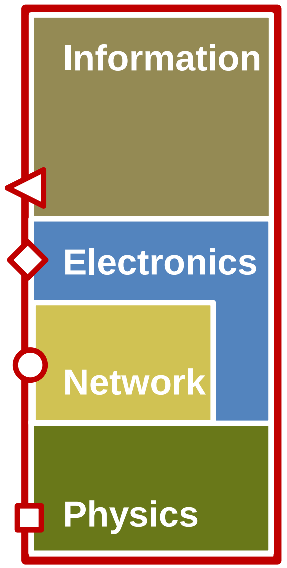
Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs







Infrastructure

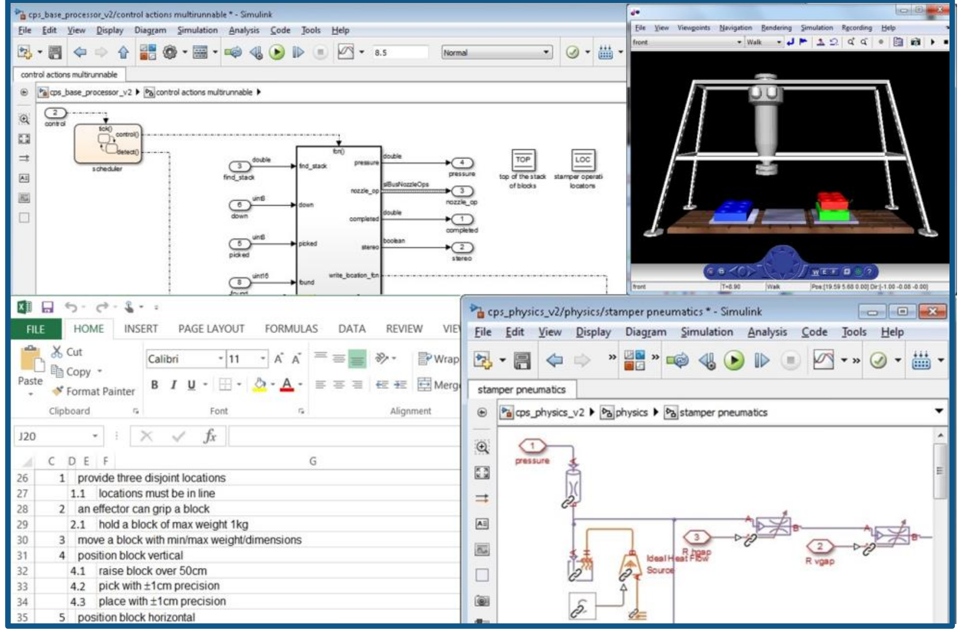
Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Design artifact sharing



Tool coupling among disparate organizations ▶

Traceability across semantic and technology adaptation, and intellectual property protection

Information extraction from obfuscated intellectual property

Relations (across abstractions, formalisms, transformations) service API, change notification API

Protected models (obfuscated, encrypted), trusted compiler

Infrastructure

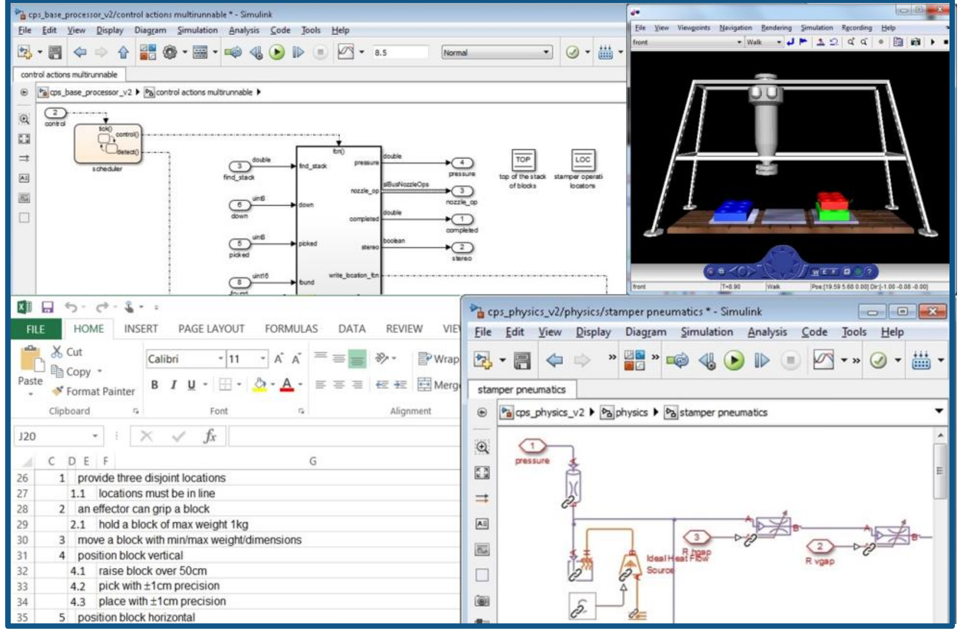
Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Design artifact sharing



Tool coupling among disparate organizations ▶

Traceability across semantic and technology adaptation, and intellectual property protection

Informatio
obfuscated i

Relations (across abstractions, formalisms, transformations) service API, change notification API

Protected m
encrypted)

Infrastructure

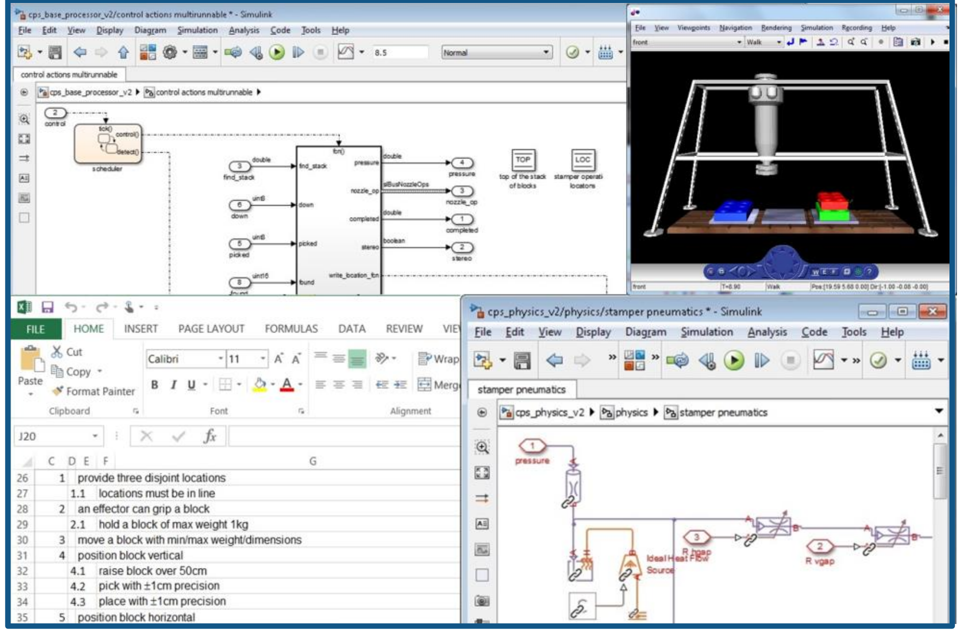
Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Design artifact sharing



Support manifold views and tools in design

Configurable view projections that are tool specific	Consistent semantics across tools by modeling the execution engines
Model generation, pattern (control) extraction, XML interexchange	Modeling numerical mathematics (integration, root finding) as dynamic system

Infrastructure

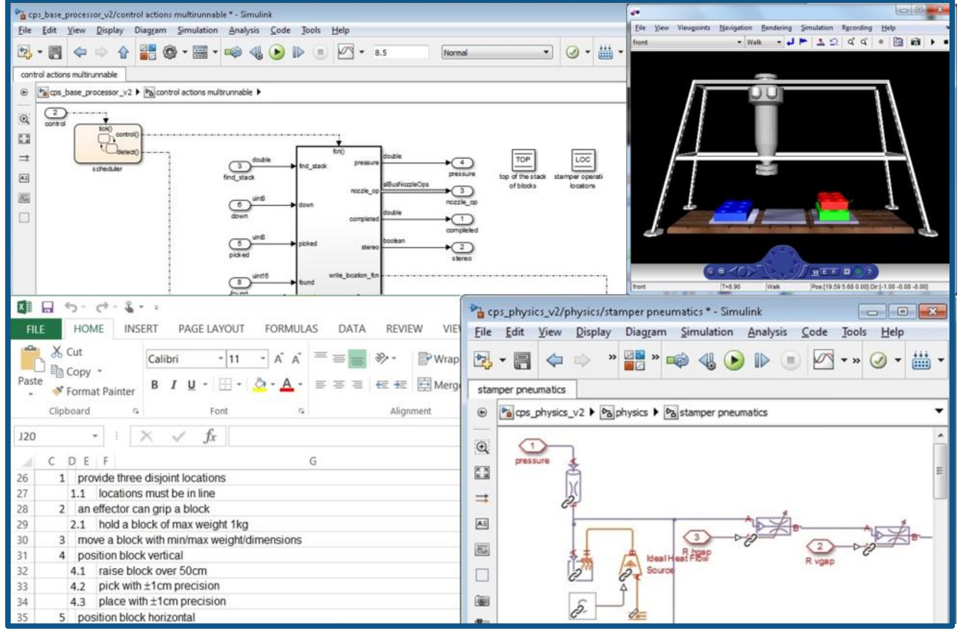
Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

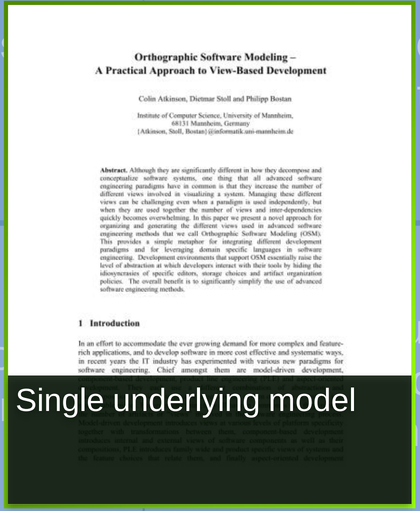
Design artifact sharing



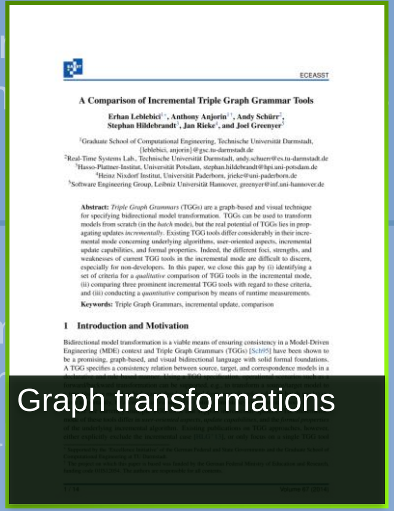
Support manifold views and tools in design

Configurable view projections are tool specific

Model generation, pattern (co extraction, XML interexcha



Single underlying model



Graph transformations

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Wireless communication

Physically aware configurable protocol stack that is IP compatible ▶

Real-time services of graded quality with a low footprint and a configurable protocol stack that includes time and location information

Communication protocol (building block) modeling, performance modeling across target hardware

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

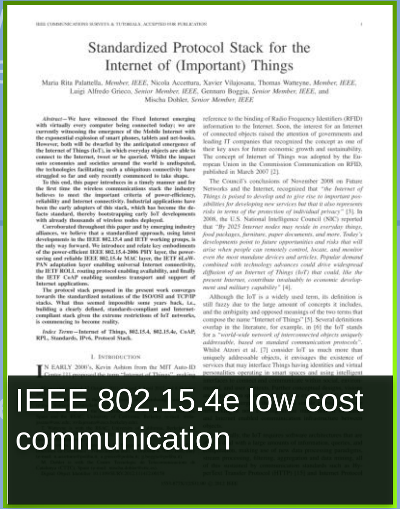
Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Wireless communication

Physically aware configurable protocol stack that is IP compatible ▶

Real-time services of graded quality with a low configurable protocol stack that includes time information

Communication protocol (building block) modeli modeling across target hardwar



Standardized Protocol Stack for the Internet of (Important) Things

Maria Rita Palattella, *Member, IEEE*, Nicola Accettura, Xavier Vilajosana, Thomas Watteyne, *Member, IEEE*, Luigi Alfredo Grieco, *Senior Member, IEEE*, Gennaro Boggia, *Senior Member, IEEE*, and Mischa Dohler, *Senior Member, IEEE*

IEEE 802.15.4e low cost communication

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Wireless communication

Precise timing and synchronization in a distributed environment

Physical layer based timing and synchronization architectures	Scheduling of periodic and aperiodic events with reliable execution times
Physical layer (RF) modeling, antenna modeling	Scheduler configuration, dynamic scheduling with guarantees, mixed synchronous and asynchronous behavior

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

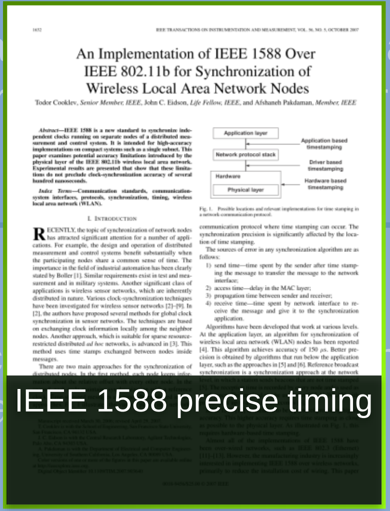
Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

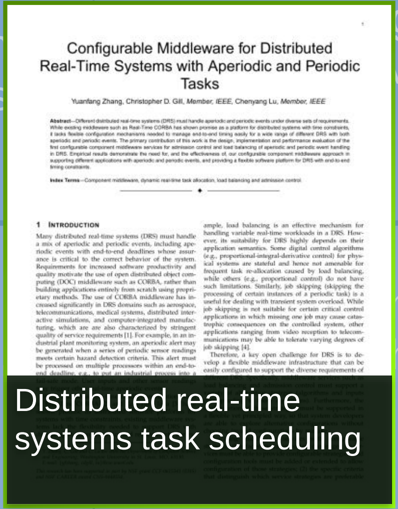
Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Wireless communication

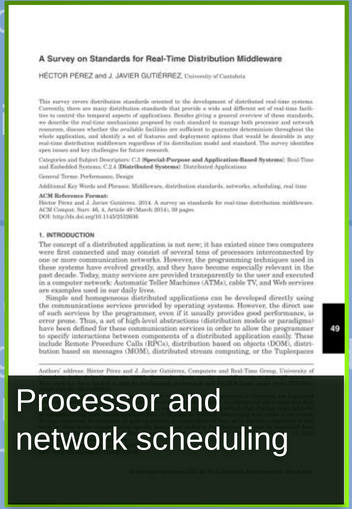
Precise timing and synchronization in a distributed environment



IEEE 1588 precise timing



Distributed real-time systems task scheduling



Processor and network scheduling

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Hardware resource sharing

Flexible and transferable embedded functionality dispatch ▶

Standardized and configurable real-time execution stack

Virtual machine (LLVM, JVM, Docker) with real-time capabilities, serialized intermediate representation of functionality

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

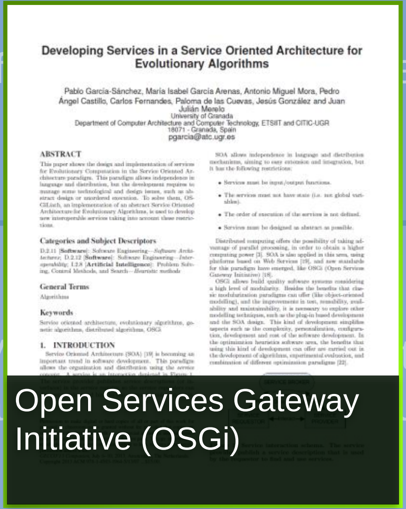
Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Hardware resource sharing

Flexible and transferable embedded functionality dispatch ▶

Standardized and conf… ec…

Virtual machine (LLVM, JV… ti…
serialized intermedia… ur…



Developing Services in a Service Oriented Architecture for Evolutionary Algorithms

Pablo García-Sánchez, María Isabel García Arenas, Antonio Miguel Mora, Pedro Ángel Castillo, Carlos Fernandes, Paloma de las Cuevas, Jesús González and Juan Julián Merelo

University of Granada

Department of Computer Architecture and Computer Technology, ETSIIT and CITIC-UGR

18071 - Granada, Spain

pgarcia@atc.ugr.es

ABSTRACT

Categories and Subject Descriptors

General Terms

Keywords

1. INTRODUCTION

Open Services Gateway Initiative (OSGi)

A State-of-the-Art Survey on Real-Time Issues in Embedded Systems Virtualization

Zonghua Gu, Qingling Zhao

ABSTRACT

1. Introduction

Real-time virtualization

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Hardware resource sharing

Performance characterization from abstract functionality ▶

Platform-based modeling of execution behavior functionality

Combine analytic and experimental target profiling (processor, hardware, FPGA -in-the-loop), interpolation estimates from historical data

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

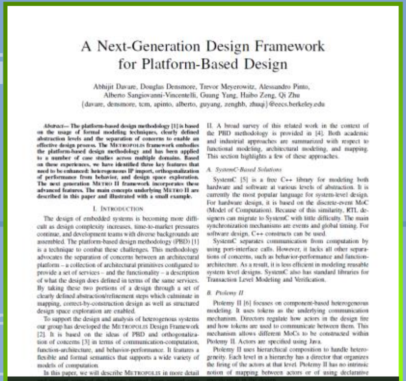
Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Hardware resource sharing

Performance characterization from abstract functionality ▶

Platform-based modeling of execution behavi

Combine analytic and experimental target prof
hardware, FPGA -in-the-loop), interpolation e
historical data



Platform-based design

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Hardware resource sharing

Determination of key test cases for different implementations ▶

Characterization of computational architectures

Static analysis methods, automatic test generation, hardware architecture behavior implementation

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

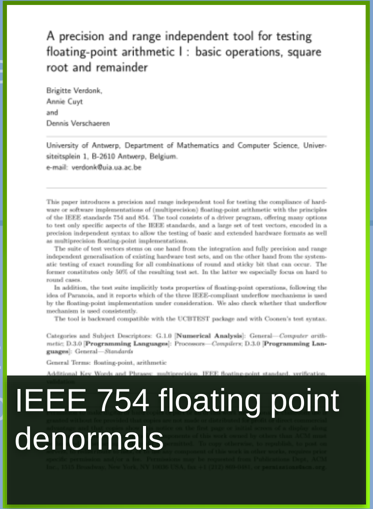
Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Hardware resource sharing

Determination of key test cases for different implementations ▶

Characterization of computational archite

Static analysis methods, automatic test generat
architecture behavior implementatic



A precision and range independent tool for testing floating-point arithmetic I : basic operations, square root and remainder

Brigitte Verdonk,
Annie Cuyt
and
Dennis Verschaeren

University of Antwerp, Department of Mathematics and Computer Science, Universiteitsplein 1, B-2610 Antwerp, Belgium.
e-mail: verdonk@uia.ua.ac.be

This paper introduces a precision and range independent tool for testing the compliance of hardware or software implementations of (multiprecision) floating-point arithmetic with the principles of the IEEE standards 754 and 854. The tool consists of a driver program, offering many options to test only specific aspects of the IEEE standards, and a large set of test vectors, encoded in a precision independent syntax to allow the testing of basic and extended hardware formats as well as multiprecision floating-point implementations.

The suite of test vectors stems on one hand from the integration and fully precision and range independent generalisation of existing hardware test sets, and on the other hand from the systematic testing of exact rounding for all combinations of round and sticky bit that can occur. The former constitutes only 50% of the resulting test set. In the latter we especially focus on hard to round cases.

In addition, the test suite implicitly tests properties of floating-point operations, following the idea of Paranoia, and it reports which of the three IEEE-compliant underflow mechanisms is used by the floating-point implementation under consideration. We also check whether that underflow mechanism is used consistently.

The tool is backward compatible with the UCBTEST package and with Coonen's test syntax.

Categories and Subject Descriptors: G.1.0 [**Numerical Analysis**]: General—*Computer arithmetic*; D.3.0 [**Programming Languages**]: Processors—*Compilers*; D.3.0 [**Programming Languages**]: General—*Standards*

General Terms: floating-point, arithmetic

IEEE 754 floating point denormals

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Hardware resource sharing

Safety of heterogeneous system ensembles

Modeling the semantics of time	Dynamically mixing safety integrity levels
Harmonic periods (integer), synchronous (single clock, discrete time), simultaneous, dense (variable step, continuous)	Certification kit for mixed Safety-Integrity Levels (SiL) of components, matching software with hardware

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

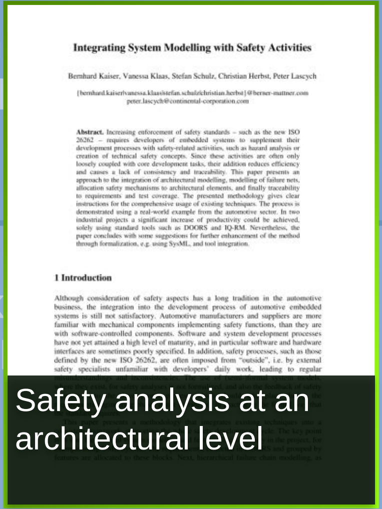
Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Hardware resource sharing

Safety of heterogeneous system ensembles	
Modeling the semantics of time	Dynamically m l
Harmonic periods (integer), synchronous (single clock, discrete time), simultaneous, dense (variable step, continuous)	Certification k Integrity components, with



Integrating System Modelling with Safety Activities

Bernhard Kaiser, Vanessa Klaas, Stefan Schulz, Christian Herbst, Peter Lascych

{bernhard.kaiser|vanessa.klaas|stefan.schulz|christian.herbst}@berner-mattner.com
peter.lascych@continental-corporation.com

Abstract. Increasing enforcement of safety standards – such as the new ISO 26262 – requires developers of embedded systems to supplement their development processes with safety-related activities, such as hazard analysis or creation of technical safety concepts. Since these activities are often only loosely coupled with core development tasks, their addition reduces efficiency and causes a lack of consistency and traceability. This paper presents an approach to the integration of architectural modelling, modelling of failure nets, allocation safety mechanisms to architectural elements, and finally traceability to requirements and test coverage. The presented methodology gives clear instructions for the comprehensive usage of existing techniques. The process is demonstrated using a real-world example from the automotive sector. In two industrial projects a significant increase of productivity could be achieved, solely using standard tools such as DOORS and IQ-RM. Nevertheless, the paper concludes with some suggestions for further enhancement of the method through formalization, e.g. using SysML, and tool integration.

1 Introduction

Although consideration of safety aspects has a long tradition in the automotive business, the integration into the development process of automotive embedded systems is still not satisfactory. Automotive manufacturers and suppliers are more familiar with mechanical components implementing safety functions, than they are with software-controlled components. Software and system development processes have not yet attained a high level of maturity, and in particular software and hardware interfaces are sometimes poorly specified. In addition, safety processes, such as those defined by the new ISO 26262, are often imposed from "outside", i.e. by external safety specialists unfamiliar with developers' daily work, leading to regular

Safety analysis at an architectural level

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Service utilization

Real-time embedded services operating in a physical environment ▶

Real-time middleware and service oriented architectures with physical capabilities

Service discovery response time (latency, averages, time-out), modal service request behavior

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

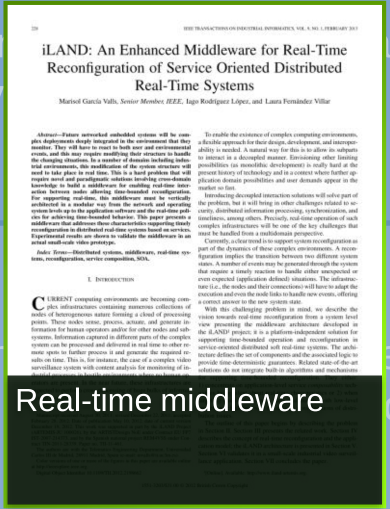
Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Service utilization

Real-time embedded services operating in a physical environment ▶

Real-time middleware ar
phy

Service discovery respor
modal se



iLAND: An Enhanced Middleware for Real-Time Reconfiguration of Service Oriented Distributed Real-Time Systems

Marisol García Valls, Senior Member, IEEE, Iago Rodríguez López, and Laura Fernández Villar

Real-time middleware

A Taxonomy of Discovery Services and Gap Analysis for Ultra-Large Scale Systems*

Joe Hoffert, Shanshan Jiang, and Douglas C. Schmidt
Institute for Software Integrated Systems
Vanderbilt University
2015 Terrace Place
Nashville, TN 37212
{joseph.w.hoffert,shanshan.jiang,d.schmidt}@vanderbilt.edu

Real-time discovery services

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Service utilization

Smart services discovery ▶

Service ontologies with taxonomies for similarity and transformability matching

Type similarity checking and conversion, semantics definition

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Service utilization

Smart services discovery

Service ontologies with taxonomies for sim
transformability matching

Type similarity checking and conversion, sema


Semantic Middleware for the Internet of Things

Zhexuan Song, Alvaro A. Cárdenas, Ryusuke Masuoka
Fujitsu Laboratories of America, Inc.
{zhexuan.song, alvaro.cardenas-mora, ryusuke.masuoka}@us.fujitsu.com

Abstract

Semantic middleware

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Service utilization

Information sharing in a heterogeneous system ensemble

Language and ontology infrastructure to support translation and transformation

Reliable model and code generation

Infrastructure

Conveniently, efficiently, and consistently collaborate between stakeholders throughout the system life cycle

Reliably configure flexible system configurations for features with varying quality of service

Contract out system resources and balance use of external resources for resiliency and runtime cost optimization

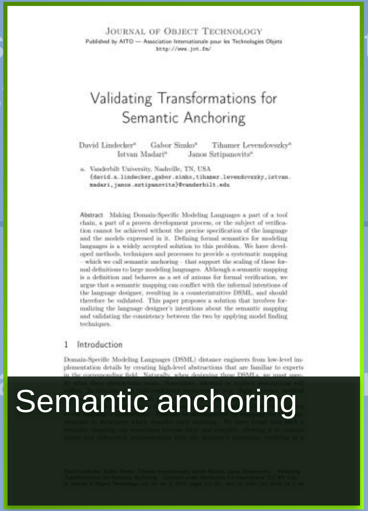
Dynamically assemble systems post-deployment and purpose available functionality to serve singular needs

Service utilization

Information sharing in a heterogeneous system ensemble

Language and ontology infras… tra… trans…

Reliable model a… n



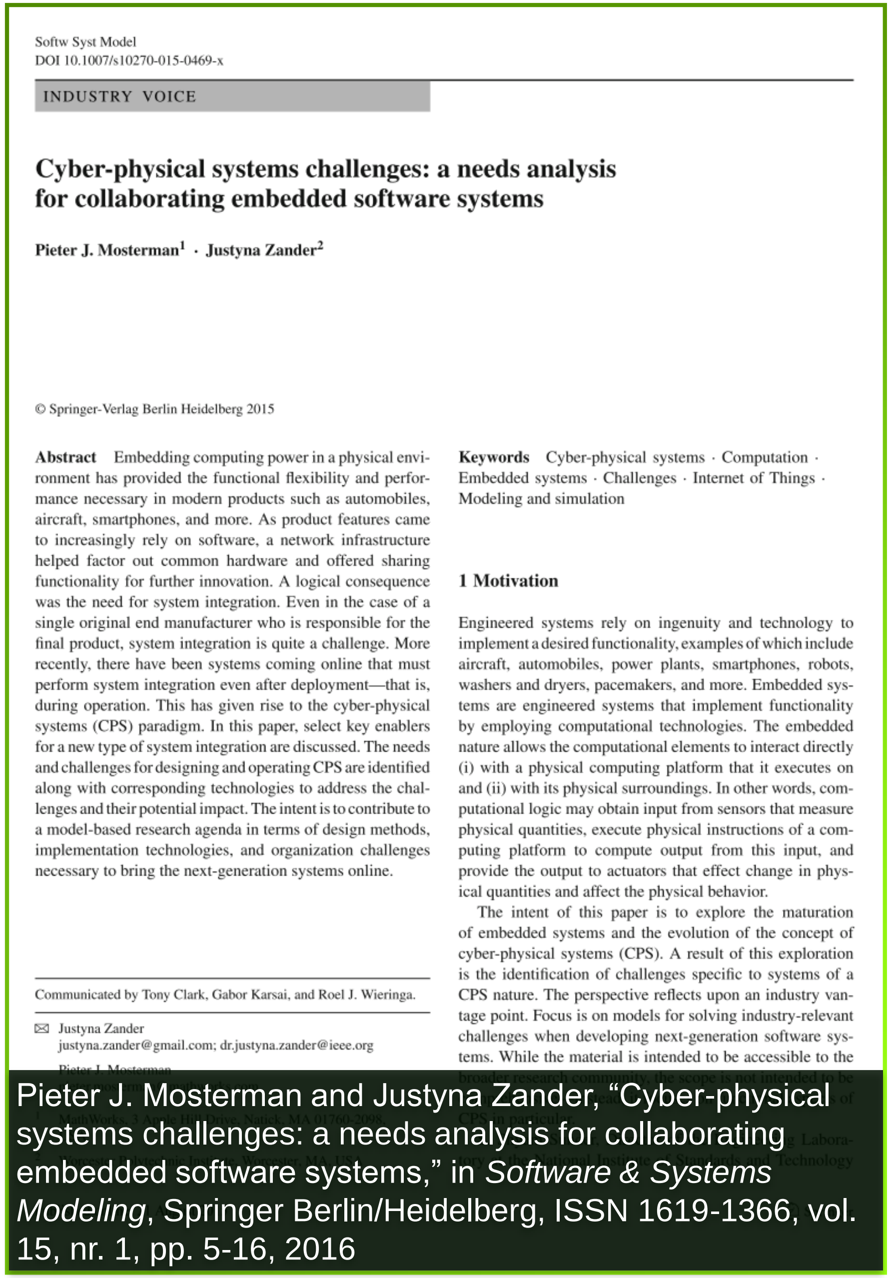
Journal of Object Technology

Validating Transformations for Semantic Anchoring

Semantic anchoring

Knowledge Representation and Reasoning on the Semantic Web: OWL

Semantic web ontology language



Softw Syst Model
DOI 10.1007/s10270-015-0469-x

INDUSTRY VOICE

Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems

Pieter J. Mosterman[1] · Justyna Zander[2]

© Springer-Verlag Berlin Heidelberg 2015

Abstract Embedding computing power in a physical environment has provided the functional flexibility and performance necessary in modern products such as automobiles, aircraft, smartphones, and more. As product features came to increasingly rely on software, a network infrastructure helped factor out common hardware and offered sharing functionality for further innovation. A logical consequence was the need for system integration. Even in the case of a single original end manufacturer who is responsible for the final product, system integration is quite a challenge. More recently, there have been systems coming online that must perform system integration even after deployment—that is, during operation. This has given rise to the cyber-physical systems (CPS) paradigm. In this paper, select key enablers for a new type of system integration are discussed. The needs and challenges for designing and operating CPS are identified along with corresponding technologies to address the challenges and their potential impact. The intent is to contribute to a model-based research agenda in terms of design methods, implementation technologies, and organization challenges necessary to bring the next-generation systems online.

Communicated by Tony Clark, Gabor Karsai, and Roel J. Wieringa.

Justyna Zander
justyna.zander@gmail.com; dr.justyna.zander@ieee.org

Keywords Cyber-physical systems · Computation · Embedded systems · Challenges · Internet of Things · Modeling and simulation

1 Motivation

Engineered systems rely on ingenuity and technology to implement a desired functionality, examples of which include aircraft, automobiles, power plants, smartphones, robots, washers and dryers, pacemakers, and more. Embedded systems are engineered systems that implement functionality by employing computational technologies. The embedded nature allows the computational elements to interact directly (i) with a physical computing platform that it executes on and (ii) with its physical surroundings. In other words, computational logic may obtain input from sensors that measure physical quantities, execute physical instructions of a computing platform to compute output from this input, and provide the output to actuators that effect change in physical quantities and affect the physical behavior.

The intent of this paper is to explore the maturation of embedded systems and the evolution of the concept of cyber-physical systems (CPS). A result of this exploration is the identification of challenges specific to systems of a CPS nature. The perspective reflects upon an industry vantage point. Focus is on models for solving industry-relevant challenges when developing next-generation software systems. While the material is intended to be accessible to the

Pieter J. Mosterman and Justyna Zander, "Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems," in *Software & Systems Modeling*, Springer Berlin/Heidelberg, ISSN 1619-1366, vol. 15, nr. 1, pp. 5-16, 2016

https://dl.acm.org/citation.cfm?id=2890224

The Smart Emergency Response System Using MATLAB and Simulink

https://www.youtube.com/watch?v=oofHMaEWwP8