Workshop Proceedings: Trustworthy Cyber-Physical Systems

J. Fitzgerald, T. Mak, A. Romanovsky and A. Yakovlev (Eds.)

Abstract

These are the Proceedings of the Workshop on Trustworthy Cyber-Physical Systems run on September 3, 2012 in conjunction with CONCUR 2012.

For more information see:

http://www.staff.ncl.ac.uk/terrence.mak/TCPS/Workshop.html
**Added entries**

NEWCASTLE UNIVERSITY  

**Abstract**

These are the Proceedings of the Workshop on Trustworthy Cyber-Physical Systems run on September 3, 2012 in conjunction with CONCUR 2012.

For more information see:  
http://www.staff.ncl.ac.uk/terrence.mak/TCPS/Workshop.html

**About the editors**

John Fitzgerald is Director of the Centre for Software Reliability (CSR) at Newcastle. He is a specialist in the engineering of resilient computing systems, particularly in rigorous analysis and design tools. In his research, he develops model-based methods and tools to help in the design of particularly challenging types of product, especially systems that require collaboration between engineering teams of differing backgrounds and disciplines. For example, he currently leads the international COMPASS project, which is developing technology for engineering complex "Systems-of-Systems" that are built from pre-existing systems that might never have been designed with collaboration in mind. On a different scale, he leads Newcastle's research into co-modelling and co-simulation in the design of fault-tolerant embedded systems (in the DESTECS project and in our EPSRC platform grant on Trustworthy Ambient Systems). John is probably most closely associated with the Vienna Development Method (VDM) which has been developed from its logical foundations to a commercial tool-supported method, with industry applications in areas as diverse as options trading and firmware design. He recently led work in the Deploy project on achieving and demonstrating dependability through the deployment of formal methods in four industry sectors. His project on the use of formal models to support collaborative modelling and simulation in the design of embedded systems (DESTESCS), started in January 2010. John studied formal proof (PhD, Manchester Univ.), before joining Newcastle, where he worked on formal design techniques for avionic systems with British Aerospace in the 1990s. He went on to study the potential for industrial application of formal modelling (specifically VDM) as a SERC Fellow and later as a Lecturer at Newcastle. He returned to the University in 2003, having established the design and validation team at Transitive, a successful SME in the embedded processor market. John is Chairman of FME, the main European body bringing together researchers and practitioners in rigorous methods of systems development. He is a Fellow of the BCS, and a member of the EPSRC College. He is a member of the ACM and IEEE.

Dr. Terrence Mak (Ph.D, Imperial) is a lecturer in the School of Electrical, Electronic and Computer Engineering at the Newcastle University. He is also an affiliated lecturer in the Institute of Neuroscience at the same University. Supported by the Royal Society, he holds a Visiting Scientist position at MIT. His current research focuses on many-core and VLSI systems design, and explores applications in biomedical and neural engineering. His researches are supported by EPSRC, TSB and The Royal Society. He is a member of the IET, IEEE and Society for Neuroscience. Previously, he worked with Turing laureate Prof. Ivan Sutherland at Sun Microsystems Labs (Now Oracle Labs) at Menlo Park. He was awarded the Croucher Foundation Scholarship and the US Naval Research Excellence in Neuroengineering Award. He has published over 40 research papers in peer-reviewed journals and international conferences and was awarded the Best Paper Award in 2011.

Alexander (Sascha) Romanovsky is a Professor in the Centre for Software and Reliability, Newcastle University. His main research interests are system dependability, fault tolerance, software architectures, exception handling, error recovery, system structuring and verification of fault tolerance. He received a M.Sc. degree in Applied Mathematics from Moscow State University and a PhD degree in Computer Science from St. Petersburg State Technical University. He was with this University from 1984 until 1996, doing research and teaching. In 1991 he worked as a visiting researcher at ABB Ltd Computer Architecture Lab Research Center, Switzerland. In 1993 he was a visiting fellow at Istituto di Elaborazione della Informazione, CNR, Pisa, Italy. In 1993-94 he was a post-doctoral fellow with the Department of Computing Science, the University of Newcastle upon Tyne. In 1992-1998 he was involved in the Predictably Dependable Computing Systems (PDCS) ESPRIT Basic Research Action and the Design for Validation (DeVa) ESPRIT Basic Project. In 1998-2000 he worked on the Diversity in Safety
Critical Software (DISCS) EPSRC/UK Project. Prof Romanovsky was a co-author of the Diversity with Off-The-Shelf Components (DOTS) EPSRC/UK Project and was involved in this project in 2001-2004. In 2000-2003 he was in the executive board of Dependable Systems of Systems (DSoS) IST Project. He has been the Coordinator of the Rigorous Open Development Environment for Complex Systems (RODIN) IST Project (2004-2007). He is now the Coordinator of the major FP7 DEPLOY Integrated Project (2008-2012) on Industrial Deployment of System Engineering Methods Providing High Dependability and Productivity.

Alex Yakovlev received D.Sc. from Newcastle University in 2006, and M.Sc. and Ph.D. from St. Petersburg Electrical Engineering Institute in 1979 and 1982. Since 1991 he has been at the Newcastle University, where he worked as a lecturer, reader and professor at the Computing Science department until 2002, and is now heading the Microelectronic Systems Design research group (http://async.org.uk) at the School of Electrical, Electronic and Computer Engineering. His current interests and publications are in the field of modeling and design of asynchronous, concurrent, real-time and dependable systems on a chip. He has published four monographs and more than 200 papers in academic journals and conferences, has managed over 25 research contracts.

**Suggested keywords**

DEPENDABILITY
SYSTEM MODELLING
VERIFICATION
Workshop Proceedings

Trustworthy Cyber-Physical Systems

John Fitzgerald, Terrence Mak, Alexander Romanovsky, Alex Yakovlev

Newcastle University

September 3, 2012
Workshop on Trustworthy Cyber-Physical Systems

in conjunction with CONCUR 2012

September 3, 2012

Workshop Background

Cyber-Physical Systems (CPS) has recently emerged as a new paradigm of thinking about complex systems that tightly integrate multiple, networked computing elements (hardware and software) with non-computing physical elements such as electrical or mechanical components. CPS research is in a formative phase in which concepts are created, subareas formed and foundations defined. This workshop brings together researchers and practitioners interested specifically in topic of ensuring the trustworthiness of CPS.

Designing a trustworthy CPS is beyond the state of the art because of the complexity of the systems whose global properties have to be verified, the dependency on off-the-shelf constituent systems, and because of the heterogeneity of the design models and disciplines involved (for example, how do you link a software engineer’s discrete mathematics with control laws described by a systems engineer, in order to gain confidence that their combination will be safe?).

The aim of our workshop is to help develop a community of interest in the science and engineering of trustworthy CPS, by which we mean CPS on which reliance can justifiably be placed.
Workshop Co-chairs
John Fitzgerald (School of CS, Newcastle University)
Terrence Mak (EEE School, Newcastle University)
Alexander Romanovsky (School of CS, Newcastle University)
Alex Yakovlev (EEE School, Newcastle University)

Technical Programme Committee
Michael Butler (University of Southampton, UK)
Benoit Caillaud (INRIA, France)
Patrick Degenaar (Newcastle University, UK)
Luciano Lavagno (Politecnico di Torino, Italy)
Frank Ortmeier (University of Magdeburg, Germany)
Maurizio Palesi (Kore University, Italy)
Isabelle Perseil (INSERM, France)
Chi-Sang Poon (MIT, USA)
Jim Woodcock (University of York, UK)
Programme

9.00 - welcome

9.10 - Invited talk
On an Evaluation Framework for Energy Aware Buildings in UPPAAL for Stochastic Hybrid Automata

Professor Kim G. Larsen, Aalborg University

10.10 coffee

10.30 - Trust Based Secure Cyber Physical Systems
Raja Waseem Anwar and Saqib Ali

11.05 - Mutual Knowledge in Multi-Domain Models
Jelena Marincic and Angelika Mader

11.40 - An Autonomous Vehicle Design for Safe Operation in Heterogeneous Environments
Sonke Eilers, Martin Franzle, Sebastian Gerwinn, Christian Kuka, Soren Schweigert, and Tobe Toben

12.15 - lunch

13.30 - The Co-Simulation of a Cardiac Pacemaker using VDM and 20-sim
Carl Gamble, Martin Mansfield, and John Fitzgerald

14.05 - Timed-pNets: A formal communication behavior model for real-time CPS system
Yanwen Chen, Yixiang Chen, and Eric Madelaine

14.40 - Trustworthiness in Hybrid Bio-Silicon Systems for Next-Generation Neural Prosthetics
Graeme Coapes, Terrence Mak, Jun Wen Luo, and Alex Yakovlev

15.15 - coffee

15.45 - panel, discussion, wrap-up

17.00 - close
Trust Based Secure Cyber Physical Systems

Raja Waseem Anwar\textsuperscript{1} and Saqib Ali\textsuperscript{2}

\textsuperscript{1}Arab Open University, Muscat, Sultanate of OMAN
\texttt{waseem@aou.edu.om}
\textsuperscript{2}Sultan Qaboos University, Sultanate of Oman
\texttt{saqib.ali@ieee.org}

Abstract. Cyber Physical Systems (CPS) is a combination of different embedded subsystems, which work independently of each other and interact with the environment. Embedded systems operate in the presence of inherent uncertainty, context dependencies and adversarial certainty. In the physical world of embedded systems, all operational environments (Cyber, Physical) are adversarial whether by incident or by users, by natural adversaries or by combination of those.

This research has identified the key area which is lack of trust and security among cyber physical systems, although there are many security approaches and methods which are available in order to secure CPS, unfortunately none of them emphasize the issue of trust based secure CPS. In order to achieve trust among Cyber Physical Systems, a two tier blanket approach for achieving the trust among Cyber Physical Systems has been proposed, which mainly consists of internal and external layers of trust.

Keywords: Cyber Physical Systems, Trust, Security, SCADA, embedded systems

1 Introduction

The term "cyber-physical systems" refers to the tight conjoining of and coordination between computational and physical resources. This research foresees that the cyber-physical systems of tomorrow will far exceed those of today in terms of adaptability, autonomy, efficiency, functionality, reliability, safety, and usability. Cyber-Physical Systems (CPSs), which are integrations of computational and physical processes, are becoming increasingly common in today’s society.

Cyber-Physical System (CPS) is a network of hybrid (self-building) or, combination of physical entities and software systems that control them [1]. Cyber-physical systems are tight integration of computation, networking, and physical objects, in which embedded devices are networked to sense, monitor, and control the physical world. CPS bridges the virtual world of computing and communication and the real world [2].

The integration of computing and physical processes is not new. Cyber-physical systems exist today, but in a much smaller scale in size and complexity than the anticipated CPS of the future. The revolution will mainly come from massive networking of embedded computing devices such as sensors and actuators [13].

CPS has many applications in a wide variety of systems including medical systems, industrial monitoring, agriculture and avionics [3]. A concept map of CPS is presented in Figure 1.

![Cyber-Physical Systems – A Concept Map](Adapted from [25])
In the last few years there is a tremendous increase in the number of various computers of everyday use. Modern computers are becoming smaller and smaller, while equipped with higher performance like computational speed and memory size, as stated by Moore’s law. As a consequence, computers are transforming into a lot of new forms. Some examples of new forms of computers include mobile phones, smart sensors, and even ordinary physical things, such as a lamp, a table and a cup [30].

In other words, many physical things will possess computing and communication capabilities at different levels, which are provided by small and (possibly) invisible computers embedded therein. This integration of networked computing and physical dynamics has led to the emergence of cyber-physical systems (CPS), which have become very popular in recent years [18] and this makes CPS distinct from other types of cyber / computational systems.

Thoshitha T. Gamage, Bruce M. McMillin, Thomas P. Roth, [19], explain that numerous integration levels within modern systems can be broadly categorized into two domains: a cyber-infrastructure (computations, control algorithms, decision engines, databases, etc.) and a physical infrastructure (physical processes and components, links and connections, etc.). Commonly known as Cyber-Physical Systems (CPSs), such systems are put together to provide better resource utilization, control, fault tolerance and performance.

One prominent feature in CPSs is that embedded computers and communication networks govern both physical manifestations and computations. This, in turn, affects how the two primary infrastructures interact with each other, and the outside world. From a functionality point of view, a CPS can be regarded as the intersection of properly built individual (cyber, physical, network) components.

This, unfortunately, is not the case in terms of security. Treating CPS security in a disjoint manner allows unintended information flow. Access-control-based methods and information-flow-based methods are the two primary approaches to system security policies and mechanisms.

1.1 Trust and Security in CPS

According to Siani Pearson and Azzedine Benameur [16], Trust is a complex concept for which there is no universally accepted scholarly definition. Evidence from a contemporary, cross-disciplinary collection of scholarly writing suggests that a widely held definition of trust is as follows [20]: “Trust is a psychological state comprising the intention to accept vulnerability based upon positive expectations of the intentions or behavior of another”. Yet, this definition does not fully capture the dynamic and varied subtleties involved: trust is a complex notion and a multi-level analysis is important in order to try to understand it. There are many different ways in which online trust can be established: security may be one of these (although security, on its own, does not necessarily imply trust) [21]. Some would argue that security is not even a component of trust: Nissenbaum argues that the level of security does not affect trust [22]. On the other hand, an example of increasing security to increase trust comes from people being more willing to engage in e-commerce if they are assured that their credit card numbers and personal data are cryptographically protected [23].

There can be differing phases in a relationship such as building trust, a stable trust relationship and declining trust. Trust can be lost quickly, as Nielsen states [24]: “It [trust] is hard to build and easy to lose: a single violation of trust can destroy years of slowly accumulated credibility”.

The estimation and control algorithms used in CPS are designed to satisfy certain operational goals, such as, closed-loop stability, safety, liveness, or the optimization of a performance function. Intuitively, our security goal is to protect these operational goals from a malicious party attacking targeted cyber infrastructure [14].

In traditional security models, a security perimeter is setup to create a trust boundary within which there is self-control over computer resources and where sensitive information is stored and processed. Moreover [17], security and trust are two tightly interdependent concepts and because of this interdependence, these terms are used interchangeably when defining a secure system [34]. However, security is different from trust and the key difference is that, it is more complex and the overhead is high.

Trust has been the focus of researchers for a long time [35], from the social sciences, where trust between humans has been studied, to the effects of trust in economic transactions [36-38]. Although intuitively easy to comprehend, the notion of trust has not been formally defined. Unlike, for example, reliability, which was originally a measure of how long a machine can be trustworthy, and came to be rigorously defined as a probability, trust is yet to adopt a formal definition.

Understanding the notion of trust is the key to model trust properly in a specific discipline. Trust is an old but important issue in any networked environment; it has been there all the time and people have been using it in their daily life interactions without noticing, that is, buying and selling, communicating, cooperating, decision-making etc. involve some sort of trust without paying attention to it as a specific phenomenon.

Security, however, also needs to deal with non-operational goals. For example, if the measurements collected by the sensor network contain sensitive private information, it must ensure that only authorized individuals can obtain these data. Since
components in Cyber Physical Systems are networked at every scale. The behavior of a Cyber Physical System is a fully-integrated hybridization of computational (logical), physical, and human action.

2 Background and Literature Review

Cyber Physical Systems have been growing for the last few years, and this is due to the advancement in technology more specifically in sensing, computing and networking, deeply embedded sensor and wireless sensors/actuator networks.

Table 1 presents the literature review and analysis on the literature carried out on Cyber Physical Systems with their areas of focus.

<table>
<thead>
<tr>
<th>Article</th>
<th>Area of Focus</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>[6] Jin Wang et al.</td>
<td>Presented an introduction of cyber physical enhanced secured wireless sensor networks (WSNs) integrated cloud computing for u-life care architecture called (CPeSC3) architecture. Their proposed CPeSC3 architecture is composed of three main components, namely communication core, computation core, and resource scheduling and management core. In security core, concatenation is applied between source sensor node and random number to provide protection against attacks and then encrypt them to provide anonymity of the source node against some attacks.</td>
<td>• In this work authors describe how to enhance secure wireless sensors networks, and integrate it to cloud computing. • Security issues are up to certain level, but nothing related to trust in CPS.</td>
</tr>
<tr>
<td>[7] Michael Kirkpatrick et al.</td>
<td>Proposed hardware-based lightweight, security technique for CPS. They have considered the use of Physically Unclonable Functions (PUFs) to bind an access request to specific hardware with device-specific keys. Their PUFs are implemented in hardware, such as SRAM, and can be used to uniquely identify the device. This technology could be used in CPS to ensure location based access control and encryption, both of which would be desirable for CPS implementations. However, PUFs are deterministic, as repeating the PUF, evaluation on the same hardware device will always produce the same value. Thus, PUFs can be used to confirm the unique identity of a hardware device.</td>
<td>• In this work, they consider hardware-based techniques to enhance CPS device based performance, which is one of major area in CPS, but again no discussion or inclusion of component in CPS.</td>
</tr>
<tr>
<td>[8] Guowei Wu et al.</td>
<td>Proposed an access control scheme called fault-tolerant emergency-aware access control (FEAC). Their control scheme provides a proactive and adaptive access control policies especially to address multiple emergencies management problem and supposes the fault-tolerant scheme for CPS applications. They have introduced Priority and Dependency-Action Generation Model (PD-AGM) to select the optimal response action path for eliminating all the active emergencies within the system. The priority and dependency relationships of emergencies are used to exclude the infeasible response action paths and relieve the emergencies combination state explosion problem.</td>
<td>• In this work, authors have discussed issues related to proactive and adaptive access control policies based on the integration of hardware and software based. • Moreover issues raised will work up to certain levels, lack of major security issues and no discussion related to Trust based CPS.</td>
</tr>
<tr>
<td>[9] Christopher</td>
<td>Developed three novel software methodologies to</td>
<td>• In this work the researcher</td>
</tr>
<tr>
<td>Reference</td>
<td>Authors</td>
<td>Methodology</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>Zimmer et al.</td>
<td>Provide enhanced security in deeply embedded real-time systems. They have attained elevated security assurance through two levels of instrumentation that enable us to detect anomalies, such as timing dilations exceeding WCET bounds. The three software methodologies that they have used are T-Rex, T-ProT and T-AxT. These methodologies detect more intrusions that harm security in CPS.</td>
<td></td>
</tr>
<tr>
<td>[10] Quanyan Zhu et al.</td>
<td>Proposed 6-layer security architecture for cyber physical systems motivated by the OSI and PRM models. First, they have addressed the security issues present at each layer and pinpointed a holistic viewpoint for security solutions in CPS. They have also proposed a game-theoretical model that builds bottom up from the physical layer and argued that the saddle-point solution to the dynamic game gives rise to a cross layer security policy.</td>
<td></td>
</tr>
<tr>
<td>[11] Nayot Poolsappasit et al.</td>
<td>Proposed a method for security risk assessment called Bayesian Attack Graphs (BAGs). Their model incorporates the usual cause consequence relationships between different network states (as in attack graphs and attack trees) and, in addition, takes into account the likelihood of exploiting such relationships. They proposed a method to estimate an organization’s security risk from different vulnerability exploitations based on the metrics defined in the Common Vulnerability Scoring System (CVSS). Finally, they modeled the risk mitigation stage as a discrete reasoning problem and used genetic algorithm to solve it. The algorithm can identify optimal mitigation plans in the context of both single and multi-objective analysis.</td>
<td></td>
</tr>
<tr>
<td>[12] Teodor Sommestad et al.</td>
<td>Described a framework for assessing the security of information systems while considering uncertainty. They presented on a model based assessment framework for analyzing the cyber security provided by different architectural scenarios. Their framework uses the Bayesian statistics based Extended Influence Diagrams to express attack graphs and related countermeasures. The approach allows calculating the probability that attacks will succeed and the expected loss of these given the instantiated architectural scenario. Moreover, the framework can handle the uncertainties that are accompanied to the analyses. Their model can be merged with architecture Meta models by using a concept called Abstract models.</td>
<td></td>
</tr>
</tbody>
</table>

Based on literature review and analysis, this research supports the initial findings that most of the studies target specific domain in CPS, while lacking the trust and security issues which are considered vital today and in future. Major characteristics of Trust are reliability, safety, security, privacy and usability.

In CPS, security should be considered as part of the CPS architecture and application development other than just applying security solutions [4]. The security issues are crucial for CPS applications because the entities within the systems not only interact with each other, but also with the physical environment, thus the security issues must be addressed [8].
2.1 Attacks in CPS:
Securing the emerging internet and information-rich CPSs is difficult because a number of cyber-security attacks. Some of the common attacks in CPS are described below, [5].

**Eavesdropping**
Eavesdropping refers to the attack that adversary can intercept data and any information communicated by the system. CPS is particularly susceptible to eavesdropping through traffic analysis such as intercepting the monitoring data transferred in sensor networks collected through monitoring. Eavesdropping also violates the user’s privacy such as a patient’s personal health status data transferred to the system.

**Compromised-Key Attack**
An attacker can gain access to a secured communication without the perception of the sender or the receiver by using the compromised key. The attacker can decrypt or modify data by the compromising key, and try to use the compromised key to compute additional keys, which could allow the attacker to access other secured communications or resources.

**Man-in-the-Middle Attack**
In man-in-the-middle attack, false messages are sent to the operator, and can take the form of false negative or false positive messages. This may cause the operator to take an action, such as flipping a breaker, when it is not required, or it may cause the operator to think everything is fine and not take action when an action is required.

**Denial-of-Service Attack**
This type of attack prevents the legitimate traffics or requests for network resources from being processed or responded by the system. The denial-of-service attack prevents normal work or use of the system. After gaining access to the network of cyber physical systems, the attacker can always do any of the following,
- Flood a controller or the entire sensor network with traffic until a shutdown occurs due to the overload.
- Send invalid data to controller or system networks, which causes abnormal termination or behavior of the services.
- Block traffic, which results in a loss of access to network resources by authorized elements in the system.

Another new type of attack, in which an attacker might be able to launch to control systems (i.e., attacks which are not possible in traditional IT systems), one of them is:

**Resonance attacks**
The attacker that has compromised some sensors or controllers will force the physical system to oscillate or change at its resonant frequency [21].

**Jamming the communication attack**
In [26], researchers assess the security of Cyber Physical Systems through experimental evaluation of two different components in target tracking applications; in the first component they assess the impact of low level jamming attacks on convergence properties in different target tracking scenarios. In the second component they assess the intelligent jamming attack technique in which an attacker aims to increase his/her reward based on various jamming attack policies and actions.

**Integrity attack**
In this type of attack in CPS, an attacker wishes to disturb the system by injecting external control inputs and fake sensor measurements [27].

3 Threats/uncertainties and protection of Assets in Cyber Physical Systems
Like any other system, security is a key challenge for the deployment of CPS. CPS needs protection from threats which in turn utilize the vulnerability and cause damage to CPS.

Referring to standard GB/T20984 [20], information systems assets can be classified into the following six classes: data, software, hardware, service, people, and other assets. Threats can be classified into hardware and software breakdowns, miss
operation, physical environment influences, management problems, malicious code, ultra vires, cyber-attacks, physical attacks, breach of confidence, falsification, repudiation and vulnerabilities are as follows: physical damages, network vulnerabilities, operating systems vulnerabilities, problems in technique and organization management, application middleware vulnerabilities, application systems vulnerabilities, and management problems.

The indicators mentioned above are very general which can be more detailed and further divided into sub indicators. For information security assessment, indicators can be selected according to the security goal of unique information system.

Alvaro A, Saurabh Amin, Bruno Sinopoli and Annarita Giani [20], highlight the key challenges for securing Cyber Physical Systems which are:

1. Understanding the threats, and possible consequences of attacks.
2. Identifying the unique properties of Cyber Physical Systems and their differences from traditional IT security.
3. Discussing security mechanisms applicable to cyber physical systems.

On the other hand, in order to understand the new classes of threats in Cyber Physical Systems such as the smart grid and SCADA (Supervisory Control and Data Acquisition Systems), it is useful to characterize the interactions based on the domain that is the origin of threat, and the domain where the impact is felt [21].

Cyber threats include those into confidentiality or integrity of data in the system, including connecting to a device on a home area network and retrieving usage data or modifying billing information etc. One major characteristic of the cyber threats is that they are scalable, i.e. they are easily automated and replicated and one should expect them to propagate freely across untrusted domains. Cyber physical threats are those that originate within the cyber domain but which have an impact on the physical characteristics of a system.

Today’s Cyber Physical Systems require a tight knit information and communication capability, because of the vulnerability of the internet communications, protecting the system will require new technology to enhance security of power system command, control and communication [22].

4 Need for trust-based secure CPS

Today Cyber Physical Systems are an emerging and growing category of systems that combine the physical systems with the computational in a holistic way [23]. The key property of these systems is that functionality and salient system properties are emerging from an intensive interaction of physical and computational components, as the computational components are aware of their physical context, they are intrinsically distributed in time synchronizing, have to cope with uncertainty of sensoric-input and need to produce real-time interactions.

CPSs are often designed as networks of interacting elements which are not limited to: Automotive systems, medical monitoring, process control systems (Smart Grid), SCADA and distributed robotics.

Due to this integrations of CPS (Cyber and Physical components), which work collaboratively as embedded systems, there are many challenges between these two (Cyber and Physical) domains, forming trust and security is one major challenge.

Currently, there are no defined procedures to evolve Cyber Physical Systems with respect to Trust based Security, threats to CPS, asset identification etc., there is dire need for such procedures.

The increasing complexity of components and the use of more advanced technologies for sensors and actuators, wireless communication and multicore processes pose a major challenge in building CPS.

In this research we have identified the main problem which is: to establish trust in relation to security among Cyber Physical Systems.

Sub-Problem 1:
Evaluate and analyze the existing security mechanisms in Cyber Physical Systems in comparison with trust based CPS.

Sub-Problem 2:
Evaluate and assess the role of Trust and Security in SCADA (Supervisory Control and Data Acquisition Systems) and within Smart Grids and to develop a comprehensive understanding of the security vulnerabilities in SCADA systems.

Sub-Problem 3:
Evaluate systems availability and scalability, since cyber and physical components depend on each other and some of the CPS are very large (such as nationwide power grids).

4.1 Significance:

Cyber Physical Systems can be applied in a wide range of domains. Potential applications of CPS include assisted living, integrated medical systems, safe and efficient transportation, automated traffic control, advanced automotive systems, autonomous search and rescue, energy conservation, energy efficient buildings, environmental control, factory automation, home automation, critical infrastructure control, distributed autonomous robotics, defense, and so on. Ubiquitous applications and services that could significantly improve the quality of our daily lives will be enabled by CPS, which will make applications more effective [2].
5 Trust-based secure cyber physical systems approach

To create trust based secure cyber physical systems, we are proposing two tier trust oriented approach, this trust approach is further classified into:

**Internal Trust:** responsible for different internal trusted entities such as sensors, actuators, communication networks

**External Trust**, responsible for physical environment or architecture of cyber physical systems

Trust levels can vary within a system depending on the specific functions on the time, context, and dynamically changing circumstances of its use. Achieving a trust in CPS is very tiresome and challenging; in our approach we create a boundary of trust internally and externally around CPS as depicted in Figure 2.

Integration of these two levels of trust will be achieved by implementing a two tier blanket approach; where this blanket covers the whole CPS and hide it from outside world. We need to further investigate about how these two trusts can be defined and achieved; their integration and coordination issues among them will be further analyzed.

![Fig. 2. Trust based secure CPS approach](image)

Cyber Physical Systems are different from desktop computing, traditional embedded /real-time systems and today’s wireless sensor networks and they have some defining characteristics.

A CPS is envisioned to be a heterogeneous system of systems, which consists of computing devices and embedded systems including distributed sensors and actuators. These components are interconnected together in a large-scale and execute autonomous tasks to link the cyber world and the physical world [28].

Following are the main components of our proposed trust based CPS approach and their roles:

- **Physical System**: Collection of actuator units
- **Sensors and actuators**: which are used to monitor and measure various physical processes. Sensors and actuators provide the interface between the physical and cyber worlds. A sensor is a device that measures a physical phenomenon, e.g., room temperature and converts physical phenomena into information which contains the attributes, sampling, timestamp and/or space stamp. An actuator on the other hand, is a device that is able to change attributes of a physical object, e.g., move a chair or physical phenomena [29].
• **Communication Network (wired / wireless):** Communication network provides the connection between Cyber and physical systems through wired or wireless network.

• **Cyber System:** Collection of control logic, sensor units and critical infrastructures.

• **Internal Trust:** Trust state among physical systems, sensors and actuators and internal communication networks.

• **External Trust:** Trust state among internal trust and external entities.

• **Boundary:** Parameterized boundary where trust will be established.

• **Attacks:** All kinds of external attacks on CPS.

Trust is an important concept with respect to security. Trust is a belief that an entity will behave in a predictable manner in specified circumstances. The entity may be a person, process, object or any combination of such components. Trust is like bonding a relationship among cyber and physical components.

Establishing a trust among internal and external components in any cyber physical system, i.e. (power grids, SCADA, Oil and Gas fields, aviation, water distribution and healthcare) is challenging due to the growing concern over cyber-attacks targeted towards these infrastructures. One major limitation of today’s cyber physical systems: is no established threat model to characterize the security needs and the level of acceptable risks.

**Trust Management and Attribution** [29]: The cyber infrastructure in power system domain can be viewed as interconnected, “islands of automation”. This interconnection brings about inherent trust concerns as vulnerabilities in other domains may abuse trust relationship. In addition, if an organization has a system affected by a security event, than that information may not be communicated to all concerned domains, therefore the decreased trust is not appropriately communicated to all other systems.

- **Trust Management Lifecycle:** The dynamic environment of the smart grid requires a trust model, which allows continual reevaluation. Since the smart grid will likely exhibit emergent behaviors, trust management must remain flexible to address continual modifications in usage and misuse patterns. Trust management policies should follow specific tailoring of these changes.

- **Formal Trust Representation:** Investigate quantified notions of trust, specifically representing impact to control and privacy data. Develop algorithms to evaluate a trust revision impact on grid reliability.

- **Insider Threat Management:** While most cyber protections attempt to limit external attacks, recent events have increased concerns from malicious insiders. Utility employees are typically highly trusted to efficiently manage and operate the grid; however, nefarious actions by these individuals could produce disastrous results.

- **Attribution:** The ability to attribute actions back to a system or user is imperative to identify malicious actors. By developing strong attribution mechanisms, the individuals responsible for a cyber-attack can be identified and penalized. Additionally attribution provides a method to deter future malicious activities.

In addition to the above we have identified the following issues which are not limited but could highly influence establishing trust among CPS components:

- Technical implications
- Practical implications
- Management implications
- Access control to vendors
- Standardization: (up to what extent)
- Vender influence
- Weakest link (people working for particular organization)

Nowadays Information and Communication Technology (ICT) services are becoming increasingly dependent upon CPS to provide automated and efficient management of essential services; care has to be taken to ensure that they are secured.

According to Eric Ke Wang [5], a highly confident cyber physical system should satisfy the following objectives, while we consider “Trust” is another important objective.

**Confidentiality:**

Confidentiality refers to the capability to prevent the disclosure of information to unauthorized individuals or systems. Confidentiality is necessary to maintain the users’ privacy in Cyber Physical Systems. Realizing Confidentiality in CPS must prevent an adversary from inferring the state of the physical system by eavesdropping on the communication channels between the sensors and the controller, as well as between the controller and the actuator.
Integrity:

Integrity refers to the trustworthiness of data or resources that cannot be modified without authorization. Integrity in CPS could be the capability to achieve the physical goals by preventing, detecting, or blocking deception attacks on the information sent and received by the sensors and the actuators or controllers. A lack of integrity results in deception.

Availability:

For any system to serve its purpose, the service must be available when it is needed. High availability of CPS aims to provide service by preventing computing, controls, communication corruptions due to hardware failures, system upgrades, power outages or denial-of-service attacks.

Authenticity:

In computing and communication process it is necessary to ensure that the data, transactions, communications are genuine. It is also important for authenticity to validate that both parties involved are who they claim to be. In CPS, the authenticity aims to realize authentication in all the related processes such as sensing, communications, actuations.

Trust Oriented:

According to Blaze, M., Feigenbaum [15], Trust is context-dependent (the trust relationships are only meaningful in the specific contexts), dynamic, non-monotonic and function of uncertainty.

Types of Trust:

- Interpersonal (agent & context specific)
- Structural (system within which trust exists)
- Dispositional (independent of agent & context)

Trust as an essential attribute in building a relationship between entities has been studied for a long time by scientists from disparate scientific fields. Every field has examined modeling and calculating trust using different techniques, and one of the most prominent and promising technique is the use of statistics, mainly probabilities to solve the problem, especially in dynamic networks where the topology is changing rapidly.

The nature of CPS is integrated and embedded (sensors, actuators, communication network). However, to achieve Trust based CPS in distributed environments, such components should have a common vocabulary to allow them to communicate with each other regarding security attacks and countermeasures in a trusted way. For that purpose, we would like to employ two tier blanket approach that allows sharing a common understanding of information about information security (security attacks and trust, in particular) among both humans and software agents.

6 Conclusion

In order to achieve secure cyber-physical systems, it is required to incorporate trust factor into account at the very start of the design process for such systems. CPS is opening up extraordinary opportunities for research and development in numerous disciplines. Our research findings highlighted the key area of lacking trust in securing cyber physical systems. This blend of trust with CPS to achieve security is still unsolved and challenging. We presented two tier trust based blanket approach to achieve security in CPS.

This research will be further extended to implement trust based approach to find out how much we succeeded in securing CPS. This research will also analyze the existing security mechanisms in cyber physical systems in comparison with trust based CPS.
References


16. “A survey of Trust Management Systems (power point presentation)”, Dalal Al-Arayed (PhD – IT Student), Supervised by: Joao Pedro Sousa, PhD Assistant Professor, CS, GMU


28. “Spatio-Temporal Event Model for Cyber Physical Systems”, Ying Tan, Mehmet C. Vuran and Steve Goddard, CSE conference and workshop papers, University of Nebraska – Lincoln


30. Feng Xia and Jianhua Ma (Natural Science Foundation of China Grant Number: 60903153, the fundamental Research Funds for Central Universities (DUT10ZD110), and the SRF for ROCS, SEM), Building Smart Communities with Cyber-Physical Systems.
Mutual Knowledge in Multi-Domain Models

Jelena Marincic
and Angelika Mader

University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science, Enschede, The Netherlands
(j.marincic, a.mader)@utwente.nl

Abstract. Keywords: co-simulation, modelling method, model validation, confidence in the model

1 Introduction

Model driven development is a popular and promising approach for improving quality and efficiency in the design and construction of embedded systems. When we come to systems of systems, however, it quickly becomes obvious that a whole system cannot be put into one single model. Systems of systems require systems of models. Furthermore, different domains, that are involved in an integral view of a system, need their own models. Each domain has its own dedicated languages and tools, where the phenomena and characteristics of the domain can be expressed and analyzed in a suitable way. When models do not have a common mathematical basis, as it is for continuous time and discrete event perspectives, the need for multi-domain models is especially urgent.

Having models in different domains describing different aspects of one system, a number of question has to be answered. One is, how these models or their results can be coupled or integrated. Another is, how independent are these models? What does a modeller in one domain has to know about models of another domain, when both describing parts of a systems of systems? We investigate this question in the context of a framework, where a continuous time model and discrete event model are coupled by co-simulation, as performed in the DESTECS project [2]. We will approach the problem using an example of a small robot. In this context, we focus on the question what a software modeller has to know about the continuous time domain and the modeling decisions taken there.

A basic claim in our work is that modelling is not a purely formal process. What abstractions to take, what the system fragment to model, what behaviour to describe, e.g., are decisions that are taken in a non-formal way. However, these decisions are based on both, structured arguments and experience. In earlier work we investigated the question what decisions have to be taken while modelling. A taxonomy of modelling decisions [10] can help to identify and acquire the relevant knowledge about a system and its environment for the modelling process.

In multi-domain modelling, as we consider it in this work, the relevant knowledge extends by additional restrictions caused by modelling decisions taken in
another model and specifics of the other model domain. For continuous time
models and discrete event models one part of an answer is rather generic, and
can be found in the literature, as, e.g. sampling time, taking into account jitter
and delays and mapping control architecture to software architecture. The non-
generic, and therefore more difficult part of the answer is, where a real plant
has to be addressed, and detailed knowledge about its specific mechanics and
electronics become relevant. We try to identify categories of modelling decisions
to take in this context of multi-domain models. Goal is that the explicit state-
ment of decision categories helps the modeller to guide the modelling process, to
identify the necessary decisions to take, and to improve the quality of the mod-
els constructed. Furthermore, we claim that by making modelling steps explicit,
also the re-use of models is supported and the modelling process becomes more
efficient.

2 Introduction to our Working Example

Before we proceed with our analysis of the modelling process, we will introduce
our running example and quickly describe the Destecs multi-domain modelling
framework.

DESTECS (Design Support and Tooling for Embedded Control Software) [2]
is a consortium of companies and research groups founded with the goal to
develop a tool and a method that supports concurrent modelling for simulation
of embedded systems. The idea of this approach is to couple the models made
by control and software engineers from early design stages. Control and software
engineers design their own models, using their mathematical formalisms and
tools, but within a joint framework that forces them to define what are the
variables and parameters shared with the other model [5][4]. The advantage of
this approach is that it lets both groups of experts work in their domain, in
which mature methods and tools exist and which are well known to experts. At
the same time it forces them to communicate early in the design process and
identify the trade-offs between different system requirements. This shortens the
integration time, in comparison when these two aspects are designed in isolation.
The DESTECS approach also pays a special attention to modelling faults and
fault-tolerance mechanisms, as they also pose requirements on both control and
software.

Control engineering is concerned with dynamical system behaviour. 'Dy-
namical' means that the behaviour over time is relevant, it implies long-term
qualitative properties of the system behaviour. For example, the system desired
behaviour is formulated in form of specifications for dynamical behaviour of tem-
perature, pressure, position, speed of the plant components and processes. An
idealised design process steps could be described as a sequence of the follow-
ing tasks: (1) Model the plant, (2) Design the control law, (3) Simulate to test
whether the control forces the plant to behave as required.

The plant model is based on the laws of physics, it is a transfer function that
describes the dependence between the plant output (controlled) signal and the
The plant input signal generated by the controller. The plant is belongs to continuous domain, and the control is in discrete domain. Digital control theory is based in the mathematical framework that supports design and analysis of a digital control system that forces the continuous plant to perform required dynamical behaviour. In the project, 20-sim [1] tool is used to model the plant, describe its control laws and simulate it.

On the discrete event side, the VDM (Vienna Development Method) tool [3] is used to model the software. VDM describes the software on an abstraction level higher than that of the implementation. This allows the designer to model the software specification and analyse the software in early design phases, before going into the details of implementation. The modern versions of VDM are object-oriented languages that also support modelling or real-time aspects, such as the scheduling policy of the underlying operating system and distribution to different processors. VDM is a formal language, with formal semantics, therefore software properties can be formally checked. Integrity properties, which are formal property descriptions can be generated automatically using VDM tools. Also, it is possible to enable test coverage documentation, using testing techniques. The third way to validate the model is to execute models together with the code, using the tool's graphical end.

2.1 Line-Following Robot

The example that we will use is the project's pilot study of modelling a line following robot. The robot is a small cart with two wheels, each moved by a servo motor. Also, each wheel is connected to an encoder that senses the wheel's angular position. At the robot's front there are two infrared sensors that sense the lightness on the floor based on infrared reflection. Also, two infrared distance sensors mounted on the robot sense objects based on reflected infrared light. A contact switch detects bumping to another object or a wall. The requirement for the robot is to follow a black line drawn on the ground, using two infrared sensors.

As it is the case in a pilot study, different models were designed, with different software and controller architecture, and while this was done, the control and software experts learnt from each other about the other domain.

The high-level model structure is shown on Fig. 2. This is just a high-level diagram, with many details abstracted away as they are not relevant for our discussion. For example, the loop control does not actually show the feedback loop, and the tool does not directly connect the contract with the two models, but they are equipped with interfaces towards the contract. Also, the DE control architecture is not given in our diagram, but will be discussed later.

The plant is modelled using domain-independent bond graphs, the environment of the robot, which is the ground with a black line is described with the file that contains the lines coordinates. The loop control is distributed to two processors, each controlling one motor. It is a PID control, with a standard control structure, which usually asks for tuning its parameters in simulation and by doing experiments.
The control that sends referent values to the wheels, that switches between control modes, is in the DE side. Also, the part of the control that deals with different faults is on the DE side.

The contract contains parameter and variables that are read by one of the models an written into by the other. The parameters shared in this particular case are the lateral and longitudinal offset of the robot. The variables that are shared are the values of encoders, which the CT side writes into, and the DE side reads. Another set of values read by the DE side, are the values of infra-red sensors. The DE side writes into the variables that set the referent speeds for the right and the left wheel (in case of a curvy path, the wheels will have different speeds).

The contract also gets the information about the faults detected from CT side, and the value of LED diodes is written by DE side, so that the operator gets the signal if there is a fault and when the robot is in regular working mode.

With the Destecs tool, the models of the plant and the control are designed, simultaneously with the software models and it is possible to simulate (and visualise) the robot movement along a previously defined line.

2.2 Communication between Experts

The information that the two models share in direct interaction is made explicit by the contract. In case of the robot, CT model will write encoder values, and the DE model will read them. The DE model will set the referent speed values that will be read by the CT model and forwarded to the PID controllers.

But, the communication between software and control (CT and DE) experts contains much more. It is necessary that the two co-modellers communicate ??.

An example of the information that was necessary to exchange between the experts is different interpretation of the angle direction - CT side used anti-clock wise direction for positive angle, while the DE side used the clock-wise direction. This reflects some of the implicit assumptions hidden in an engineering discipline culture.

At the moment, it is being investigated how to improve the contract to achieve consistency between the models and whether more detailed documentation would
help in this communication. This is especially important when the two teams are working in geographically isolated locations.

3 A Non-Formal Framework for Communicating Modelling Decisions of CT and DE Modellers

The goal of our work is to give a non-formal framework for communicating modelling decisions for CT and DE models, to uncover hidden assumptions made by one modeller about the other model. Perhaps, there are sources of inconsistencies between the two models that can be formalised, by designing an ontology, like for example in the work of Lickly et al. [7], or a meta-model, as done in the work of Lee et al. [6].

But, before formalising, it is important to investigate what needs to be compared and what are the general sources of conflicts between these two models.

In our previous work, we explored the ways to deal with non-formal aspects of modelling decisions. Modelling is only partially formal activity, as there are a number of decisions that cannot be formalised and made part of a tool or a general-purpose language. To structure these decisions, we developed a taxonomy and used it to analyse modelling decisions a posteriori, to come up with guidelines to develop next system generation [8]. We also used the taxonomy to structure validation of the model that raises the confidence of the modeller and the model stakeholders that the model adequately represents the system with respect to the requirements [9].

Our previous experience in analysing and designing models of two industrial systems showed that the general categories of our taxonomy are useful only when they are refined into problem-specific modelling steps. These steps are refined to suit the problem in hand, but general enough to be reused in similar products, like in case of product lines of product generations.

If for a modelling process a number of issues are relevant: efficiency, understandability, repeatability, and the competence of the resulting model. In the context here, we look closer at the model competence, and find that consistency is a closely related concept. Possible inconsistency is an issue for any set of submodels. When submodels are in different domains the possibilities for formalisation of inconsistencies get even less. Within the DESTECS project we investigate the question how to achieve competence and how to avoid inconsistencies of the two co-models. Our approach is to look at the modelling decisions made by control and software engineers from the perspective of our taxonomy.

Competence is about a model representing the system correctly, with respect to the requirements of the system for which the model is investigated (sometimes these are referred to as a purpose of the model). Competence is a relationship between a model and the system that the model represents, whereas consistency is the relationship between the models. For example, models in the two domains may be constructed with the same wrong assumption. They may be consistent under this shared wrong assumption, but the composition does not lead to an competent model of the system. Also, two models may be inconsistent even
though each of them, analysed in isolation, is competent. This might happen when two models are designed to show different properties of a system, e.g. one is made for showing stability when slowing down, the other one to show that a robot can follow a straight line. Possible inconsistency is an issue for any set of submodels. When submodels are in different domains the possibilities for formalisation of inconsistencies get even less.

Both inconsistency and incompetence of submodels can lead to incompetent co-models, where the reason is lack of knowledge about the other domain (model).

![Diagram](image)

Fig. 2. Distinguishing model’s adequacy and consistency.

### 3.1 Classes of incompetence and inconsistency

Below we give a list of sources of inconsistency and incompetence. From a logical perspective it is easier to describe when a model is inconsistent (or incompetent), than when it is consistent (competent): the number of reasons for a model to be inconsistent (incompetent) is huge. We cannot assume that we cover all of them, but a model is only consistent (competent) if it avoids all possibilities of inconsistency (incompetence).

Ideally, we would like to have all sources of inconsistencies and incompetences to be automatically detected by a tool. Unfortunately, this is not possible in general. Some can be found by a tool straightforwardly (syntax check), for some we could add information to make them automatically detectable, as, e.g., by extended types. Still a major part cannot be automatically treated. For these we suggest to use the list below as guidelines, in the sense that if we address issues listed there explicitly, we decrease the chance of getting inconsistent and incompetent models.

Incompetence and inconsistencies can be introduced in initial models or during the evolution of models. As this does not make a principle difference for the sort of incompetence and inconsistencies, we do not distinguish these two phases in our analysis.
**Contract level syntactic inconsistency:**
A co-model is syntactically inconsistent if both models do not address the same set of variables, parameters and events, with identifier names, types and ranges.

**Contract level semantic inconsistency:**
If the variables, parameters and events do not address the same phenomena in both models, the co-model will be inconsistent, e.g. a variable that represents speed of a wheel is interpreted differently in the sub-models. In the motion systems and autonomous vehicles, it is very important to always interpret correctly spatial information. Studying the documentation of the robot case, we found that the two groups of experts interpret differently the angle values, the CT modeler assigned the positive angle in the anti-clock wise direction, whereas the DE modeller assumed that the angle goes up in the clock-wise direction. Therefore, for these kinds of systems it should be established beforehand a referent system and the initial robot position and orientation in this referent system.

In the robot two encoders are connected to the right and the left wheel, and it is important that their connected with the correct connectors of the interface card, which is a piece of hardware used to connect the encoder and the CPU. Otherwise their identities can be mixed (the left wheel can be read as the right while and vice versa). In the robot case, there are only two wheels to control, but in the systems where many units of the same kind (e.g. beds in hospital connected to a central computer that monitors patients) can be easily accidentally wired wrongly.

Also if the measurement units do not coincide, models are inconsistent.

**Variable propagation inconsistency:**
It might be the case that at the interface/contract variables are interpreted consistently, but , e.g., while passing them to other components of the control software, the semantic interpretation of the variable changes, leading to inconsistency.

In the robot model, we noticed that the DE model reads the variables values and then distributes these values from one variable to another. In copying the value from variable ‘a’ to variable ‘b’, and then copying the value of the variable ‘b’ to variable ‘c’ and so on, it is extremely important to keep in mind what this value represents. Theoretically, it could happen that the encoder value is copied into the variable that represents the speed or the variable that represents the value of a different encoder. In case of robot, if it would follow a straight line, this would not be easily noticed, as the two wheels will always have the same speed.

**Initialisation phase inconsistency:**
For the initial phase of a simulation run (and also a real system execution) assumptions are made on the initial values of variables, about start-up procedures (e.g. a control may assume that a robot starts up with the robot head in zero-position. In reality the position of the robot-head is where it was at the end of the last robot operation. Finding out the real position of the head may be part of a start-up procedure that is called homing), and the availability of values in buffers.
**Mode change incompetence:**
A plant may operate in different modes, e.g. line following mode and turning mode for a robot. When changing modes we also have assumptions on variable values for the new mode starting. In a generalized view, this may also be considered as an initialisation phase inconsistency, but we want to distinguish it here for explicitness.

Usually there is a start up mode, during which the plant moves to stationary, working state; then there is the working mode that also can consist of different modes; finally, shut-down mode puts the vehicle into a safe position.

Control theory assumes linear systems within of a mode, but the switch from one mode to another can potentially be non-linear, which may bring unwanted behaviour. The physical behaviour of a plant is possibly not in a steady state immediately after mode change, but needs a time to achieve a steady state. In control engineering we speak "bumpless transfer" when these transition phases are properly addressed. An example is the robot in the line following mode, that will need some time to reach the intended speed. This is a common phenomenon occurring in practical control design. If it is not addressed, it will lead to incompetent models. If only addressed in the plant model, and not in the control model, we will have inconsistent submodels.

An example of the transfer between modes may mean unwanted sudden changes of currents and voltages, but it can also have consequences in not controlling the plant parts. For example, a vehicle similar to our robot was designed to carry packages along a partly inclined surface. Two different control modes were used to control the vehicle, and as the vehicle climb with a high speed, the packages bounced off it, when it reached the horizontal surface. The bouncing was an uncontrolled phenomena, left there as it did not pose any danger. But, these 'in-between' modes behaviours need to be carefully examined, and sometimes control has to change to avoid unwanted or unpredictable behaviour.

**Sampling time incompetence:**
Control theory states that for the discrete control of a continuous plant a certain sampling period is necessary to ensure that the plant behaves as intended. The sampling time depends in the first place on the dynamical plant behaviour and the physical properties of the plant. As there is an infinite set of possible dynamic plant behaviours, the requirements determine the ones of interest. If the correct sampling time is not considered in the control model, the overall model is not competent.

Computation time incompetence: Limitations of the control platform (hardware and operating system) may lead to jitter, i.e., variation of delays in reading sensor values and writing actuator values. This may lead to violation of the hard real time constraints of the control law of the computation, i.e., the next steering value is not available before the next sampling moment. (see also footnote 1 below for more explanation of the problem). There is a number of solutions to the problem. If the problem is not addressed, the submodels may not be competent. If the chosen solution is not communicated from the control engineer
to the software designer, then there may occur inconsistencies within the DE submodel.

The robot control law takes into account the calculation time, so the calculation based on the measurements in the moment k*T, is calculated to be written into actuators in the (k+1)*T moment in time, which is the next sampling time.

Quantization incompetence:
A sensor measuring some quantity gives typically an analog value as result (voltage). By quantization, i.e., digitalization of an analog value, the accuracy of the value decreases (i.e. the accuracy depends on the (size of the) representation). If the DE model does not cope with this inaccuracy, the DE model will not be competent. An example may be that we integrate about (inaccurate) speed to derive the position of an object, which will by "wrong" to the extent of the inaccuracy of the speed.

Boundary incompetence:
When a quantity to measure is beyond the maximum level of the sensor, then there are different ways how the sensor writes its measurement. One possibility is that the sensor keeps writing its maximal value. If this maximal value is, e.g., misinterpreted as a constant quantity, then this may lead to an incompetent or inconsistent control, i.e. an incompetent or inconsistent DE model.

System/environment inconsistency:
Often, we do not model a "complete" system, but only a fragment of it (e.g. when modeling a car, a nuclear power plant, etc.). It has to be well defined what the borders of the system that we model are, and what assumptions we have about the environment (e.g. when there are shared resources, about the availability of these resources, e.g. for the robot: what is friction of the ground, is the ground flat, is there an inclination? e.g. for a automotive system: is the ECU used available often enough, do the buses used have enough capacity?) Both sub-models have, of course, to address the identical fragment of the system. This may be trivial for toy examples, but for real cases it is non-trivial.

Assumption inconsistency:
Assumptions are in most cases implicit. Documenting assumptions are not actually part of the tool, but should be part of a modelling methodology. Examples for inconsistent assumptions can, e.g., be found in the previous paragraph on system/environment inconsistency. In general, assumptions may refer to all parts of a system and its environment, the plant the control and their surrounding (the operator, humidity, temperature etc.).

Abstraction inconsistency:
Abstraction is one of the basic techniques of modelling, "leaving away" non-relevant aspects. It has as a consequence that all models are "wrong", but what we try to achieve is that we still can deduce some properties of the real system from the properties of the model. Obviously, if we make different abstractions in the sub-models, the co-model may become inconsistent. As an example we take here the case where friction and torque are modelled in the CT model of the robot, where it is possible that under certain circumstances slip occurs (wheels rotating without the robot moving). A control model on the DE side that has
abstracted away from slip, may interpret the movement of the wheels wrongly and come to wrong conclusions about speed and position of the robot.

**Idealisation inconsistency:**

Idealization is similar to abstraction. But, whereas in abstraction we leave something away, for an idealisation we introduce simplified behaviour. The classical example is the point mass modelling in physics. In reality, there are no point masses, but for modelling it is a useful idealisation, that allows for a simpler mathematical description, and gives satisfactory results. A typical idealisation in control is that there is no delay in reading, writing or computing. If idealisations on delay are different in submodels, this may lead to inconsistency.

An idealisation typical for control design is idealising the plant as linear. The plant model is derived from the first principles (established laws of physics), or empirically (in black box modelling process) and it is not entirely accurate. There are two reasons for having an inaccurate plant model. First, it is sometimes impossible to derive an accurate plant model. Second, even when this is possible, such a model would be very complex, too complex to manipulate it and to derive the control. So, a simplified model is derived, simple enough to manipulate it, but complex enough to give predictions that are accurate enough.

This does not only enormously simplifies the control design, but it also makes it a lot easier to reason about the system. It would be very hard to understand the dynamics of a system without first focusing to the special case of linear systems. When modelling a system composed of electrical, hydraulic, mechanical, optical and so on components that interact with each other, the modeller starts with linear systems, and then, if necessary, extends the model with non-linear components.

If we would extend our robot to be a part of a network of autonomous robots that can communicate with each other, there would be a potential danger of electro-magnetic interference of the vehicles’ sensors. The plant may have to be re-modelled, which as a consequence would need a different control laws, which would again affect the higher control layers modelled on DE side.

**Idealisation incompetence:**

A typical example from control engineering is a robot that bumps into a wall. Physically, this is a phenomenon with very fast, but continuous dynamics. Physically, this is a phenomenon with very fast dynamics. The robots speed and acceleration during bumping change with a very high frequency. If the details of this transition from one steady state (before bumping) to the next steady state (after bumping) are not of interest, bumping into the wall can be represented in the model in a much simpler way. For example, in hybrid systems community, it is common to model it as an event that changes the speed and acceleration values from some value to zero, instantly.

**Modelling trick inconsistency:**

Tools provide languages to model a system. Typically, these languages are limited in their expressiveness, and often we have to squeeze a model into the modelling language of a tool. Experienced users of a tool know “the way how to model for
this tool”. A modelling trick describes a different behaviour that the "natural" one, simply because of the expressiveness of the tool. If the modelling trick is not made explicit to the other domain modeller, an inconsistence co-model may be the consequence.

In Uppaal, for example, we add additional variables, states or automata just to be able to formulate logical expressions that check properties of interest. If we would use the model in combination with another model, it should be made clear that these additional elements do not represent elements of the system, even if they look like they do.

4 Discussion and Conclusion

The aim of our framework for communication between the CT and DE modeller is to help to improve the modelling process. It contains two different kinds of sources of inconsistency and model incompetence. One kind reflects the general differences between models used in control theory and in computer science. It is about how timing, quantization, limitations of hardware that implements control laws, as well as the software.

Another kind of possible sources of conflicts between the two models comes from different interpretation of the data exchanged in the contract. In our running example, the encoder readings are the data that the DE model receives, and this is further interpreted as the knowledge about the robot movement, position and so on.

The reason to have a non-formal classification is that not all of this knowledge can be formalised. Some of these elements can be formalised and become part of a language, or an ontology or a pattern. But, some of the knowledge will always stay non-formal. Having these classes should help find the formalisable ones and refine the others into some sort of a checklist or guidelines for modellers.

The categories we talked about are not something unknown to the experts. Some of these things are known to control engineers and not to software engineers. But each of these groups is deeply settled into their own conceptual worlds and consequently to their own mathematical frameworks, that seeing the systems from the perspective of the other discipline is very difficult. Taking a pragmatic view, until there is a common semantics or a mathematical framework that covers both worlds, it is useful to have a list of things that one expert needs to understand from the other expert world.

Acknowledgements: We thank Xiaocheng Zhang and Jan Broenink of Twente University for discussions and feedback on the robot case and control engineering problems in general.

References

An Autonomous Vehicle Design for Safe Operation in Heterogeneous Environments

Sönke Eilers\(^1\), Martin Fränzle\(^1\), Sebastian Gerwinn\(^1\), Christian Kuka\(^1\), Sören Schweigert\(^1\), and Tobe Toben\(^2\)

\(^1\) OFFIS, 26121 Oldenburg, \(<\text{firstname}\>\text{.}\text{lastname}@offis.de}
\(^2\) BTC Embedded Systems AG, 26135 Oldenburg, \text{tobe.toben@btc-es.de}

Abstract. As autonomous vehicles are increasingly deployed outside closed environments, safety consideration for their use in outdoor environments is of crucial importance. A complete safety case for such systems covers uncertain perception of the environment, motion planning within such an environment and finally potential failures in the execution of the planned behavior. Here, we present a design for safety of an autonomously operating transport vehicle considering all of these three aspects thereby providing clear arguments for such a safety case that this system is acceptably safe to operate on publicly accessible spaces and in heterogeneous outdoor environments.

1 Introduction

Autonomous vehicles are increasingly used in logistics and factory automation. With the evolution of navigation capabilities and obstacle avoidance techniques, they have reached a degree of autonomy that allows their operation beyond closed work environments accessible to authorized personnel only. In such heterogeneous outdoor environments autonomous systems face new challenges, such as the avoidance of moving obstacles and navigation under changing environment conditions. Traditionally, automated vehicles are considered as machines and considerations on functional safety follow mechanical engineering principles, where the focus is on component failures and the primary goal is to increase the component reliability. After integrating these components the safety risks of the system are assessed and if necessary, a stage of risk reduction is performed, \textit{e.g.}, by installation of additional safety devices. Those devices are usually certified or evaluated, such that their failure probability can be estimated. According to EN 1525, a dedicated safety standard for automated guided vehicles, safety devices on an automated vehicle have to be able to detect test bodies of a certain size and comply to certain safety categories (EN 954 and ISO 13849). With a higher speed and dynamic trajectory planning, verifying the safety of an autonomous vehicle becomes a challenging task. The new standard ISO 26262 for the automotive sector requires the definition of a safety case that provides evidence for a system’s safety. For autonomous vehicles, similar requirements in future standards can be envisioned. Kelly \textit{et al.} \cite{6} state: “The safety case is the document, or set of documents, presenting the argument that a system is acceptably
safe to operate in a given context.” Here, we will present a design for safety for an autonomous transport vehicle (ATV) together with the corresponding safety considerations and thereby providing the basis for the necessary arguments of such a safety case of an ATV following this design.

2 Design for Safety

We are interested in a logistics application in a scenario in which the autonomous system is equipped with more sensors (mobile and stationary) than the traditional automated guided vehicles. The concept mainly builds on the fact that the sensed area is increased due to the use of multiple sensors and hence has the potential of driving at higher speeds while still maintaining a high level of safety illustrated in Fig. 1.

2.1 Architecture

For the ATV we assume a common architecture as illustrated in Fig. 2 as a basis for the following safety considerations. First, information about the environment is gathered via on-board and/or external sensors to build up a context model. This context model is then used to make predictions about future changes in the environment, other traffic participants in particular. Based on this prediction, a trajectory is planned to follow a route to a given goal. As the ATV has to act on incomplete and possibly erroneous information about the environment, we have to show that such incomplete knowledge in the environment cannot lead to collisions, i.e., only safe decisions are taken by the ATV. Specifically, we consider three different sources of partial knowledge (marked as red in Fig. 2):

1. Measurement uncertainty: Sensor measurements are often subject to noise, which has to be accounted for in the data fusion process. Here, we assume that the noise also reflects the possibility of not detecting objects.

2. Movement uncertainty: Future movement directions of other traffic participants are unknown to the system and therefore a worst case has to be considered.

3. The implemented control action from the actuator might differ from the desired ones such that planned trajectories might lead to collisions due to these differences. Therefore, this uncertainty has also to be considered during trajectory planning to ensure collision avoidance.
In the following argument, we will demonstrate how these non-determinisms can be treated safely, such that the resulting actions/trajectory can be guaranteed to be collision-free. At this stage we will neglect technical failures due to individual components such as the actuators. These have to be analysed separately and can be dealt with standard methods such as Fault Tree Analysis, see [7].

The overall arguments is as follows: First, we will show, how uncertain sensor information can be used to obtain an upper bound on the occurrence probability of an object at a particular position at the current point in time. Second, we will define a dynamic model, which again gives an upper bound on this occurrence probability, however at any point in time in the future. Finally we have to show, how this information can be used to plan a collision free trajectory, which leads to the desired goal.

2.2 Context model

In this section, we show, how sensor information can be incorporated into an occupancy map ensuring a safe over-approximation. Specifically, we calculate a probability of different cells $c_{xy}$ being occupied. Thus $P_t(c_{xy} = 1)$ indicates the probability that the cell at position $xy$ is occupied at time $t$, whereas $c_{xy} = 0$ means that the cell is free. As stated previously, the goal is to compute an upper bound on this probability based on the information provided by the sensors. As the probability varies over time, we have to condition on a point in time, for which the probability is estimated. Technically, we represent the sensor measurement and dynamic evolution of the objects as a dynamic state model in which the occupancy grid is the state variable, which evolves over time and from which we can only make noisy observations. The underlying model is similar to the one used in [1,5] and can be described by the following two aspects: (1) Definition of the dynamics, (2) Incorporate sensor information. Each of these steps have to result in an occupancy grid, which needs to be pessimistic, i.e., is only allowed to overestimate and must not underestimate the occupancy probability.

System Dynamics. To define the dynamics pessimistically, we define dynamics by assigning an upper bound on the transition probability $P\left((x', y') \xrightarrow{\Delta t} (x, y)\right)$ from a cell $c_{x',y'}$ to another cell $c_{x,y}$ within the duration $\Delta t$. Depending on the distance between source and target cell, the bound will decrease to zero. We denote the support of this transition probability with $R_{xy}$.

If we start with a given probability map at time $t$, we can then calculate the pessimistic update at time $t + \Delta t$:

$$P_{t+\Delta t}(c_{xy}) = 1 - \left( \prod_{(x',y') \in R_{xy}} \left(1 - P_t(c_{x',y'})P\left((x', y') \xrightarrow{\Delta t} (x, y)\right)\right) \right) \cdot (1 - P_t(c_{xy})\lambda_{xy}) \left(1 - P_{xy}^s\right)$$

$$\leq 1 - \prod_{(x',y') \in R_{xy}} \left(1 - P_t(c_{x',y'})\right) \left(1 - P_t(c_{xy})\lambda_{xy}\right) \left(1 - P_{xy}^s\right)$$

(1)

For example a disk with radius given by the maximal allowed velocity
where $\lambda_{xy}$ is a predefined rate with which objects might generate new objects to model a person leaving a car or a car leaving a garage. $P_{xy}^s$ is the probability with which new objects might spawn at location $xy$. Both rates are free parameters and have to be set reasonably. For example the spawning probability might be higher at the border of the map, where new objects can enter the grid for the first time. For equation (1), we used the worst case bound by using the trivial bound $\leq 1$ on the transition probability. However, we could also obtain a tighter bound and be more efficient, if we used the trivial bound on the acceleration instead and thereby bounding the probability distribution over velocities. Note that, by using this pessimistic dynamical model, probability mass will be generated, reflecting aging of knowledge and decreasing reliability of occupancy information in absence of fresh sensory information. This process can, however, be counterfeited by fresh information in terms of sensor information. As the operation has to be performed at every time step $\Delta t$, we have to guarantee that it can be carried out in real time. For this, we note that if we represent equation (1) in the log-domain, it can be written as a filter operation commonly used in image processing (see also equation (4)).

**Probabilistic Sensor Information.** To incorporate the measurements from different sensors, we use a Bayesian approach \[1,5\]. Many sensors, e.g., laser scanners, do not directly measure grid-cells and only provide partial information. Specifically, for the laser scanner, the (uncertain) information about occupied cells must be inferred from distances to objects along a beam originating from a specific position. To this end, we first transform the current occupancy grid (given by $P_t(c_{xy})$) into a polar occupancy grid, centered on the laser scanner. As several cells from the original grid might overlap with a cell in the newly constructed grid $P_t(c_{r,\psi})$, we assign the maximal occupancy probability of all overlapping cells to this polar-grid cell. To incorporate a measurement from a laser scanner in the form of a measured distance $d_i$ along a beam with angle $\psi_i$, we need upper and lower bounds on the likelihood $p(d|r,\psi) \leq p(d|r,\psi) \leq p(d|r,\psi)$. These bounds can be obtained from worst and best case environmental conditions for the sensor, or by accessing the systematical and statistical error typically given by the technical specifications of a sensor. To calculate the posterior distribution $p_t(r|d,\psi)$ we first have to calculate the current distribution over distances $p_t(r)|\psi)$ first which in turn can easily be calculated from the current occupancy grid $P_t(c_{r,\psi})$. To obtain an upper bound on the posterior probability, we exploit the bound:

$$p_t(r|d,\psi) = \frac{p_t(r,\psi)p_t(r|\psi)}{\sum_{d'} p_t(d',\psi)p_t(r'|\psi)} \leq \frac{p_t(d,r,\psi)}{\sum_{d',\psi} p_t(d',\psi)p_t(r'|\psi)}$$

(2)

Note that the right hand side can be larger than one, which we can safely reduce to one. However, the efficiency of such a bound mainly depends on the quality of the bounds on likelihood $p(d|r,\psi)$ and the lower bound in particular. For the updated (polar) occupancy grid, we have:

$$P_t(c_{r,\psi}|d,\psi) = \underbrace{p_t(r|\psi)}_{p_t(c=r|c)=p_t(r=c)} \cdot \left(\sum_{s<r} p_t(s|d_i,\psi_i)\right) + P(c_{r,\psi}) \cdot \left(\sum_{s<r} p_t(s|d_i,\psi_i)\right)$$

(3)

This update can be done for each laser-scan beam individually, as the corresponding cells do not overlap, i.e., provide independent information. To obtain an occupancy grid in the Cartesian coordinate system, we use the same over-approximation as before,
that is we assign each cell \( c_{xy} \) the maximal probability among the overlapping cells of equation (3). If information of more than one sensor is to be incorporated into the occupancy grid, we can update the grid sequentially as long as the different sensors are independent of each other (with respect to their causes of errors).

2.3 Prediction

For the reachability, we can use the same dynamic model as for the pessimistic update in equation (1).

\[
\log(1 - P_{t+\tau}(c_{xy})) = \sum_{x',y' \in R_{xy}^\tau} \log(1 - P_{t}(c_{x',y'})) + \log(1 - P_{t}(c_{xy}) \lambda_{xy}) + \text{const}, \quad (4)
\]

where we made the dependency of the reachability region \( R \) on the planning horizon \( \tau \) explicit. Furthermore, we note that \( P_{t+\tau} \) not only gives a bound on the occurrence probability at the time-point \( t + \tau \), but for the complete period \([t, t + \tau]\), because of the pessimistic bound on the transition probability. To bound the computational complexity, the reachability region \( R^\tau_{xy} \) can be computed beforehand (possibly for different types of dynamic objects) and then equation (4) can be implemented as a convolution with a fixed kernel which has to be carried out during runtime. Having calculated this prediction of possibly occupied cells, we can directly evaluate (upper bound) the collision probability of an ATV at future times by summing all cells the ATV will cross during a planned trajectory.

2.4 Trajectory Planning

Safety Assessment of Trajectories. As calculated in the previous sections, we already have a safe upper bound on the occupancy probability for future times and places as given by the occupancy grid. To assess the safety level of a trajectory (given by a sequence of acceleration and steering actions), we have to sum all the grid cells which the ATV will cross during the execution of such a trajectory. At this step, we have to take inaccuracies of the controller, as indicated in Fig. 2, into account. To do so, we have to calculate reachability regions due to these inaccuracies using a model of the car dynamics:

\[
\dot{x}(t) = v(t) \cos(\theta(t)), \quad \dot{y}(t) = v(t) \sin(\theta(t)), \quad \dot{v}(t) = u_1 a, \quad \dot{\theta}(t) = \frac{v(t)}{L} \sin(\phi_{\text{max}} u_2),
\]

where \( u_1, u_2 \) are angular and longitudinal acceleration actions, \( a \) is a friction coefficient, \( \phi_{\text{max}} \) is the maximal steering angle, and \( L \) is the length of the car. \( v, \theta, x, y \) are velocity, angular velocity, and positions respectively. Using this model, we can calculate the worst case deviation in \( x, y \) position, if we assume inaccuracies in \( u_1, u_2 \). Again, these regions can be calculated beforehand and only during runtime we have to add these regions to the planned path to obtain a worst case zone in which the ATV is guaranteed to be during the execution of the planned action sequences. For an explicit algorithm to calculate these regions, see [4]. To bound the collision probability of a sequence of actions, we sum all occurrence probabilities in grid cells covered by the reachability regions.
Existence of Safe Trajectories. Once we have access to the collision probability at any point in space and time in the future, we are in the position to validate the trajectories which correspond to different actions such as acceleration and/or steering adjustments. For the safety case, we have to show that we can always find a combination of actions such that the collision probability is smaller than a given threshold. The argument for the existence is analogue to the concept of a safety zone [2]. The sketch of the argument is the following. Initially, we can assume that we have found such a trajectory at the last execution of the planner module (We can assume to start our mission from a safe position with zero velocity, hence the action 'no acceleration' is always safe). Second, we assume, that the time between two execution steps of the planner module is smaller than a given threshold $h$. Let $t_{\text{max\ stop}}$ be the time duration needed for the ATV to achieve a full stop starting from maximal velocity, which defines the safety zone. We assume to have found a trajectory in the last execution step, which is safe for the next $t_{\text{max\ stop}} + h$ time units for a sequence of steering and acceleration actions. As the trajectory from the last step is safe under pessimistic assumptions, it will be safe for the next $t_{\text{max\ stop}}$ time units. Hence, we always have the possibility to brake and will stop within these $t_{\text{max\ stop}}$ units. In [2] a more detailed discussion as well as a formal argument on these real-time constraints for the safety zone can be found. However, we can also choose another action/steering combination, given the safety level as computed in the previous section is acceptable. If we cannot find such a trajectory in time, we have to fall back to the emergency braking maneuver.

Efficiency Considerations. Although the argument above shows, that we can always find a safe trajectory, the procedure is not necessarily efficient. Specifically, as we cannot probe all possible trajectories and evaluate their safety level, we need an efficient procedure to explore the action space. As the focus is on the safety aspects, we do not go into details on how to explore the action space. In general, there are two ways to tackle this efficiency task. On the one hand, one can evaluate the risk associated with a specific action directly during the trajectory generation. Schröder et al. [8] use a modified $A^\star$-Algorithm to generate paths for an autonomous vehicle from path primitives. Their use of a risk map allows the search algorithm to penalize vehicle states which lie in cells with a high associated risk, and thereby directing the search away from critical zones. Although they used this technique for the generation of paths, it can also be used for the generation of actual action sequences, if penalties are introduced. If the above quantification of the safety level is used as risk measure and no trajectory is selected which exceeds the predefined threshold, the corresponding safety level is automatically guaranteed without any further evaluation. On the other hand, one can postpone the risk evaluation completely and only consider possible action sequences during the generation of trajectories [3]. Although a subsequent assessment of the safety level is necessary in order to guarantee a bounded collision probability, it has the advantage of possibly fewer risk evaluations and therefore has the potential of being more efficient.

3 Conclusion

We have presented a design for autonomous transport vehicles together with the necessary safety considerations needed for a complete safety case. The subsequent verification, as required in the safety case, thus can follow a decomposition approach, as the
design facilitates a separation into several safety subtasks. For each subtask we have shown that the corresponding processing step computes safe over-approximations of risk incurred or justifies its action selections w.r.t. such, i.e., bounds the collision probability from above. For this upper bound of collision probability to be valid, we have made several assumptions. First, we assumed an initial probability grid. Although the initial probability map has to be given the probability of a cell being occupied increases exponentially fast over time to 1, if the cell is not measured by any sensor. Second, we have assumed that we have access on bounds for the reliability of the sensors. Finally, the presented safety concept is only valid, provided the processing steps can be performed within a guaranteed duration, finding a suitable trajectory in particular. The timely execution of the processing step has to be verified in a subsequent analysis step. However, as most of the operations needed for the safety considerations here (kernel filtering, evaluation of different trajectories w.r.t. collision probability) scale linearly and can be parallelized, such real time constraints mainly depend on the hardware used.

**Acknowledgments** This work has been supported by the German Federal Ministry of Economics and Technology (BMWi) under the grant 01MA09037 and by the European Commission under the project MoVeS, FP7-ICT-257005.

**References**

The Co-Simulation of a Cardiac Pacemaker using VDM and 20-sim

Carl Gamble, Martin Mansfield, and John Fitzgerald

Newcastle University, Newcastle upon Tyne, NE1 7RU, UK,
FirstName.LastName@ncl.ac.uk

Abstract. Collaborative modelling and analysis are essential for the effective design of trusted cyber-physical systems, including those used in medical applications. We describe the co-modelling and co-simulation of discrete-event descriptions in the Vienna Development Method (VDM), with continuous-time descriptions in 20-sim. We present a co-model of the McMaster-Boston Scientific cardiac pacemaker problem linking a VDM model of a cardiac pacemaker controller combined with a 20-sim model of heart electrophysiology. This is augmented with descriptions of potential heart failures, allowing the level of correction delivered by the pacemaker controller to be assessed.

Keywords: Cyber-physical systems, model-based formal methods, simulation

1 Introduction

In order to gain confidence in the trustworthiness of Cyber-Physical Systems (CPSs), it is necessary to combine heterogeneous models in a disciplined way. For example, ICT and software engineers who naturally use rich discrete-event (DE) models must collaborate with engineers who use continuous-time (CT) techniques suited to the description of the controlled plant and environment. The goal of our work is to facilitate collaborative modelling and simulation, in which engineers to work in the most expressive for the cyber and physical elements of the particular problem, allowing trade-offs across the DE/CT divide. The DESTECs project\(^1\) has developed a framework for collaborative designs (called co-models) composed of models of DE (typically loop and supervisory controller) and CT (typically plant/environment) elements expressed in different formalisms. Reconciled operational semantics for the two formalisms allows the constituent models to run in their own simulators, while a co-simulation engine manages the synchronisation of time and data between them under the control of an external script that can represent environment or user choices. We have developed methods for the construction of co-models [1], patterns for fault tolerance co-modelling [2] and support for design space exploration by multiple co-simulation and ranking of co-models [3]. The approach and tools are being validated in industry on applications including transportation, heavy machinery and high-speed paper processing.

Medical applications are an important domain for CPS technology. In order to provide a common basis for evaluating emerging methods and tools, McMaster University

\(^1\) http://www.destecs.org
and Boston Scientific released a natural language specification for a previous generation artificial cardiac pacemaker\(^2\), inspiring several studies on the use of formal DE-only modelling [4–7]. However, it is appropriate to consider co-modelling in this context, since confidence in such a medical CPS requires models of diverse aspects such as software, electrophysiology and even fluid flow. This paper reports the first attempt to apply co-modelling to the pacemaker, and the first application of DESTECS technology (Section 2) in the medical devices domain. Our objective has been to build a proof-of-concept co-model composed of an abstract CT model of heart electrophysiology in 20-sim and a VDM model of a pacemaker controller (Section 3). We model known heart faults in the CT-side model and demonstrate through co-simulation how the pacemaker design provides resilience to these (Section 4). We draw initial conclusions and suggest future work in Section 5.

### 2 Collaborative Modelling and Co-simulation

A co-model consists of a DE model of a controller and a CT model of the plant, with a contract describing controlled and monitored variables, named events (raised in the CT model and handled by the DE model), and shared design parameters. During co-simulation, user and environmental interactions are governed by a script. We use the Vienna Development Method (VDM) formalism for DE-side, and bond graphs for CT-side modelling. VDM is a rich language with features for modelling object-orientation and concurrency [8], and real-time distributed embedded systems [9]. VDM models define state variables over abstract data types, and functionality via state-changing operations. VDM simulation is supported by the Overture tool\(^3\). CT models are built, simulated and visualised using the 20-sim\(^4\) tool [10]. It allows the dynamics of the plant to be modelled in several ways, including signal type connections between equation blocks, and the powerful domain-independent bond graph [11] notation.

Previous work in embedded systems design, such as BODERC [12] and Modelica [13] provides modelling environments and libraries for simulating physical and computing components. Approaches to DE/CT co-simulation are defined by Nicolescu et al. [14], and there are several co-simulation architectures including Cosimate\(^5\) and HLA [15]. Ptolemy II [16] offers DE and CT simulation within a single tool, though lacking the object-orientation offered by VDM and the component libraries offered by 20-sim. Work on time synchronisation between DE and CT models is described in hybrid systems literature, e.g. [17]. The DESTECS approach is distinctive in including a rich but abstract DE-side modelling language, and in managing co-simulation of heterogeneous models in their “native” tools.

---

\(^2\) [http://sqrl.mcmaster.ca/pacemaker.htm](http://sqrl.mcmaster.ca/pacemaker.htm)

\(^3\) [http://www.overturetool.org](http://www.overturetool.org)

\(^4\) [http://www.20sim.com](http://www.20sim.com)

\(^5\) [http://www.chiastek.com](http://www.chiastek.com)
3 Heart and Pacemaker Co-model

To pump blood around the body the heart must rhythmically contract and relax its upper and lower chambers (atria and ventricles respectively). The atria contract first pumping blood into the ventricles which contract shortly afterwards, pumping blood around the lungs and the rest of the body. The contraction timing is controlled by a pair of electro-chemical nodes, the Sinoatrial Node (SA node) and the Atrioventricular Node (AV node). An electrical discharge from the SA node causes a contraction of the atria and, after a short delay while traversing an internodal pathway, leads to a discharge of the AV which causes contraction of the ventricles. When the system of natural regulation of heartbeat is impaired, an artificial pacemaker may be used (we will use the term “pacemaker” to refer to the artificial device unless stated otherwise). Pacemakers may have one or two electrodes and can be configured for the individual patient. The electrodes can both sense and stimulate the heart. By sensing the heart, the pacemaker can detect when natural stimulation has occurred so as not to interrupt a healthy intrinsic cycle. When natural nodal activity is inadequate, the pacemaker delivers an electrical charge to the heart to stimulate muscle contraction.

The purpose of our co-model is to enable exploration of the effects of pacemaker designs on the heart’s electrical activity. Our co-model therefore consists of a VDM model of the pacemaker and a 20-sim model of heart electrophysiology as this reflects the cyber and physical characters of these two elements (but this is not the only way to structure the co-model - see Section 5). To support simulation in the DESTECS tools, these models are connected by a contract which effectively models each of the two electrodes using a monitored variable (SA/AV Charge) and controlled variable (SA/AV Pacing) pair, Fig. 1. This allows the pacemaker to monitor the electrical discharges within the heart and also administer electrical stimulation via the controlled variables.

Heart Model

Fig. 2 shows the CT model of heart electrophysiology. It contains four distinct areas. The SA and AV nodes have a similar structure in which an integration block represents the store of electrical charge. The integration blocks receive a steady rate of charge, sufficient to give each node its intrinsic rhythm. Each node also contains a discharge condition block which contains the logic controlling when the node discharges and, in the SA node, passes charge to the next node. The feedback path in each node facilitates discharging of the integration block. There is also an area representing the internodal pathway, through which charge passes from the SA to the AV node. The artificial pacemaker block, instead of containing a CT representation of the (cyber) controller, contains the co-model interface used to link to the DE model via the contract.
We model two cardiac arrhythmias as faults in the CT model. In *sinus bradycardia* the SA node discharges less frequently than required, leading to a slow heartbeat. When activated, this fault is modelled by reducing the value of the primary rate in the SB_Fault block. *Third degree AV block* prohibits transmission of charge from the SA node to the AV node and is modelled by the AV_Block fault when activated.

**Pacemaker Model**

The DE pacemaker model monitors the discharges of both heart nodes and stimulates either node as necessary. Under normal conditions the controller alternates between waiting for intrinsic atrial discharge and waiting for intrinsic ventricular discharge (Fig. 3). These discharges are detected by monitoring shared node charge values and looking for a sharp negative gradient (see *Detect_Ventricular_Maxima* operation in Fig. 4). In the event that an expected discharge does not occur the pacemaker administers stimulation to the appropriate node via a non zero value on its pacing lead (*Stimulate_V* operation in Fig. 4), thus compensating for the faulty behaviour.

### 4 Co-Simulation Results

Co-simulation of the pacemaker and heart models in the DESTECS tool yields results presented in Fig. 5 as a graph (generated by 20-sim) of atrial and ventricular node activity in the heart model, where pacemaker interaction can be seen by its effects on these outputs. Figs. 5(a) and 5(b) compare a typical ECG for a healthy heart with the output
from the co-model. Atrial contraction commences at (A) and ventricular contraction at (B). The interval between atrial and ventricular contraction (C) is regular, and the interval between successive heartbeats (D) is regular and within the bounds of 60-100 BPM. Point RP in Figure 5(a) is indicative of cells recharging, but is ignored by the model which focuses only the discharges monitored by the pacemaker.

The effects of fault introduction can be seen in Fig. 5(c), where the output represents a heart with symptoms of AV block. Here the output shows no synchronisation between atrial and ventricular contraction. Fig. 5(d) shows the pacemaker interacting with the heart at points P, where spikes show the presence of artificial stimulation. This stimulation restores synchronisation between contractions, and provides a healthy rhythm.

5 Conclusions and Future Work

We have briefly introduced the DESTECS approach to co-modelling and co-simulation and presented our first co-model of the pacemaker controller. We believe that this is the first attempt at a formal co-model of the McMaster/Boston Scientific pacemaker.
The process of co-model construction is interesting. In order to build a first model of the controller, we began with a rough DE approximation of the heart electrophysiology which was later replaced with the CT model. This “DE-first” approach [1] is one of several methods. For example, a CT-first approach stresses the physics from the outset, eventually replacing controller blocks with richer descriptions in DE formalisms.

The placement of the DE/CT boundary is significant and is governed by the purpose for which the co-model is constructed. In our co-model, the contract models the controlled plant (the leads), and the heart is the environment. If we wanted to focus more signal propagation in the leads themselves, it might be appropriate to model them in greater detail and move them to the CT side.

The fidelity of our initial co-model might be improved. We would like to better represent the charging behaviour of the nodes. The ECG output may be good enough for some analytic purposes but we would like to move from a square wave to curves and spikes, with pulse heights that better represent real values such as in [18]. We plan to model more realistic beat timing variation, and to model more heart blocks including slow conduction and partial loss of electrical charge, tachycardias and junctional arrhythmias. Our DE model is less comprehensive than the existing “DE-only” models in its coverage of pacing modes and rate modulation, but could be readily extended. Overall, although our co-model is still preliminary, we believe that it forms a useful step towards co-modelling of medical cyber-physical systems.

Acknowledgements Our work is supported by the EC 7th Framework Programme under grant agreement 248134 (DESTECS) and the UK EPSRC Platform Grant “Trustworthy Ambient Systems: Resource-Constrained Ambience”. We are grateful to Ken Pierce for many fruitful discussions.

References

Timed-pNets: A formal communication behavior model for real-time CPS system *

Yanwen Chen¹,², Yixiang Chen¹, and Eric Madelaine²

¹ MoE Engineering Research Center for Software/Hardware Co-design Technology and Application, East China Normal University, Shanghai, China
² INRIA, CNRS, I3S, University of Nice Sophia-Antipolis, Sophia Antipolis, France

Abstract. We propose a semantic model named timed-pNets to define hierarchical structures for CPSs as well as its communication behaviors semantic with time constraints. Logical clocks relations are introduced to describe the partial order of event occurring. After setting (time)-boundaries and designing properties, we use the TimeSquare tool to simulate the system and check its properties.

1 Introduction

Nowadays, cyber physical systems have received much attention since the next generation of computing revolution is integration of computation, control and communication [7]. In CPSs, heterogenous embedded devices are connected by wire or wireless networks to communicate among each other via sets of sensors and actuators. One typical application domain is intelligent transportation systems (ITS), in which cars and infrastructures are equipped with sensors and actuators. They communicate with each other to update physical information and accomplish remote controlling. Currently, Research and Innovative Technology Administration (RITA)[10] in U.S. Department of Transportation has started research work on it to achieve a vision of national transportation by feature a connected transportation environment among vehicles, infrastructures and passengers’ portable devices. They raised the importance of real-time communication among these distributed nodes since the data out of date would make big mistakes even sometime could lead to car accidence. For example, the delay of sending global traffic information to cars may result to a wrong guiding for car to choose its best way. Also the delay of information exchanging among cars may cause a car accidence especially when they cannot see each other at cross.

Our aim is to build a low-level semantic model for the system and then analyse its properties. Since each distributed device has its own clock(s) and the communication delay between devices is uncertain, synchronous and asynchronous communication behavior will be considered in our model. We propose timed-pNets, which is an extension of pNets [2], to describe the communication behavior by adding logic clock relation. Time-pNets absorbs many advantages of pNets

* This work was partially funded by the Associated Team DAESD between INRIA and ECNU, Shanghai; by ECNU for oversea visiting funding, China 973 project(No.2011CB302802) and NSFC(No.610211004).
such as well hierarchical structure, flexible communication models, compact format, expressiveness, etc. By introducing clock relations which is defined in CCSL [8], the asynchronous communication behavior can be well defined which leads to well checking for real-time properties. Finally, we use the TimeSquare tool to simulate the clock relation of models and then check the correction of time logic as well as properties.

The main contributions of our work in the paper include: 1) Introduce logical clocks into pNets, 2) Propose a model language timed-pNets, 3) Use timed-pNets to model ITS, 4) Use TimeSquare to simulate clock relations and check its properties.

In section 2 we introduce the related work. In section 3 we give the definition of timed-pNets. In section 4, we propose a simple use case and describe how we use timed-pNets to model it. In section 5, we use time square to simulate the clock relations of the system and check its properties. In the last section, we give an conclusion for our current work.

2 Related Work

Prior research works related to building model for ITS applications include:

- Timed-automata[1] can be used to model the behavior of real-time systems. They provide a simple, and yet powerful, way to annotate state-transition graphs with timing constraints using finitely many real-valued clocks. Closure properties, decision problems as well as automatic verification of real-time requirements of finite-state systems are considered in timed-automata, and are supported by a number of tools, e.g. in UPPAAL [4].
- Globally Asynchronous Locally Synchronous (GALS) systems [9] combine the benefits of synchronous and asynchronous systems, in which each embedded node is modeled as a FSM (Finite State Machine) and the communication between them as buffers. This architecture provides a methodology for combining concurrent embedded systems within loosely coupled systems.
- The BIP framework [3], which can be used to model heterogeneous real-time components. BIP provides a powerful mechanism for structuring interaction for layered components of which synchronous and time systems are particular classes. The BIP framework produces a very fine grained formal computational model of the system functional level and executes the model semantics at runtime. In parallel, recent evolutions of the BIP framework allow the use of timed models and provide a real-time BIP Engine.
- Spatio-temporal consistence language(STeC) [5], which provides a location-triggered specification as well as operational semantics for describing distributed system with time and location constraints.

Compared to these previous works, our approach uses logical clocks instead of physical clocks, so that our approach is flexible enough to describe a communication delay by clock relation rather than by concrete time units. This gives us a very flexible way to specify interaction of systems with different clocks.
3 Timed-pNets

In this section, we define Timed-pNets, which are an extension of pNets (parameterized networks of synchronized automata), a very expressive and flexible semantic model developed by the Oasis team at INRIA for the modeling and verification of (untimed) distributed systems [2].

Definition 1 (Timed-Actions). Let $T$ be a set of discrete time variables taken from non-negative natural numbers $\mathbb{N}$. $\mathcal{B}_T$ is the set of boolean expressions (guards) over time variables and $\mathcal{L}_{A,T}$ is an action set built over $T$, in which each action has a free time variable $t \in T$. We call $a^tB \in \mathcal{L}_{A,T}$, with $B \in \mathcal{B}_T$ a timed action, in which $t$ describes a time delay before the action can be executed.

We set $a^0 = a$ that means the action $a$ is always ready. As an example, $a^{t \mid 1 \leq t \leq 3}$ means the action $a$ cannot be executed until $t$ units of time are passed.

Logical Clocks of timed-Actions Logical clocks [8] represent a relaxed form of time where any events can be taken as a reference for counting. It can be used for specifying classical and multiform real-time requirements as well as formally specifying constraints on the behavior of a model.

Definition 2 (Logic Clocks). A clock $C_a$ consists of an infinite set of discrete ticks of timed-action $a^t$. We write $C_a = \{(a^t)_1, (a^t)_2, ... \mid k \in \mathbb{N}, a^t \in \mathcal{L}_{A,T}\}$, in which $(a^t)_k$ denotes the $k^{th}$ instance of clock $C_a$.

Clock Constraints Clock constraints are predicates built from binary relations between clock expressions. We take the syntax and semantics of clock relations from [8], which is a language to express multi-clock time specifications by defining clock relations of time models for real-time systems. The clock relations include: $\subseteq$ (subclock), $\sharp$ (exclusion), $=$ (coincidence), $\prec$ (strict precedence), $\preceq$ (precedence) and defined as:

- $a_1 \subseteq a_2$ ($a_1$ is a subclock of $a_2$), which means each instance of $a_1$ must be coincident with an instant of $a_2$.
- $a_1 \nsubseteq a_2$ ($a_1$ excludes $a_2$), which means none of their instances coincide.
- $a_1 = a_2$ ($a_1$ coincides with $a_2$), which means the action $a_1$ ticks if and only if the action $a_2$ ticks.
- $a_1 \prec a_2$ ($a_1$ strictly precedes $a_2$), which means for every instant $k (k \in N)$, the $k^{th}$ instant of $a_1$ strictly precedes the $k^{th}$ instant of $a_2$.
- $a_1 \preceq a_2$ ($a_1$ precedes $a_2$), which similar to the previous one. The only difference is the action $a_1$ can tick as late as when $a_2$ ticks.
- New relations or expressions can be define by combining these basic relations with arithmetic and boolean. For example: $a_1 - a_2 \prec a_3$

The next definition extends the classical pNets definition from [2] with timed-actions and clock constraints. From pNets, we retain the hierarchical structure that is essential in structuring our heterogeneous systems, but also the parameterization of subnets: holes in a pNet can be instantiated by a variable number
of subnets (as e.g. a number of cars in the forthcoming case-study). Then synchro-
nisation vectors allow very flexible and expressive multi-way synchronisation
mechanisms, that naturally we extend here with clock constraints.

**Definition 3 (Timed-pNets).** A Timed-pNet is a tuple \( <P, A_G, J, C, \bar{R}_J, \bar{V} >, \)

where:

- \( P = \{p_i/p_i \in \text{Dom}_i\} \) is a finite set of parameters
- \( A_G \subseteq \mathcal{L}_A \) is a set of global actions
- \( C \) is a set of clocks for all actions
- \( R_G \) is a set of relations between actions taken from each subnet
- \( J \) is a countable set of argument indexes: each index \( j \in J \) is called a hole
  and is associated with a sort \( O_j \subseteq \mathcal{L}_A \) and a set of clock constraints \( \bar{R}_J \)
- \( \bar{V} = \{\vec{v}\} \) is a set of synchronous vectors of the form:
  * (binary communication) \( \vec{v} = ...,[a_{[k_1]}^1, ... , ?a_{[k_2]}^2, ... ] \rightarrow (a_g^s) \), in which
    \( a_g^s \in A_G, k_1 \in \text{Dom}_1, k_2 \in \text{Dom}_2, !a^1_{[k_1]} \in O_{i_1}, ?a^2_{[k_2]} \in O_{i_2}, t_g = \max\{t_1, t_2\} \)
  * or (visibility) \( \vec{v} = ...,[a^1_{[k_1]}], ... \rightarrow (a_g^s) \), in which \( a_g^s \in A_G, k_1 \in \text{Dom}_1, a^1_{[k_1]} \)

4 Case Study

In this part we illustrate our approach with a simple use case called speed con-
trolling system taken from [10]. Here cars’ speed are monitored by infrastructure
that collects information from cars and sends brake signal back to cars under
some global decision procedure. The communication protocol is described as
following:

- Cars send heartbeat signals "I’m here" with parameters "(location, speed)".
- Infrastructure collects heartbeat signals from cars.
- Infrastructure sends "brake" signal to the car if it is over the speed limitation.
- A car reduces its speed when it gets the "brake" signal.

Fig. 1 presents its architecture in which cars and infrastructures are distribut-
ed nodes. A cars consists of three sub components: a sensor, a controller and a
brake system. The car sensor is used to detect its current location and speed
and to receive control signals received from the infrastructure. The Car con-
troller gets signals from the sensor and then call the relevant systems to execute
brake operations. The local communications between the sub components of cars
are synchronized, which means that the sending event and receiving event coin-
cide. These sub components’ LTS are shown in Fig. 1. The car sensor is modeled
by two LTSs: one defining the periodical sending of heartbeat signals to report
its location and speed, the other describing its reactions to control signals.

Now, we describe how to use timed-pNets to model the communication be-
bavior and how to build clock relations in and between components. By lack of
space we only explain the top-level synchronisation elements, and the behaviour
of the car’s subprocesses.
4.1 Formalisation of timed-pNets Architecture

The building of timed-pNets is hierarchical. For our example, we generate separately a pNet structure for the toplevel assembly, and for each of the Car and Infrastructure components. The top-level pNet has 2 holes, the first one receiving an arbitrary number of Cars, the second one a single Infrastructure. Within each “second layer” pNet local communications will be synchronous, while at toplevel communications are asynchronous. This shows in the following pNet, where clock relations on the “hb” and “ctrl” links are defined as precedence.

\[
\begin{align*}
&P = \{ k : \mathbb{N} \}, \ A_G, R_G, J, \tilde{C}_J, \tilde{O}_J, \tilde{R}_J, \bar{V} > \\
&A_G = \{ CI_{hb}^{k}(loc, speed), CI_{ctrl}^{k}(brake) \} \\
&J = \{ car[k], infrastructure \} \\
&O_{Car} = \{ !hb^{k}(loc, speed), ?c_{ctrl}\langle k\rangle(brake), call(brake), T_s, \ldots \} \\
&O_{infrastructure} = \{ ?I_{hb}^{k}(loc, speed), !I_{ctrl}\langle k\rangle(brake), \ldots \} \\
&R_G = \{ !hb^{k}\subseteq I_{hb}^{k}(loc, speed); \}
&\quad !ctrl\langle k\rangle(brake) \subseteq !ctrl\langle \rangle(brake); \}
&\quad \bar{V} :< O_{car}[k], O_{infrastructure} \rightarrow AC_{Car\_infrastructure} \\
&= <!hb^{k}\subseteq I_{hb}^{k}(loc, speed), ?I_{hb}^{k}(loc, speed) \rightarrow CI_{hb}^{k}(loc, speed); \\
&\quad <?c_{ctrl}\langle k\rangle(brake), !I_{ctrl}\langle k\rangle(brake) \rightarrow CI_{ctrl}^{k}(brake). \}
\end{align*}
\]

An interesting point is that the Infrastructure receives independent heartbeats for the Cars, that are subsequently interleaved within the Infrastructure structure. This is expressed by a clock relation on the link between Infrastructure sensors and control: \(?I_{hb}^{k}\subseteq I_{hb}^{k}(loc, speed) \subseteq !hb_{\bar{V}}^{k}(loc, speed)\), telling the each Car heartbeat clock transmitted by the Sensor is a subset of the (single) heartbeat clock received by the Control.
Finally, we give an example of the set of Clock constraints that we generate from a pLTS, here for the Car Sensor sub component:

\[ R_{\text{CarSensor}} = \{ \text{hb(loc, speed)} \overset{\text{idealClock discretizedBy rate}}{\Rightarrow} (1); \]
\[ \tau_{\ast}^{\text{ctrl}}(\text{brake}) \prec (2); \]
\[ ?c_{\text{ctrl}}(\text{result}) \prec (\tau \wedge !\text{ctrl}(\text{brake})) (3); \]
\[ !\text{ctrl}(\text{brake}) \prec ?T_s (4); \]
\[ (?T_s[i] \vee \tau[i]) \prec ?c_{\text{ctrl}}(\text{result})[i + 1] (5); \]

where (1) describes that the heartbeat signal is sent periodically; (2) indicates that events \( \tau \) and \( !\text{ctrl}(\text{brake}) \), from different paths of the same LTS, are exclusive; (3) denotes that the event \( ?c_{\text{ctrl}}(\text{result}) \) always precedes the event \( \tau \) and \( !\text{ctrl}(\text{brake}) \); (4) tells us that event \( !\text{ctrl}(\text{brake}) \) precedes the event \( ?T_s \); (5) explains that the events in the \( i^{th} \) cycle precedes those in the \( (i + 1)^{th} \) cycle.

5 Simulation

We use TimeSquare [6] to simulate the clock relations and check its logic correction. The input of TimeSquare is a CCSL file including clock relations, bound requirements and properties. The tool proceeds with a symbolic simulation, and generates a trace model (one partial order satisfying the specification). Output files (text and graph) are generated to display the traces and eventually show the property violations.

In our use-case the clock relations generated from the pNets model. Then we representing the communication and computation delays by fixing delay bound in timed-action guards. We set heartbeat interval \( hi = hb.(i + 1) - hb.i \). The minimum and maximum communication delay between car and infrastructure is set as \((1/5)hi\) and \((3/5)hi\). For the computation delay, we assume that it takes at most \((2/5)hi\) for each action transition so that for instance we can set \((?c_{\text{ctrl}}(\text{result}) - ?T_s) \leq (2/5)hi\). For deadline, assuming each heartbeat signal should be processed before sending next heartbeat, then for each action \( a.i \), we set \( \text{deadline} = hb.(i + 1) \). All these boundaries are merged into the TimeSquare CCSL input file. We expressed a “boundary liveness property” for this simulation to see if the system satisfies the real-time property under the hypothesis of boundaries. \( (?T_s.(i) \prec hb.(i + 1)) \wedge (\tau.i \prec hb.(i + 1)) \) denotes each heartbeat signal finally will be processed before deadline.

After simulation, we got the result as Fig.2, which shows that the real-time property is not satisfied since the clock \( !\text{ctrl}(\text{brake}) \) is over the deadline. The boundary condition we set in previous part was too large. After we modify the maximum computation boundary, the simulation shows no more violations.

6 Conclusion

In this paper, we have added time constraints to the pNets behavior semantic model. In timed-pNets, logical time relations in lower-level (synchronous) components are derived from the corresponding label transition systems. Then
Fig. 2. property checking

communication delays, processing delays and required global properties are defined by the user. We illustrate our approach on a simple use case from Intelligent Transport Systems, show how our Timed-pNets models are constructed, and how timed properties are validated through simulations in the TimeSquare tool.

References

Trustworthiness in Hybrid Bio-Silicon Systems for Next-Generation Neural Prosthetics

Graeme Coapes, Terrence Mak, Jun Wen Luo, and Alex Yakovlev

School of Electrical and Electronic Engineering, Newcastle University, Newcastle, UK

Abstract. The development of next-generation neural prosthetics will see closed-loop interactions form between computational elements and biological systems, with the aim of restoring or repairing standard behaviour functionality. This cyber-physical connection places a huge amount of trust into the computation, due to the catastrophic consequences of potential failures. This paper discusses the progression of these prosthetic devices and highlights potential causes for discussion when considering the trustworthiness of the system.

1 Neural Prosthetics

Brain-machine interfaces (BMI) involve the coupling of electronic components to an animal’s nervous system, typically with the objective of either recording activity or stimulating the nerves.

Monitoring nerve activity allows for investigations into the behaviour of biological systems and has led to some great leaps forward in our understanding of the function of the nervous system and brain. The stimulation of biological tissue started with similar investigatory goals, but it has progressed to allow the development of medical neural prosthetic devices. These devices offer potential treatment for previously incurable disease and ailments, such as deafness [1], blindness [2] and tetraplegia [3].

In recent years we have seen the advancement of BMI technology, whereby the coupling between machine and body is greater than ever before and the interaction is becoming a closed-loop system. More often we are seeing the functionality of the nerve stimulation systems rely upon the recorded neural activity. For instance, the neural prosthetic device for tetraplegia described by Moritz et al. relies upon neuron activity in the motor cortex to control stimulation of the patient’s arm muscles [4]. Also, in 2011 we saw a brain-machine-brain interface developed by O’Doherty et al [5]. This involved not only a robotic arm controlled via a primate’s motor cortex, but also a sensory feedback signal being delivered back into the brain of the animal, which changed depending upon the movement of robotic arm.

The next-step in this technological arena could be to develop cognitive prosthetic devices whereby, damaged individual neuron cells, or even large brain regions are substituted by silicon neurons, in order to overcome conditions such
as stroke or epilepsy. By connecting bio-realistic neuron models to the damaged brain tissue it may be possible to restore, repair or replace the damaged nerve cells to regain functionality [6], an example of a such a next-generation neural prosthetic is shown in Fig. 1.

![Fig. 1. An example of a next-generation cellular neural prosthetic. A closed-loop interaction is formed between the biological nerve cells and the silicon neuron model through various interface techniques.](image)

The behaviour of some neural circuits relies upon the presence of a pacemaker cell, acting much like a clock within a silicon system. If the pacemaker is damaged and becomes inactive or faulty, the neural circuit may also cease to function. By modelling the damaged cell and interfacing with the biology, the behaviour may be restored.

We have demonstrated this concept in practice with real nerve tissue dissected from a crab. Fig. 2 shows the results of this experiment. First of all, the original circuit is intact and functioning correctly. Then, when two of the cells are anaesthetized the behaviour dramatically alters. By reintroducing two silicon nerve cells, including the pacemaker cell, the network functionality can be restored.

![Fig. 2. An example of a next-generation cellular neural prosthetic. A closed-loop interaction is formed between the biological nerve cells and the silicon neuron model through various interface techniques.](image)
By extending this concept of cellular rehabilitation to large-scale systems, complex cognitive behavioural prosthetics may be introduced, perhaps allowing for treatment for memory or speech defects.

Recently such a cognitive prosthetic device has been demonstrated in practice in live mice by Liu et al [7]. Within this experiment a mouse was taught to fear a particular scenario. When this scenario occurred the mouse instinctively froze and became aware of danger. Liu et al. showed how by optical stimulating the same neurons in the hippocampus that were involved in the initial memory formation the mouse once again froze and became wary. This proved how targeted stimulation of a select group of neurons is able to influence cognitive behaviour.

2 Trust in Neural Prosthetics

It is clear that although the potential for application is enormous, there are serious hurdles to overcome in order to make such systems dependable and secure and hence, trustworthy. The amplitude of these hurdles are magnified by the requirements of the application and the general behaviour of neural systems, which due to their nature, are inherently noisy, undefined and occasionally erratic.

For a BMI system to be trustworthy there is a dependence upon the machine to provide the correct service to the brain system. Hence, we may judge the machine by its ability to satisfy the criteria and definition of dependence [8]:

* Availability - A silicon neural model must be ready to operate in the time window that the biology expects. This requires that all neural models must run in real-time under all operating conditions. This perhaps may be a challenge for a neural network which is susceptible to bursting, whereby a rapid pattern of activity may cause a high computational and communicational load.

* Reliability - The machine must provide continued service with extremely, preferably zero, failure rates. Two major challenges exist here: 1) the wide range of possible neural input and 2) from an engineering point of view the interface must be able to operate in the long-term without degradation in performance, such as that caused by build-up of glia surrounding electrodes.

* Safety - Not only must neural prosthetic devices stay within human-tolerable operating conditions, such as current and temperature limits, but there must be guarantees that what is perceived as regular neural stimulus does not cause catastrophic changes in neural circuit activity. Perhaps, in order to guarantee integrity the machine components of a BMI must be aware of its influence upon the service of the biology, and be able to refrain from stimulus that causes unwanted behaviour.

* Integrity - Cognitive prosthetics offer the potential to restore behaviour and functionality to brain circuit, but they also invite the opportunity for unwanted outside influences to alter circuit behaviour, with potentially catastrophic damage. BMI systems must be impenetrable to unauthorized accesses and system design errors.
Maintainability - This may be a particular challenge for implanted devices, whereby change or alteration could require surgery to the patient. Therefore, ideally devices should be externally programmable or reconfigurable such that authorized alterations may be made without severely impacting upon the user or patient.

Confidentiality - Cognitive behaviour in animals is encoded within the neural activity patterns. How can we prevent too much information about this activity from being read and decoded by unwanted external factors, who perhaps may have perverse motives.

Although, it is imperative that the machine aspect of a BMI be trustworthy and dependable, there can be no guarantees that the biological elements will fulfil their obligations with regards to the metrics described above. The service provided by the biology may not always be what the silicon expects.

For instance, a consistent stimulus from the machine may have wildly different biological responses due to the natural plasticity and learning processes within brain circuits. Hence, this makes the requirements on the silicon components even greater, as they must satisfy their obligation to be trustworthy and dependable in a wide-regime of behaviours and dynamics.

References