The final publication is available at Springer via http://dx.doi.org/10.1007/s10270-015-0493-x

Industry 4.0 as a Cyber-Physical System Study

Pieter J. Mosterman[1], **Justyna Zander**[2]

[1] MathWorks, 3 Apple Hill Drive, Natick, MA 01760-2098, USA
pieter.mosterman@mathworks.com
[2] Worcester Polytechnic Institute, Worcester, MA, USA
dr.justyna.zander@ieee.org

Received: date / Revised version: date

Abstract Advances in computation and communication are taking shape in the form of the Internet of Things, Machine to Machine (M2M) technology, Industry 4.0, and Cyber-Physical Systems (CPS). The impact on engineering such systems is a new technical systems paradigm based on ensembles of collaborating embedded software systems. To successfully facilitate this paradigm, multiple needs can be identified along three axes: (i) online configuring an ensemble of systems, (ii) achieving of a concerted function of collaborating systems, and (iii) providing the enabling infrastructure. This work focuses on the collaborative function dimension and presents a set of concrete examples of CPS challenges. The examples are illustrated based on a pick and place machine that solves a distributed version of the Towers of Hanoi puzzle. The system includes a physical environment, a wireless network, concurrent computing resources, and computational functionality such as, service arbitration, various forms of control, and processing of streaming video. The pick and place machine is of medium-size complexity. It is representative of issues occurring in industrial systems that are coming online. The entire study is provided at a computational model level with the intent to contribute to the model-based research agenda in terms of design methods and implementation technologies necessary to make the next generation systems a reality.

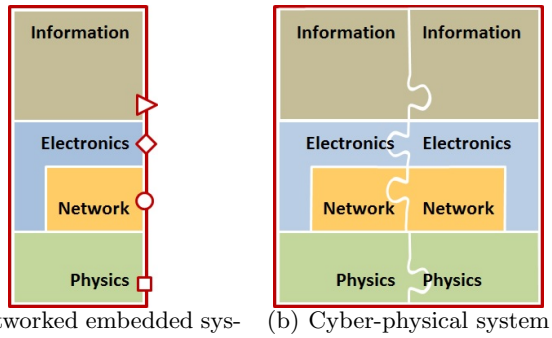
1 Introduction

With wirelessly connected mobile devices, an age of unparalleled connectivity has descended upon society. We are now able to communicate with most anybody at almost any time and from nearly everywhere. This degree of connectivity and communication is increasingly finding its utility in technical systems. Examples of this trend are the Internet of Things (IoT) [27], Machine to Machine (M2M) [7] technologies, Industry 4.0 [1], and Cyber-Physical Systems (CPS) [2]. All of these exploit the configurability and combined functionality of communicating systems. For all the opportunity a set of needs must be recognized, though, with challenges that must be overcome to design and operate such systems [6].

Previous work [17] inventoried and analyzed needs to address when embedded software systems collaborate to form systems of an open nature such as those in IoT, M2M, Industry 4.0, and CPS. In particular, a difference in technical system design and operation between networked embedded systems and CPS was highlighted. Figure 1 illustrates this difference. In Fig. 1(a) a networked embedded system is conceptualized as a combination of a physical part, a network part, an electronics part, and an information part. These various parts can each be extensive and may be provided by many organizations. Ultimately, however, on original equipment manufacturer (OEM) is responsible for the comprehensive system and the integration of the respective parts. Interaction with the system surroundings can be captured at the various modalities indicated by different type ports in the illustration. These surroundings do not fall under the purview of the OEM. The system is then designed to maintain operational behavior irrespective of the interaction, for example, by considering the interaction a disturbance that must be regulated away or by establishing very restricted and thoroughly tested interface behavior.

In Fig. 1(b) a CPS approach is conceptualized. Here, two systems that are the responsibility of one or two OEMs are shown to closely connect and combine into a new system of systems, an *ensemble* of systems [24]. The different systems in such an ensemble share computational functionality, electronics hardware, network connectivity, and a physical world. Functionality of the ensemble emerges by combining the functionality of the systems that constitute the ensemble. As such, these ensembles of systems consist of collaborating systems and

Send offprint requests to:



(a) Networked embedded system (b) Cyber-physical system

Fig. 1 A networked embedded system view versus a cyber-physical system view. *Published with permission, © 2015 MathWorks. All Rights Reserved.*

Fig. 2 An embedded system. *Published with permission, © 2015 MathWorks. All Rights Reserved.*

may come into (temporary) existence post deployment. For example, automobiles that approach an intersection may communicate with each other to negotiate which one crosses first. A system ensemble with intersection functionality emerges while the cars are interacting with one another, but it does not exist before and after. The difficulty of designing and operating such novel systems should be immediately evident especially in the face of the necessary compatibility between the range of automobiles fielded by all the different automobile manufacturers.

In such a CPS paradigm there is no opportunity to 'iron out the kinks' and 'get things to work' by a solid integration engineering effort. Instead, at design time it must be clear that a system will behave as intended even in a collaborative feature implementation and while sharing (part of) the implementation fabric.

Taking an industry vantage point, this work presents an example from the Industry 4.0 domain that is of medium-size complexity with industry relevance. Focus is on the CPS aspects, which pertains to the interaction across the physical space. The extensive information processing and analytics aspects of Industry 4.0 are beyond the scope of this work. As an example of an Industry 4.0 application, mymuesli.com allows a user to configure an individual muesli mix. As the muesli package is moving through the factory, the 'smart package' communicates to each of the machines how much of each of the corresponding ingredient should be filled.

In this paper, a pick-and-place machine as shown in Fig. 2 provides pick and place services to a set of 'smart blocks.' The blocks combine each of their local plans so that they emerge on a single stack in a sorted order (i.e., the system ensemble is a distributed form of the Towers of Hanoi puzzle). In the ensemble, various communicating systems interact in a physical environment, across a shared resource, each providing functionality to be combined into a feature responsible for a desired emerging behavior. The study serves as an archetypical CPS by embodying the corresponding paradigm of systems collaborating in cyberspace while operating concurrently in a physical environment and brings out a number of specific challenges that are found across a broad range of applications such as:

- An intersection with automobiles automatically managing their right of way.
- A smart port with autonomous vehicles for automatic cargo loading, routing, and unloading (cf. the port of Rotterdam).
- A Smart Emergency Response System (SERS) with automated delivery by autonomous drones (e.g., SERS description [14] and SERS software [5]).
- An office building purchasing auctioned energy supply based on predicted and controlled energy needs.

Designing for new and unknown system functionalities that may become required or active not sooner than during the deployment of the system challenges the traditional approach of systems engineering a system. Multi-vendor systems that configure post deployment are mostly restricted to data sharing and rely on standards such as the Internet Protocol (IP), American standard code for information interchange (ASCII), the hypertext markup language (HTML), and more recently the extensible markup language (XML). While more sophisticated service based technology is coming online, closing the loop with machines operating in physical spaces puts forward many challenges, a number of which are illustrated in this work. From a higher level of abstraction down, agent-based approaches (e.g., [26]) are a potentially viable foundation to build the necessary framework and technology on and this work provides a concrete system that may serve as a challenge problem.

Novel system engineering approaches become a necessity, which, in turn, rely heavily on enabling system architectures and powerful design-time approaches to conceptualize behavior at various levels of detail in different parts of the system [25]. Concrete challenges are highlighted that must be overcome as part of a new type of methodological thinking for successful CPS design, with emphasis on post-deployment system integration issues.

In previous work [17] the transformation of networked embedded systems into CPS is discussed. On the one hand, the transformation establishes a functionality enriching paradigm, on the other hand, it is accompanied by exacerbated system integration challenges. Starting from the notion of CPS as an ensemble of system, Section 2 describes a distributed Towers of Hanoi system. Concrete examples of challenges are presented in Section 3 as coordinating distributed control for emerging behavior, Section 4 as distributed multirate architectures for data sharing, Section 5 as feature interaction for functionality sharing, and Section 6 as a comprehensive quality assurance strategy for collaborative functionality testing. Section 7 completes the study with conclusions.

2 Distributed Towers of Hanoi

This section introduces a study to illustrate some of the characteristics of the CPS paradigm shift. Figure 2 shows a representation of an industry process (e.g., for manufacturing purposes) with a supervisory control and data acquistion (SCADA) system that can sort a stack of blocks akin to the Towers of Hanoi puzzle [18,19]. An original stack of blocks may be provided with the blocks in an arbitrary order of color, size, etc. An operator then initiates a sorting operation by executing a sequence control program as part of the supervisory control. The sequence control moves a nozzle in the horizontal and vertical direction to pick and place the set of blocks one block at a time. It may be preprogrammed which block to pick and where to place in various different sequences of control. The operator then picks the particular sequence to execute based on the initial order.

In the new paradigm, which is at the foundation of Industry 4.0, the blocks to be sorted may have built-in functionality that is an integral part of the overall sorting feature functionality. In this configuration, the control algorithm does not include a set of fixed sequence control programs. Instead, the system provides pick and place services that can be accessed over a network by each of the blocks. Before the start of the sorting operation, the blocks devise an individual plan on how they each should be picked and placed. During operation, by having the blocks request pick and place services based on their individual plans (their local control) a global behavior emerges that has the final stack of blocks sorted according to color, size, etc. In such a configuration, the overall sorting feature functionality is created post deployment as the blocks are being provided in the field (and by various different suppliers). The increase in flexibility of such a system should be immediately apparent (e.g., there is no limitation for an operator to resort to pre-programmed sequence control) while a broad array of business opportunities may become intuitive [21] (e.g., a partner of the system operator may provide smart decals with the respective sorting algorithms as compatible pick and place service requests).

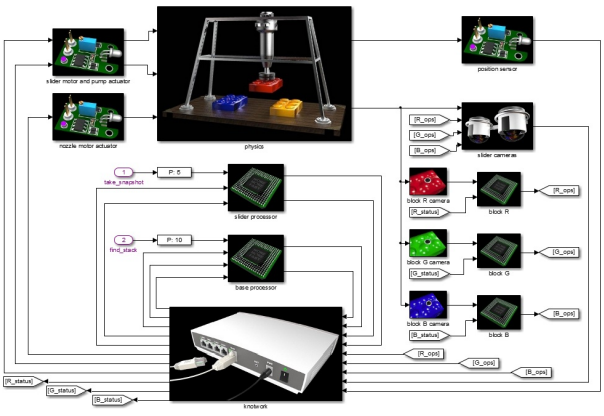


Fig. 3 Simulink® Model of the Distributed Towers of Hanoi. *Published with permission, © 2015 MathWorks. All Rights Reserved.*

A fully executable model of the distributed Towers of Hanoi puzzle has been designed using the MATLAB® and Simulink® family of products [13] and is publicly available.[1] The top level of the model hierarchy is shown in Fig. 3. The system integration issues in the distributed Towers of Hanoi model are resolved as in the traditional design paradigm with a single responsible OEM the way it is currently practiced in industry. The example is explored to identify and highlight some of the particular challenges that are faced when migrating to a CPS approach and how this differs from a more traditional single OEM paradigm.

An architecture of the overall system is shown in Fig. 4, which illustrates the elements and their main functionalities. The physics model includes the gantry structure as the *machine*. The *nozzle motor* and the *slider motor* enable vertical and horizontal motion, respectively. These *dc* motors are controlled by network capable (NCAP) actuators called *voltage actuator*. An NCAP *position sensor* measures the horizontal position of the slider. Each of the three blocks (*red block*, *green block*, and *blue block*) is modeled along with a *light sensor* for each of them as well as a microprocessor μP to provide computing power and a network interface (akin to a smartphone with its camera). A pair of *image sensors* with corresponding *stereo camera* hardware supports streaming stereoscopic video to the *wireless network*. This network further connects two electronic computing units (ECU). The *slider ECU* implements the *stereo analysis* slider functionality and the *base ECU* implements the *supervisory control*, *slider control*, *detection logic*, and *force profile selection* functionality. The force profile is communicated to the nozzle motor NCAP as the *sequence control* profile while the slider control reference is communicated to the slider motor NCAP as the *position control* reference input. Finally, each of the blocks contains functionality to *detect light* to indicate

[1] The model can be downloaded from the MATLAB Central File Exchange or GitHub [18].

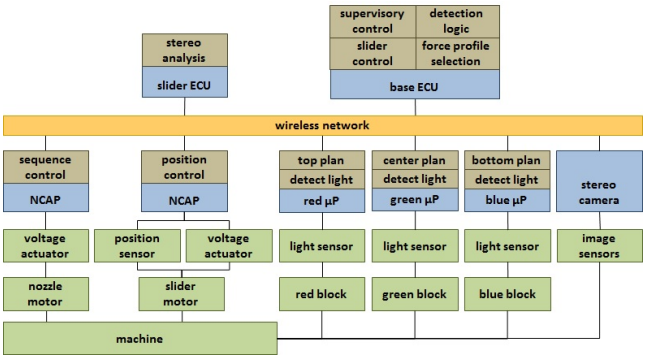


Fig. 4 Towers of Hanoi CPS architecture. *Published with permission, © 2015 MathWorks. All Rights Reserved.*

Need	Challenge
Emerging behavior design	Collaborative planning, guidance, and control
Data sharing	Multirate architectures
	Extracting and deriving specific value from general information
Functionality sharing	Multi-use functionality post deployment
	Feature interaction
Collaborative functionality testing	Systematic test suite generation and automated test evaluation
	Test results reproduction under minimum uncertainty

Table 1 Needs analysis for concerted function of collaborating systems (excerpt from [17])

that the block is on top of a stack (and so can be moved) as well as a service invocation plan for each of the desired position in the final stack (*top plan*, *center plan*, and *bottom plan*).

In previous work, needs to further the CPS paradigm were classified along three axes: (i) dynamic configuration needs, (ii) collaborating systems needs, and (iii) infrastructure needs [17]. Here, the term configuration refers to an ensemble of systems with various points and modalities of interaction among the systems. These points of interactions and the specific modalities may change during operation, in particular post deployment. The work presented here concentrates on the collaborating systems needs, however, which are reproduced in Table 1 along with their corresponding identified challenges. For each of the needs, a challenge will be discussed followed by a concrete illustration as found in the distributed Towers of Hanoi.

3 Emerging Behavior Design

To design emerging behavior a single most important challenge was identified to be *collaborative planning, guidance, and control*.

3.1 The Challenge

Coordinating a number of controlling entities such that they operate according to a well-understood concerted global mechanism is a distinct challenge (e.g., [26]). Nevertheless, it is feasible during a system integration phase if the cases are limited in their interaction scenarios. In a CPS paradigm, the coordination must be encoded in collaborating functionality and protocols. This adds to the complexity and implementation effort of collaborating entities and degrades the performance characteristics of the overall system. For example, how to factor out functionality that is encoded in a global coordinating mechanism into local functionality (such as bilateral agreement exchanges) is an open-ended question.

Once a solution becomes available, the trade offs can be studied. On the one hand, the more robust implementation of local exchanges is desired, on the other hand, a more efficient global scheme has distinct advantages. Similarly, efficiency can be gained when synchronization between concurrently executing logic inherently (and implicitly) operates correctly on a closed and given platform. However, when the overall system becomes open, express synchronization becomes essential. While the necessity for such synchronization must be systematically determined, it also introduces further overhead in order to grant the overall behavior the necessary level of robustness. A methodology to achieve emerging behavior by (efficiently) coordinating controlling entities online is an important topic of research. Drawing on software engineering techniques, a potential approach may be to characterize dynamic behavior and reconfigure interacting functionality accordingly to account for optimization opportunity (e.g., removing a synchronization point if execution times are determined to enable this, possibly adorned with an assertion).

3.2 An Example

In the distributed Towers of Hanoi, each of the blocks has its individual pick and place plan. Each block is equipped with a camera to determine whether it is on top of the stack it is in such that pick and place services based on the following plans can be requested:

- Red block
  1. Move one spot to the left
  2. Move one spot to the right
  3. Move two spots to the left
- Green block
  1. Move one spot to the left
  2. Move one spot to the left.
- Blue block
  1. Move two spots to the left.

When these local pick and place sequences are properly interleaved, exhaustive testing of the 6 possible permutations validates that the emerging behavior is an RGB

(red, green, blue) sorted order. Because the blue block must be at the bottom, as soon as this block can be picked from the original stack location it can be moved two spots over to its final location. If the green block can be picked, the blue block may still be underneath and, therefore, the green block first moves one spot to the left to an intermediate location. If the blue block were underneath the green block, the blue block can then be moved to its final location before the green block is moved one more spot to the left from the intermediate location to the final location. If the red block were underneath the green block as well and on top of the blue block, the red block must first be moved off the blue block by moving the red block one spot to the left. Then the blue block can be moved to its final location. The red block must then be moved one spot to the right back to its original location so the green block can be moved one spot to the left to its final location. After this, the red block can be moved two spots to the left to its final location.

These distributed plans merge properly only by merit of dynamic priority assignment of the services requested from the shared resource (the pick and place machine with its SCADA component). For example, consider the case where in the initial stack the blocks are sorted (from top to bottom) as green, red, blue as shown in Fig. 5.

1. With the green block on top, its request for service to be moved one spot to the left (GL) with priority 3 (<3>) is honored (SRV G), which results in the machine moving the top block in location 3 to location 2 (MV3-2). Upon completion (COMP) a message *complete* is sent to the green block, where a synchronization stage (SYNC) awaits the negative of *complete* to synchronize the state of the block and the state of the service routine.
2. Next, both the red block and the green block can be moved and they both request service to be moved one spot to the left, RL1 with priority 2 and GL1 with priority 1, respectively. Because request by the red block has higher priority, it is selected for service (SRV R) and the machine moves the top block in location 3 to location 2 (MV3-2). Upon completion, the synchronization occurs and the priority of the red block is lowered. Because the red block is now on top of the green block, the green block suspends its service request GL1.
3. Now that the red block and the blue block can be moved, they request to be moved one spot to the right and two spots to the left, RR1 with priority 1 and BL2 with priority 4, respectively. Because of the higher priority, the blue block is selected for service (SRV B) and the machine moves the top block in location 3 to location 1 (MV3-1).
4. After synchronization, the service request by the red block (RR1) is the only one still active, selected for service (SRV R), and so the machine moves the top block in location 2 to location 3 (MV2-3). The priority of service requests by the red block is then lowered to 0. Because the red block is moved off the green block, the green block can now request service again.
5. At this point, both the red block and the green block request to be moved to the left by 2 spots (RL2) and 1 spot (GL1), respectively. Because of the change in priority, in contrast with the second stage, the green block now takes precedence over the red block (SRV G) and the machine moves the top block from location 2 to location 1 (MV2-1).
6. Finally, the request by the red block to be moved 2 spots to the left (RL2) can be honored (SRV R, which has the machine move the top block from location 3 to location 1 (MV3-1) and the sorting completes.

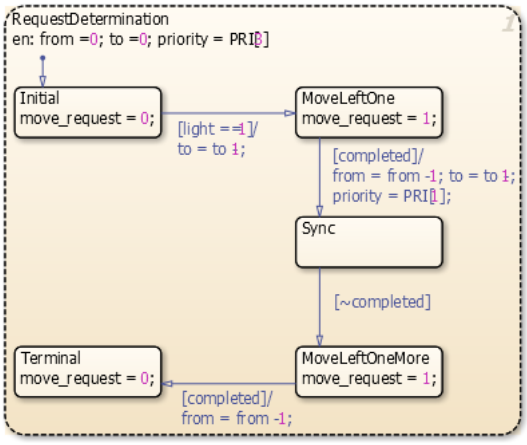


Fig. 6 Green block service request. *Published with permission, © 2015 MathWorks. All Rights Reserved.*

Note that the priority order between the green and the red block changes dynamically. Otherwise, either the green block would have been moved in stage 2 or the red block in stage 5 (see Fig. 5). This dynamic priority assignment is modeled in a state transition diagram that implements the plan of the green block, shown in Fig. 6. In this diagram, upon entry of the *RequestDetermination* state, the priority of the service requests by the green block is set to the value *PRI[3]*. Once the block detects that it can be moved because the condition *light == 1* is satisfied (i.e., it is at the top of the stack), the block plan moves into the state where it requests to be moved one spot to the left. After the block has been moved, the priority is then set to the lower value *PRI[1]* so any moves of the red block can take place first.

The service module of the machine consists of a coordinating mechanism with a priority resolution scheme that communicates with the red block via channel 1, the green block via channel 2, and the blue block via channel 3. When the machine is available to perform a pick and place action it is in the *ready* state. When in the *ready* state, the channel that currently has the highest priority is determined. If the channel with the highest priority is indeed requesting the corresponding block to be moved then the service is initiated. If not, the channel with the next highest priority is determined and the cor-

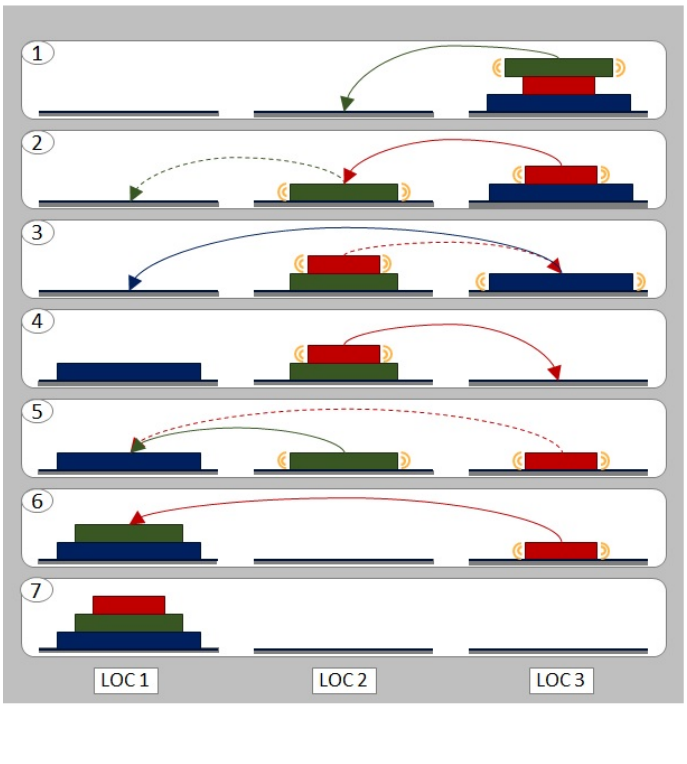


Fig. 5 One full sorting scenario with emerging behavior from distributed plans. *Published with permission, © 2015 MathWorks. All Rights Reserved.*

responding request to be moved is evaluated. Finally, if the second highest priority does not request to be moved, the lowest priority channel is evaluated.

The combination of the local plans, the dynamic prioritization, and the complex priority based service resolution results in an overall substantial complexity. Moreover, the distributed nature of the architecture calls for synchronization mechanisms between events that may not be immediately intuitive. For example, the service request handling captured by the state transition diagram in Fig. 7 moves into a *chan?Service* state after the priorities have been resolved. While in this state the requested move operation is executed. When the operation is completed, a flag *complete* is set to *True* and the state transition diagram moves out of the *chan?Service* state. The *complete* flag is also communicated to the block that is being operated on so that the block can move to the next state in its plan. For example, in Fig. 6, when in the *MoveLeftOne* state, once the condition *completed* is *True*, the green block moves out of this state. Instead of moving into the next phase of its plan, the *MoveLeftOneMore* state, the plan requires a synchronization with the service handler to reset the *completed* flag. Without this *Sync* state, the state machine in Fig. 6 may move into and out of *MoveLeftOneMore* before the concurrently executing state machine in Fig. 7 has reset the *complete* flag. Note that without the *Sync* state, depending on the execution time of the concurrently executing logic of the other blocks and the timing between them,

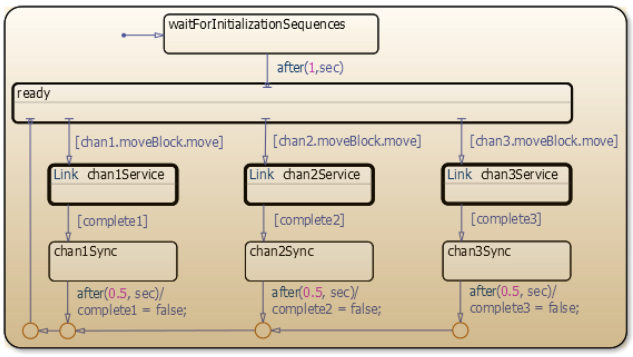


Fig. 7 Service request handling. *Published with permission, © 2015 MathWorks. All Rights Reserved.*

the system may appear to behave properly and the erroneous design would be difficult to evidence by testing only.

4 Data Sharing

The identified need to share data comes with two main challenges: (i) distributed multirate architectures and (ii) extracting and deriving specific value from general information. The distributed multirate architectures challenge is discussed in detail. Examples in the robotics domain of extracting information can be found in related work [28].

4.1 The Challenge

Multirate architecture comprises elements in software or hardware (e.g., tasks, processes, hardware peripherals, etc.) that operate at different rates and that share data. These elements often execute at discrete points in time with a periodic (at different rates) and synchronous execution in the sense that they share a single execution resource with one clock. More generally, and typical in CPS, elements of a multirate architecture include asynchronous periodic (periodic behavior on concurrent resources), nonperiodic, and continuous-time executions.

Timing issues in multirate architecture can be gnarly and exasperating to identify and remedy, especially in the face of distributed resources. It has to be known (i) whether there are delay issues, (ii) where to find to find the potential for these issues, (iii) whether the delays indeed impact behavior, and (iv) in case of adverse effects how to remedy this. In particular, the temporal effects of logical schemes such as double buffering are far from immediately obvious and require much experience in order to track down the root cause. While ultimately the culprit may be identified and a resolution implemented (cf. retiming analyses [33]) a more fundamental approach is required for CPS.

Networked embedded systems often allow tight control over timing complexity and may limit the concurrency in a system, prevent multiple rates, if there are multiple rates enforce harmonic rates, tightly restrict nonperiodic functionality, present asynchronous behavior because of multiple clocks, or carefully calibrate a global timing mechanism. In CPS, there is much less if any control over such implementation choices and so the effects (e.g., nonharmonic rates, drift, small phase shifts, and so forth) must be accounted for. Moreover, the systems that collaborate as an ensemble generally are designed to operate in a broad range of conditions. As such, timing may be affected by dynamic clock scaling (e.g., to preserve power) and multi-use functionality (with different quality of service trade offs such as throughput vs. latency). Furthermore, the necessary infrastructure to support dynamically configuring an ensemble introduces timing challenges because of the use of an open network (compared to a bus or local area network) and the service layer between systems, both of which come with their own timing characteristics (e.g., variation and uncertainty). In addition, a service layer does not allow direct access deep down the software stack (which is practiced in industry to resolve timing challenges).

In the face of these timing challenges, the dynamic configurability of CPS does not support study and experimentation to find and (re)solve all timing issues prior to deployment. Therefore, timing should be addressed at an architectural level. Current system design approaches do not support the necessary technological facilities and abstraction layering that are necessary for enabling online timing analysis and retiming synthesis. As a recognized need [25], much effort ought to be dedicated to this as a key building block in CPS design.

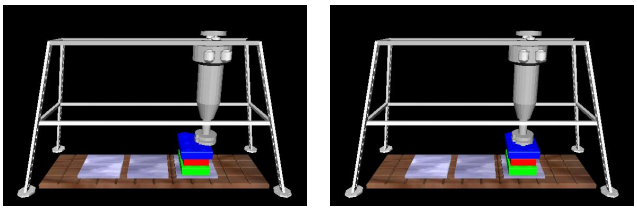
4.2 An Example

The computations for the various features in the distributed Towers of Hanoi execute with three different time periods. The feedback control loop (i.e., the positioning of the slider in the horizontal direction) and the feedforward control (i.e., the positioning of the nozzle in the vertical direction) execute with a time period of 5 ms. The service feature that allows blocks to request being picked and placed executes at 50 ms. A stereopsis feature to locate where the original stack of blocks is placed is computationally the most intensive and is performed every 100 ms.

When a value that is sampled at a given rate is communicated between two concurrent tasks, in order to preserve determinism, a communication scheme such as double buffering [20] may be employed. The effect is that the reader task reads a value that was written by the writer task during the previous time period. Thus, every time a value is deterministically communicated an additional delay of a full time period is introduced.

In case of a multirate system the implication is that data that is computed with different time periods (e.g., the sampled video streams for stereoscopic analysis at 100 ms and the sensed slider location for feedback control at 5 ms) becomes misaligned in time. For the stereopsis feature the effect is that after one deterministic communication the image pair in a video stream that is being analyzed is already 95 ms older than the slider location measurement. This temporal misalignment can lead to a disastrous effect. For example, when the stereoscopic analysis allows for detecting the stack of blocks and the location measurement that is available at that time is stored as the location of the stack of blocks. The result is a location that is much beyond the actual location of stack of blocks. Thus, time alignment should be given particular attention to avoid such a situation.

Figure 8 shows the difference in behavior between two cases. One is a straightforward integration of the various systems in the distributed Towers of Hanoi, which results in a misalignment in time. The other is an integration that accounts for the double buffering communication delays. Figure 8(a), shows how the misalignment leads to a pick action at an incorrect location. Figure 8(b) shows the pick action at the desired location. Where in a traditional design, such timing issues are resolved during system integration, as a CPS the timing must be properly composed post deployment, during operation [10].



(a) Without time compensation (b) With time compensation

Fig. 8 Misalignment in time of stereoscopic analysis and location measurement. *Published with permission, © 2015 MathWorks. All Rights Reserved.*

5 Functionality Sharing

The need for functionality sharing comes with challenges of (i) multi-use functionality post deployment, and (ii) feature interaction. The latter of these will be illustrated in detail followed by a brief example of the former.

5.1 The Challenge

While in an offline system integration phase issues that arise because of interacting features can be resolved, performing the careful design, calibration, and analysis after deployment is far from straightforward. Much more robust is a methodological assessment of acceptable uncertainties, critical quantities with tolerances, and introducing performance and quality bounds. It is a necessity for CPS to provide such a more comprehensive systems view. In this manner, experts can share the domain knowledge and proactively design for it. A corresponding challenge is to determine the most appropriate level of abstraction as well as formalism to support and enable such a multi-discipline specific systems view. As such, Computer Automated Multiparadigm Modeling [15] is a natural direction to explore.

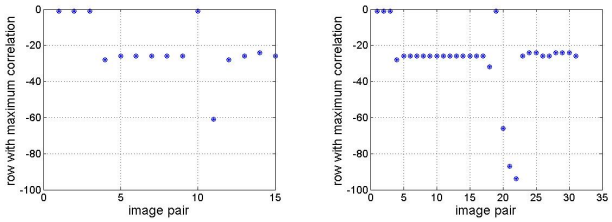
5.2 An Example

Because of the online adaptability of CPS, situational awareness is a key enabling element [25]. In the distributed Towers of Hanoi, this is reflected in a possibly unknown location of the original stack of blocks. This location is determined based on a stereopsis feature. The feature input comprises sensory information obtained by a camera system that observes the environment under the nozzle The stack of blocks is found by first moving the slider with attached nozzle and cameras across its entire horizontal range while analyzing the corresponding video stream (which, in the model, is a synthesized stream from a virtual reality scene). The location where the stereoscopic analysis finds objects nearer to the cameras is then interpreted as the location of the blocks. To find the distance of the closest object, an image from the left camera is compared to an image from the right camera. More specifically, both the left and right camera provide an image of 400x240 pixels. For the left image, a subimage of 150x240 pixels is extracted and this image is compared against a set of 100 subimages extracted from the right image, each consecutive subimage offset by an additional row. The row at which the subimages are maximally correlated (based on a cumulative *exclusive or* across all 150x240 pixels) provides a measure of closeness such that the larger the offset, the closer the nearest object in the image pair is.

From the description of the stereopsis feature it is noticed that there is a delicate dependency on the slider control feature. More specifically, feedback control moves the slider in the horizontal direction by a *dc* motor. This feature senses the current actual location of the slider and based on the desired location (in sweep mode, the end of the horizontal range) a control force acting on the slider is computed. The particular control force depends much on the desired type of control (state feedback, H-infinity, proportional-integral, adaptive control, etc.) and the performance characteristics used as criteria in the control design (e.g., rise time, overshoot, settling time, etc.) [23].

The combined system then relies on stereoscopic analysis of images in a video stream where the analysis must be carefully calibrated to detect when objects are present. The number of total images in the video stream is, however, determined by how fast the slider with attached cameras moves, which depends on the parameters of the slider control feature. For example, Fig. 9 shows the stereoscopic analysis profile for two different control laws with image pairs on the horizontal axis and correlation on the vertical axis. For the fast control, the stereopsis has only access to 15 image pairs across the slider range and the offset with maximum correlation is at row -61 (because the consecutive displacement reduces row numbers, a larger offset corresponds to a more negative row number). For the slow control the stereopsis has access to 31 image pairs (almost twice as many as for the fast control) and the offset with maximum correlation is at row -97. When calibrating the stereopsis feature, a threshold level of -80 may be used to determine whether the stack of block has been found. However, when a new control feature is then employed, for example because it becomes available post deployment, if its control is too fast the stereoscopic analysis functionality fails because there is no image pair with row result that exceeds the threshold. To support such reconfigurability, online calibration [11] is indispensable.

Note that the interaction is exacerbated when functionality is used in multiple features. For example, if the video stream is also used to enable monitoring by an operator then the multi-function video streaming is subject to different performance criteria (or different *quality of service* constraints) [12]. To detect the stack location real-time behavior with low latency is essential whereas



(a) Fast control (b) Slow control

Fig. 9 Stereoscopic analysis for a stream of image pairs of two different control implementations. *Published with permission, © 2015 MathWorks. All Rights Reserved.*

to monitor pick and place behavior throughput is more important. Reconciling the different quality of service constraint may in general be challenging and lead to suboptimal solutions for the individual functionality uses.

6 Collaborative Functionality Testing

As CPS become more common in our society and they directly interact with humans, it must be assured that they robustly behave as intended [30], which has a particular bearing when artificial intelligence (AI) is utilized [22]. The challenges for collaborative functionality testing are classified into *(i) systematic test suite generation and automated test evaluation* and *(ii) test results reproduction under minimum uncertainty.*

6.1 The Challenge

For CPS, system quality assessment must be reliably supported even after a system has been deployed and entered into operation. Moreover, quality assurance methods must adapt to the dynamically changing configurations of CPS. Technical solutions for integrating architecture-based testing [3], behavioral testing, and safety analysis [9] into a comprehensive approach are required yet lacking. All artifacts that are produced to assure quality must be consistent, traceable, and reusable [8]. The resulting test suites should be easily extendable and reidentifiable in the system architecture.

Functional testing is of particular importance for CPS. The currently available methods must be transformed from the embedded system domain to the broader CPS paradigm. Much research on model-based testing contributes to solving the challenge of systematically *generating functional test suites* [31]. The test generation, however, must be directly integrated with the test evaluation as initiated in previous work [32]. While *automated test evaluation algorithms* have been investigated in related work [4], they only provide offline test evaluation. That is, they are not yet suitable for systems that configure themselves post deployment and are operating continuously.

Further, CPS demand an online approach to the analysis of system architecture where artifacts reorganize into various test architectures during the system deployment. The test behavior should be adjustable to these new test architectures (also called test harnesses). Furthermore, new coverage metrics must be defined to measure the efficiency, reliability, and effectiveness of the test methods.

In the following, specific validation and fault mitigation strategies are exemplified. First, an example of systematic automated test evaluation is presented. Then, safety analysis for the selected scenario is illustrated to complement the test artifacts. Finally, an example fault mitigation strategy is discussed.

6.2 An Example

For the distributed Towers of Hanoi, functionality of the green block is studied. First, a validation function for an abstract scenario is semiformalized. This formalization follows the approach introduced in previous work [32]. An implementation of a Model in the Loop Test of Embedded Systems blockset [29] allows design of the validation function in Simulink using automatic transformations. The validation function consists of a preconditions (p) and assertions (a) tuple. A similar approach is then applied for safety analysis to identify a fail state of the light sensor located on the block. Lastly, a mitigation strategy for handling this particular fault is presented to complete the overall quality assurance strategy.

In the model of the distributed Towers of Hanoi, an assumption holds that each block introduced into the system ensemble is equipped with a sensor to detect light. A stack may consist of one or more blocks. Hence, if light is detected by the block, as represented by the light condition evaluating to *true*, this block is located on the top of a stack of blocks.

The validation function for the scenario evaluating the green block functionality is presented in Table 2. The assessment is performed in the context of the architecture where three types of blocks (green, red, and blue) combined with the pick and place machine constitute the system ensemble.

Assuming that each time when the following preconditions hold: (p) the system ensemble is ready to perform a pick and place action (i.e., *systemState* is equal to *ready*) and (pp) the light sensor indicates (i.e., the light *lightForGreenBlock* is equal to *true*) that the green block issues a movement service request (i.e., *greenBlock_state* is not *completed* and *moveRequest* is equal to *true*), and (ppp) no such action occured yet (i.e., *moveLeftOne* did not occur at any time before), then the set of assertions holds: (a) the system should perform a pick and place action in the context of global movement priority (i.e., *movementGlobalPriority* decreases by one), (aa) and after a predefined time duration (i.e., after time

Evaluation of Preconditions::	
IF	the system consists of three types of blocks such as, *greenBlock* and *redBlock* and *blueBlock*
AND	*systemState* is equal to *ready*
AND	*greenBlock_state* is not *completed*
AND	*moveLeftOne* did not occur anytime before
AND	the light *lightForGreenBlock* is equal to *true*
AND	*moveRequest* is equal to *true*
Validation of Assertions::	
THEN	*movementGlobalPriority* decreases by one
AND	after time duration of *timeParameter* ms
	the state *greenBlockState* is equal to *completed*

Table 2 A validation function for testing the green block functionality

Fault Detection::	
IF	the *data store memory (DSM)* detects that two blocks are located on the same stack
AND	both blocks send a *movementRequest*
Prevention and/or Reaction::	
THEN	at least one block sensor is broken
AND	the system cannot autonomously operate
AND	stop accepting service requests

Table 3 Safety analysis for checking sensor integrity

Fault Confirmation::	
IF	at least one block light sensor in the system ensemble is faulty
Fault Isolation and Mitigation::	
THEN	move the top block one spot over
AND	pick up the second block from the original stack
AND	put this second block on top of the original top block
AND	determine whether both blocks still sense light
AND	continue this process until it is determined which block is faulty
AND	continue not accepting service requests
UNTIL	the faulty sensor is replaced

Table 4 Mitigation strategy for a faulty sensor

duration of *timeParameter* miliseconds) the green block should be relocated (i.e., *greenBlockState* is equal to *completed*).

The safety analysis serves as a means to check the integrity of the light sensors on any block. The listing of fault detection constraints and prevention/reaction mechanisms is shown in Table 3. Always when the *data store memory (DSM)* of the supervisory controller on the Base ECU detects that two blocks are located on the same stack and both blocks send a pick and place request (i.e., both blocks send a *movementRequest*), then at least one sensor lost its integrity and is faulty, and the entire system ensemble cannot operate autonomously. The pick and place service requests cannot be accepted anymore either.

A mitigation algorithm for eliminating the scenario that includes a faulty sensor is listed in Table 4.

The above validation process of the green block functionality relates to a simplified scenario. A deeper look at the test scenarios indicates, though, that further-reaching analysis is required in a CPS paradigm. For example, if the light sensor is broken what implication does that have on the ability to locate the original stack of blocks? In case the stereopsis feature fails, the block cameras could be combined with the slider location measurement to determine where the stack is by detecting when the slider is over the top block. Further, if the light sensor of the green block is faulty because of a faulty power supply, the other blocks should be informed that their power supplies must be checked with respect to their integrity as well.

A complete quality assurance framework should include both test and safety harnesses. Ideally, they could be automatically created and their core specification should be reused using each others contents. They then have to separately be extended to cover specific metrics for test and for safety standards. Further, architectural elements must be considered. A systematic, comprehensive, and automated methodology that moves beyond the presented solutions is required to properly handle the reliability and robustness of CPS.

7 Conclusions

The unabated proliferation of computing power combined along with seemingly omnipresent wireless communication capabilities have come to drive a new technical systems paradigm. This paradigm is based on collaborative functionality of embedded software systems. Exponents of this paradigm are the IoT, M2M technologies, Industry 4.0, and CPS. Previous work [17] presented a needs analysis for the design and operation of CPS as collaborating embedded software systems. Three main categories were identified: (i) online configuring of an ensemble of systems, (ii) concerted function of collaborating systems, and (iii) infrastructure needs.

This work presented concrete examples of challenges for CPS as ensembles of collaborating systems that attain a concerted function. The study centers around a distributed version of the Towers of Hanoi puzzle as an Industry 4.0 smart manufacturing example. The example system includes models of the physics, electronics, network, computational functionality, and the physical geometry as a virtual reality scene.

As an example of multiparadigm modeling [16], a range of modeling formalisms with both imperative and declarative semantics are combined. In addition to finite state machine behavior, behavior in the form of ordinary differential equations (ODEs), differential and algebraic equations (DAEs), and difference equations are part of modal dynamics. There are three periodic rates in the system with multiple tasks executing on concurrent (multi-core) computing resources. Synthetic video

of 400x240 pixel images from the virtual reality scene is streamed from 2 viewpoints in 3 colors each (red, green, blue), represented by unsigned 8 bit integers, at a logical rate of 10 Hz (every 100 ms), or 400·240·2·3·8 bits per 0.1 second = 46 megabit per second (Mbps). Stereoscopic analysis is performed by image processing using array-based semantics with for loop iterations scheduled to execute in parallel on multiple cores. Composite data is exchanged between model components with variability, where the components may include causal and noncausal models, the latter in multiple physics domains (pneumatic, mechanical, electrical). Each of the model components executes in a separate thread structure. Communication and various forms of control (feedback control, switched control, feedforward control, distributed control, supervisory control, sequence control) are included as well.

As such a system of medium-size complexity, the distributed Towers of Hanoi system is representative of many facets and issues that exist in systems found in industry. Examples of illustrated challenges include collaborative control, feature interaction, multirate distributed architectures, and an automated test evaluation approach combined with safety analysis.

The presented study is based entirely on computational models and so accessible for and amenable to theoretical study. Specific challenges faced when moving to a CPS paradigm are discussed and intended to motivate model-based research of specific cyber-physical challenges necessary to make the next generation systems an operational reality.

References

1. acatech Final report of the Industrie 4.0 Working Group. Securing the future of german manufacturing industry recommendations for implementing the strategic initiative industrie 4.0. acatech—National Academy of Science and Engineering, Munich, Germany, April 2013.
2. acatech Position Paper. Cyber-physical systems. driving force for innovation in mobility, health, energy and production. acatech—National Academy of Science and Engineering, Munich, Germany, December 2011.
3. Aitor Arrieta, Goiuria Sagardui, and Leire Etxeberria. A model-based testing methodology for the systematic validation of highly configurable cyber-physical systems. In *The Sixth International Conference on Advances in System Testing and Validation Lifecycle*, pages 66–72, 2014.
4. Eckard Bringmann and Andreas Krämer. Model-based testing of automotive systems. In *Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation*, ICST '08, pages 485–493, Washington, DC, USA, 2008. IEEE Computer Society.
5. SERS Consortium. Smart Emergency Response System: GitHub Software Repository. Aug 2014. doi: 10.5281/zenodo.13978.
6. Robert B. France and Bernhard Rumpe. The evolution of modeling research challenges. *Software and System Modeling*, 12(2):223–225, 2013.
7. A. Gyrard, C. Bonnet, and K. Boudaoud. Enrich machine-to-machine data with semantic web technologies for cross-domain applications. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pages 559–564, March 2014.
8. Michaela Huhn, Martin Mutz, Karsten Diethers, Bastian Florentz, and Michael Daginnus. Applications of static analysis on UML models in the automotive domain. In Eckehard Schnieder and Géza Tarnai, editors, *Formal Methods for Automation and Safety in Railway and Automotive Systems (FORMS/FORMAT 2004)*, pages 161–172, Braunschweig, Germany, December 2004.
9. Bernhard Kaiser, Vanessa Klaas, Stefan Schulz, Christian Herbst, and Peter Lascych. Integrating system modelling with safety activities. In *Computer Safety, Reliability, and Security, 29th International Conference, SAFECOMP 2010, Vienna, Austria, September 14-17, 2010. Proceedings*, pages 452–465, 2010.
10. Edward A. Lee. Cyber physical systems: Design challenges. In *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, May 2008. Invited Paper.
11. Jesse Levinson and Sebastian Thrun. Automatic online calibration of cameras and lasers. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.
12. Jie Liu and Feng Zhao. Towards service-oriented networked embedded computing. Technical Report MSR-TR-2005-28, Microsoft Research, February 2005.
13. MathWorks®. *MATLAB® and Simulink® product families*, September 2012.
14. Pieter J. Mosterman, David Escobar Sanabria, Enes Bilgin, Kun Zhang, and Justyna Zander. A heterogeneous fleet of vehicles for automated humanitarian missions. *Computing in Science & Engineering*, 12:90–95, August 2014.
15. Pieter J. Mosterman, Janos Sztipanovits, and Sebastian Engell. Computer automated multi-paradigm modeling in control systems technology. *IEEE Transactions on Control System Technology*, 12(2):223–234, March 2004.
16. Pieter J. Mosterman and Hans Vangheluwe. Computer automated multi-paradigm modeling in control system design. In *Proceedings of the IEEE International Symposium on Computer-Aided Control System Design*, pages 65–70, Anchorage, Alaska, September 2000.
17. Pieter J. Mosterman and Justyna Zander. Cyber-physical systems challenges—a needs analysis for collaborating embedded software systems. *Software and System Modeling*, x(x):x–x, 2015.
18. Pieter J. Mosterman and Justyna Zander. GitHub Repository: Towers of Hanoi in MATLAB/Simulink for Industry 4.0. Jan 2015. doi: 10.5281/zenodo.13977.
19. Pieter J. Mosterman, Justyna Zander, and Zhi Han. The towers of hanoi as a cyber-physical system education case study. In *Proceedings of the First Workshop on CPS Education*, Philadelphia, PA, April 2013.
20. Marco Di Natale, Liangpeng Guo, Haibo Zeng, and Alberto Sangiovanni-Vincentelli. Synthesis of multitask implementations of Simulink models with minimum delays.

IEEE Transactions on Industrial Informatics, 6(4):637–651, November 2010.
21. National Institute of Standards and Steering Committee Technology. Strategic vision and business drivers for 21st century cyber physical systems. *Report of the Steering Committee for Foundations in Innovation for Cyber-physical Systems*, January 2013.
22. Stuart Russell, Daniel Dewey, and Max Tegmark. Research priorities for robust and beneficial artificial intelligence. *Research priorities for robust and beneficial artificial intelligence: an Open Letter*, January 2015.
23. Tariq Samad and Anuradha Annaswamy, editors. *The Impact of Control Technology*. IEEE Control Systems Society, February 2011.
24. Jeff W. Sanders and Graeme Smith. Emergence and refinement. *Formal Aspects of Computing*, 24(1):45–65, 2012.
25. Steering Committee for Foundations in Innovation for Cyber-Physical Systems. Foundations for Innovation: Strategic Opportunities for the 21st Century Cyber-Physical Systems—Connecting computer and information systems with the physical world. Technical report, National Institute of Standards and Technology (NIST), March 2013.
26. Adelinde M. Uhrmacher and Danny Weyns. *Multi-Agent Systems: Simulation and Applications*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2009.
27. Ovidiu Vermessan and Peter Friess. Internet of things: From research and innovation to market deployment. River Publishers Series in Communication, 2014.
28. Markus Waibel, Michael Beetz, Raffaello D'Andrea, Rob Janssen, Moritz Tenorth, Javier Civera, Jos Elfring, Dorian Gálvez-López, Kai Häussermann, J.M.M. Montiel, Alexander Perzylo, Björn Schießle, Oliver Zweigle, and René van de Molengraft. RoboEarth - A World Wide Web for Robots. *Robotics & Automation Magazine*, 18(2):69–82, 2011.
29. Justyna Zander. Model in the Loop Test for Embedded Systems: (MiLEST) Blockset Repository. Jan 2009. doi: 10.5281/zenodo.13983.
30. Justyna Zander and Pieter J. Mosterman. *Computation for Humanity: Information Technology to Advance Society*. CRC Press, Inc., Boca Raton, FL, USA, 2013.
31. Justyna Zander, Ina Schieferdecker, and Pieter J. Mosterman. *Model-Based Testing for Embedded Systems*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2011.
32. Justyna Zander-Nowicka, Ina Schieferdecker, and Abel Marrero Perez. Automotive validation functions for online test evaluation of hybrid real-time systems. In *Autotestcon, 2006 IEEE*, pages 799–805, Sept 2006.
33. Nacer-Eddine Zergainoh, Katalin Popovici, Ahmed A. Jerraya, and Pascal Urard. Matlab based environment for designing DSP systems using IP blocks. In *Proceedings of the Workshop on Synthesis And System Integration of Mixed Information technologies*, pages 296–302, Kanazawa, Japan, October 2004.