Extensions to Pattern Formats for Cyber Physical Systems

Antonio Maña
University of Málaga, Spain
Bulevar Louis Pasteur, 35, 29071, Málaga, Spain
amg@lcc.uma.es

Ernesto Damiani
University of Milan
Via Bramante 65, 26013 Crema, Italy
ernesto.damiani@dti.unimi.it

Sigrid Gürgens
Fraunhofer SIT Institute, Germany
Rheinstrasse 75, 64295 Darmstadt, Germany
sigrid.guergens@sit.fraunhofer.de

George Spanoudakis
City University London
Northampton Square, London, EC1V 0HB, UK
g.e.spanoudakis@city.ac.uk

ABSTRACT
Cyber Physical Systems (CPSs) are becoming not only increasingly complex but also increasingly important. Traditionally, CPSs were self-contained monolithic systems operating in a well-defined environment. CPSs today have evolved to large-scale systems including multiple interacting components. Due to the inherent complexity of the physical environments where CPSs operate and the ever changing operational and context conditions of these environments, CPSs need to be able to adapt in a semi- or fully autonomic manner. At the same time, in the course of adapting themselves, they need to satisfy key quality, security and privacy (QSP) requirements and be resilient to threats even if those were unknown at the time the CPS was designed. Engineering CPSs with such capabilities is a challenging problem that requires expert knowledge. Our approach to addressing this problem is based on the use of QSP-preserving patterns, i.e., patterns that encode proven design and engineering solutions providing particular QSP properties. We argue that existing pattern formats miss some essential features of CPSs. As a step towards the realization of approach, in this paper we present a new pattern format that enables the representation of most important characteristics of QSP preserving patterns for CPS. Our format provides an initial basis for defining a full pattern language for engineering QSP preserving CPS.

Categories and Subject Descriptors
Software and its engineering~Software design engineering; Software and its engineering~Patterns; Software and its engineering~Unified Modeling Language (UML); Computer systems organization~Embedded and cyber-physical systems

General Terms
Security Patterns, Security modeling and engineering, Pattern-based system engineering.

Keywords
Computer-processable knowledge representation

1. INTRODUCTION
Traditional Cyber-Physical Systems (CPSs) were isolated, self-contained systems operating in well-defined environments (e.g., embedded in vehicles or electronic appliances) and exposed to limited threats. Recently, however, CPSs have evolved into large-scale systems, including multiple interacting components, connected with other cyber and physical systems (e.g., energy infrastructures). CPSs are exposed to a large number of known and unknown threats. They also need to operate under continually changing conditions and to adapt and be resilient to them. Such adaptive behavior is hard to guarantee at design time; therefore, engineering CPSs is becoming an increasingly complex task.

To illustrate the challenges that the development of CPSs entails, consider electricity provisioning. With the advent of regenerative energy, electricity can be obtained from many different sources besides traditional power plants (sunlight, wind, rain, tides, waves and geothermal heat). Source diversity has fostered a major change from a centralized grid topology to a highly distributed one. Smart grids, for example, involve sensors that measure electricity production and consumption, controllers that collect and report measurements, services that control the generation, distribution and storage of energy and several cyber components enabling the processing and communication of this information.

In addition to functional requirements, some important non-functional ones need to be considered, most notably regarding quality, security and privacy (which we will refer to in the paper as QSP properties). For instance, the insertion of forged (i.e., non-authentic) information about the amount of available electricity that a wind turbine can produce, can lead to the alteration of an electricity distribution control service, which in turn can potentially lead to a breakdown of the entire electricity grid.

As this example suggests, CPSs are complex, composite systems that deal with multiple physical parameters in a coordinated and coherent way. To do so, CPSs include highly interconnected and interdependent cyber-components that provide critical functions. Components of CPSs may evolve independently from each other in order to offer new capabilities and/or to adapt to their changing environments. Some of these cyber-components have important

security and safety requirements, and may interact with humans. They may also operate within the realm of different and not always harmonized jurisdictions and may need to transfer data across them. Furthermore, cyber-components of CPSs may have diverse computational features and roles.

Further, both cyber and physical components of CPSs operate under distributed ownership and control, and within uncontrolled and unprotected physical environments. These may be characterized by changing operational conditions and constraints (e.g., changing temperatures, physical damage, changes to power supply etc.). As a consequence of these factors, CPSs may be deployed in adverse operating conditions, related to the unpredictable nature of the physical world, which can compromise the availability and security of some of their components. They are very vulnerable to security attacks through their ICT or physical components. Local sensors, for example, can be destroyed or tampered with. This may result in losing essential information (e.g., the correct amount of energy that is being consumed at a given point in time) or transmitting misleading information that may be used to attack these systems. Likewise, network and application level cyber-components may be subject to cyber attacks that may cause the CPS to violate security or privacy requirements (Chen et al., 2011).

The composite structure of CPSs implies that attacks to any of their components may compromise the overall security. CPSs may also be subject to conflicts with the goals of other interacting CPSs, which must be solved cooperatively. They may also generate, make use of and inter-relate massive amounts of personal data in ways that can potentially breach legal and privacy requirements (Lioudakis, Kaklamani, and Venieris, 2009). Finally, the infrastructures that a CPS relies on are often outside its control and may change frequently. Infrastructural changes can also compromise the security, resilience and availability of CPS operations and the service(s) that they offer.

In order to cope with these challenges and to support the engineering of CPSs that meet functional as well as QSP requirements, we propose a pattern-based approach. In particular, the patterns that underpin our approach define generic ways for composing and coordinating different types of components (e.g., computer hardware, networks, sensors, actuators) in such a manner that they provably preserve specific QSP properties. Thus, we refer to them as QSP patterns. One important aspect of our QSP patterns is that they support the preservation of QSP properties at runtime: On the one hand, they include information about conditions to be monitored in order to detect (i) changing operational conditions, and (ii) violations and threats to their desired QSP properties. On the other hand, in case such monitoring reveals the need to adaptation, the QSP pattern defines the necessary actions to be taken. In this paper, we introduce a pattern format for describing QSP patterns and present an example of a QSP pattern that shows the viability of this format.

The rest of the paper is structured as follows: Section 2 explains the challenges that need to be addressed in order to provide adequate support for engineering CPS satisfying specific QSP properties, Section 3 introduces a motivating example pattern that will be used to define extensions to existing pattern formats that we deem adequate to cover CPS-specific needs. Section 4 describes these extensions and uses them for a renewed representation of our example. Finally, Section 5 presents conclusions and introduces future work.

2. QUALITY, SECURITY AND PRIVACY ENGINEERING FOR CPS

Please Preserving QSP properties in CPSs under the circumstances discussed above is a challenging scientific and engineering problem. It involves determining the exact compositional structure of a CPS and the functional and non-functional characteristics of its individual components. It further involves identifying and dealing with complex dependencies between system-wide and/or component level QSP properties. Guaranteeing one of such properties may have a negative effect on others (e.g., encryption-based confidentiality vs. energy consumption) which in turn requires defining trade offs. All these processes have a strong dependence on expert knowledge. Hence, engineering CPSs in ways that can simultaneously guarantee a set of QSP properties of interest demands for an integrated QSP-aware and expert knowledge-based design process addressing both system development and runtime operation.

Current design methods, techniques and tools have evolved to accommodate some aspects of the design and evolution of CPS. However, new artifacts and tools are required for ensuring QSP-preserving operation of a CPS at the system level, while at the same time preserving QSP properties at the level of the individual components.

We claim that an effective CPS engineering approach needs to take into account the structure of CPS, the features of their components at all layers (physical –non ICT– components, sensors/actuators, networks, software, and human participation), structural and functional relationships between these components, and contextual conditions of their operation, as all these elements can affect the preservation of QSP properties. It should take into account typical constraints of CPS devices and components, and should consider requirements related to CPS interoperability, adaptability and optimization.

A CPS engineering approach needs also to be aware of and support effectively the composition of cyber and physical components. This composition is inherently different from traditional software systems composition. Indeed, software centric approaches to composition (e.g., software service/components orchestration workflows) do not address the physical layers of CPS systems. A successful CPS engineering approach needs to support within a single framework multiple alternative forms of composition, integrating different and heterogeneous components and devices in a composite CPS and considering different perspectives (e.g. cyber, physical, human). Moreover, it has to include powerful abstraction mechanisms to tame the complexity of the hierarchical composition of CPSs. From a QSP perspective, this approach should also take into account the quality and security characteristics of the underlying mechanisms that are used to realize CPS compositions (e.g., network management middleware, messaging middleware, software components orchestration middleware).

Understanding the above factors demands expert knowledge from diverse disciplines such as computer science, analog and digital communication and physics. Such knowledge is often captured and expressed in multi-perspective domain models that need to be properly interrelated. The sheer amount of effort and skills needed for applying that expert knowledge to capture and express such models from scratch is hindering CPS development and leads often to domain specific vertical solutions.

Our proposal is based on capturing such expert knowledge in a way that is usable by engineering-assistance tools. Among the artifacts that we use for capturing the expert knowledge required for the successful engineering of CPSs, we present in this paper the artifact used to capture QSP solutions, which we call QSP-patterns and is materialized as an enhancement of current security patterns.

At the software architecture level, security patterns have already proven their usefulness in capturing re-occurring solutions and thus supporting software development. We claim that patterns particularly dedicated to the CPS specific challenges discussed above can effectively support a model driven approach to CPS engineering. Moreover, the overarching nature of some recent modeling formalisms for embedded systems, and in particular SysML (OMG, 2012), and to a lesser extent MARTE (OMG, 2011), has shown that it is possible to provide solutions that are accepted by different industries, and has paved the way for a pattern-based and QSP-aware engineering of CPS.

We argue that in order to provide a solid foundation for addressing the challenges described above, there is a need for:

- providing abstraction mechanisms to model the inherent complexity of CPSs and their alternative and complex compositional structures;
- reusing capabilities to address the needs for reducing time-to-deployment for CPS by a significant percentage while ensuring QSP requirements;
- increasing trust in solutions to ensure that engineers have solid and rigorous foundations for designing their CPSs;
- fostering inter-sector applicability in order to de-verticalize solutions and to break the trend of solutions being too specific due to the adoption of vertical technologies and frameworks, which "isolate" devices and CPS and do not promote device and system interoperability across different application sectors and markets;
- providing separation of responsibilities to ensure that different actors along the value chain play an adequate role, and that the result of the expertise and work of each of them is coherently integrated into an integrated engineering approach; and
- encoding the above in the form of machine-readable and –processable patterns. Such patterns should be applicable in various CPS design, engineering and runtime management activities (e.g., CPS design verification and adaptation, model-driven CPS development, runtime CPS modification).

3. QSP PATTERNS FOR CPS

Security patterns have been successfully used in the past to support system engineering activities, such as designing software architectures with known QSP properties and trade-offs, to verify whether service oriented software system designs satisfy required security properties, and even to adapt them if they do not –see (Pino et al. 2014, Pino and Spanoudakis, 2012)–. Furthermore, they have proven their value as an instrument for communicating design good practice, and educating software architects. However, the security patterns used for the above purposes have focused on computing systems (i.e., cyber-systems) rather than CPSs, and therefore miss aspects that are important for CPSs, such as the physical models of some of the CPS components. Also in some of the approaches, they are expressed in informal and semi-formal languages that do not allow their use in reasoning mechanisms that could assist the engineering process. This limitation has already been identified and remains an important impediment for pattern based CPS engineering. Existing attempts to document patterns for CPS, as for instance (Fernandez et al., 2009), lack a CPS-specific format and result in missing some of the important essential characteristics, in particular, their physical aspects. So, in spite of on-going promising work towards defining computer-supported and engineering-oriented security patterns, called COmputer-Supported Security Patterns (COSSP) (Mana et al., 2013, Arjona, Ruiz and Mana, 2014), and the fully automated pattern processing approach followed in (Pino et al. 2014, Pino and Spanoudakis, 2012) to the best of our knowledge to date there does not exist an approach that addresses all necessary aspects. Hence our objective in this work is to overcome these limitations.

3.1 Synchronously-controlled distribution line (SCDL) pattern: an informal example

To clarify our ideas, we introduce a QSP pattern for synchronously controlled gas pipelines whose description is based on a variant of the Fowler Form (Fowler, 1996). It is important to note that our approach is not based on this form, but describes extensions (see section 4.3) that can be applied to any existing pattern form. In particular, we have chosen this form to reduce the existing descriptions to the minimum and to highlight the extensions. We also provide a brief discussion of how a developer could use this pattern in system design. Our example enables us also to highlight some open problems that will be tackled by the paper.

Title
Synchronously-controlled distribution line (SCDL)

Problem
In order to avoid failures or attacks, a control system for a distribution pipeline must ensure that it will operate in virtual synchrony and with the desired density of readings. If this cannot be ensured, the asynchronous nature of the sensors used to obtain information from physical variables can originate failures or impose the system to attacks. The purpose of this pattern is precisely to guarantee that a physical system (a distribution pipeline, say for gas or electricity) controlled by a distributed asynchronous sensor system that is subject to attacks/failures will operate in virtual synchrony and with the desired density of readings. The pattern allows the entire system logic to work as if it was executed on a synchronous platform robust with respect to attacks.
The elements involved in the pattern are the PIPELINE and a number of SENSOR nodes (PHYSICAL part) and one or more CONTROLLER nodes (CYBER part).

The ***arguments*** of the pattern are *(i)* a matrix ***M*** expressing inter-sensor *distances, (ii*) the minimum density **r** of sensors per meter required for a reliable control, *(iii)* the *frequency f* of readings from each sensor node to the controller.
The ***physical system*** must operate under three constraints:

- *Bounded Local Clock Error* - Each sensor *i* has access to an approximation of the true global time *t* via a local clock C_i where the maximum error (of each local clock) is ε, i.e.,

$$| C_i - t | < \varepsilon.$$

- *Bounded Reading Time Variation* - Each node i executes its reading in a time T_i having a maximum variation w.r.t. a nominal value d, i.e.,
$$| T_i - t | < d.$$
In other words, the time required for the read action lies in the interval $t \pm d$.
- *Minimum density* – The number of physical readings per meter must never fall below r.

The ***ICT system*** includes the following components:

- *Reliable Bounded Message Delivery* protocol. Messages are: (1) delivered to their destinations reliably, and (2) they are time-stamped by sources (the SENSORS) with a locally computed timestamp based on d and e, so that each received message can be mapped to an end-of-period (EOP) time by the CONTROLLER.
- *Interpolation* routine. Attacks could result in the CONTROLLER receiving different message sets, even though the network delivers each one correctly; the pattern specifies (i) a mechanism to control if the density r is achieved and (ii) a routine for computing linear *interpolation* of the closest sensors at the same EOP to substitute the missing readings if the minimum density is not achieved.

Solution

Use a *synchronizing controller* that stores readings received from sensors and processes them periodically in fixed intervals. All readings received in a period are consumed at the end of the fixed interval. The controller is connected to the sensors using a *Reliable Bounded Message Delivery* protocol that ensures data origin authenticity. Additionally, the controller uses an Interpolation routine to compute linear *interpolation* of the closest sensors at the same EOP to substitute missing or corrupted readings. Therefore, the ***guarantees*** [1] after applying the pattern are:

(1) CONTROLLER will consume readings at approximately the same times, with period $P = 1/f$. More specifically, readings generated during period P_k are consumed by CONTROLLER at time EOP(P_k) So, it looks like outputs of sensors are received by the CONTROLLER synchronously.
(2) Readings received and accepted by the CONTROLLER originate from legitimate sensors.
(3) At each EOP(P_{ks}), the CONTROLLER will be able to detect whether the source density is enough for reliable control.

4. TOWARDS A QSP PATTERN FORMAT TO BUILD A PATTERN LANGUAGE FOR CPS

4.1 Goals for QSP Patterns

Our QSP patterns will support the development of CPSs in several respects. First, they will identify all components (e.g. individual services, sensing and communication components, etc.) needed for the system to satisfy the functional requirements. Additionally, they will specify the requirements to be satisfied by these components in order for the CPS as a whole to meet specific QSP requirements. In other words, QSP pattern-based development can focus on ensuring that CSP components and their composition comply to the specification provided by the pattern. There is no need to reason about resulting QSP properties, these are assured by the pattern, i.e. hold by-design. Finally, QSP patterns being machine-readable enables tool-assisted engineering processes: The developers select adequate patterns that in turn can induce the query for and the selection and composition of the components specified by those patterns.

QSP patterns will further support the preservation of QSP properties at runtime. More specifically, a QSP pattern will include information about the conditions that need to be monitored and/or dynamically tested at runtime in order to preserve the QSP properties guaranteed by the pattern. The monitoring/testing information allows detecting changing operational conditions, as well as violations and threats to their desired QSP properties. It further allows verifying that CPS components satisfy specific QSP properties and configurations that had been used as assumptions in proving the QSP properties guaranteed by the pattern to hold. Finally, QSP patterns contain information about actions to be taken at runtime in order to adapt the compositional structure of the CPS in ways that preserve or restore the desired QSP properties.

4.2 Using QSP Patterns

QSP patterns will be used to support both the development and the runtime management of cyber physical systems. During the development of a CPS they can be used to

a) analyze a given design of a CPS in order to check if it satisfies a desired QSP property; or to
b) generate a partial design of a CPS that is guaranteed to satisfy a given property.

At runtime, QSP patterns can be applied in order to

c) determine and monitor the conditions that need to be satisfied by the components of a CPS for it to satisfy a given QSP property; and to
d) adapt the compositional structure of a CPS in ways that can guarantee the desired QSP property.

QSP patterns could be used under (a) to check if a CPS design as a whole (or parts of it) satisfies a given QSP property. In our example SCDL pattern, for instance, a gas distribution CPS design can be analyzed based on the pattern to check if the topology of pressure sensors required by the SCDL pattern for guaranteeing a minimum accuracy of pressure readings is preserved by each of the different pipelines of the gas distribution CPS. In this case, based on the SCDL pattern, the analysis would check that the distance between sensors on each of the system's pipelines does not exceed the maximum threshold established by the pattern in order to guarantee the accuracy property. Specific components of the gas distribution CPS design could also be checked according to the pattern to ensure the preservation of other properties. The controllers of different pipelines or the general controller of the CPS could, for instance, be checked to establish whether they are capable of consuming sensor readings with the minimum frequency, a property that is important for guaranteeing the freshness of pressure information within the gas distribution CPS. The main benefit arising from the use of QSP patterns for such analysis is avoiding to carry out reasoning based on first-principles (i.e., using the analytic model that underpins the conditions regarding the topology of sensors in order to check and

[1] NOTE: In our pattern, these guarantees are expressed by a single property called Synch-Complete-Communication.

guarantee the accuracy of sensor readings). As a consequence of not having to reason from first principles, pattern based analysis is scalable to large systems models.

As an example of (b), the physical model of the SCDL pattern could be used to dictate where exactly to place sensors on different pipelines of the gas distribution CPS. Similarly, once the structure of pipelines and sensors has been established, the pattern could be used to dictate the number and types of controllers that will be required in order to process the readings from the sensors, and how sensors should be allocated to them. This could be derived from the characteristics of different types of controllers (e.g. max signal processing frequency per time unit) and sensors (e.g. signal frequency). Further, the ICT (cyber) model of the pattern would specify the actual communication protocol to be used in order to guarantee the reliability of messages consumed by the controller.

As an example of (c), the SCDL can be used to establish monitoring conditions regarding the aliveness and faultless operations of sensors on different pipelines. These conditions can be derived from the fact that the pattern requires sensors to be placed on a maximum distance from each other on a pipeline. It is thus possible to figure out the minimum number of sensors that should be alive at an instance of time on a pipeline of given length in order to guarantee the accuracy of readings. Furthermore, as the pattern establishes the maximum difference between the pressure readings of sensors placed at a given distance from each other, it is possible to derive conditions for cross checking the pressure readings of adjacent sensors and ensure that they remain within the boundaries indicating that they are correct.

Regarding (d), assuming a scenario where there is some localized or wider failure of controllers, the SCDL pattern can be used to trigger the search, binding, and activation of new controllers into the gas distribution CPS as well as the dynamic assignment of sensors to them (the latter would be based on an analysis similar to what underpins case (b) above but at runtime).

4.3 Extensions to pattern formats required for QSP Patterns

Traditional pattern description formats, like the Alexandrian, POSA, GoF, or Schumacher and Fernandez Security Patterns, or even recent computer-readable formats like COSSPs, fall short of providing detailed information for systematically capturing the characteristics of CPSs. In particular, they do not support the identification of and differentiation between physical and ICT components, and (more importantly) the description of their relations. Further, these generic pattern formats are very flexible and therefore it is highly likely that patterns produced by different authors are neither consistent nor comparable. Yet in order to enable the CPS developer to choose an adequate pattern, aspects like marking the physical and ICT components, describing their relations, supporting the systematic and consistent cross-referencing of those elements and relations, among others, need to be supported by the pattern format.

More specifically the following extensions to existing pattern formats will be necessary for QSP patterns to serve effectively as a tool for engineering CPS systems:

- *Component characterization.* To adapt to the composite and socio-technical nature of CPS we add a section to describe the components (both physical and cyber components are considered here). Each component is described by a unique *ID* (identity), a set of *scopes* (e.g. Physical, ICT, Human), a set of *types* (e.g. sensor, actuator, control, interface), and the QSP properties they provide. In order to ensure comparability and interoperability of the pattern descriptions, in this paper we will assume that the possible values are standardized, but our goal is to later introduce the use of ontologies to support more flexible schemes for this purpose.
- *Component relations.* To cover the highly interconnected nature of CPS we add a section describing the different relations between CPS components. Each relation has a unique identifier (ID), points to the involved components, and defines the nature of the relation and its role in the pattern.
- *Component orchestration.* To adequately represent dynamically-adaptive CPSs, a QSP pattern must incorporate a description of the architecture and orchestration of the components. In this paper we will describe it using natural language but we envisage the use of more formal descriptions such as BPEL or SysML.
- *Physical model.* A QSP pattern must also describe the physical aspects of the solution it represents. The focus of this section is to describe the physical processes that are controlled by the CPS.
- *Cyber model.* This section will be used to describe the ICT aspects of the solution presented in the pattern. The focus of this section is to describe the ICT components that are used in the CPS and the ways they are connected. This section will also deal with the interface components (e.g. sensors and actuators).
- *QSP Properties.* This section will be used to specify the QSP properties provided by the pattern (as a whole), given that the components satisfy the QSP properties listed in their respective characterizations.

Examples of the use of these extensions are included in the revised description of the SCDL pattern in section 4.4 below. It should be noted that the above list of extensions is not exhaustive. Indeed, our expectation is that in the course of modeling additional examples of CPS systems, we will identify the need for further extensions or for extending or adapting the ones presented above.

4.4 The Synchronously-controlled distribution line (SCDL) pattern revisited

In this section we present the example pattern in a more formal way using the proposed extensions. We also discuss why the extensions and modifications introduced above result in an accurate description of the pattern and how they enable the pattern to be used to actually build a specific system. In order to compare the improvements and advantages introduced by the proposed extensions, we will use again the Fowler Form as a basis.

Title
Synchronously-controlled distribution line (SCDL)

Problem
In order to avoid failures and counter attacks, a control system for a distribution pipeline must ensure that it will operate in virtual synchrony and with the desired density of readings. However, the asynchronous nature of the sensors used to obtain information from physical variables can originate failures or impose the system to attacks if the control system has to operate with an unpredictable density of readings. The purpose of this pattern is

precisely to guarantee that a physical system (a distribution pipeline, say for gas or electricity) controlled by a distributed asynchronous sensor system subject to attacks/failures will operate in virtual synchrony and with the desired density of readings. This pattern allows the entire system logic to work as if it was executed on a synchronous platform robust with respect to attacks.

Solution

Use a synchronizing controller that stores readings received from sensors and processes them periodically in fixed intervals. All readings received in a period are consumed at the end of the fixed interval. The controller is connected to the sensors using a *Reliable Bounded Message Delivery* protocol that ensures data origin authenticity. Additionally, the controller uses an Interpolation routine to compute linear *interpolation* of the closest sensors at the same EOP to substitute missing or corrupted readings. Therefore, the ***guarantees*** [2] after applying the pattern are:

(1) CONTROLLER will consume readings at approximately the same times, with period $P = 1/f$. More specifically, readings generated during period P_k are consumed by CONTROLLER at time EOP(P_k) So, it looks like outputs of sensors are received by the CONTROLLER synchronously.

(2) Readings received and accepted by the CONTROLLER originate from legitimate sensors.

(3) At each EOP(P_{ks}), the CONTROLLER will be able to detect whether the source density is enough for reliable control.

The elements involved in the pattern are the PIPELINE and a number of SENSOR nodes (PHYSICAL part) and one or more CONTROLLER nodes (CYBER part).

The ***arguments*** of the pattern are *(i)* a matrix ***M*** expressing inter-sensor *distances, (ii*) the minimum density of sensors per meter required for a reliable control **r** *(iii)* the *frequency f* of readings from each sensor node to the controller.

Component characterization

Pipeline
ID: Pipeline_1
scopes: Physical
types: physical component

Sensors
ID: Sensor_1 … Sensor_n
scopes: Physical, ICT
types: interface

Controller
ID: Controller_1
scopes: ICT
types: controller

Component relations

Pipeline_1-Sensor_i ($1 \le i \le n$)
ID: Pipeline_1-Sensor_i
nature: physical_to_ICT
components: Pipeline_1, Sensor_i

Sensor_i-Controller_1 ($1 \le i \le n$)
ID: Sensor_i-Controller_1
nature: ICT, asynchronous
components: Sensor_i, Controller_1

Component orchestration

(1) Components are statically connected in this pattern. All n sensors (Sensor_1 … Sensor_n) are connected to the pipeline (Pipeline_1) through appropriate physical connections (Pipeline_1-Sensor_1 … Pipeline_1-Sensor_n) and are connected to the controller (Controller_1) using a *Reliable Bounded Message Delivery* protocol (Sensor_1-Controller_1 … Sensor_n-Controller_1).

Physical model

The physical system model specifies three constraints.

- *Bounded Local Clock Error* - Each sensor (*Sensor_i*) has access to an approximation of the true global time t via a local clock C_i, where the maximum error (of each local clock) is ε, i.e.,
$$| C_i - t | < \varepsilon.$$
- *Bounded Reading Time Variation* - Each sensor (*Sensor_i*) executes its reading in a time T_i having a maximum variation w.r.t. a nominal value of d, i.e.,
$$| T_i - t | < d.$$
In other words, the time required for the read action lies in the interval $t \pm d$.
- *Minimum density* – The number of physical readings per meter must never fall below r

Cyber model

The SCDL pattern's ICT model includes the following components:

1. *Model of Reliable Message Delivery* protocol. Messages are (1) delivered to their destinations reliably and (2) time-stamped by sources (the SENSORS) with a locally computed timestamp based on d and ε so that each received message can be mapped to an end-of-period (EOP) time by the CONTROLLER.
2. *Model of density control*. SENSOR failures could result in the CONTROLLER receiving different message sets at each period; the *density control model* represents (i) a mechanism to control if the density r is achieved and (ii) a routine for computing linear interpolation of the closest sensors at the same EOP to substitute the missing readings if the minimum density is not achieved.

An STS is a finite state automaton that consists of states and transitions between states, labeled with actions, guards, and update mapping. In our pattern, the message delivery protocol can be modeled via two STSs, one for the SENSORs and one for the CONTROLLER. The CONTROLLER-side STS may consist of just four states (IDLE, RECEIVING, COMPUTING and ERROR). The transition from IDLE to RECEIVING is triggered by the arrival of a message from a SENSOR, while the opposite one (RECEIVING to IDLE) is triggered when the message has been entirely received[3]. The transition from IDLE to COMPUTING is triggered by the CONTROLLER's internal clock at each EOP and enabled by the guard [*density* > *r*]. Another

[2] NOTE: In our pattern, these guarantees are expressed by a single property called Synch-Complete-Communication.

[3] We leave it to the reader to consider an STS model for SENSORs so that transmission is enabled only when CONTROLLER is IDLE.

transition, from IDLE to ERROR, is enabled by the opposite guard [*density* ≤ *r*]. Finally, a transition from ERROR to IDLE enables going back to receive messages from the next period.

.

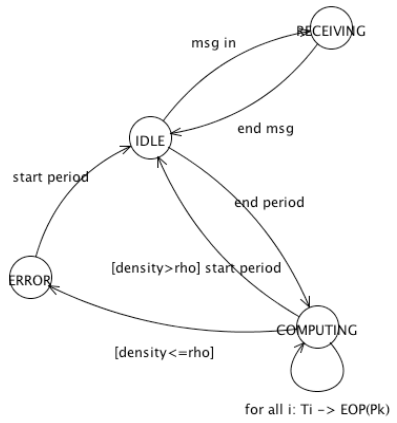

Fig. 1: The CONTROLLER's sample STS

It is important to note that the complexity and detail of the cyber model in the pattern depend on the attack/failure representation. The simple STS described above for the CONTROLLER supports guarantee (2) in the case the only possible sensor failures are on-off ones, where a faulty sensor (e.g., disabled by a physical attack) is not able to send messages during one or more periods. A model representing only this on-off failure would be simpler, but only adequate for this pattern in the case perfect message security may be assumed for some reason (e.g., physically unbreakable cable connections between SENSORS and CONTROLLERS). A more suitable model would include sophisticated attacks, like some attacker tampering with sensor readings, thus allowing us to test/prove that the system is robust with regards to those attacks. Likewise, the cyber-model may not model data connections, but assume that the required connections already exist and introduce bounded latency μ, $\mu_{min} \leq \mu \leq \mu_{max}$.

A pattern supporting guarantees in the case of *on-off* failures would be adequate to the design of a real system only in the (improbable) case message integrity may be assumed for some reason (e.g., an unbreakable cable connection between SENSORS and CONTROLLER). An SCDL++ pattern would include a more complex STS (Anisetti et al., 2013) model featuring more sophisticated attack models, like an attacker tampering with the messages. This way, SCDL++ would allow us to test (or prove) that a system designed according to SCDL++ is robust with respect to tampering.

The inclusion of the simple STS illustrated in Figure 1 in the pattern is not just a help to the design: it enables generating test cases supporting the SCDL pattern's guarantee (1), i.e. test cases consisting of messages containing erroneous or missing timestamps. Also, the STS supports testing the pattern's guarantee (2), generating test cases where CONTROLLER receives a variable number of messages, i.e. where sensors fail in all possible ways resulting in *density* greater or smaller than *r*. Likewise, it can be used to derive monitoring rules for the pattern.

Note that STS is only one possibility to model CPSs. Other possibilities include the Security Modeling Framework SeMF (Gürgens, Ochsenschläger and Rudolph, 2005) that provides means to prove QSP properties of a CPS using the QSP properties of its components as assumptions, without the need to specify an attack model.

Presenting the STS model (or any other model describing the CPS) in a computer-processable way (e.g. as extended *Symbolic Transition Systems* – eSTS) allows for tool-supported generation of test cases and monitoring rules, given that the extended pattern format we propose includes the computer-processable characterization of the CPS's components, their relations and orchestration.

Our proposed pattern format fosters precision, homogeneity and interoperability and enables CPS developers to identify the requirements to be satisfied by the CPS components, thus enabling them to verify whether the components to be used for a concrete CPS comply with these requirements. Further, it supports the composition of components by specifying the exact way of composing these components in order to achieve certain QSP properties both from the physical and ICT perspectives. The pattern format further enables the comparison of patterns with respect to specific characteristics of the components and/or their composition, thus supporting the developer in choosing an adequate pattern.

5. CONCLUSIONS

This paper has presented some challenges that arise when trying to describe patterns for QSP-enabled CPS, has identified the aspects of current pattern formats that need to be extended or improved, and has proposed a set of extensions to adequately capture the important aspects of CPS. Additionally, we have illustrated the new format using a running example that has also served to show what elements are needed for the adequate description of this type of patterns. Based on our example, we have described some extensions and improvements that we consider necessary in order to adequately support the engineering of CPS. We have then used the proposed format to present the motivating example as a first pattern of a future pattern language for CPS. We have the goal of producing a pattern language composed of a relevant number of patterns for QSP-enabled CPS and to describe them using our proposed extensions to pattern formats. Therefore, we will also continue improving the proposed format as we develop the collection of patterns. In particular, we plan to follow the approach proposed by COSSPs of adding computer-oriented extensions to the proposed pattern format. One line of improvement will focus on using the novel capabilities introduced by SysML for representing problems and rationales as a way to improve the integration between the pattern description and the pattern diagrams. It is also our interest to extend the use of QSP Patterns to runtime situations and in particular to support monitored operation of CPS.

6. ACKNOWLEDGMENTS

We would like to thank the members of our Writers' Workshop at PLoP'14 for their work in analyzing this paper and their help in improving it, and very especially our shepherd Ralph Johnson, for his guidance and support during the shepherding of this paper.

The work of A. Maña is supported by the CUMULUS and PARIS projects. The work of E. Damiani is supported by the CUMULUS project. The work of S. Gürgens is supported by the ASSERT4SOA project. The work of G. Spanoudakis is supported by the CUMULUS project.

REFERENCES

[1] Chen, D., Chang, G., Jin, L., Ren, X., Li, J., Li. F. A Novel Secure Architecture for the Internet of Things. 2011 Fifth International Conference on Genetic and Evolutionary Computing:311—14, 2011. 2011 Fifth International Conference on Genetic and Evolutionary Computing, 29 Aug.-1 Sept. 2011, Xiamen, China.

[2] Lioudakis, G.V., Kaklamani, D.I., Venieris, I.S. Personal data protection under the InfoCity lights. 2009 European Wireless Technology Conference (EuWIT):104—7, 2009. 2009 European Wireless Technology Conference (EuWIT), 28-29 Sept. 2009, Rome, Italy.

[3] Slomka, F., Kollmann, S., Moser, S. and Kempf, K. A Multidisciplinary Design Methodology for Cyber-physical Systems.

[4] Object Management Group (OMG). OMG Systems Modeling Language (OMG SysML). Version 1.3. June 2012. Available online at http://www.omg.org/spec/SysML/1.3/PDF

[5] Object Management Group (OMG): Modeling and Analysis of Real Time and Embedded systems, version 1.1 (MARTE). June 2011. Available online at http://www.omg.org/spec/MARTE/1.1/

[6] Pino L., Spanoudakis G., Fuchs A., Gürgens S., Discovering Secure Service Compositions, 4th International Conference on Cloud Computing and Services Sciences, Barcelona, Spain, April 2014

[7] Pino L., Spanoudakis G.: Constructing Secure Service Compositions with Patterns, 8th IEEE World Congress on Services, Hawaii, USA, June 2012

[8] Mana, A. Fernandez, E.B., Ruiz, J.F. and Rudolph, C. Towards Computer-oriented Security Patterns The 20th International Conference On Pattern Languages Of Programs PLoP'13. 2013

[9] Arjona, M., Ruiz, J.F and Mana, A, Security Patterns for Local Assurance in Cloud Applications. International Workshop on Engineering Cyber Security and Resilience ECSaR'14 in The Third ASE International Conference on Cyber Security 2014.

[10] Fernandez, E. B., Wu, J., Larrondo-Petrie, M. M., Shao, Y. On Building Secure SCADA Systems using Security Patterns. Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies. 2009.

[11] Fowler, M. Analysis Patterns: Reusable Object Models. Addison-Wesley. ISBN 0-201-89542-0. 1996.

[12] Anisetti, M., Ardagna, C. A., Damiani, E., Saonara, F. A test-based security certification scheme for web services. TWEB 7(2): 5 (2013)

[13] Gürgens, S., Ochsenschläger, P., Rudolph, C. On a formal framework for security properties. Computer Standards & Interfaces 27(5): 457-466 (2005)