DESIGN BY CONTRACT OF CYBER-PHYSICAL SYSTEMS DRIVEN BY SIMULATION AND BASED ON PROPERTIES MODELING

Andrea Tundis, Max Mühlhäuser

Telecooperation Lab, Department of Computer Science, Technische Universität Darmstadt
Hochschulstrasse 10, 64289, Darmstadt, Germany

tundis@tk.tu-darmstadt.de, max@tk.tu-darmstadt.de

ABSTRACT
Requirements elicitation and analysis is the basis for the successful development of a Cyber-Physical Systems (CPS). The misunderstanding of one or more requirements, due to different skills and knowledge between stakeholders and engineers, could com-promise the success of an entire project with harmful consequences. Usually, agreements on the system to be delivered and related expected results are based on textual requirements with a big lack of not being computationally verifiable and difficult to trace. To this purpose, the employment of innovative engineering tools for supporting the modeling and the verification of system requirements represent a viable solution. In this context, the pa-per proposes the exploitation of a Properties Modeling (PM) approach combined with Simulation techniques as Design-by-Contract method for CPS. In particular, PM is adopted for sup-porting the definition and the representation of system requirements and constrains as computable entities, whereas a Simulator is developed and exploited for enabling their automatic verification. Such combination is used as tool for defining requirements and conditions and verify their fulfillment before the sys-tem deployment. The results gathered from the simulation represent the contract on which the parties can agree for the realization of the actual system. The approach is exemplified in the Smart Grid domain.

Keywords: Properties Modeling, Requirements Specification, Simulation-based Verification, Design-by-Contract, Cyber Physical Systems, Smart Grids

1. INTRODUCTION

According to the International Council on Systems Engineering (INCOSE 2017), the causes that determine the success or the failure of a product, a service or an entire project, usually rely in the bad management of those factors related to its life cycle (INCOSE Book 2015). Since cyber-physical systems (CPS) (Danda, Rawat, and Rodrigues 2015) become more and more complex, the requirements to be fulfilled, both in terms of functionality and performance, are of primary interest. As a consequence, there is need to include them among the operational constraints from the beginning of the design stage. Especially, in application domains such as power plants, medical appliances, aerospace, and automotive, some non-functional requirements (such as reliability, availability, maintainability, safety and security) have to be guaranteed and comply to standard specifications and regulations (Lahtinen, Johansson, Ranta, Harju, and Nevalainen 2010; Rierson 2013; Furfaro, Garro, and Tundis 2014; Furfaro, Gallo, Garro, Sacca, and Tundis 2016). Indeed, the violation of some requirements can generate the failure of a project whose impact can be measured in terms of: (i) economic and temporal; (ii) motivational; (iii) individual and organizational stress; (iv) the destruction of value and corporate reputation, and even worse as (v) loss of human lives.

Unfortunately, because of the high heterogeneity of CPS in terms of system components and functionalities to be provided, the management and the manual checking of the requirements is a challenging task to be performed. Thus, maintaining the compliance between the requirements and the actual system becomes increasingly difficult and unproductive to be performed. So, there is the need, from one hand, to clearly define constraints and requirements, and from the other hand to be able to verify them, possibly before the realization of the system or even before an advanced stage of its development is reached. This, in turn, implies to address some important challenges ranging from (Garro and Tundis 2015; Seshia, Hu, Li, and Zhu 2016; Falcone, Garro, and Tundis 2014) (i) identification of concepts and notations for modeling requirements; (ii) approaches for integrating design and requirements; (iii) automatic mechanisms that provide indications on the level of fulfillment of requirements during the system development.

In this panorama, the use of engineering tools in terms of innovative methods and techniques represent a profitably solution. Particularly interesting is the Design by Contract (DbC) method, typical of software engineering (Ozkaya and Kloukinas 2013). According to the DbC, the involved entities in the design have obligations towards other entities on the basis of well-formalized rules. A functional specification, called *contract* is created for each software module before it is implemented. Finally, the overall program execution is seen as the interaction among the various modules bounded according to these contracts (Klaeren, Pulvermüller, Rashid, and Speck 2001).

In this paper, the main purpose is to exploit the DbC method (a) as a contracts–based validation tool to support the development process of cyber-physical systems by simulation; and (b) to reuse the results gathered from the simulation as *contract* to verify the actual system operation after its deployment. To this aim, a possible solution is represented by Properties Modeling (PM) approach (Otter et. Al 2015; Nguyen 2014; MODRIO 2016) for modeling *contracts*, combined with the use of Simulation techniques to automatically verify them. Specifically, PM is recently considered among the major research fields of SE (INCOSE 2017). It deals with system requirements and constraints from the early stages of the system development process, passing through its release, up to its operational maintenance (Garro and Tundis 2015; Nguyen 2014; Rubio-Medrano, Ahn, and Sohr 2013). The combination of system properties along with simulation techniques allow to support and evaluate the goodness of the system through the virtual verification of the requirements. In this case, the modeling of *requirements* in terms of *properties* represent the *contract* among stakeholders and designers. The results gathered from the simulation represent "formalized requirements", that the actual system have to comply.

The rest of the paper is organized as in the following. Section 2 provides related work and motivation; Section 3 describe the proposal and how to combine the Properties Modeling and Simulation to enable the Design by Contract method for supporting the cyber physical system development. A case study along with a Simulator is described in Section 4, whereas a discussion is reported in Section 5. Conclusions are drawn in Section 6.

2. RELATED WORK AND MOTIVATIONS

2.1. Background on Design by Contract

Design by Contract is a typical methodology for supporting software design and development (Meyer 1992). Usually, designers and stakeholders collaborate to define the software specifications in terms of abstract data, routines and functions clearly and testable. Generally, the contract is defined through a set of rules called *pre-conditions* and *post-conditions*. In particular, *pre-conditions* state what a data, a function or method has to satisfies before it is used (e.g. the expected input); whereas the *post-conditions* state the status that the output (the data or the routine) will be comply after it has been used. For example, a contract takes the following general form:

```
if pre-conditions (A) == true
    then post-conditions (routine(A)) == true;
```

So, if the caller satisfies the initial conditions then a "correct" output is guaranteed, but if the initial conditions are violated then the correctness of the output cannot be guaranteed. So using a composite approach, the correctness of each software module can rely, in theory, on the correctness of the software modules that are used, as long as their preconditions are satisfied (Jazequel and Meyer 1997).

A lot of interest is shown towards this approach as evidenced by the different research efforts already available in literature. As an example, in many application domains where the software engineering development process is involved, the DbC is strictly related to the programming language. A popular example is represented by the DbC in Java language (Zimmerman and Kiniry 2009), where introduction of specific keywords, based on the concept of *assertion*, are introduced into the language. The main advantage in this case is that the developer and the designer share the same notation, so the level of ambiguity and misunderstanding is reduced. From the other side, the code to build the software and the one to check its correctness results mixed and highly coupled. As a consequence, software readability and maintainability become harder.

Another contribution is represented by the DbC with JML (Leavens, Baker, and Clyde 2006; Leavens and Y. Cheon 2013), a Behavioral Interface Specification Language (BISL). Besides pre- and post-conditions, it also allows assertions to be intermixed with Java code, and it is designed to be used by working software engineers. However, whereas Java expressions lack some expressiveness that makes more specialized assertion languages convenient for writing behavioral specifications, JML solves this problem by extending Java's expressions with various specification constructs, such as quantifiers.

Another approach to enable DbC is based on a Temporal Logics (Cimatti and Tonetta 2016). An implementation of it is available in OCRA (Cimatti, Dorigatti, and Tonetta 2013) that is very suitable to represent temporal relationships among events. It provides a support to contract-based design, ranging from the formal specification of the architecture and contracts to the automatic analysis of the refinement, implementations, and safety of the contract specification (Nuzzo, Sangiovanni-Vincentelli, Bresolin, Geretti, and Villa 2015).

In (Heckel and Lohmann 2005) is provided another contribution regarding the employment of the DbC method, by representing contracts in terms of graph transformation rules, for testing web services against their description. In (Seshia, Hu, Li, and Zhu 2016) is discussed an important research effort based on optimization techniques; whereas a more recent discussion is provided in (Murthy 2016; Ozkaya and Kloukinas 2013), where the authors highlight possible perspectives and argue the exploitation of the DbC as method to increase the dependability, in terms of correctness and robustness, of a software by support different phases in its development process.

2.2. Main Objectives

As explained, the DbC is a very powerful and wide adopted method in Software Engineering. Indeed, all the approaches, languages and methods mentioned in the previous Section are meant to be employed for supporting the development process and verification of software.

However, particular interest is shown nowadays for using the DbC also in CPS domain due to its own intrinsic nature (Sreram, Buonopane, Srinivasan, Subathra, and Ayyagari 2015; Derler, Lee, and Törngren 2013). Indeed, according to NIST perspective (NIST 2017), Cyber-Physical Systems are co-engineered interacting networks of physical and computational components (such as Smart Grids, Internet of Things, Smart Cities etc.). These systems represent the basic of critical infrastructures, for building smart services, and improve the quality of the society. Furthermore, most of their part are physical components (e.g. Electrical, Mechanical and Hydraulic components), whose dynamics in terms of inputs, outputs and evolution is well-know. This means that:

- their behavior can be represented without ambiguity through algorithms, functions and mathematical expressions;
- their output values, in terms of data and physical flows, can be combined and evaluated quantitatively.

As a consequence, these characteristics can be employed to define formal and computable *contracts*, on which the parties can agree, before the actual realization of the system.

In fact, a preliminary evaluation and approval of its functionalities through virtual tool, represents an important step not only to identify gap in design but also to distribute "responsibility" as well as to clarify the objectives by reducing ambiguities due to the natural language and different background among the involved parties (stakeholders, designers and developers). Specifically, the use of simulation allows to support domain experts and engineers during the design of the CPS, as a testing tool, and for the evaluation of design choices in terms of constraints violation, according to the stakeholders' expectation, agreements and regulations. The output of the overall process represents the *contract* between stakeholders and system engineers, that is the basis on which the actual system has to be build. To this aim, next Section illustrates how to exploit Properties Modeling combined with Simulation as enabling approach.

3. ENABLING DESIGN BY CONTRACT THROUGH PROPERTIES MODELING

This Section describes how to extend the concept of DbC as method to support the development process of cyber-physical systems. In particular, it wants to be used as a preliminary assessment tool, based on the definition, evaluation and virtual validation of requirements, by defining and suitably calibrating system parameters before the actual implementation of the system. In fact, in this context, a model-based and simulation-driven verification approach can be adopted not only to model requirements in a formal way but also to compute them in order to discover emergent system behavior that is not typically identifiable through the classic requirements analysis.

3.1. Combining the Properties Modeling approach and Simulation Techniques

The Design by Contract method for CPS is based on the combination of the Properties Modeling (PM) approach and Simulations Techniques (ST). Specifically, the aim of the PM approach is not only to allow in a more formal way the requirement's representation, but also to enable their monitoring. From the other side ST and related tools enable not only to run and emulate the system under consideration but also to trace every requirement and to be notified *where*, *how* and *when* one of them (requirements) is violated.

The Design by Contract of a CPS can be defined as *DbC (CPS)=<C, Rc, S, P, Rp, Rpc>,* where:

- *C* represents a set of Components*;*
- *Rc* is a set or Relationships among the Components C.
- By using *C* and *Rc* a System Design model *D* of the Cyber Physical System can be defined.
- *S* represents a set verification Scenario, that is, the flow of actions that can be triggered manually or timed to be carried out, in order to stress and stimulate the system;
- *P* is a set of (System) Properties that are used to validate the design *D* against the scenario *S*;
- *Rp* is a set or Relationships among the Properties *P*.
- *Rcp* is a set or Relationships among the Components *C* and Properties *P*.

As a consequence, the *contract* on a *CPS* is accepted by the parties when *DbC(CPS)=true*, that is, when none of the *P* properties is violated against the scenario S.

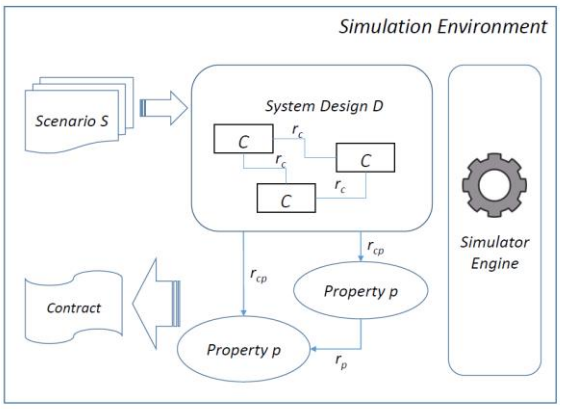


Figure 1: Simulation Model

Figure 1 shows the proposed *Simulation Model (SM)* where the involved *C components, P properties, Rc, Rp* and *Rpc relationships* and *S scenarios* are used for modeling the overall system. This is, in turn, wrapped into a *Simulation Environment* for enabling the simulation-based design validation of the system by properties evaluation against inputs defined in terms of verification scenarios.

In particular, beside the *System Design* model, system requirements are formalized as computable components in terms of *System Properties* that can be verified. It is worth noticing that the *properties* are fed only with the values coming from the *System Design* model, without interfere with the behavior of the system; whereas the *Scenario* is exploited as input to the system and a consequence as a verification scenario to be checked. The output of the simulation represents the *contract*, that is generated from the automatic execution and evaluation of the *system properties*, through the virtual environment. The next Section describes how the process takes place step by step.

3.2. Reference Methodology

In this Section, a proposed methodology for supporting the Design by Contract method in cyber-physical systems domain is described. It is based on the involvement of different types of Actors, as represented in Figure 2 and described in Table 1, who are exploited throughout the overall process: *Stakeholder, Property Designer (Designer), System Designer* (*Designer*) and *System Developer*.

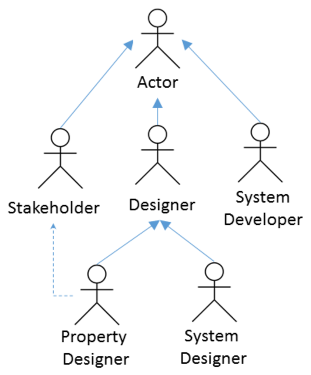


Figure 2: Hierarchy of the involved actors

The proposed methodology is centered on a 6-phases process: *Requirement Specification, Properties Modeling, System Modeling, Virtual Design Modeling, Results Assessment, System Realization and Deployment*, as shown in Figure 3.

Specifically, in the *Requirement Specification* phase, the *Stakeholders* provide all the necessary details and describe textually the system according to their needs and expectations, in terms of functionalities, performances and expected results. The output produced at the end of this phase are two work-products*:* textual *System Requirements* and the *System Conditions & Constrains.*

Table 1: Involved Actors and related description

Involved Actor	Description
Stakeholde r	Domain expert who has interest in building the system under consideration. Typically, with limited technical knowledge. Stakeholder express they requirements and constrains in their own language. Different stakeholders can express conflicting requirements.
Property Designer (Designer)	It is a Designer. He has both: (i) enough expertise in the stakeholder domain, in the context of the system under development; and (ii) technical competences about the system development environ-ment. He is the responsible to translate and provide a formal definition of requirements from the textual version provided by the stakeholder to computable component expressed in terms of System Properties.
System Designer (Designer)	It is a Designer. He is an actor with high level multi-disciplinary competences (e.g. electrical, informatics, hydraulic, etc.). He is in charge to define and design the overall system in terms of components their relationships, functions and input/output values.
System Developer	He is the technical expert with low level of knowledge about the domain. Based on the system design he is in charge the development and/or the integration of system components and system property in the simulation environment, in order to analyze the system behavior and get results from the Simulation.

Starting from such outputs, two phases can get started and proceed in parallel. In particular, the *Properties Modeling* phase takes in input only the *System Requirements*. It is performed by the *Property Designers*, who have similar competencies of the *Stakeholders* and who are able to interpreter correctly the *System Requirements* produced by the *Stakeholders*. In this phase the *System Requirements* are transformed into formalized and computable components (*Properties*) by using mathematical notation, algorithms or functions. The work-products produced in this phase are represented by a set of *System Properties* that able to (i) read and elaborate values originated from the system under consideration and (ii) provide in output a quantitative evaluation according to the requirements.

In parallel can take place the *System Modeling* phase, which is performed from different actors called *System Designers*, who, by using both *System Requirements* and *System Conditions & Constrains*, are able to (i) define the *System Design* in terms of system components, internal behavior and functionalities provided by each component as well as the interaction among them in order to build the overall system; (ii) derive the verification *Scenario* to be used as input for the system.

It is worth noticing that these two phases are completely decoupled and they do not have to interfere to each other, as well as the *Properties Designers* and the *System Designers* are not supposed to communicate and influence each other.

Starting from the outputs produced from the previous phases, during the *Virtual Design Modeling* phase, technical actors called *System Developers* are in charge to build virtually the system simulation environment. In particular, in this phase the *System Design* and the *Properties* are mapped in order to enable their validation against the defined *Scenario*. The output of this phase is represented by an integrated design model. By executing the simulation, different results are automatically generated against the input *Scenario*.

Such results are used and analyzed in the *Results Assessment* phase by the *Stakeholders*, who can approve them completely or partially by sending feedback backward to the *System Designers* in the *System Modeling* phase, based on the output of the *Properties* that codify the requirements. Here, the *System Designer* can use such feedback to improve the *System Design* in order to meet the requirements encoded into the *Properties* that will be again evaluated. The process iterates as long as all the *Properties* are not violated. This means that all the requirements specified by the *Stakeholders* are fulfilled and as a consequence the *System Design* is validated. Once the *Result Assessment* phase ends the *contract* takes place by relying on the fulfillment of the *Properties*, then the last phase can start.

At this point, the *System Realization* and *Deployment* phase of the real CPS can be done. In this phase real system is built, according to the specification clearly identified, and specified preliminary in terms of *Properties* and whose expected outputs have been already analyzed through the Simulation.

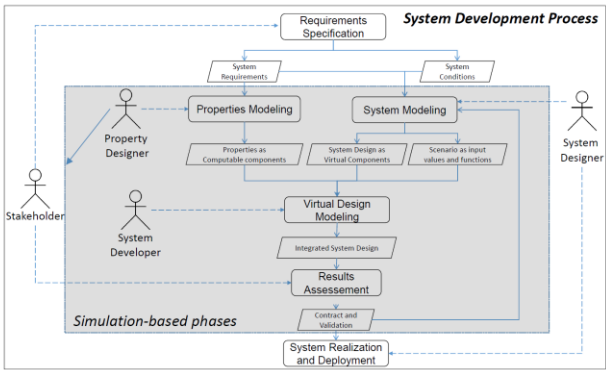


Figure 3: Reference Process and Work-products

In the next Section the proposal is exemplified through a case study in Smart Grid.

4. EXPERIMENTING THE DESIGN BY CONTRACT IN SMART GRID DOMAIN

In this Section, a case study in Smart Grid environment, that illustrates the exploitation of the Design by Contract approach, driven by Simulation and based on Properties Modeling, is presented. In particular, after a brief overview on Smart Grids, the Property Modelling approach is adopted for representing computable requirements; whereas, a Smart Grid Simulator, that implements the properties model, has been ad-hoc developed for supporting their validation through simulation.

4.1. Smart Grid description and Main Objectives

The CPS under consideration consists of a Smart Grid (SG) (Karnouskos 2011), a modern electric power grid infrastructure. It relies on smooth integration of renewable and alternative energy sources through automated control and modern IT technologies for improving its management. A SG integrates both a cyber-part that encompasses computing and networking resources, and a physical part consisting of physical processes such as mechanical and electrical.

Some of the SG components are *Energy Providers (EPs)*, such as Power Plants, Wind Turbines, Solar Panels, whereas other are *Energy Consumers (ECs)* such as Houses, Cars, Hospitals and so on. In particular, *EPs* are responsible to provide the required amount of energy, according to specific agreements, in order to satisfy the needs of the *ECs*, as efficient as possible.

Furthermore, a SG has to be able to (i) find an optimal balance of energy production for a dynamic demand, (ii) collect data from devices within the grid to manage and discover information, (iii) organize either small micro grids or continental-scale grids, and (iv) integrate heterogeneous devices ranging from big transformers and power plants to smart household appliances (Seo, Lee, and Perrig 2011).

4.2. Requirements Specification

This case study has been defined by cooperating with domain experts in the context of a German research project called PolyEnergyNet (PEN 2017), whose main objective is to support the increase of resilience in Smart Grid environments.

It is worth noting that, there are different types of requirements. Some of them are perceived by the end user in terms of provided services and functionalities; whereas, other requirements are more transparent to users to whom the services are provided.

Nevertheless, they are essential to guarantee a certain level quality and performance. In this phase, the specifications provided by the *Stakeholders* are analyzed and the textual *Requirements* are extracted. Table 2 reports 9 identified requirements, some of which comply the National Electrical Manufacturers Association (NEMA 2017) specifications.

According to the proposed method, some of the above mentioned requirements, are used by the *System Designers* in order to define the *System Design* in terms of Smart Grid structure (e.g. component involved in the design, connections, input and output) and behavior (e.g. actions, process, and events intra- and inter-components) as well as the verification *Scenario*, as described in Section 4.3. Other requirements, instead, are exploited by the *Property Designers* who are in charge to identify and define SG *System Properties*.

Table 2: Smart Grid System Requirements

Requirement	Description
Requirement_01	During operation, the SG has to ensure the functioning of a minimum number of energy producers (Solar Panels, Wind Turbines, etc.).
Requirement_02	The SG has to ensure, for each typology of involved component, that the sum of their output (i.e. the sum of the energy produced) do not drop below a certain threshold.
Requirement_03	When an energy producer is turned on (or turned off), it has to reach its maximum production level (or stop producing) within a certain time (otherwise there could be the risk of burn some SG component and, in general, affect the operation of the overall system).
Requirement_04	The output produced by each energy producer can fluctuate within specific and limited boundaries. For example, the temperature t or the pressure p must fluctuate/oscillate within a certain range.
Requirement_05	After an energy producer is activated in the grid, its operation (and as a consequence its service) has to be guaranteed for at least a specific period of time (e.g. hours, days, years).
Requirement_06	In case of the disservice of the SG, the time of undersupply of each energy consumer must not exceed a certain time interval.
Requirement_07	The total duration of undersupply for each specific consumer, must not exceed a certain limit of annual hours.
Requirement_08	The Smart Grid has to include at least one Power Plant and one House.
Requirement_09	The Smart Grid could include Solar Panels, Wind Turbines, Hospital, and Vehicles.

4.3. Design Modeling

As mentioned in the previous Section, in this phase, the *System Designers* extract the requirements that they need for defining both the structure of the SG as well as for modeling its functional- and non-functional behavior in terms of (i) type of components, (ii) components cardinality, (iii) connections and interactions (iv) constrains such as level of priority of being supplied with energy, and so on.

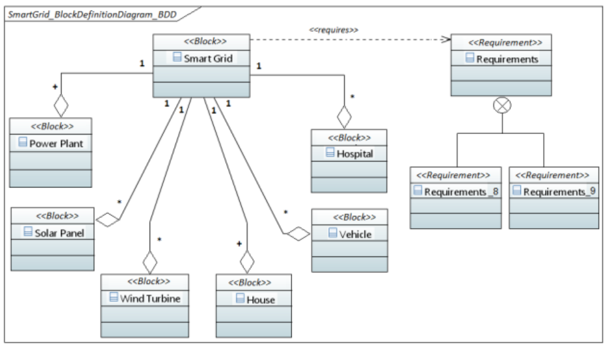


Figure 4: Block Definition Diagram of the main Smart Grid components

As an example, Figure 4 shows a possible high level structural of a Smart Grid, derived from *Requirement_8* and *Requirement_9* that provides information about the involved type Energy Producers and Energy Consumers and their cardinality. The SG components are represented through SysML Block by using a SysML Bock Definition Diagram (BDD) (OMG 2017) in the left side, whereas the related requirements are reported in the right side.

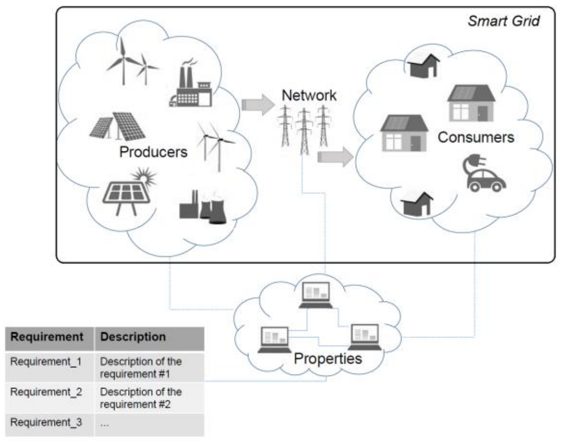
It is worth notice that, typically, requirements are defined textually without verifiable relationships. Indeed, it is not trivial to establish verifiable relationships in order to proof the fulfillment of textual requirements on the basis of the behavior of system components.

Moreover, specific verification *Scenario* are also identified in this phase. As described before, a *Scenario* model catches a specific sequence of actions that are used to stimulate the *System Design* model in order to induce a particular reaction. So each verification scenario is defined, based on requirements, with the purpose to test the *System Design*. In the following, some of the *Scenarios*, defined starting from the *Requirements Specifications* are reported:

- *Scenario_1*: decreasing or increasing of the level of energy of a Power Plant under or over the allowed thresholds;
- *Scenario_2*: switching-off time a Power Plant. To verify the time that a Power Plant takes to move from the Max level of energy production to 0;
- *Scenario_3*: switching-on time of a Power Plant. To verify the time that a Power Plant takes to move from 0 to the Max Level of energy production.

4.4. Properties Modeling

According to the proposed methodology, this phase aims to define the P*roperties*. Specifically, the *Property Designers* use specific requirements, or extract specific information from them, to define a middle layer of computable *properties*, that allows to bind the *System Design* with the *Requirements*. A graphical representation of the binding between the design of the Smart Grid under consideration and the *Requirements* through *System Properties* is shown in Figure 5.



Requirement	Description
Requirement_1	Description of the requirement #1
Requirement_2	Description of the requirement #2
Requirement_3	...

Figure 5: Binding Smart Grid with Requirements through System Properties

Example of *properties*, identified starting from the *requirements* specified in Section 4.2 and defined by using a mathematical notation, are the following:

- *Property_1 (*derived from *Requirement_1)*: It is related to the *Components Cardinality* of a specific type of Component C. In particular, given a set of *n* components of type C $\{c_1,...,c_m,\ldots,c_i,\ldots,c_n\}$. Given the function $W(c_i)=1$ if the component ci is working and $W(c_i)=0$ if ci is not working. Then the property states that at least *m* out *n* components must be working as specified by the equation (1).

$$\sum W(c_i) \geq m \qquad i=1\ldots n \qquad (1)$$

- *Property_2 (*derived from *Requirement_2)*: It is related to the *Cumulative Value* CV_C provided by a specific set of Energy Producer component of type C.
It states that given a set of *n* producer components of type C $\{c_1, c_2,\ldots,c_i,...,c_n\}$ and given the function $P(c_i, t)$ that provides information on the current amount of energy provieded from c_i at the time *t*, then the total amount of the energy produced from the components CV_C has to be, at any time, at least more than the minimum value P_{min} specified according to the equation (2).

$$CV_C = P(C) = \sum P(c_i, t) \geq P_{min} \qquad i=1\ldots n, \forall\, t \in T \qquad (2)$$

- *Property_3 (*derived from *Requirement_3)*: It is related to the *Variation Time* of energy production of Energy Producer component.
Given an initial value of energy $P_{Initial,}$ that is roduced from a component *c* at time t_0, $P(c,t_0)= P_{Initial}$. Then the final value of energy to be produced from c, at time t_f, $P(c,t_f)= P_{Final}$ has to be reached within a specific time Δt, $|\, t_f - t_0| \leq \Delta t$. As a consequence, the variation of energy $\Delta P(c)$ produced by a component c must be reached within Δt according to equation (3).

$$|P(c,t_0)- P(c,t_f)| = \Delta P(c), \qquad \text{with } |t_f - t_0| \leq \Delta t \qquad (3)$$

- *Property_4 (*derived from *Requirement_4)*: It is related to *Threshold Value*. It state that given *k* Energy Producer component C $\{c_1, \ldots c_j,\ldots,c_k\}$. The value of energy $P(c_j)$ produced in output from each producer c_j has to be kept between an Upper Bound (V_{max}) and a Lower Bound (V_{min}), according to equation (4).

$$V_{min} \leq P(c_j) \leq V_{max} \quad \forall c_j \in C \ j=1\ldots k \qquad (4)$$

- *Property_5 (*derived from *Requirement_5)*: It is related to the *Service Life Time* of a component. The property state that if a component *c* is ON at time t_0, the service S provided must be hold for at least a *period* of time t_{period}. So no variation of the service has to occur in t_{period} according to equation (5).

$$|S_{ON}(c, t_{i+1}) - S_{ON}(c, t_i)| = 0, \forall\, t_i, \ t_{period} > t_i > t_0 \qquad (5)$$

- *Property_6 (*derived from *Requirement_6)*: It is related to *Grid Resilience*. If a Energy Consumer component c is undersupplied at time t_0, the system must be able to recover within a time Δt. Given a function $U(c,t_i)$ that is equal to 1 when a consumer *c* is undersupplied at time t_i, and $U(c, t_i)$ is equal to 0 when *c* is supplied. Then the duration of the disservice has to be no longer than Δt according to (6).

$$\forall\, U(c,t_i) = 1 \rightarrow t_{i+1}- t_i \leq \Delta t \qquad \forall t_i \in T \ i= 0..k \qquad (6)$$

Starting from the above produced work-products, defined in terms of models (e.g. *System Design*, *Properties* and *Scenarios*), the *Simulation Environment* can be defined.

4.5. Virtual Design Modeling

In this phase the Smart Grid is virtually represented by the *System Developers*, by modeling first the SG components and connections according to the design provided by the *System Designer*, and then by integrating the *Properties model* provided by the *Properties Designers*. An example of smart grid configuration is shown in Figure 6 through the user interface of the Smart Grid Simulator (SGS) that has been implemented according to the defined models (System Design, Properties and Scenarios).

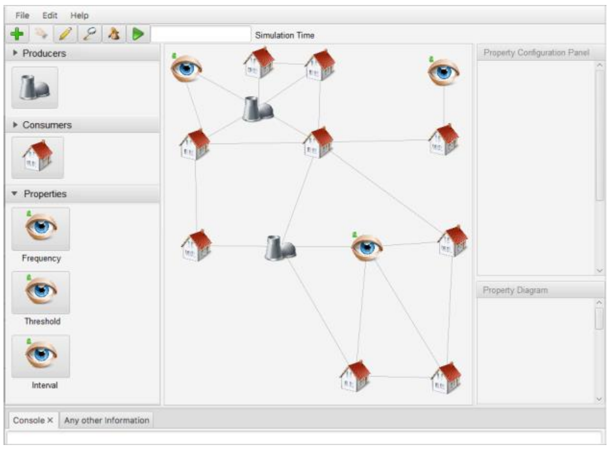


Figure 6: Smart Grid design extended with Properties

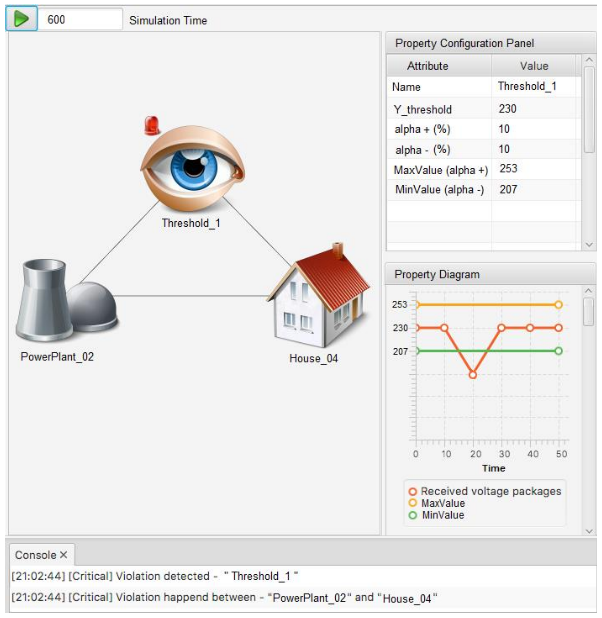
In the left side of the Graphic User Interface (GUI), different kind of components to represent SG configuration are provided and grouped as *Producers* (e.g. Power Plant, Solar Panel and Wind Turbine) and *Consumers* (e.g. House, Hospital, and so on). Each component can be configured to send and receive physical data flows as well as digital signals. Their configuration and the way to interact are enabled via data lines to form a network, that determine the actual behavior of the overall grid.

Furthermore, the *Property model* is integrated into the *Simulation Environement* by implementing the *Properties* described in Section 4.4 as additional network components for validating the SG configuration.

As we can see, in the end, the SG *System Design is* extended by introducing verifiable *requirements* defined in terms of *properties* that create the matching between textual requirements and the SG System Design. Once the overall configuration is defined, it is possible the execute the simulation in order to perform different experiments. This allows to evaluate quantitatively the fulfillment of the requirements on the basis of the behavior of the smart grid.

4.6. Results Assessment

One of the experiment has been conducted by validating the SG System Design against the *Scenario_1*, introduced in Section 4.3. To this aim the *Property_4*, called *Threshold Value*, has been employed. By using the zooming-in feature of the SGS simulator, a closer view on each specific property can be provided. As example, Figure 7 shows a *Threshold Value* property defined on two components: a Power Plant as Energy Producer and a House as Energy Consumer.



Attribute	Value
Name	Threshold_1
Y_threshold	230
alpha + (%)	10
alpha - (%)	10
MaxValue (alpha +)	253
MinValue (alpha -)	207

```
[21:02:44] [Critical] Violation detected -  " Threshold_1 "
[21:02:44] [Critical] Violation happend between - "PowerPlant_02" and "House_04"
```

Figure 7: Threshold Value Property

The Property Configuration Panel shows how the *Threshold Property* is configured. In particular, the Y_threshold = 230V represents the desired value of Voltage to be transmitted, whereas the deviation parameters ”alpha +” and ”alpha -” are configured with the same percentage ± 10%. These initial values generate automatically the MinValue (V_{min}) and MaxValue (V_{max}). The property is fulfilled if, for the overall transmission time, the amount of energy is between MinValue = 207V and MaxValue = 253V. As shown from the Property Diagram in Figure 7, a violation of the property is detected at the simulation time around t=200, because the amount of transmitted Voltage is less than the required MinValue. The property is then highlighted in red color, whereas the components responsible of violation are printed out in the console. This implies the existence of anomaly in the SG System Design and in particular because the Energy Producer behavior do not respect the required energy that the Energy Consumer needs receive.

This means that the implementation of some functionalities of the SG do not comply the requirements, as a consequence the contract is not respected. As a consequence, as long as exists a property violated, at least a requirement is not fulfilled. This implies that it is necessary to re-iterate the modeling process of the Smart Grid in order to improve its System Design.

Further information is shown in Table 3 which reports the properties used in the verification *Scenario_1* and their related results. In particular, some properties are Not Violated, whereas others are Violated. Moreover, Table 3 shows, also, when a property was Violated by reporting its Violation Timestamps, and duration of each violation (Violation Length).

Table 3: Simulation Results based on Properties as Contracts

Property	Property Status	Violation Timestamps	Violation Length
Threshold_1	*Violated*	*VT_[20]*	*VD_[1]*
Grid Resilience_1	*Not Violated*	*VT_[:]*	*VD_[:]*
Threshold_2	*Violated*	*VT_[3,7,26]*	*VD_[4, 2, 5]*
…	…	…	…

When all properties are not violated that means that all the contracts are fulfilled and the process can move towards the *System Realization and Deployment* phase.

According to the proposed methodology, the inputs of this phase are not anymore represented by only textual requirements, but they are also accompanied by a well-defined model of the Smart Grid that represents how the System Design should look like in reality and how services have to work; whereas the *System Properties* can be reused as monitoring tool that observe the SG during its operation and to compare the deviations among real values against the simulated ones.

So, the specifications formalized in terms of software (Design and Properties) and the simulation results represent quantitative indicators, to be used as *contracts*, for evaluating the level of fulfillment of requirements starting from the early stages of the development process of a CPS.

5. DISCUSSION AND ADVANTAGES

As above described, the proposal extends the traditional system development process by introducing virtual modeling and evaluation phases based on simulation where requirements become computable system entities. This allows to obtain quantitative evaluations of the system design by creating formal matching based on variables as well effective automatic way to trace and verify the fulfillment of requirements using *Properties*. Moreover, thanks to the decoupled level of development between the *System Design* and the *System Properties* the fulfillment of requirements through properties is not biased.

Furthermore, from the point of view of the actors involved, such approach is beneficial both for the *System Designers* and for the *Stakeholders*. In fact, *System Designers* know concretely what to develop. They know not only the inputs and outputs of the system, but also their relationships and how the inputs need to be combined and processed in order to obtain the outputs according to the *Stakeholder* expectations; *System Designers* also have a way to prove that the real system behaves as the virtual one, on which the agreement (the *contract*) was established. On the other hand, the approach is also beneficial for the *Stakeholders* who have the certainty that the requirements have been correctly understood. In addition, this allows *Stakeholders* to claim the development of capabilities without ambiguity, but rather with "evidence", in case of the real system behaves differently from the virtual one thanks to the virtual model and related gathered results, on which the agreement (the *contract*) was established.

Finally, once the system is realized and deployed, properties may be used, as a control tool, for observing its real behavior during its operation in order to detect anomalies and generate notifications and alerts.

6. CONCLUSION

The paper has proposed a Systems Engineering approach for enabling the Design by Contract method in cyber-physical systems domain. It is focused on the combination of a Properties Modeling approach and Simulation techniques, in which the former is used for enabling in a more formal way the representation of system requirements and constrains, whereas the latter is exploited as tool to objectively check their correctness.

The approach is based on a methodology centered on 6 phases (*Requirement Specification, Properties Modeling, System Modeling, Virtual Design Modeling, Results Assessment, System Realization and Deployment*). It aims at supporting the process for the definition of *contracts* by formalizing the requirements in terms of *System Properties* which in turn are exploited to validate the System Design before actual realization of a system.

A first experimentation has been conducted on a case study in the field of Smart Grid. In particular, a Smart Grid Simulator has been developed to enable the automatic verification of system requirements. The simulator implements not only the main SG components and their related behavior but also it allows to include computable requirements defined as system properties among the operational constraints as well as automatically evaluate them against specific verification scenarios.

Based on the work-products(models) gathered from the overall process and from the simulation results, the approach allows *Stakeholders* and *Engineers* to obtain a preliminary evaluation and agree on specific *contracts* that rely on the validation of the requirements driven by simulation.

Ongoing works are devoted to (i) enrich the approach by defining and integrating a specific Attack Scenario Models based on a Faults and Failures Generator Model able to support the injection of external attacks and faults in the Smart Grid, and (ii) experiment the approach in other application domains.

ACKNOWLEDGMENTS

The work in this paper was partially performed in the context of the PolyEnergyNet (PEN) research project and partially funded by the Germany Federal Ministry for Economic Affairs and Energy (BMWi) under grant no. "0325737E".

REFERENCES

INCOSE Book, 2015. Systems Engineering Handbook A Guide for Systems Life Cycle Processes and Activities, 4th Edition, Wiley.

Jazequel J. M. and Meyer B., 1997. Design by contract: the lessons of Ari-ane, in Computer, vol. 30(1), January, pp. 129-130.

Meyer, 1992. Applying design by contract, in Computer, vol. 25(10), pp. 40-51.

Garro A. and Tundis A., 2015. Modeling of system properties: Research challenges and promising solutions, Proceedings of the first IEEE International Symposium on Systems Engineering (ISSE), Rome, Italy, pp. 324-331.

Zimmerman D. M. and Kiniry J. R., 2009. "A Verification-Centric Software Development Process for Java," Proceedings of the Ninth International Conference on Quality Software, pp. 76-85.

Leavens T., Baker L., and Clyde Ruby. 2006. Preliminary design of JML: a behavioral interface specification language for java. SIGSOFT Software. Engineering. 1-38.

NIST (National Institute of Standards and Technology) Cyber Physical Systems, 2017 - available online at https://www.nist.gov/el/cyber-physical-systems.

Leavens G. and Y. Cheon, 2003. Design by contract with JML.

Cimatti A. and Tonetta S., 2016. A Temporal Logics Approach to Contract-Based Design. In: Architecture-Centric Virtual Integration (ACVI), Venice, pp. 1-3.

Cimatti A., Dorigatti M., and Tonetta S., 2013. OCRA: A Tool for Checking the Refinement of Temporal Contracts, in ASE, pp. 702–705.

Nuzzo P., Sangiovanni-Vincentelli A., Bresolin D., Geretti L., and Villa T., 2015. A Platform-Based Design Methodology With Contracts and Related Tools for the Design of Cyber-Physical Systems, Proceedings of the IEEE, vol. 103(11), pp. 2104–2132.

Heckel R. and Lohmann M., 2005. Towards Contract-based Testing of Web Services, In Electronic Notes in Theoretical Computer Science, vol. 116, 19, pp. 145-156.

Murthy PVR., 2016. Reliability by Construction using Design by Contract Methodology. In Proceedings of the 9th ACM India Software Engineering Conference (ISEC '16), New York, USA.

NEMA (National Electrical Manufacturers Association) 2017. [Online]. Available: https://www.nema.org/.

PEN (PolyEnergyNet) 2017. [Online]. Available: http://www.polyenergynet.de/.

OMG (System Modeling Language), 2017. [Online]. Available: http://www.omg.org/spec/SysML/.

Otter M., Thuy N., Bouskela D., Buffoni L., Elmqvist H., Fritzson P., Garro A., Jardin A., Olsson H., Payelleville M., Schamai W., Thomas E., and Tundis A., 2015. Formal Requirements Modeling for Simulation-Based Verification, Proceedings of the 11th International Modelica Conference, Versailles, France.

Nguyen T., 2014. FORM-L: A MODELICA Extension for Properties Modelling Illustrated on a Practical Example, Proceedings of the 10th International Modelica Conference, Lund, Sweden.

MODRIO (ITEA 3 Model Driven Physical Systems Operation) Project, 2016. Available: https://itea3.org/project/modrio.html.

Rubio-Medrano C. E., Ahn G., and Sohr K, 2013. Verifying Access Control Properties with Design by Contract: Framework and Lessons Learned, Proceedings of the IEEE 37th Annual Computer Software and Applications Conference, Kyoto,Japan, July 22-26.

Ozkaya M. and Kloukinas C., 2013. Towards Design-by-Contract Based Software Architecture Design, Proceedings of the 12th IEEE International Conference on Intelligent Software Methodologies, Tools and Techniques. Budapest (Hungary) September 22-24.

Lahtinen J., Johansson M., Ranta J., Harju H. and Nevalainen R., 2010. Comparison between IEC 60880 and IEC 61508 for certification purposes in the nuclear domain, Proceedings of the 29th International Conference on Computer Safety, Reliability and Security. Vienna, Austria, September 14-17.

Rierson L., 2013. Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance.

Sreram B., Buonopane F., Srinivasan S., Subathra B., and Ayyagari R. 2015. Verification of Design Contracts for Cyber-Physical System Design Using Evolutionary Optimization. Proceedings of the IEEE International Conference on Circuit, Power and Computing Technologies.

Seshia S.A., Hu S., Li W., Zhu Q., 2016. Design Automation of Cyber-Physical Systems: Challenges, Advances, and Opportunities, Transaction on Computer-Aided Design of Integrated Circuits and Systems.

Seo D., Lee H., and Perrig A., 2011. Secure and Efficient Capability-Based Power Management in the Smart Grid, Proceedings in the Parallel and Distributed Processing with Applications Workshops (ISPAW), pp. 119–126.

Karnouskos S., 2011. Cyber-physical systems in the SmartGrid, Proceedings of the IEEE International Conference on Industrial Informatics (INDIN), pp. 20–23.

Danda I.S., Rawat B. and Rodrigues Joel J.P.C., 2015. Cyber-Physical Systems: From Theory to Practice.

Derler P., Lee E.A., and Törngren M., 2013. Cyber-Physical System Design Contracts, Proceedings of the International Conference of Cyber Physical System, Philadelphia, PA, USA, April 8-11.

INCOSE (International Council on Systems Engineering), 2017 - [Online]. Available: http://www.incose.org/.

Klaeren H., Pulvermüller E., Rashid A. and Speck A. 2001. Aspect Composition Applying the Design by Contract Principle, In: Butler G., Jarzabek S. (eds) Generative and Component-Based Software Engineering. LNCS, vol. 2177. Springer, Berlin, Heidelberg.

Furfaro A., Garro A., and Tundis, A. 2014. Towards Security as a Service (SecaaS): On the modeling of Security Services for Cloud Computing. Proceedings of the 48th Annual IEEE International Carnahan Conference on Security Technology (ICCST 2014), Rome, Italy, October 13 – 16.

Furfaro A., Gallo T., Garro A., Sacca D., and Tundis, A., 2016. Requirements specification of a cloud service for Cyber Security compliance analysis. Proceedings of the International Conference on Cloud Computing Technologies and Applications, (CloudTech 2016) pp. 205-212. doi: 10.1109/CloudTech.2016.7847700.

Falcone A., Garro A., Tundis A., 2014. Modeling and simulation for the performance evaluation of the on-board communication system of a metro train. Proc. Of the 13th International Conference on Modeling and Applied Simulation, (MAS 2014), Bordeaux, France, September 10 – 12, pp. 20-29.