Declarative Dependency Specification for Inter-connected Large-scale Cyber-physical Systems

Jing-Quan XIE¹,a, Thomas DOLL¹,a
¹Fraunhofer IAIS, Sankt Augustin, Germany
ajingquan.xie@iais.fraunhofer.de

*Corresponding author

Keywords: Cyber-physical Systems, Declarative Specification, Federated Simulation, Dependency Analysis, Complex Event Processing.

Abstract. Cyber-Physical Systems (CPS) are inter-connected software-hard ware artefacts ranging from small-scale bio-electrical implants, medium-scale smart cars and intelligent production lines, to large-scale critical infrastructures with power grids, telecommunication networks and railway systems. Large-scale CPS plays a central role in modern society by delivering essential services for human daily life. Depending on the delivered services, any disruption or mal-function of these systems with a duration of hours or days will have severe societal and economic impacts and consequences, in particularly when the inter-dependencies between these systems are considered: the total consequences resulted by cascading effects is usually much larger than the one caused by the original failure. Federated simulation provides an effective means to analyse the overall dynamics of inter-connected large-scale CPS. In a typical federated simulation environment, dedicated domain-specific simulators, both commercial and open-source, are applied and inter-connected together to provide realistic system behaviours of the complete CPS networks. Specifying the dependencies among different simulator models is however a challenging task, especially for networks involving multiple large-scale CPS. In this paper, a declarative approach for dependency specification based on data stream processing is proposed. Preliminary results show the benefits comparing to the traditionally imperative approaches in terms of flexibility, scalability and usability.

Introduction

The term Cyber-Physical Systems (CPS) was first coined by Helen Gill at NSF for systems combining computation, networking and physical processes in 2006 [6]. CPS covers a wide application areas ranging from small-scale bio-electrical implants to large-scale national Smart Grids [3]. Critical infrastructures (CI) like national or regional electric networks, telecommunication networks and railway systems are large-scale CPS with grave societal and economic impacts in modern civilisation. Any malfunctioning of these components will have substantial consequence to the society, especially if second-order and even third-order chaining effects during crisis are considered [14, 8]. Sophisticated dependencies exist between different large-scale CPS - either explicit or implicit. Identifying and analysing these dependencies can substantially improve the resilience of these systems in operation environment. Since the proposed approach focuses on large-scale CPS, in the following text of this paper, without explicit explanation, the term CPS refers to large-scale CPS.

This paper is structured as follows: Section 1 provides introductory material and background information of the proposed approach. It is followed by Section 2 with an elaboration of modelling and simulation based CPS analysis, focusing on the core problem to be addressed - dependency modelling. Section 3 describes the proposed approach for declarative dependency specification. A
concrete case study is presented in Section 4 to help the reader further understand the benefits of the proposed approach. Finally, related work and conclusion are given in Section 5 and Section 6 respectively.

Analysis of inter-connected CPS

Due to the significance of CPS in modern society, comprehensive analysis needs to be performed for CPS. To analyse inter-connected CPS that are widely deployed in modern society, federated simulation proves to be a feasible approach to accomplish this task [15]. One of the core problems to improve the resilience of these CPS networks is the identification of (inter-)dependencies between CPS, hence reduce cascading effect during crisis situations. These dependencies need to be identified, modelled and analysed in a systematic and effective way - ideally with high-level of re-usability.

Federated Simulation

Federated Simulation [15] provides a promising approach to dealing with sophisticated simulation involving multiple CPS. Conceptually, integrating these simulators in an appropriate way can provide realistic dynamics of CPS. Technically, most of these simulators provide Application Program Interfaces (API) so that other systems can control the simulator (start, stop, step, etc.) and retrieve the relevant information during the simulation running. In this paper an improved version of the DIESIS [17] federation middleware will be discussed. The major improvement is depicted in Figure 1, where the diagram on the left side denotes the original DIESIS federated simulation middleware, where each simulator communicates with others directly through dedicated adapters. The diagram on the right hand illustrates the improved version with event processing in mind. A rule base, embedded in the rule engine, is constructed to handle most of the federation logics in a declarative fashion. Since the federation logics are relocated from the dedicated adapters to the event engine, the original adapters with hardcoded imperative code for federation logics are becoming lightweight declarative rules.

![Figure 1. The original DIESIS federation middleware on the left and the improved version with event processing and declarative rule base on the right.](image)

Dependency Analysis in CPS networks

Four categories of dependencies between CPS have been discussed in previous works [14]: physical, cyber, geographical, and logical. In this paper we focus on the logical dependencies between CPS. A CPS can be decoupled as a set of elements:

$$CPS = \{ET_1, ET_2, ET_3, \ldots\}$$

For instance, a regional electric distribution network REDN contains multiple substations SS, transformers TF, loads LD, power lines PL, etc. Similarly a regional telecommunication network RTMN includes numerous routers RT, base stations BS, and cables CB.
REDN = \{SS1, SS2, TF1, LD1, LD2, LD3, PL1, PL2, \ldots\}

RTMN = \{RT1, RT2, BS1, BS2, BS3, CB1, CB2, CB3, \ldots\}

The internal dependencies of a domain are handled by separate domain-specific simulators. They deliver specific system dynamics like the load flow calculation done by SINCAL. The federated simulation on the other side focuses on the cross-CPS dependency analysis, i.e. if the state of an element in one CPS is changed, how it influences the behaviour of other CPS in an inter-connections environment. For two CPS, it can be formed as a total function $DP^2$:

$DP^2 : CPS_1 \times CPS_2 \rightarrow \{0, 1\}$

It is defined as a total function instead of partial, because in the proposed approach all dependencies must be explicitly identified beforehand. If the dependency between two elements is not clear, fuzzy approach can be applied - instead of Boolean values, probability should be used. This is however out of the scope of this paper and will be elaborated in another forthcoming article.

Declarative Dependency Specification

The motivation for the declarative approach lays in the fact that the federated simulation can be viewed as an event-driven application, where the state transition of elements in simulators can be considered as events. On the other side, comprehensive research work has already been conducted in both Complex Event Processing [7] and Data Stream Processing [10] communities. Leverage the results from there and further adapt them to handle federated simulation can substantially reduce the complexity of dependency modelling.

Federated Simulation Event Streams

An event stream $S$ in the proposed approach is an ordered list of federated simulation events

$S = \langle E_1, E_2, E_3, \ldots \rangle$

A federated simulation event is defined mathematically as a tuple:

$E=\langle ID, T_0, T_1, SRC, PLD \rangle$

Where $ID$ uniquely identifies an event, $T_0$ is the occurrence time represented as simulation time, $T_1$ is the arrival time represented as the real wall time, $SRC$ is an identifier of the simulator that generates the event, $PLD$ is the event payload with a list of key-value pairs. $T_0$ uses simulation time and $T_1$ uses wall time, this distinction is critical to correctly model the temporal semantics of a federated simulation event. Depending on the performance and loads of computers, simulators with the same model can take longer or shorter time to perform the calculation. Use simulation time in $T_0$ guarantees the same result if the federated simulation is repeated. Event payload is a list of key-value pairs which contains essential contextual information about the event.

Dependency as Declarative Rules

Formally, a declarative event handling rule $R$ is defined as a partial function:

$R : \{E_i, E_j, \ldots \} \times C \rightarrow E_k$

where $E_i, E_j, \ldots$ are input events that are either from the same event stream or from different event streams. Input events can be one single event or multiple events. $C$ denotes the context information that is outside of the simulated domains. Examples of context information are e.g. weather conditions, traffic situations, etc. $E_k$ is the output event - only one event can be derived from a rule. The derived event is inserted into the target event stream and can be further used as input events for other rules.
Some of $E_k$ are modelled as actionable events, i.e. events contain command information to change the physical systems - either in simulated or real world. The execution of these events are however application-specific. It depends on the interfaces exposed by physical devices and simulators.

**Event Processing Network**

All the rules form the rule base for a given federation simulation. Together with the event streams, an event processing network $\mathcal{EPN}$ is established:

$$\mathcal{EPN} = \{\{S_1, S_2, \ldots\}, \{R_1, R_2, \ldots\}\}$$

An $\mathcal{EPN}$ is a graph with the event streams $\{S_i\}$ as vertices and the rule base $\{R_i\}$ as edges. A vertex denotes all event instances in a stream. A rule denotes how events are derived and connected.

One benefit provided by the declarative approach is its flexibility. If the dependencies are changed, only the rule base needs to be modified; meanwhile, the rules are declarative, i.e. the whole dependencies between CPS can be adapted at runtime without re-building the simulation middleware.

**Case Study**

In this section, a simplified example is presented to demonstrate the capability and flexibility of the proposed declarative approach. Two simulators SINCAL [16] and ns-3 [13] are included in this example. SINCAL is used for load flow calculation and ns-3 for network package simulation.

The electric network model in SINCAL consists of one feeder FD1, one two-winding transformer TM1, two power lines LN1 and LN2, and two loads LD1, LD2. The internal dependencies of these elements are illustrated in the left part of Figure 2. Similarly, the telecommunication network model of ns-3 consists of three routers RT1, RT3, and RT3, two base stations BS1 and BS2. The internal dependencies are illustrated in the right part of Figure 2. The pseudo code below demonstrates the definition of event streams and the corresponding event processing rules. Five streams are defined below. The Event stream is the base stream gathering all events from external. The SincalStream and NS3Stream are for events from SINCAL and NS3 respectively. The two update streams with URL as payloads are streams with *actionable* events: RESTful endpoints will be:

```plaintext
create schema Event (sim, comp, status);
create schema SincalStream (comp, status);
create schema NS3Stream (comp, status);
```

Figure 2. Simplified SINCAL and ns-3 model where LD1 provides power to RT2 and BS1; while LD2 provides power to RT1, RT3 and BS2.

Invoked if one event is inserted into these streams.
```plaintext
create schema Event (sim, comp, status);
create schema SincalStream (comp, status);
create schema NS3Stream (comp, status);
```
create schema Sinca1UpdateStream (comp, status);
create schema NS3UpdateStream (comp, status);

Two basic re-dispatching rules: If an event is sent by SINCAL with sim=1, derive a new event and insert it into the Sinca1Stream; similarly If an event is sent by ns-3 with sim=2, derive a new event and insert it into the NS3Stream:

insert into Sinca1Stream select comp, status from Event where sim = 1 ;
insert into NS3Stream select comp, status from Event where sim = 2 ;

The following rules are for CPS dependency handling. After the load flow calculation in SINCAL, if LD1 is out of operation, RT2 and BS1 will lose their power supply (suppose no backup power supply is equipped).

insert into NS3UpdateStream select "RT2" as comp, "failed" as status from Sinca1Stream
where comp="LD1" and status="failed";
insert into NS3UpdateStream select "BS1" as comp, "failed" as status from Sinca1Stream
where comp="LD1" and status="failed";

Similarly, after the load flow calculation in SINCAL, if LD2 is out of operation, RT1, RT3, and BS2 will lose their power supply (suppose no backup power supply is equipped).

insert into NS3UpdateStream select "RT1" as comp, "failed" as status from Sinca1Stream
where comp="LD2" and status="failed";
insert into NS3UpdateStream select "RT3" as comp, "failed" as status from Sinca1Stream
where comp="LD2" and status="failed";
insert into NS3UpdateStream select "BS2" as comp, "failed" as status from Sinca1Stream
where comp="LD2" and status="failed";

Inter-dependencies can also be modelled in a similar way. For instance, if the distribution network is operated with a SCADA system that needs the Internet services provided by the routers in the telecommunication network, additional dependencies from telecommunication network to the distribution network should also be modelled. In the end, depending on the network configuration, the federated simulation will converge and all CPS elements could be out of operation.

Related Work

Investigating dependencies between CPS like critical infrastructures has been a research topic for more than a decade [14, 12]. Various methods have been proposed to model and analyse the dependencies, including Bayesian Network based approaches [4], federated simulations [15], Ontology-based modelling [17-18], etc. The proposed approach focuses on extending the federated simulation by introducing the event processing components with declarative rules. Several standards have already been proposed for interoperability of CPS simulators like the High Level Architecture (HLA) [5] that is widely used in military applications. Event processing is however not explicitly addressed in these standards. Dedicated federated simulators for CPS exist as well, like I2Sim [9]. They need however a common model for all involved domains, which means existing models cannot be re-used and huge efforts are needed to build the models. On the other side, to our best knowledge, little work has been done in leveraging declarative rules to facilitate event-driven federated simulation. Several relevant works [2] are done in the area of cyber defence.

Conclusion

This paper briefly introduced the motivation of using declarative rules to model the (inter-)dependency of large-scale CPS. The proposed approach further develops the federation
middleware in the DIESIS project by introducing the concept of event streams and declarative rule bases. A small use case demonstrates the basic idea and potential benefits of using declarative rules in modelling CPS dependencies: flexibility, maintainability and scalability. One of the major limitations of the proposed declarative approach is that it is difficult to guarantee the consistency of large rule base: as the model increases, more rules are needed for dependency modelling. Effective algorithms need to be developed to automatically verify the rule base.

To further improve the interoperability of the proposed approach, semantic approaches with explicit semantics can be applied. For example, the common vocabulary CIS0 proposed in [18] can be used to construct a set of event processing rules that can be shared between different scenarios. In addition, fuzzy dependencies in different domains exist as well. The expressivity of the proposed declarative rules should be enhanced to support them. Supporting data provenance is also an important topic. After the federated simulation is finished, some kind of explanation should be provided why certain events are derived: which rules are triggered, what is the triggering events, etc. This is critical for debugging these declarative-based solutions.

Advanced tool support should also be provided to facilitate the development of these declarative rules. For instance, dedicated rule editor with syntax highlight and grammar checking is essential to accelerate the rule development. Graphical programming based systems like the one provided by IBM Amit [1] would be helpful to develop the Event Processing Network (EPN) in a user-friendly fashion. Integrated development environments containing fully-fledged text editors, graph editors and rule base verification tools can substantially reduce the barrier to adopt the proposed declarative approach.

**Acknowledgements**

The research leading to these results was funded by the European Commission within the Seventh Framework Programme project CIPRNet and PREDICT. The authors thank all of the project partners for many constructive discussions and useful suggestions.

**References**


