Complex Adaptive Systems Modeling

METHODOLOGY Open Access

A formal specification framework for smart grid components

Waseem Akram and Muaz A. Niazi*

*Correspondence:
muaz.niazi@gmail.com
Computer Science Department, COMSATS University, Park road, Islamabad 45550, Pakistan

Abstract

Purpose: Smart grid can be considered as the next step in the evolution of power systems. It comprises of different entities and objects ranging from smart appliances, smart meters, generators, smart storages, and more. One key problem in modeling smart grid is that while currently there has previously been a considerable focus on the proof of concept aspect of smart grid, there have been very few modeling attempts and even lesser attempts at formalization. To the best of our knowledge, formal specification has not been applied previously in the domain of smart grid.

Methods: Using a state-based formal specification language namely Z (pronounced as 'Zed'), we present a novel approach to formally modeling and specify smart grid components.

Results: The modeling exercise clearly demonstrates that Z is particularly suited for modeling various smart grid components.

Conclusions: The presented formal specification can be considered as a first step towards the modeling of smart grid using a Software Engineering formalism. It also demonstrates how formal specification can be used to model complex systems in general, and the smart grid, in particular.

Keywords: Formal specification, Smart grid, Complex adaptive system

Background

A smart grid can be considered as an advanced and radically evolved version of traditional power systems. The term 'smart' in the smart grid exemplifies the use of advanced technology such as bi-directional communication, artificial intelligence (Tokody et al. 2018), Complex systems theory (Iantovics et al. 2018), modeling and simulation, and more, all employed with the goal of converting the legacy power grid into an advanced proactive and reactive system. At the lowest level, the Smart grid can be considered as an integrated system made up of a variety of interacting components—ranging from smart appliances and smart storages to smart generators, Internet of Things (IoT) (Fortino et al. 2017), and beyond. Another key focus area of the Smart grid is in the integration of renewable energy resources, such as, but not limited to, wind turbines and solar panels (Wong and Pinard 2017). By integrating advanced communication and information systems, Smart grid components can communicate and coordinate with each other with the goal of constructing a sustainable and efficient energy production system (Gungor et al. 2011) for the future.

An increase in the size of smart grid has unexpected consequences. This increase is coupled with a corresponding increase in its intrinsic complexity (Cintuglu et al. 2017). Firstly, it is important to understand that smart grid implies complexity (Monti and Ponci 2010). To understand this complexity, consider the fact that, in any modern large-scale power system, each component is often times, dynamic in nature. As such, the states of the system also vary temporally. The reason for that is that the states are not only a function of, but also, are the result of emergent properties of the same numerous interacting components. Thus the result of what we see at the macro-level cannot thus be easily discernible as a direct function of the micro-level components. This behavior can thus be considered as the outcome of numerous interactive events occurring in relation to each component—quite similar to a natural complex adaptive systems (CAS) (Iantovics 2013; Amin and Wollenberg 2005).

With so much complexity at hand, it is clear that modeling the smart grid certainly needs considerable number of practical examples and case studies (Pagani and Aiello 2014). It is also evident that developing various types of formalisms for the domain is needed. This will allow for the selection of better and more elegant solution to models—which in turn can be used to develop a better understanding of the complex domain. Essentially, modeling any system can be considered as an activity which allows for a better understanding of the system. In the smart grid domain, better modeling approaches can not only simplify system complexity but also allow for a better understanding and implementation of the system. Besides, it can also allow for ensuring a reduction in system failures.

Formal methods provide facilities for the modeling of each component of any complex system (Hall 1990). It allows for developing models for each component of the system allowing for a clear focus on understanding consistency as well as semantic correctness. The behavior of each system can be analyzed and observed with the help of these formal models. A key benefit to this approach is that it helps in the detection of faults and flaws in the design phase of system development, thereby considerably improving system reliability.

In previous studies, formal specification framework has been successfully applied for the mathematical modeling of different CAS ranging across various domains. Some key examples of such work includes a formal specification used for the modeling of AIDS spread using agent-based modeling (Siddiqa and Niazi 2013). Likewise, it has been developed for modeling the progression of researchers in their domain (Hussain and Niazi 2014), and for the modeling of wireless sensor networks (Niazi and Hussain 2011a). The use of formal specification models for modeling CAS also include studies such as (Zafar and Afzaal 2017; Afzaal and Zafar 2016). Suggestions to use formal specification for the Smart grid have also previously been mentioned in literature (Hackenberg et al. 2012). Another example is the use of state machine formalism (Turner 2014). However, to the best of our knowledge, the same approach has not been applied much in the domain of the smart grid domain. It is thus clear that there is a growing need to model the key components in a smart grid by means of an elegant formal framework among other tools such as noted previously (Rohjans et al. 2014). Such a prudent approach allows for a better understanding of the domain besides allowing for systems to be verified using the given specification.

In this paper, we present first steps towards a basic formal specification modeling framework for smart grid components. We first consider different types of entities and then elaborate their detailed formal specifications.

The rest of the paper is structured as follows: first, basic concept of a formal framework and a smart grid scenario is discussed. This is followed by formal specification of different entities of smart grid system. The paper ends with a conclusion section.

Theoretical foundation

In this section, we present the theoretical foundations needed for understand the presented formal specification model.

Formal frameworks

A formal specification is a software engineering approach that is used for mathematical modeling of different components of a system (Hall 1990). During engineering of any system, it is important to ensure that all the system components are integrated and working correctly without any error. A formal specification provides this facility to accurately specify each requirement of the system before going to the real implementation (Woodcock and Davies 1996).

A formal specification decomposes the large system into the subsystem. Then provides a specification for each individual subsystems. It follows two approaches; one is the algebraic approach in which each operation and relationships can be described, second is a model-based approach which concerns with the state and transitions of each individual components. The model-based approach uses the Z language specification which involves the utilization of mathematical notations, sets, sequences, and states (Bowen 1996).

Scenario of the smart grid

We can think of a smart grid as a complex system in which different consumers and generation units are connected through power communication lines (Milanovic and Zhu 2017). Generation units generate power and transmit toward consumer's side. Consumers demand energy according to their usage profile and generation units response according to the consumer's demand (Yang et al. 2017).

On the consumer's side, the process of monitoring and controlling consumer's demand energy usage profile is called home energy management (HEM) (Hosseini et al. 2017). This process involves the use of smart meters, smart appliances, information and communication technology etc. the smart meters collect data about consumer's demand at the different time period and transmit this collected data to the power generation unit. On the power generation unit side, this data is analyzed and response according to the consumer's demand. Sometimes, consumer's demands exceed from available energy power at the grid side. This creates unbalance situation of the power system. To handle this issue, the concept of demand response management (DRM) (Soares et al. 2017) has been introduced.

DRM is the process of adjusting consumer's demand in response to the price of energy and incentives from grid side. The generation unit sends their energy cost pattern of different time periods. The time in which power cost is high is called high peak hour and the time in which power cost is low is called off peak hour. So during high peak hour, consumers can keep their flexible appliances off and run at low peak hour.

To reduce burden on generation unit, the concept of renewable energy sources (RES) is also introduced in a smart grid application (Park et al. 2017). RES can be in the form of photo-voltaic (PV) or wind energy sources. These energy sources can be installed and provide energy to the consumer's side. Consumers can use this energy at an off-peak hour, so this solves the energy unbalance as well as high-cost issues. The energy collected from RES can be stored using storage devices. Sometimes, if the available energy is larger than consumer's demands, then it can be sold back to the grid unit by using some buyback mechanism (Chiu et al. 2017). However, RES has unpredictable nature and depends on weather and time. PV energy can only be produced during the daytime i.e. at sunny day. Wind energy can only be produced in windy environments.

Regarding RES, a new concept has also been included in a smart grid called electric vehicles (EVs) (Ahmadian et al. 2017). These EVs are using energy storage devices which can be charged either using PV or electric station. The stored energy is then used for drive vehicles. In case if the energy demands of consumers exceed at grid unit. These EVs can also provide energy to the grid unit by using the vehicle to grid (V2G) mechanism.

Formal specification framework

In this section, we present formal specification models for the smart grid components. In our study, we consider four different entities in a smart grid system. These entities are: appliance, solar, turbine, and storage devices. Each entity has different states as well as associated events that cause state transitions. In Table 1, summary of each object, states and events are given.

Smart appliance

An appliance object has three distinct states along with events associated with states. As we can see in Fig. 1, Whenever an unplugged event occurs, the appliance enters in the disconnected state. By plugged-in event, the appliance enters in the connected state. And whenever an in-use event occurs, the appliance remains in the running state.

Table 1 Smart grid components, their states and events

Object	States	Events
Appliance	Disconnected	Unplugged
	Connected	Plugged-in, not in-use
	Running	In use
Turbine	Not running	No wind
	Slow running	Slow wind
	Fast running	Fast wind
Solar	No energy generation	Night time
	Partial energy generation	Day, cloudy
	Full energy generation	Day, sunny
Storage	Charging	Store energy
	Discharging	Consume energy
	Not-in-use	Remove storage

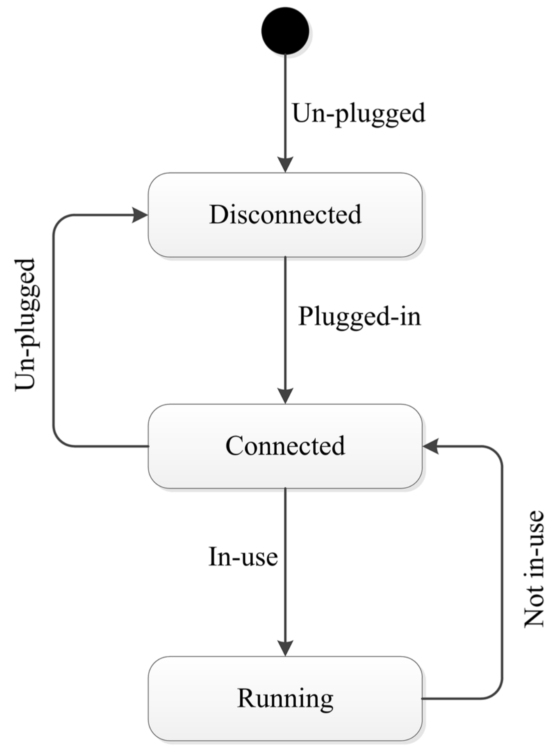

Fig. 1 Statechart depicting the life of an appliance

Next, we present formal specification for an appliance object.

A free type "APPLIANCESTATE" is used to represent different states of an appliance that are disconnected, connected and running.

$$[APPLIANCESTATE] == \{disconnected, connected, running\}$$

Next, we define an *appliance* schema that contains "appState" variable of type "APPLIANCESTATE". The value of "appState" can be either disconnected, connected or running.

$Appliance$

$appState : APPLIANCESTATE$

As we have defined appliance state and appliance schema. Now we can move towards operational schemas.

First, we start by presenting initialization schema named as *InitAppliance*. In this schema, we declare "Appliance" schema as a variable. In predicate section, we say that on initialization, the state of an appliance must be equal to disconnected.

$InitAppliance$

$Appliance$

$appState = disconnected$

As the plugged-in event occurs, the state of an appliance becomes connected. To show this transition, we define an operational schema called *PluggedInAppliance*. In the schema, the changing state of an appliance is shown by a delta sign (Δ). In predicate, we say that the state of an appliance is changed from disconnected to the connected state.

PluggedInAppliance
$\Delta Appliance$

$appState = disconnected$
$appState' = connected$

The state of an appliance is also changing whenever an in-use event occurs. Then the appliance remains in a running state. This transition is shown by defining *InUseAppliance*. Here, again we declare the changing state of an appliance with a delta sign. In predicate, first an appliance must be in the connected state, then we say that the state changes to the running state.

InUseAppliance
$\Delta Appliance$

$appState = connected$
$appState' = running$

As we noted before, when an unplugged event occurs, an appliance goes to the disconnected state. This transition is shown by another schema called *UnPluggedAppliance*. In the predicate section, we can see that the state of an appliance changes from connected to the disconnected state.

UnPluggedAppliance
$\Delta Appliance$

$appState = connected$
$appState' = disconnected$

An appliance can be in connected but not in use state. This operation is shown by introducing another schema called *NotInUseAppliance*.

NotInUseAppliance
$\Delta Appliance$

$appState = running$
$appState' = connected$

There can be some scenario in which a schema can be success or there may some error. To understanding the success and error conditions of schemas, we present free type definitions in the form of Table 2. In the given table, for each schema we define the success and failure conditions.

Wind turbine system

A wind turbine object has three distinct states named as turbine not running, turbine slow running, and turbine fast running. There are also different events associated with each state that causes a state transition. In Fig. 2 we can see that when there is no wind, the turbine enters in the "turbine not running" state. When there is slow wind, turbine runs slowly. And if there is fast wind, the turbine is running fast.

Next, we present formal specification for a turbine object.

First, we start by defining a free type "TURBINESTATE" set. This set comprises of a turbine different states that are "turbineNotRunning, turbineSlowRunning, and turbineFastRunning".

Table 2 Success and error outcomes of the schemas for an appliance entity

Schema	Pre-condition for success	Condition for error
PluggedInAppliance	Appliance is disconnected *appState = disconnected*	Already connected *appState = connected*
InUseAppliance	Appliance is connected *appState = connected*	Already in running state *appState = running*
UnPluggedAppliance	Appliance is connected *appState = connected*	Already in disconnected state *appState = disconnected*
NotInUseAppliance	Appliance is running *appState = running*	Already not in not running state *appState ≠ running*

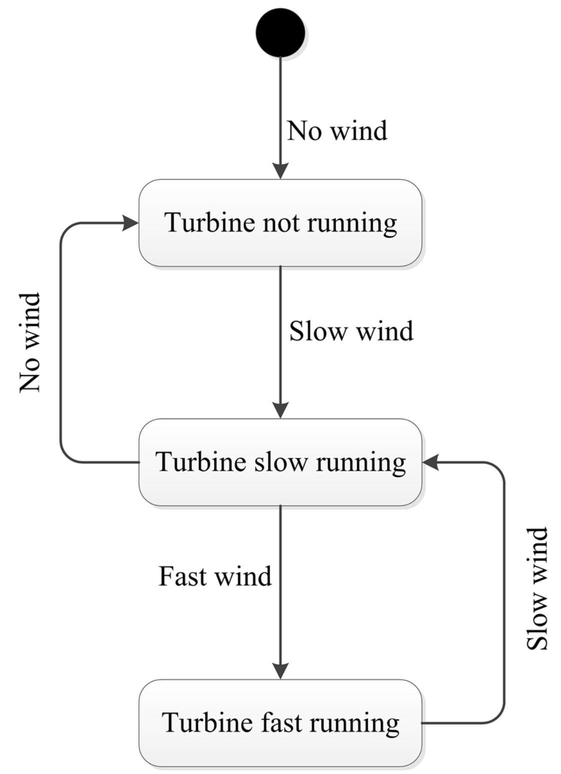

Fig. 2 Statechart depicting a Wind turbine life

$[TURBINESTATE] == \{trubineNotRunning,$
$turbineSlowRunning, turbineFastRunning\}$

Now, we define a wind turbine system schema by presenting *WindTurbine*. This schema consists of a variable "trbState" of type *TURBINESTATE*. The value of this variable can be either "turbineNotRunning", "turbineSlowRunning", or "turbineFastRunning".

```
WindTurbine
trbState : TURBINESTATE
```

As we have defined turbine's states and turbine's schema, now we move to operational schemas.

First, we start by initializing a turbine entity by presenting *InitTurbine* schema. In the schema, first, we call the *WindTurbine* schema. Then in the predicate, we say that the initial state of a turbine must be equal to "turbineNotRunning" state.

InitTurbine
$WindTurbine$

$trbState = turbineNotRunning$

As we noted in the state diagram when during slow wind, the state of a turbine becomes "turbineSlowRunning". This transition is shown by defining *SlowWind* schema. In the schema, we call the *WindTurbine* schema with a delta sign. In predicate, we say that the state of a turbine is changing from "turbineNotRunning" to the "turbineSlowRunning".

SlowWind
$\Delta WindTurbine$

$trbState = turbineNotRunning$
$trbState' = turbineSlowRunning$

In case of fast wind, a turbine runs fast. To show this process, we present another schema called *FastWind*. In the schema, again we call *WindTurbine* schema with a delta sign. In predicate, we say that the state of a turbine changes from slow to fast running.

FastWind
$\Delta WindTurbine$

$trbState = turbineSlowRunning$
$trbState' = turbineFastRunning$

A turbine's state also changes to not running when there is no wind. This transition is shown by introducing *NoWind* schema. The schema shows that a turbine's state is changed from slow running to not running state.

NoWind
$\Delta WindTurbine$

$trbState = turbineSlowRunning$
$trbState' = turbineNotRunning$

The schemas of the turbine entity can be success or fail when errors occur. So to understand the success and error conditions of each turbine's schemas, we present free type definitions in Table 3.

Table 3 Success and error outcomes of the schemas for a wind turbine entity

Schema	Pre-condition for success	Condition for error
SlowWind	Turbine not running $trbState = notRunning$	Already in slow running state $trbState = turbineSlowRunning$
FastWind	Turbine is running slowly $trbState = turbineSlowRunning$	Already in fast running state $trbState = turbineFastRunning$
NoWind	Turbine is running slowly $trbState = turbineSlowRunning$	Already in not running state $trbState = turbineNotRunning$

Solar system

A solar object has three distinct states named as "no energy generation, partial energy generation, and full energy generation". There are also different events associated with each state that causes a state transition. The states and it's associated events have shown in Fig. 3. Next, we present formal specification for a solar object.

First, we begin by introducing a free type "SOLARSTATE" that is used for presenting different states of a solar entity. This set comprises of "noEnergyGeneration, partialEnergyGeneration, and fullEnergyGeneration" states.

$$[SOLARSTATE] == \{noEnergyGeneration,\\ partialEnergyGeneration, fullEnergyGeneration\}$$

Here, we present a solar system by introducing *SolarPanel* schema. The schema takes a variable "slrState" of type *SOLARSTATE*. The value of this variable can be either "noEnergyGenreation", "partialEnergyGeneration" or "fullEnergyGeneration".

```
┌ SolarPanel ─────────────────────────
│ slrState : SOLARSTATE
└─────────────────────────────────────
```

As we have defined solar state and schema, now we move towards the operational schemas.

The initialization process is shown by presenting *InitSolar* schema. First, the schema calls the *SolarPanel* schema in the declaration part. In predicate, we defined that the state of a solar is "noEnergyGeneration" at the initial time.

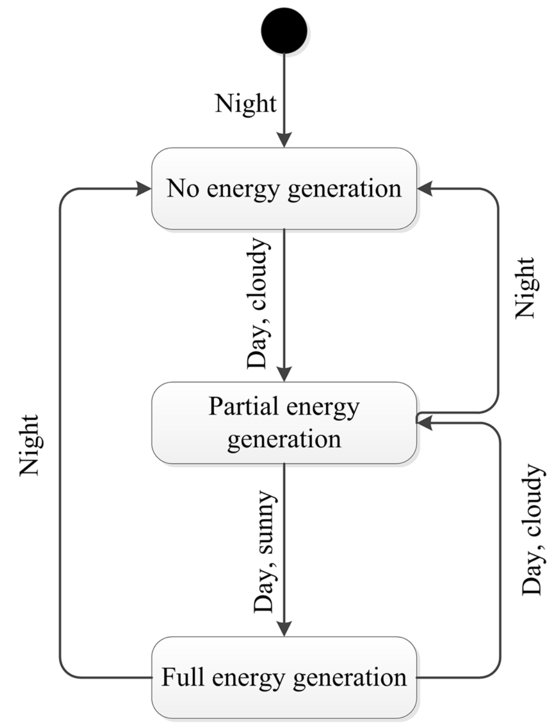

Fig. 3 Statechart for a solar energy system

$$\begin{array}{|l} \textit{InitSolar} \\ \textit{SolarPanel} \\ \hline \textit{slrState} = \textit{noEnergyGeneration} \\ \hline \end{array}$$

As we know that when there is a cloud at daytime, then the solar generates partial energy. During this time period, the solar remains in a partial energy generation state. This process is shown by introducing *DayAndCloudy* schema. In the schema, the changing state of a solar is shown by a delta sign. In predicate, it shows that the state of solar changes from no generation to the partial generation.

$$\begin{array}{|l} \textit{DayAndCloudy} \\ \Delta \textit{SolarPanel} \\ \hline \textit{slrState} = \textit{noEnergyGeneration} \\ \textit{slrState}' = \textit{partialEnergyGeneration} \\ \hline \end{array}$$

In case of the sunny day, a solar remains in a full generation state. This process is shown by defining *DayAndSunny* schema. Here, we again define the changing state of a solar with delta sign. In predicate, we say that the state of solar changes from partial to full energy generation.

$$\begin{array}{|l} \textit{DayAndSunny} \\ \Delta \textit{SolarPanel} \\ \hline \textit{slrState} = \textit{partialEnergyGeneration} \\ \textit{slrState}' = \textit{fullEnergyGeneration} \\ \hline \end{array}$$

As we know that a solar does not generate any energy during night time. At this time period, solar remains in a no generation state. Here, we present another schema called "Night". In the schema, we say that the current state of a solar can be partial or full energy generation. Then we change solar state to the no energy generation state.

$$\begin{array}{|l} \textit{Night} \\ \Delta \textit{SolarPanel} \\ \hline \textit{slrState} = \textit{partialEnergyGeneration} \lor \textit{fullEnergyGeneration} \\ \textit{slrState}' = \textit{noEnergyGeneration} \\ \hline \end{array}$$

The success and failure conditions for the solar's schemas is summarized in Table 4.

Storage system

A storage device also has different states as well as events that cause the transition of states. As we can see in Fig. 4 that there are three states named as "charging, discharging, and not in use" of a storage object. When a store energy or battery low

Table 4 Success and error outcomes of the schemas for a solar entity

Schema	Pre-condition for success	Condition for error
DayAndCloudy	Solar is in no generation state *slrState* = *noGeneration*	Already in partial generation state *slrState* = *partialEnergyGeneration*
DayAndSunny	Solar is in partial state *slrState* = *partialEnergyGeneration*	Already in full generation state *slrState* = *fullEnergyGeneration*
Night	Solar is in partial or full state *slrState* = *PartialEnergyGerention*∨ *fullEnergyGeneration*	Already in no generation state *slrState* = *noEnergyGeneration*

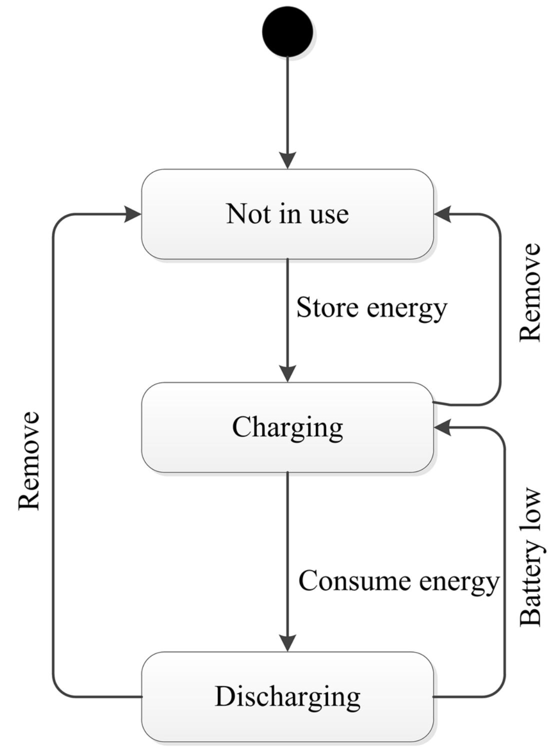

Fig. 4 Statechart for a storage system

event occurs, the storage enters in the charging state. In case of energy consumption, the state of storage becomes discharging, In case remove event, the state of storage becomes not in use.

Next, we present formal specification for a storage system.

First, we start by defining a free type *STORAGESTATE* set that comprises of different state of a storage device.

$[STORAGESTATE] == \{charging, discharging, notInUse\}$

Here, we define a storage system schema. In the schema, we declare a variable "strState" of type *STORAGESTATE* that has any single value from the storage state set.

```
StorageDevice
strState : STORAGESTATE
```

As we have defined storage state and system schema. Now, we move towards operational schema.

First, we start by introducing an initialization schema called *InitSorage*. This schema calls the *StorageDevice* schema as a variable. In predicate, it shows that the state of a storage is "notInUse" at the initial time.

```
── InitStorage ──────────────────────
│ StorageDevice
├──────────
│ strState = notInUse
└─────────────────────────────────────
```

As we noted before, when a store energy event occurs, a storage goes to the charging state. This process is shown by *StoreEnergy* schema. The changing of a storage is shown by a delta sign. In predicate, it shows that the current state of a storage is "notInUse" which is changed to the "charging" state.

```
── StoreEnergy ──────────────────────
│ ΔStorageDevice
├──────────
│ strState = notInUse
│ strState' = charging
└─────────────────────────────────────
```

We also know that when a consume energy event occurs, a storage device goes to the discharging state. This process is shown by presenting *ConsumeEnergy* schema. In the schema, we can see that the state of a storage changes from charging to the discharging state.

```
── ConsumeEnergy ────────────────────
│ ΔStorageDevice
├──────────
│ strState = charging
│ strState' = discharging
└─────────────────────────────────────
```

A storage can also enter in the charging state whenever the storage state is low. This process is presented by means of *BatteryLow* schema.

```
── BatteryLow ───────────────────────
│ ΔStorageDevice
├──────────
│ strState = discharging
│ strState' = charging
└─────────────────────────────────────
```

When a storage is removed from the system, then it goes to the "notInUse" state. This process is shown by introducing another schema called *RemoveStorage*. In the schema, we can see a storage can be in either charging or discharging state. Then it's state changes to the "notInUse" state.

```
── RemoveStorage ────────────────────
│ ΔStorageDevice
├──────────
│ strState = charging ∨ discharging
│ strState' = notInUse
└─────────────────────────────────────
```

The schemas of the storage devices can be success and may be there some errors at the execution time. So there is also need to define the success and error criteria for schemas. Here, in Table 5, we summarized all success and failure conditions of the storage's schemas.

Comparison with previous work

In this section, we discuss different related work carried out in the smart grid domain using formal method approaches. We highlight their objectives and key contributions to the smart grid technology. However, the previous studies do not consider modeling of

Table 5 Success and error outcomes of the schemas for a storage entity

Schema	Pre-condition for success	Condition for error
StoreEnergy	Storage is in not-in use $strState = notInUse$	Already in charging state $strState = charging$
ConsumeEnergy	Storage is in charging state $strState = charging$	Already in discharging state $strState = discharging$
BatteryLow	Storage is in discharging state $strState = discharging$	Already in charging state $strState = charging$
Remove	Storage is in charging or dischargig state $strState = discharging \vee charging$	Already in not-in-use state $strState = notInUse$

different key components of the smart grid as presented in this paper. Some of the key papers are discussed as follows.

In paper (Pózna et al. 2018), the author used a colored Petri net approach for modeling electrical network. The network is composed of feeders and sources. The proposed work effectively diagnose and locate unbalance situation on the network.

In the paper (Mhadhbi et al. 2018), a distributed energy management model is proposed that capable of handling neighbors surplus energy. The system is modeled using colored Petri net approach which achieved system balancing.

The smart microgrid is an effective way of using renewable energy resources. As the number of component increases, its complexity also increases. To handle the complex scenario of the smart microgrid, in the paper (Halim 2018), the author proposed the new hybrid Petri net approach. The proposal modeled power balance in a bus. This work showed various operations strategies that are useful for achieving network balancing.

Another use of Petri net approach is discussed in the domain of smart grid (Jiang et al. 2018). In this work, a hybrid distributed control system for modeling smart grid network. They used supervisors and local solutions for fault handling. The proposal is capable of detecting and locating fault using colored Petri net approach.

In paper (Loia et al. 2017), a multi-agent approach is used to monitor states of the smart grid as well as its components. The objective was to define optimal power flow on the network. The Fuzzy transform approach is applied on a large operational dataset of the smart grid. This work reduces storage needs in the network. By deploying agents, a reliable system is achieved.

In paper (Sultan et al. 2017), the author proposed the use of the formal method for smart grid components. In the study, two components that are smart meter and transformer are used. The states and operations of the components are modeled using UML notations. In this work, the VDM-SL language is used.

In previous studies, the smart grid architecture approach is used for modeling, requirement engineering, and analysis in different literature. In the paper (Neureiter et al. 2016), SGAM is integrated with a formal method for security requirement. In this study, different security classes, risk, and vulnerabilities are identified and modeled using a formal method along with SGAM.

In the smart grid, there involves heterogeneous components and complex interaction among them. In a large-scale power system, the power flow and interaction among different entities lead to the emergent behavior of the system. In paper (Kolen et al. 2018), the author focused on the distribution aspect of the smart grid. This study proposed

DistAIX modeling paradigm for power distribution system. The proposed model comprises of power lines, communication, control, interaction.

Discussion and future work

The idea in this paper was primarily to identify and specify key components of the smart grid. This included an identification of their different states, in addition to events resulting in a state transition. In this regard, we have considered appliances, solar system, turbine, and storage as the key components and presented formal models of each component. The formal specification using the 'Z' language is a state-based approach useful for defining and developing mathematical models of different entities with dynamic behavior. However, there is more work that needs to be performed especially in the domain of integration of different entities and components into a single system. In cases such as smart homes, additinal techniques such as artificial intelligent algorithms (Crișan et al. 2017) in general and agent-based computing (Niazi and Hussain 2011b) in particular can be especially useful. Agent-based modeling allows to develop an abstract model of any complex system allows for a clear understanding of the model, behavior of each component, thus results in understanding the global behavior of the system. The optimization approaches are needed to create optimal decisions on energy consumption at the consumer side.

Conclusions and future work

In this paper, we focused on formal specification of smart grid components. We considered four different components of a smart grid system. We identified states and events which cause state transitions of each individual component. Then we modeled each component of the smart grid system by using a formal specification approach. This provides a clear understanding the behavior of each component involved in the system. The presented formal framework clearly demonstrates the utility of using a formal specification approach to modeling complex systems including technological systems such as the smart grid.

Here we would like to mention that the presented framework currently considers only four key smart grid components. However, the same can be further expanded for a large number of devices. For example, the same approach can be applied to the smart home energy management system. A smart home is an integrated energy system comprises of different and a large number of smart objects such as sensors, smart meters, smart appliances etc. The smart home also integrates solar panels and storage devices Thus, a smart home can be considered as an integrated system consists of various objects and system. In such a system, each component communicates and interact with each other. The users demands and available energy needed adaptive strategies. In such scenarios, the Petri net approach is also useful. Petri net approach is an interval-value based specification approach and deals with uncertain situations that is useful for modeling integrated systems such as a smart home. Modeling of the smart home leads to clear understanding the overall behavior of the smart grid.

Authors' contributions
Overall, the authors contributed equally to the paper. MN suggested idea for the paper. MN and WA designed the schemas for each entities of the smart grid system. Both authors wrote the paper. Both authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Funding
Not applicable.

Publisher's Note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 20 March 2018 Accepted: 24 August 2018
Published online: 03 September 2018

References
Afzaal H, Zafar NA (2016) Formal analysis of subnet-based failure recovery algorithm in wireless sensor and actor and network. Complex Adapt Syst Model 4(1):27
Ahmadian A, Sedghi M, Elkamel A, Fowler M, Golkar MA (2017) Plug-in electric vehicle batteries degradation modeling for smart grid studies: review, assessment and conceptual framework. Renew Sustain Energy Rev 8:2609–2624
Amin SM, Wollenberg BF (2005) Toward a smart grid: power delivery for the 21st century. IEEE Power Energy Mag 3(5):34–41
Bowen JP (1996) Formal specification and documentation using Z: a case study approach. International Thomson Computer Press, London
Chiu TC, Shih YY, Pang AC, Pai CW (2017) Optimized day-ahead pricing with renewable energy demand-side management for smart grids. IEEE Internet Things J 4(2):374–383
Cintuglu MH, Mohammed OA, Akkaya K, Uluagac AS (2017) A survey on smart grid cyber-physical system testbeds. IEEE Commun Surv Tutor 19(1):446–464
Crişan GC, Pintea CM, Palade V (2017) Emergency management using geographic information systems: application to the first romanian traveling salesman problem instance. Knowl Inf Syst 50(1):265–285
Fortino G, Gravina R, Russo W, Savaglio C (2017) Modeling and simulating internet-of-things systems: a hybrid agent-oriented approach. Comput Sci Eng 19(5):68–76
Gungor VC, Sahin D, Kocak T, Ergut S, Buccella C, Cecati C, Hancke GP (2011) Smart grid technologies: communication technologies and standards. IEEE Trans Ind Inform 7(4):529–539
Hackenberg G, Irlbeck M, Koutsoumpas V, Bytschkow D (2012) Applying formal software engineering techniques to smart grids. In: 2012 international workshop on software engineering for the smart grid (SE4SG). IEEE, New York, pp 50–56
Halim A (2018) New hybrid Petri net application for modeling and analyzing complex smart microgrid system. J Eng Appl Sci 13(9):2713–2721
Hall A (1990) Seven myths of formal methods. IEEE Softw 7(5):11–19
Hosseini SS, Agbossou K, Kelouwani S, Cardenas A (2017) Non-intrusive load monitoring through home energy management systems: a comprehensive review. Renew Sustain Energy Rev 79:1266–1274
Hussain A, Niazi M (2014) Toward a formal, visual framework of emergent cognitive development of scholars. Cogn Comput 6(1):113–124
Iantovics B (2013) An agent-based hybrid medical complex system. Int Inf Inst (Tokyo) Inf 16(6):3709
Iantovics LB, Rotar C, Niazi MA (2018) MetrIntPair—a novel accurate metric for the comparison of two cooperative multiagent systems intelligence based on paired intelligence measurements. Int J Intell Syst 33(3):463–486
Jiang Z, Khalgui M, Al-Ahmari A, Li Z, Wu N, Zhou M (2018) Automatic supervisory control for the self-healing of smart grids based on colored Petri nets. IEEJ Trans Electr Electron Eng. https://doi.org/10.1002/tee.22726
Kolen S, Dähling S, Isermann T, Monti A (2018) Enabling the analysis of emergent behavior in future electrical distribution systems using agent-based modeling and simulation. Complexity. https://doi.org/10.1155/2018/3469325
Loia V, Tomasiello S, Vaccaro A (2017) Using fuzzy transform in multi-agent based monitoring of smart grids. Inf Sci 388:209–224
Mhadhbi Z, Zairi S, Gueguen C, Zouari B (2018) Validation of a distributed energy management approach for smart grid based on a generic colored Petri nets model. J Clean Energy Technol. https://doi.org/10.18178/jocet.2018.6
Milanovic JV, Zhu W (2017) Modelling of interconnected critical infrastructure systems using complex network theory. IEEE Trans Smart Grid. https://doi.org/10.1109/TSG.2017.2665646
Monti A, Ponci F (2010) Power grids of the future: Why smart means complex. In: Complexity in engineering, 2010. COMPENG'10. IEEE, New york, pp 7–11
Neureiter C, Eibl G, Engel D, Schlegel S, Uslar M (2016) A concept for engineering smart grid security requirements based on SGAM models. Comput Sci Res Dev 31(1–2):65–71
Niazi MA, Hussain A (2011a) A novel agent-based simulation framework for sensing in complex adaptive environments. IEEE Sens J 11(2):404–412
Niazi M, Hussain A (2011b) Agent-based computing from multi-agent systems to agent-based models: a visual survey. Scientometrics 89(2):479

Pagani GA, Aiello M (2014) Power grid complex network evolutions for the smart grid. Phys A Stat Mech Appl 396:248–266
Park L, Jang Y, Cho S, Kim J (2017) Residential demand response for renewable energy resources in smart grid systems. IEEE Trans Ind Inform. https://doi.org/10.1109/TII.2017.2704282
Pózna A, Fodor A, Gerzson M, Hangos K (2018) Colored Petri net model of electrical networks for diagnostic purposes. IFAC-PapersOnLine 51(2):260–265
Rohjans S, Lehnhoff S, Schutte S, Andren F, Strasser T (2014) Requirements for smart grid simulation tools. In: 2014 IEEE 23rd international symposium on industrial electronics (ISIE). IEEE, New York, pp 1730–1736
Siddiqa A, Niazi M (2013) A novel formal agent-based simulation modeling framework of an aids complex adaptive system. Int J Agent Technol Syst 5(3):33–53
Soares J, Ghazvini MAF, Borges N, Vale Z (2017) A stochastic model for energy resources management considering demand response in smart grids. Electr Power Syst Res 143:599–610
Sultan M, Pir A, Zafar NA (2017) UML based formal model of smart transformer power system. Int J Adv Comput Sci Appl 8(11):304–310
Tokody D, Tor M, Szűcs E, Flammini F, Iantovics LB (2018) On the development of intelligent railway information and safety systems: an overview of current research. Interdisc Descr Complex Syst INDECS 16(1):176–185
Turner GK (2014) A formalized method for state machine software implementation in smart microgrid control systems. The University of Texas at Arlington, Arlington
Wong S, Pinard JP (2017) Opportunities for smart electric thermal storage on electric grids with renewable energy. IEEE Trans Smart Grid 8(2):1014–1022
Woodcock J, Davies J (1996) Using Z: specification, refinement, and proof. Prentice Hall, Englewood Cliffs
Yang Z, Xiang J, Li Y (2017) Distributed consensus based supply-demand balance algorithm for economic dispatch problem in a smart grid with switching graph. IEEE Trans Ind Electron 64(2):1600–1610
Zafar NA, Afzaal H (2017) Formal model of earthquake disaster mitigation and management system. Complex Adapt Syst Model 5(1):10. https://doi.org/10.1186/s40294-017-0049-8