Formal Application Description of Autonomous and Cooperative M2M Application Services

Michael Steinheimer[1,2], Ulrich Trick [1], and Bogdan Ghita[2]
[1] Research Group for Telecommunication Networks, Frankfurt University of Applied Sciences, Frankfurt/M., Germany
[2] Centre for Security, Communications and Network Research, University of Plymouth, Plymouth, UK
Email: {steinheimer, trick}@e-technik.org; bogdan.ghita@plymouth.ac.uk

Abstract —This publication presents a novel concept for designing M2M applications on end user level. A formal description language is introduced that enables formal description of M2M application semantic based on statemachines. The evaluation of several modelling languages for describing statemachine-based application semantic are analysed with the result that UML Statemachine Diagrams form the best fitting approach for the introduced formal description language. The defined concept of behavioural modelling M2M application through end users by means of statemachines forms a generalised, intuitive and platform independent methodology to define the semantic of M2M applications.

Index Terms—Formal description, statemachine, M2M

I. Introduction

End user devices, such as domestic appliances are becoming more and more intelligent. These devices include complex functionality for monitoring and control and many devices are equipped with communication functionality enabling remote device access. M2M systems realise the integration of such intelligent devices and provision of specific M2M applications. Publications, such as [1] or [2] have identified that it would be advantageous if the potential, respectively the resources available in the end user domain, could be made accessible for external entities as a service, so that they can be integrated into external applications or processes. For this, it is necessary to integrate end users actively into the service provision process, by having the option to define M2M applications for their personal environment and additionally having the possibility to make their applications and M2M device resources available to external entities. [1] introduced that the resources, respectively M2M services available in end users domain could be combined in order to realise complex, distributed M2M applications. It has also been identified in [2] that both local and distributed M2M application service provision should be realised without any centralised entity in M2M system architecture in order to avoid dependencies on single stakeholders or platform components.

This publication introduces the methodology to enable end users to design M2M application semantics in a convenient way and introduces a new formal description language describing the application semantic to enable automatic processing and M2M application exchange. Section II introduces the principles of proposed modelling methodology. Section III analyses several formal notations and their applicability in the proposed concept. Section IV presents the approaches for modelling M2M applications using Statecharts and finally section V gives a conclusion of the presented methodologies and approaches.

II. Formal M2M Application Behaviour Modelling by End Users

Integration of end users implies that the end users have the possibility to create M2M applications for their personal environment (e.g. their Smart Home). An intuitive development of M2M applications, as illustrated in Fig. 1, could be realised by a graphical development process and by modelling the behaviour of an M2M application independently from underlying technologies [3], [4].

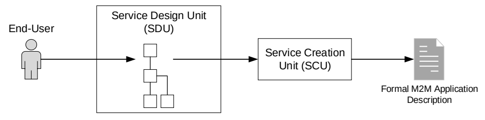
End-User
Service Design Unit (SDU)
Service Creation Unit (SCU)
Formal M2M Application Description

Fig. 1. M2M application development process

It is proposed, that a Service Design Unit (SDU), located in end user's environment, provides the interface for graphically designing M2M applications via a Graphical User Interface (GUI). The definition of an M2M application is done by the end users by modelling the behavior of the M2M application through the combination of graphical building blocks. The graphical building blocks represent the control and monitoring functionalities of M2M devices (available in their personal environment) as well as M2M resources (M2M applications or M2M device functionality) provided as a service by other end users. The definition of M2M application semantic is abstracted from technical realisation meaning the end user defines the application logic graphically and the designed application is automatically transformed into a formal description representing the application semantic [4], [5]. The convenient graphical design of M2M applications enables end users to create M2M applications according their individual requirements and satisfies the requirement of

Manuscript received May 1, 2018; revised December 20, 2018.
doi:10.12720/jcm.14.1.33-39

end user integration into the process of M2M application design.

This project proposes an abstraction mechanism for service executables. The graphically designed M2M application logic is not transformed to application code that is dependent on the execution environment and therefore dependent on the platform. The Service Creation Unit (SCU) (refer to Fig. 1) automatically transforms the application logic into a formal application description using a unified, standardised and machine-readable formal description language describing the application semantic. The formal application description can be parsed, interpreted, and the application logic described can be executed independently of the underlying execution environment. This makes the formally defined M2M application independent of the execution environment and can be easily ported to other execution environments, which meets the requirements of platform independence. Each execution environment that contains a parser for the unified and standardised formal description language is able to execute the application. The machine readability of the formal description language enables the fully automated application execution. The application of unified mechanisms for M2M application description using a standardised formal description language supports the realisation of an application description parser on different platforms.

The proposed system architecture contains a Service Runtime Environment (SRE) responsible for execution of the formally described M2M applications (refer to Fig. 2).

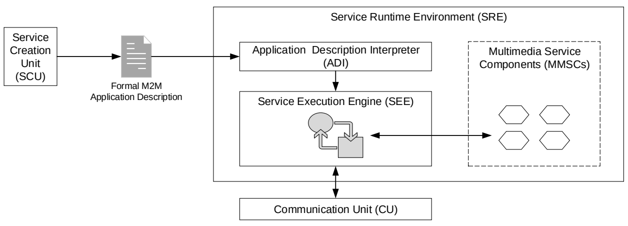
Service Runtime Environment (SRE)
Service Creation Unit (SCU)
Formal M2M Application Description
Application Description Interpreter (ADI)
Multimedia Service Components (MMSCs)
Service Execution Engine (SEE)
Communication Unit (CU)

Fig. 2. Service Runtime Environment (SRE).

The SRE contains an Application Description Interpreter (ADI) and a Service Execution Engine (SEE) for realisation of the M2M application execution. The SRE receives the formal application description from the SCU and executes the defined application logic on behalf of the ADI (parsing and interpreting the formal application description) and the SEE (triggering the defined actions) [2; 5].

This research proposes a methodology for application creation following MDA (Model Driven Approach) principles derived from [6] by defining a platform independent application model [4]. Such an abstract application model is expressed by a modelling language describing the behaviour of an application, separated from the technology-specific realisation of it [6], [7]. The definition of the abstracted model is independent of the realising platform and can therefore be modelled without specific knowledge about the platform that implements the application [8]. The executing system has to interpret the abstract description of the application and convert it into a specific structure appropriate to the executing system. In the concept designed in this project end users graphically create a behaviour model of the M2M applications, which allows them to define the M2M application semantic without having specific knowledge of the executing platform [9].

In order to specify an adequate methodology how end users can intuitively model the behaviour of an M2M application, this project proposes that end users design statemachines describing the activities of an application by connecting building blocks representing M2M devices and M2M application services [10]. According [11] statemachines describe the behaviour of a system that is specified using states. A statemachine describes how a system residing in a specific state acts at specific events. This approach to describe the behaviour of a system has been derived because end users are able to understand the principles behind (i.e. describing sequence of activities) as described subsequently.

Devices have specific functionalities executed when the devices are triggered (e.g. powered on). The functionality of a device is used to perform an activity that corresponds to the functionality of the device. The end user knows in principle how a device works and is familiar with the use case that devices are connected with each other such as illustrated in Fig. 3. It shows that if the end user triggers a device (switch button by pushing), the switch button activates the lighting.

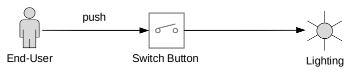

Fig. 3. Use case switch lighting.

Transferring this behaviour to M2M devices/ services they have also inputs from which an activity results generating outputs. At this point, three classes of devices are specified that can be present in the personal environment of an end user:

- Actuators – Actuators control actions depending on the inputs and supply output values if necessary.
- Sensors – Sensors are used to acquire data which they supply as output values, possibly triggered by an input.
- Combined – These devices cannot be assigned directly to one of the other groups, since they offer both, possibility to control and supply of sensor values.

Fig. 4 illustrates the general structure of M2M devices/services defined to specify a general perspective of them.

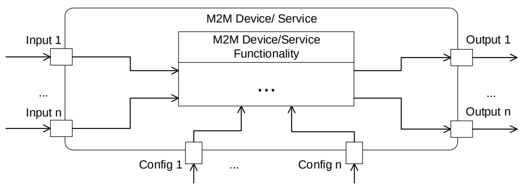

Fig. 4. General structure of M2M device and services

From a general perspective, end users can define M2M applications by connecting the Output parameter of one M2M device/service with the Input parameter of another M2M Device/service. In general, the behaviour of an application can be defined as devices/services that are in a particular state and depending on the input data they generate an output, which in turn is passed on to another device or service. A deterministic Finite State Machine (FSM) specifying this behaviour is used in the concept designed in this project. To model an application consisting of linked M2M devices and services with a FSM, the components of M2M applications are mapped to the elements of the FSM as specified in Table I.

TABLE I: M2M APPLICATION COMPONENT FSM ELEMENT MAPPING

M2M Application Component	FSM Element	Description
M2M Device/Service Component	State	M2M Device/Service components are assigned to the states of a FSM representing them inside the FSM.
M2M Device/Service Component Connection	Transition	Connections between the M2M Devices and services are assigned to transitions connecting the states of a FSM and representing the information flow between M2M Devices/ Services inside the FSM.

III. FORMAL M2M APPLICATION NOTATION

The previous section introduced the methodology how end users can model M2M applications in the form of a statemachine. This section determines a concept for statemachine-based modelling and describes how a specific M2M application can be modelled and formally described using Statecharts.

Describing an M2M application with a FSM allows to map M2M devices/services to the states of an FSM and to map the connections between them to state transitions. To identify an appropriate modelling language enabling the formal description of the graphically modelled application using statemachine concept, the following first defines requirements according to [9] for selecting an optimal modelling language for this purpose.

- Standardised language – It should be a standardised modelling language to ensure portability.
- M2M Device/Service Mapping – The modelling language must provide elements that enable representation of M2M devices/Services and connection between them. The connections between the M2M devices/services, i.e. information flow between them or activation should be equipped with a condition.
- Intuitive usability – Since the system is to be used by an average technically experienced end user after a short training, the complexity of the graphical notation should be low. Complex and non-intuitive forms of modelling reduce or eliminate usability.
- Parallel flows – Within an application, it should be possible to define parallel sequences to realise concurrent tasks.
- Synchronisation of states – To synchronise parallel flows there must be a synchronisation possibility.
- Machine readability – Since the M2M application should be generated automatically after graphical modelling and automatically processed by the M2M platform, the modelling language should be a formal language which is machine-readable.
- State parametrisation–The elements to be combined should be able to be parameterised (definition of input/output/configuration parameters). Therefore, the states in the statemachine must also be parameterisable.
- Existing parser/interpreter implementation – To be able to process the formal language automatically, an implementation of a corresponding parser or interpreter should exist.
- Domain independent–The modelling language cannot be a domain specific language to prevent limiting the scope to the specific domain.

An application according [12] can be modelled with three different views onto the system:

Functional View – The Functional View specifies the processes, activities, and functions of a system. It includes inputs and outputs of activities, which is the information flow between the internal and external activities. The Functional View of a system is described using Activity Diagrams as Modelling Language [12]. An Activity Diagram describes complex processes, focuses on the task of the system that has to be divided into single steps, and answers the question “how a system realises a specific behaviour” [11].

Behavioural View – The Behavioural View specifies the behaviour of a system. It specifies how a system acts on defined conditions and when the activities defined in the Functional View become active and when the information flow between the activities take place. The Behavioural View of a system is described using State Diagrams as Modelling Language [12]. A State Diagram describes the behaviour of a system using states and transitions between states that are triggered by external or internal events. A state diagram answers the question “how the system behaves when residing in a specific state and a specific event occurs” [11].

Structural View – The Structural View specifies the modules and subsystems “constituting the real system and the communication between them”. The Structural View is considered as the “physical model” of the system describing the specific hardware and software implementations [12].

Since the end user should describe the application in an abstract way and do not need to know any hardware- and software-specific details, the application cannot be modelled according to the Structural View. Because modelling the behaviour of an M2M application is in the focus of the concept described in this research, formal descriptions based on the Behavioural View appear to be a suitable approach. According [11], [13] and [14] statemachines (Behavioural View) and Activity Diagrams (Functional View) are equivalent and can describe both the same behaviour of a system, therefore also modelling languages for process-based modelling are considered in the evaluation for the optimal modelling language.

Based on the above specified requirements, the potential candidates Business Process Model and Notation (BPMN) [15], UML Activity Diagram (UML AD) [16], UML StateMachine Diagram (UML SMD) [15] have been analysed with regard to the defined requirements. The first two modelling languages represent candidates for Functional View (describing process-oriented modelling, workflows). The last modelling language represents a candidate for Behavioural View (describing behaviour-oriented modelling). All of these modelling languages specify a set of elements that can be combined graphically to describe a process or the behaviour of an application. Table II summarises the evaluation results of the modelling language candidates and compares it with the general FSM concept.

TABLE II: M2M APPLICATION COMPONENT FSM ELEMENT MAPPING

Requirements	Modelling Language			
	Functional View		Behavioural View	
	BPMN	*UML AD*	*UML SMD*	*FSM*
Standardised Language	+	+	+	-
M2M Device/MMSC Mapping	+	+	+	+
Intuitive Usability	o	o	+	+
Parallel Flows	+	+	+	-
Synchronisation of States	+	+	+	-
Machine Readability	+	-	+	-
State Parametrisation	+	+	+	-
Existing Parser/ Interpreter	+	-	+	-
Domain independent	o	+	+	+
Requirements Evaluation: += satisfied; o= partially satisfied; -=not satisfied				

The result of the evaluation is that UML SMDs fully satisfy the requirements. SCXML (Statechart XML) enables describing UML SMDs via XML as a standardised description language and is therefore proposed as basis for describing M2M application behavior via Statecharts. BPMN and UML AD represent both very extensive modelling languages allowing a fine granular and detailed description of processes, whereby BPMN serves more for the description of business processes. BMPN also fulfils almost all requirements for the formal description language. However, the fact that WSBPEL is limited to Web service environments prevents WSBPEL from being used for the application description in a RESTful M2M environment.

IV. M2M APPLICATION MODELLING USING STATECHARTS

To determine how a modelled M2M application can be formally described by means of Statecharts, it is necessary to define a general structure for an M2M application and to transfer this structure to the modelling principles using Statecharts. For this purpose, the structure of a Statechart was derived and transferred to the structure of an M2M application. Thus, an M2M application can consist of different M2M devices/services connected sequentially (OR-connections), or in parallel sequences (AND-connections). Through the generally defined, reusable structure of an M2M application description by means of SCXML, a possibility is created to formally describe M2M applications according to a unified pattern. This generally defined structure serves for the formal M2M application description process (described subsequently) to transform the graphically modelled application into a formal description.

To automate the formal description of the M2M application, the following section presents the process generating a formal description from the graphically modelled application. The Service Creation Unit (SCU) illustrated in Fig. 1 performs this process receiving as input the graphical notation of the application description and generates the formal notation from it.

To define an adequate algorithm performing the generation of the formal description, a General M2M Application Model (see Fig. 5) and a General State Model (see Fig. 6) have been derived from the above-described principles of M2M application modelling by means of Statecharts.

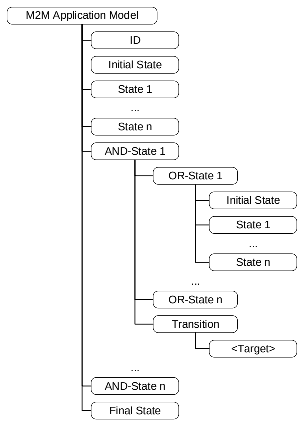

Fig. 5. General M2M application model

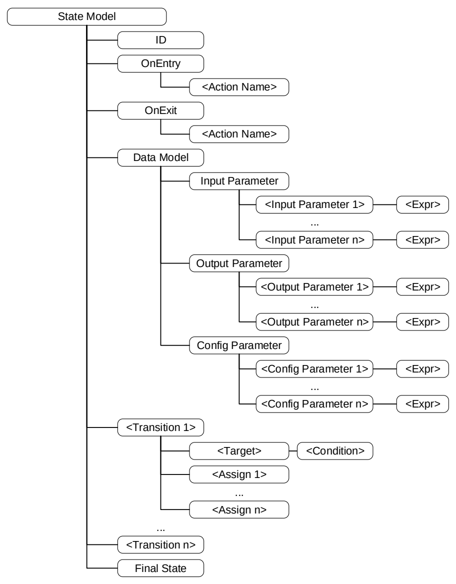

Fig. 6. General state model

The General M2M Application Model describes the generalised structure of an M2M application. This defines that M2M applications always have an ID uniquely identifying them. Furthermore, the entry point must be defined for each M2M application realised by the Initial State definition. An M2M application also requires a defined exit point marking the end of the application realised by the Final State element. Additionally to these mandatory elements, an M2M applications have states as optional elements. These can be simple states not containing other states or complex states describing parallel sequences in M2M applications. Parallel sequences are modelled using AND-States, which have individual sections executed in parallel. For each of the parallel sections, it is necessary to define an OR-State that includes the states to be executed in parallel. For each OR-State, it must also be defined which of the contained Simple States is the entry point (defined as Initial State). Additionally AND-States have a transition to integrate them into an application flow.

The General State Model describes the generalised structure of a simple state. They have an ID to uniquely identify the state or the M2M device. Via OnEntry and OnExit elements of a state, an action can be defined by name that is executed when the state is entered or exited. Furthermore, a state has a Data Model (input/output/config parameters) that specifies the interfaces to an M2M component.

A value can be assigned to each Input/Output/Config parameter using the <Expr> element of the parameter. Connections between M2M components are represented by the Transition elements of a state. The Transition elements are used to specify which state is the target of the transition defined by the <target> element. For each Transition, a condition can be defined specifying when the transition is executed. The condition is defined by the <Condition> element of the transition. If value assignments should be made to the M2M Components during a transition, this is specified via the <Assign> elements of the Transitions. A state can be declared as a Final State. If so, the state contains a Final State element in the specification.

The SCU uses the General M2M Application Model and the General State Model to generate the formal M2M application description applying the algorithm described in Fig. 7. First, the SCXML frame is generated based on the General M2M Application Model. Where the name of the modelled application is specified as the ApplicationID. Furthermore, the start element is defined as initial state. Afterwards, all graphically modelled states are captured. Each of these states is analysed and inserted into the SCXML frame based on the result of the analysis. It is first checked whether it is a simple state or a complex state (parallel flow). If it is a simple state, a state element is generated based on the General State Model. Fig. 8 shows the mapping of the elements from graphical notation into the formal description of the state according to [4]. If it is a parallel sequence element, a parallel state element is first created in SCXML. All parallel sections are then captured. For each parallel section, an OR-State element is created and all states that are contained within the parallel section are captured. The collected states are simple states that are analysed as described above. After the formal definition of the state is done, the state is inserted into the OR-State element. After all states are captured and added to the OR-State element, the OR-State element is added to the previously generated parallel state element. As soon as all parallel sections have been processed, the parallel state element is added to the SCXML frame.

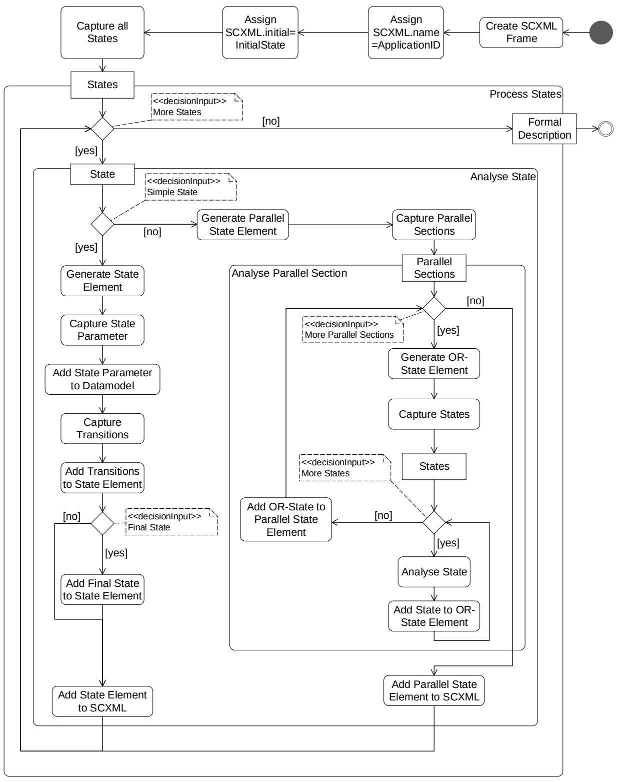
Create SCXML Frame
Assign SCXML.name =ApplicationID
Assign SCXML.initial= InitialState
Capture all States
States
Process States
<<decisionInput>> More States
[no]
Formal Description
[yes]
State
<<decisionInput>> Simple State
Analyse State
[no]
Generate Parallel State Element
Capture Parallel Sections
[yes]
Generate State Element
Parallel Sections
Analyse Parallel Section
[no]
<<decisionInput>> More Parallel Sections
[yes]
Capture State Parameter
Generate OR-State Element
Add State Parameter to Datamodel
Capture States
Capture Transitions
<<decisionInput>> More States
States
Add Transitions to State Element
Add OR-State to Parallel State Element
[no]
[no]
<<decisionInput>> Final State
[yes]
[yes]
Analyse State
Add Final State to State Element
Add State to OR-State Element
Add State Element to SCXML
Add Parallel State Element to SCXML

Fig. 7. Formal M2M application description generation process

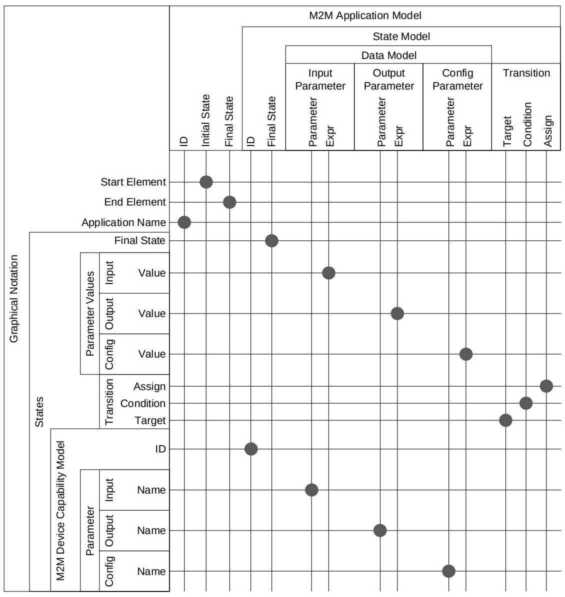

Fig. 8. Assignment graphical Notation Elements to M2M Application Model.

V. Conclusion

This publication introduced the principles of M2M application definition by means of modelling the behaviour of the M2M application. It has been illustrated that the behaviour of an M2M application can be modelled by defining statemachine-based application flows. To achieve this a general structure of M2M device/services containing input/ output/ config parameters has been defined and mapped to the states of a statemachine. This enables the representation of an M2M device/service inside the application behaviour model. The connections of M2M devices/services have been mapped to the transitions of the statemachine and representing the connection between them. The defined concept of behavioural modelling by means of statemachines forms a generalised, intuitive and platform independent methodology to define the semantic of M2M applications.

The modelling languages BPMN, UML ADs, and UML SMDs for statemachine-based modelling have been evaluated regarding defined requirements. The result of the evaluation was that UML SMDs satisfy the defined requirements and therefore has been selected as graphical modelling language for M2M applications. The semantics have been defined to model and formally describe M2M applications using Statecharts. For this a general M2M Application Model and a General State Model were defined enabling the formal description of M2M applications using SCXML. For automatically generating the formal M2M application description, a process has been defined that uses the General M2M Application Model and General State Model to create the SCXML description out of the graphical notation of the M2M application.

The concept includes a loose coupling of application description and application execution. The presented concept uses standardised notation for application modelling (UML SMDs/Statecharts) and SCXML as standardised formal description language for application description. It is platform independent and adaptable because M2M application logic is defined at a higher level than realisation by means of specific programming languages and compilation of application executables. By using a modelling and description of applications independent of the target language, the approach is independent of the technical realisation/programming. This makes it portable to other M2M platforms and does not require reimplementation or recompilation of application/service logic. The advantage of using a formal description that is based on an official standard is that there is a loose coupling between the application generation tool and the executing environment. Therefore, another GUI or approach for application definition, such as plain text-based application design, could replace the GUI.

Acknowledgment

The research project P2P4M2M providing the basis for this publication is partially funded by the Federal Ministry of Education and Research (BMBF) of the Federal Republic of Germany under grant number 03FH022IX5. The authors of this publication are in charge of its content.

References

[1] M. Steinheimer, U. Trick, W. Fuhrmann, B. Ghita, "P2P-based community concept for M2M Applications", Proceedings of the *Second International Conference on Future Generation Communication Technologies (FGCT 2013)*, pp. 114-119, November 2013, London, UK, IEEE.

[2] M. Steinheimer, U. Trick, W. Fuhrmann, B. Ghita, "P2P based service provisioning in M2M networks", *Second Spanish-Geman Symposium on Applied Computer Science (SGSOACS)*, Invited Talk, December 2015, Frankfurt, Germany.

[3] M. Steinheimer, U. Trick, and P. Ruhrig, "Energy communities in Smart Markets for optimisation of peer-to-peer interconnected Smart Homes", Proceedings of the *2012 8th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP)*, pp. 1-6, July 2012, Poznan, Poland, IEEE.

[4] M. Steinheimer, U. Trick, W. Fuhrmann, B. Ghita, "Decentralised System Architecture for autonomous and cooperative M2M Application Service Provision", Proceedings of the *2017 IEEE International Conference on Smart Grid and Smart Cities (ICSGSC 2017)*, pp. 312-317, July 2017, Singapore, IEEE.

[5] M. Steinheimer, U. Trick, P. Ruhrig, P. Wacht, R. Tönjes, M. Fischer, D. Hölker, "SIP-basierte P2P-Vernetzung in einer Energie-Community" (translated title: "SIP-based P2P Networking inside an Energy-Community"), Proceedings of the *Eighteenth VDE/ITG Mobilfunktagung*, pp. 64-70, Mai 2013, Osnabrück, Germany, VDE.

[6] Object Management Group, OMG, "OMG Model Driven Architecture", Available at: http://www.omg.org/mda/[accessed 25th January 2017], OMG.

[7] Object Management Group, OMG, "Model Driven Architecture (MDA) MDA Guide rev. 2.0", Version 2.0, Boston, USA, June 2014, OMG.

[8] R. Petrasch, O. Meimberg, "Model Driven Architecture", dpunkt.verlag, Heidelberg, Germany, ISBN: 978-3-89864-343-6, 2006.

[9] M. Steinheimer, U. Trick, W. Fuhrmann, B. Ghita, G. Frick, "M2M Application Service Provision: An autonomous and decentralised Approach", *Journal of Communications*, Vol. 12, no. 9, pp. 489-498, 2017.

[10] M. Steinheimer, U. Trick, W. Fuhrmann, B. Ghita, "Autonomous decentralised M2M Application Service Provision", Proceedings of the *Seventh International Conference on Internet Technologies and Applications (ITA 17)*, pp. 18-23, September 2017, Wrexham, UK, IEEE.

[11] C. Rupp, S. Queins, B. Zengler, "UML Glasklar – Praxiswissen für die UML-Modellierung (3rd edition)" (translated title: "UML Crystal Clear – Practical Knowledge for UML Modelling (3rd edition)"), Hanser, Munich, Germany, ISBN 978-3-446-41118-0, 2007.

[12] D. Harel and M. Politi, “Modeling Reactive Systems with Statecharts: The Statemate Approach”, McGraw-Hill, New York, USA, ISBN: 0-07-026205-5.
[13] ISO/IEC 19514, International Standard, “Information Technology – Object Management Group Systems Modeling Language (OMG SysML), First Edition, ISO/IEC.
[14] Object Management Group, OMG, “Information Technology – Object Management Group Systems Modeling Language (OMG SysML)”, Boston, USA, Mai 2017, OMG.
[15] Object Management Group, OMG, “Business Process Model and Notation (BPMN)”, Version 2.0, Boston, USA, January 2011, OMG.
[16] Object Management Group, OMG, “OMG Unified Modeling Language (OMG UML)”, Version 2.5, Boston, USA, March 2015, OMG.

Michael Steinheimer received the B.Sc. degree from the University of Applied Sciences Darmstadt, Germany, in 2008 and the M.Sc. degree from the University of Applied Sciences Darmstadt, Germany, in 2011, both in computer science. He is currently pursuing the Ph.D. degree with the Centre for Security, Communications and Network Research, University of Plymouth, UK. His research interests include decentralised service provision, Machine-to-Machine Communications, and Peer-to-Peer Networks.

Ulrich Trick received the Dipl. Ing. degree from the University of Kaiserslautern, Germany, in 1983 in Electrical Engineering - telecommunications. He received his Doctoral degree from the University of Kaiserslautern, Germany, in 1987. Since 2001 he is Professor for Telecommunication Networks with the Department for Computer Science and Engineering at Frankfurt University of Applied Sciences, Germany. His research interests include NGN, M2M, IoT, P2P and virtualisation.

Bogdan Ghita received the Dipl. Eng. from Politehnica University of Bucharest, Romania, in 1998 and his PhD from Plymouth University, UK, in 2005. He is Associate Professor at Plymouth University and leads the networking area within the Centre for Security, Communications, and Network research. His research interests include computer networking and security, focusing on the areas of network performance modelling and optimisation, wireless and mobile networking, and network security. He has been principal investigator in several industry-led, national, and EU research projects. He is the chair of the International Networking Conference series.