Electronic Communications of the EASST
Volume 51 (2012)

Proceedings of the 5th International Workshop on Petri Nets, Graph Transformation and other Concurrency Formalisms (PNGT 2012)

On Modelling Communication in Ubiquitous Computing Systems using Algebraic Higher Order Nets

Susann Gottmann, Nico Nachtigall, Kathrin Hoffmann

12 pages

Guest Editors: Julia Padberg, Kathrin Hoffmann
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

On Modelling Communication in Ubiquitous Computing Systems using Algebraic Higher Order Nets

Susann Gottmann[1], **Nico Nachtigall**[2], **Kathrin Hoffmann**[3]

[1] susann.gottmann@uni.lu
[2] nico.nachtigall@uni.lu
Interdisciplinary Centre for Security, Reliability and Trust
Université du Luxembourg, Luxembourg

[3] kathrin.hoffmann@haw-hamburg.de
Hochschule für Angewandte Wissenschaften Hamburg, Germany

Abstract: Ubiquitous computing systems (UCSs) are designed to participate almost imperceptibly in everyday life. To ensure a solid operation, a UCS heavily depends on a reliable and efficient communication between its distributed computing components. Moreover components can join and leave the system at any time. In order to guarantee high quality systems, the use of models is inevitable especially at an early stage of the development process where models are the only possibility to address a system which does not yet exist in reality. Petri nets and graph transformation systems are established, theoretically well-founded concepts for modelling and analysing complex systems.

This paper presents a formal approach for modelling core aspects of the communication in UCSs by using Algebraic Higher Order Nets with Individual Tokens and graph transformation. The approach is suitable to cover the different aspects of communication and enables the analysis of specific properties. The approach and its suitability are illustrated based on a running example. The feasibility of embedding the approach in a broader context of modelling is demonstrated in applying it to a real world system: the Living Place Hamburg.

Keywords: Ubiquitous Computing Systems, Systems Modelling, Algebraic Higher Order Nets, Petri Nets with Individual Tokens, Graph Transformation

1 Introduction

The continuous development of technical possibilities provides new opportunities to assist people in everyday life. In this regard, a new computational paradigm emerged called ubiquitous computing [Wei99]. Ubiquitous computing systems (UCSs) are designed to participate in everyday life in a nearly imperceptible manner and are applied to various domains, e.g., vehicular traffic control [DV09], assisted learning [JH12] and ambient assisted living [FA12] including home-based healthcare [RY09] and mobile healthcare [CRL12]. A complex UCS is composed of a large number of computing devices which form a dynamically reconfigurable communication network where a device can join and leave the network at any time. In order to ensure a

solid operation, a UCS heavily depends on a reliable and efficient communication between its distributed devices.

Modelling is an inevitable task for the development of high quality UCSs [Mil06] especially at an early stage of development where models are the only possibility to address a system, which does not yet exist in reality. This paper presents a formal approach for modelling core aspects of the communication in UCSs, i.e., synchronous communication through shared channels and asynchronous communication following the publish/subscribe scheme, by using a special type of Petri nets, called Algebraic Higher Order Nets with Individual Tokens (AHOI nets) [MGE+10], and graph transformation [EEPT06]. In considering the fact that UCS devices can join and leave the communication network of a system at any time, we set up the following requirements on communication: (1) communication partners may be unaware of each other so that some concept is needed that links devices for communication without including their identities, (2) one-to-one as well as one-to-many communication is required, (3) asynchronous communication between devices that may not be present at the same time and (4) synchronous communication between devices that are present simultaneously. Synchronous communication through shared channels and asynchronous communication following the publish/subscribe scheme together can satisfy the requirements and are widely used in UCSs. AHOI nets in combination with graph transformation are suitable techniques for an integrative modelling of both communication models. Both techniques are theoretically well defined, which enables the analysis of specific properties concerning the reliability and efficiency of the communication. The feasibility of embedding the approach in a broader context of modelling is demonstrated in applying the approach to the modelling of a real world system: the Living Place Hamburg [LP12].

Sec. 2 explains core aspects of communication in UCSs based on a running example. In Sec. 3, the Living Place and the modelling approach are presented and applied to the running example, Sec. 4 introduces the formal and analysis techniques, Sec. 5 discusses related work and Sec. 6 provides a conclusion concerning the suitability of the approach and future work.

2 Communication and Running Example

Synchronous communication through shared channels and asynchronous communication following the publish/subscribe scheme are the core aspects of communication in UCSs that are considered in this paper. Both communication models support one-to-one as well as one-to-many communication. In short, synchronous communication means that the event of sending a message and the event of receiving that message necessarily occur simultaneously [CMT96], i.e., the send and receive operations may block as long as no device is ready to receive or send. In contrast to that, in asynchronous communication the events of send and receive information occur independently of each other, possibly at different points in time, i.e., without the need of blocking. Consequently, an intermediate component is needed that provides a buffer for storing the messages. Synchronous channel-based one-to-one communication implies that two devices can synchronously communicate through a shared named channel. In publish/subscribe based asynchronous communication, devices can send (publish) information of specific types and can receive information of specific types by announcing their interest in (subscribing to) those types first [EFGK03]. A central component manages the subscriptions and buffers and distributes

the information. Publish/Subscribe communication appears in different variants: topic-based, content-based and type-based (c.f. [EFGK03]). Both communication models provide a concept for linking devices for communication without including their identities, i.e., the concept of a named channel that is used (shared) by several devices to communicate with each other and some concept of topic, content-identifier or type that is used by the devices for publishing and subscribing (compare requirement (1) in Sec. 1). In one-to-one communication a device sends information to exactly one device whereas in one-to-many communication a device sends information to several devices which are interested in receiving that information.

The following running example illustrates a scenario for communication in UCSs. It is set up in the domain of ambient assisted living. We consider three communicating devices: a blood pressure sensor, which asynchronously sends measured values to a logging device that stores all values, so that the doctor in charge can analyse the data at a later moment. In addition, the blood pressure sensor synchronously informs a monitoring device about striking values to alarm the doctor in charge. This second communication is synchronised to ensure the reliable reception of the striking values. In Fig. 1 the communication between all three devices is illustrated. The blood pressure sensor on the first row is sending its data simultaneously or successively to the monitoring device on the second and the logging device on the third row. The vertical arrows for synchronous communication indicate that the events of send and receive data occur simultaneously whereas the arrows for asynchronous communication depict that the events of send and receive may occur at different points in time.

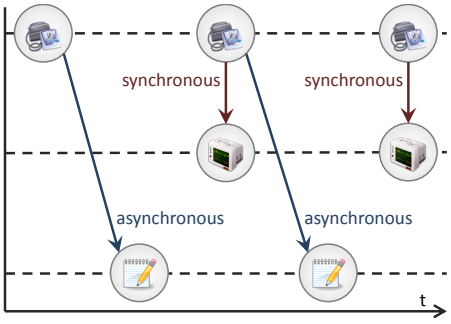

Figure 1: Communication in running example

3 Modelling and Real World Application

In this section, we present a formal modelling approach of synchronous and asynchronous communication that has been embedded in a broader context of modelling, the modelling of the system of the Living Place Hamburg [LP12]. The Living Place is a smart home for ambient living that is developed and constructed by the Hochschule für Angewandte Wissenschaften Hamburg. The apartment is equipped with several sensors and "intelligent" computing devices that together form an UCS enabling the seamless interaction between the inhabitant and the smart home. It is used as laboratory for applied research, like for performing user studies about the behaviour of a person in such an environment. Our approach for modelling communication is contained in the model of the system behaviour of the Living Place case study as developed in [GN12].

For creating a model of the communication in UCS as in the Living Place Hamburg, we decided to use Algebraic High Level Nets with Individual Tokens (AHLI nets) [MGE+10], a special type of extended Petri nets, and Algebraic Higher Order Nets with Individual Tokens (AHOI nets), a restriction of AHLI nets (c.f. Sec. 4). The decision is based on the general requirements for communication in UCS we set up in Sec. 1, but also on the following demands: The model should be able to represent the flow of any kind of data between devices. The behaviour of the

system is dynamic, i.e., components can enter or leave the system at any time. We decided to create a hierarchical model in order to strictly separate especially the communicating devices from the system managing the communication between these objects. The lowest abstraction level in our model is the *data level* representing abstract data types and formalised in signatures and associated algebras.

The device specific processing including communication is modelled by an AHLI object net token for each device with a superior AHOI net for the modelling of the topic-based publish/subscribe communication. All available AHLI net tokens are also called object nets. Here, object nets either represent a device in the UCS or the AHLI net for storing data and subscriptions. Together with all AHLI transformation rules for the reconfiguration of AHLI object nets as tokens, they form the *object level*.

The outermost level is the *system level*. It manages the communication, i.e., the distribution of messages between connected and activated devices. The system level is modelled as AHOI net with AHLI nets and suitable transformation rules as tokens.

In our model, all individual tokens on the system level, i.e. object nets and AHLI transformation rules, seem to be always assigned to the same place, i.e., they are only used for applying AHLI transformation rules. However, this model is embedded into a broader case study of the Living Place presented in [GN12], where the object nets move on the system level, e.g., object nets representing devices can move from a place for online devices to a place for offline devices. The individual tokens of the AHLI nets on the object level are formed by the elements of the carrier sets of an algebra from the data level.

Furthermore, we decided to model transformation rules on AHLI nets as individual tokens assigned to places on the system level. Another possibility is to include AHLI transformation rules on object nets within the signature and algebra and include them in the set of transition conditions in the system net. In [GN12] we defined an additional layer, where rules for the transformation of the AHOI net (system level) are defined to reflect user interactions with the system, like adding, removing or manipulating object nets. E.g., user rules are defined for adding devices to the system by creating tokens assigned to the place *Devices* in Fig. 2. For better transparency and in order to provide the possibility of extending the net so that transformation rules can be added, removed or modified easily by the user, we decided to model transformation rules as individual tokens.

In the following, we introduce the model of asynchronous and synchronous communication in UCSs. We do not mention the corresponding signatures and algebras explicitly in this paper, because of limited space. Similar signatures and algebras are given in [GN12]. We illustrate all rules in compact notation, i.e., parts marked with "−−" will be deleted and "++" will be added, all other parts stay unchanged.

The main place on the system level is *Devices*. Assigned to this place are individual tokens, whose data have the structure of a tuple containing an AHLI net N and a set of transitions. The set of transitions include those transitions out of N that are used for synchronous communication. Each AHLI net within each tuple represents one device connected to the UCS.

On the system level of our model, the transitions *exchange information*, *subscribe topic*, *unsubscribe topic* and *send data to device* (c.f. Fig. 2) contain the same set of equations as transition conditions. Equation $m : Mor$ defines a matching morphism m, whose codomain *cod(m)* is the disjoint union of the input AHLI nets, i.e., for a given morphism $f : A \to B$, *cod(f)* delivers B.

The equation *applicable(r,m) = tt* checks, if the input rule r can be applied on the input net n via match m; *transform(r,m)* applies rule r and returns a disjoint union of two nets. To split the disjoint union in order to return each (maybe modified) net back to its previous place, we use the operation $isomorphic_{PTNET}$, which gets the original AHLI net and the modified AHLI net as input values. The operation $isomorphic_{PTNET}$ flattens both nets to the structure of PTNets, i.e., it forgets the whole algebraic elements and also all individual tokens. Then, the structure of both inputs can be compared, because our model does not modify the internal structure of an object net, instead only the marking of the net is changed.

The transition *receive data from device* contains all previously mentioned equations extended by *createRule*, because this transition gets an interaction scheme as input and needs to generate an amalgamated rule, a parallel rule with maximal matching, first, before applying it.

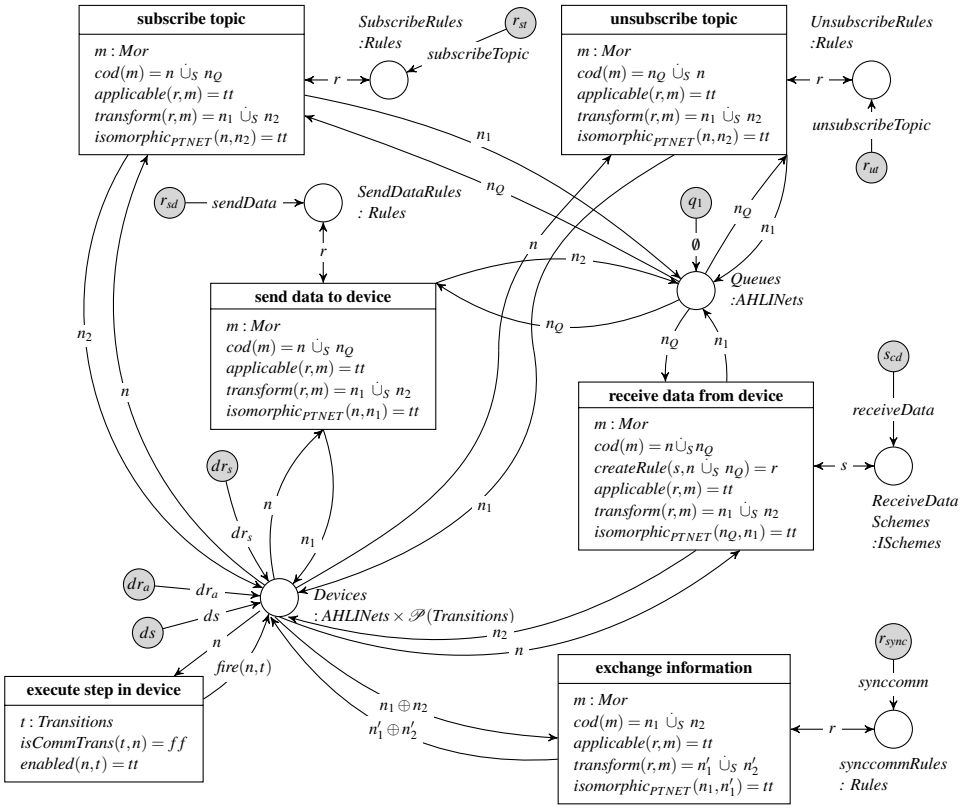

Figure 2: Communication on the system level

A special transition is *execute step in device* (c.f. Fig. 2), which initiates the firing of a transition within an object net in executing the operation $fire$. In our model, *execute step in device* can only be applied to object nets representing devices in the UCS, because this transition is only connected to the corresponding place *Devices*. To ensure synchronous communication, we would like to prevent the uncontrolled firing of the transitions involved in synchronous communication. For that, we equip each object net assigned to *Devices* with an additional set of transitions. This set contains the transitions involved in the synchronous communication. The condition *isCommTrans(t,n) = ff* checks, if a selected transition *t* (*t:Transitions*) is not part of the

corresponding set of the transitions and *enabled(n,t)* checks, if t is enabled. Then, *execute step in device* can execute and *fire* is applied to the selected object net.

In our modelling we use different data types, like *InputData* and *Data* (c.f. Fig. 5), but they have the same elements within the algebra. We define operations on the object level (d_{rcv}, t_{indata}) to cast from one data type to the other. We need to distinguish between these data types in order to keep our transformation rules simple, and at the same time, ensure the desired match.

Synchronous Communication: The main characteristic of synchronous communication is the simultaneous sending and receiving of a message (c.f. Sec. 2), which is fulfiled by our model, and satisfies requirement (4) (c.f. Sec. 1). The transition *exchange information* (c.f. Fig. 2) is involved in this synchronous communication on the system level. In Fig. 3 the pattern for the modelling of AHLI nets representing the behaviour of a synchronous sender (left) and a synchronous receiver (right) are illustrated. We hide the device specific behaviour within the black-box *Device specific processing*. Within this black-box further transitions, places and tokens need to be modelled. According to our running example in Sec. 2, the synchronous communication takes place between the blood pressure sensor, which is represented by the token ds assigned to *Devices*, and the monitoring device, expressed by dr_s. The net ds includes the pattern of a synchronous sender. The net dr_s includes the pattern of a synchronous receiver. The sender provides a message for a specific receiver, which is indicated by the same channels represented by individual tokens within both devices. In addition, the receiver has to show its readiness for receiving messages (see token assigned to *isReady*). Then, the transition *exchange information* on the system level is able to fire, which uses the rule *synccomm* (c.f. Fig. 4) to check the just mentioned pre-conditions and to copy the message from the sender to the receiver simultaneously. Furthermore, the status indicating the readiness for receiving messages will be removed, because the receiver will reset this flag every time it is ready to perform a new data transfer. The synchronous communication is guaranteed in our model, because the firing behaviour of an object net can be controlled on the system level, i.e., it is possible to ensure the synchronous firing of two transitions.

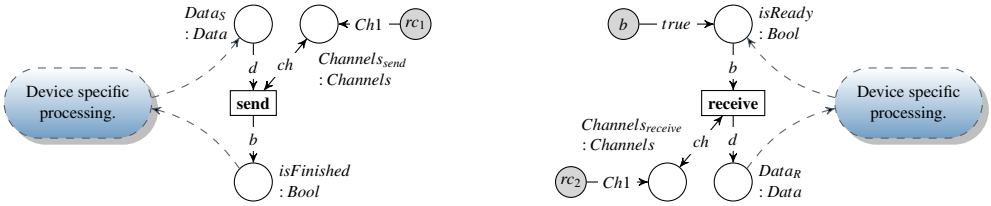

Figure 3: AHLI nets for synchronous devices

Similar to the one-to-one approach, we are able to model the synchronous one-to-many communication. For that, we need a transition similar to *exchange information* with the following changes: Instead of the rule *synccomm*, it uses an interaction scheme adapted to that rule in order to address several devices that are ready to receive at the same time. To create an amalgamated rule out of this interaction scheme, the transition needs to collect all tokens assigned to *Devices*, i.e., the inscription of the corresponding edge needs to be adapted to the number of individual tokens assigned to *Devices*. Furthermore, the set of equations within the transition will be extended by *createRule*. In the one-to-many approach, at least one device, which is ready to receive, will get the message. If several receivers are ready to receive, they will all get the message.

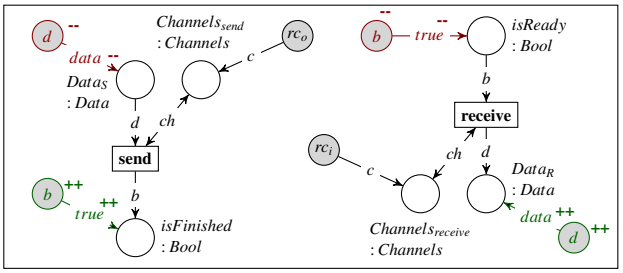

Figure 4: Rule synccomm

Asynchronous Communication: In Sec. 2, we introduced the main characteristics of the asynchronous topic-based publish/subscribe communication, which are observed by our model and fulfil requirement (3). The content- and type-based variants of publish/subscribe can be modelled analogously.

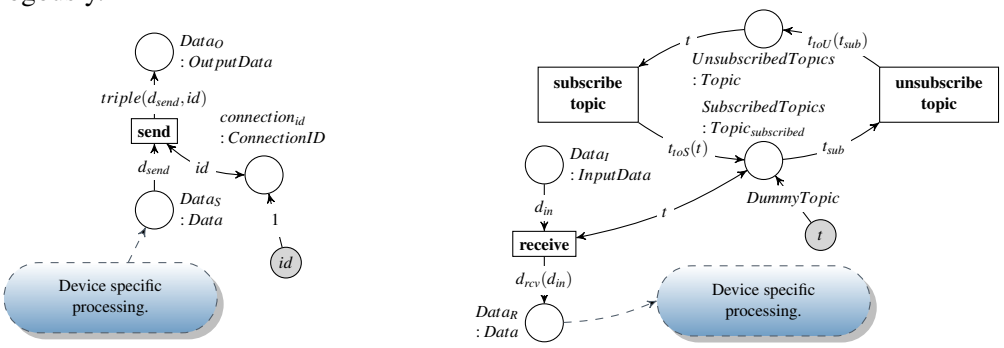

Figure 5: AHLI nets for asynchronous devices

The asynchronous communication described in Sec. 2 is performed between the blood pressure monitor, represented by the device net ds, and the logging device dr_a (c.f. Fig. 2). Fig. 5 presents patterns of asynchronous devices on the object level, which are able to send (left) and receive (right) messages asynchronously. For the asynchronous transmission of a message, different steps are necessary. First, each device, which intends to receive information asynchronously, needs to register for certain topics. The registration will be initiated by the device itself, which assigns the desired topic as token to *SubscribedTopics*, and will be performed on the system level by the transition *subscribe topic* and the rule *subscribeTopic* illustrated in Fig. 6. The system level stores the subscription of all components globally within an AHLI net assigned to the place *Queues*. The transition *unsubscribe topic* and the rule *unsubscribeTopic* (c.f. Fig. 7) are performing the removal of a subscription. The asynchronous transmission of a message is executed in two steps: The process of fetching data from a publisher, whereas the publisher decodes the topic of the message in the individual token assigned to $Data_O$ in its device-specific net. This token is a triple containing the data itself, the topic and the individual connection id of the publisher. To fetch a message from a publisher, the system level uses transition *receive data from device* and rule *receiveData* shown in Fig. 9. The topic and the actual data (with the connection id of the publisher) of the message triple are separated in using operations *topic* and t_{data} and distributed to the global message queues for each corresponding subscriber. At

last, the system level forwards the data from the corresponding global message queue to each subscriber in firing transition *send data to device* and therefore applying rule *sendData* depicted in Fig. 8. Our model allows one-to-one and one-to-many communication for the synchronous and the asynchronous communication. Thus, it satisfies requirement (2).

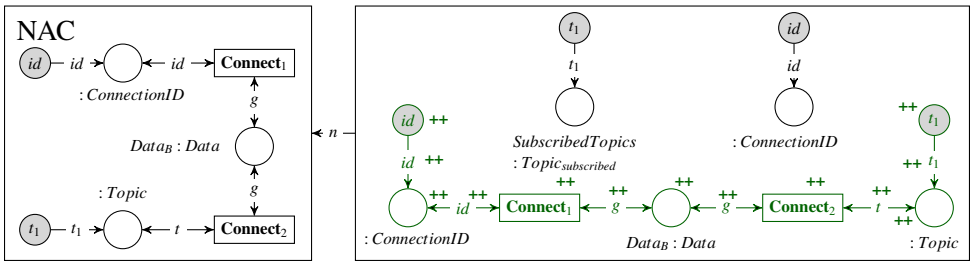

Figure 6: Rule subscribeTopic

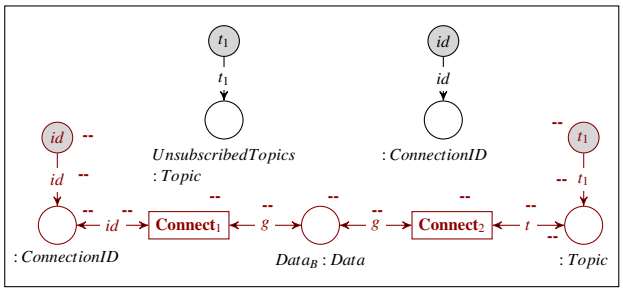

Figure 7: Rule unsubscribeTopic

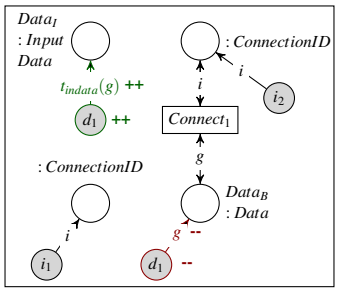

Figure 8: Rule sendData

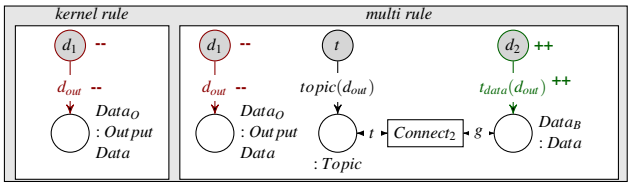

Figure 9: Interaction scheme receiveData

4 Formal Framework

To model the communication in UCSs, we use Algebraic High Level Nets with Individual Tokens (AHLI nets) (c.f. [MGE+10]), a special type of extended Petri nets that help to fulfil the requirements described in Sec. 1 for modelling the communication in UCSs. They enable the modelling on different abstraction layers, they are theoretically well-founded and therefore provide different possibilities for the analysis of the system behaviour. Furthermore, they can be described visually and the firing behaviour of AHLI nets allow the simulation of the model. In the following, we briefly introduce AHLI nets and AHOI nets and create a connection to the model introduced in Sec. 3.

An AHLI net, as defined formally in [MGE+10], consists of a signature Σ including sorts,

operation symbols and variables, and an Σ-algebra. Furthermore, an AHLI net is defined through places typed over the sorts of the signature, transitions, and arcs linking places and transitions. Arcs contain terms of the signature as inscriptions. For each transition a finite set of equations over terms of the signature is defined as firing conditions. AHLI nets also provide a set of individual tokens, which can be infinite, and the individual tokens are assigned to data elements on the places of the AHLI net by a marking function. A transition is enabled, if its set of firing conditions is fulfiled under consideration of the set of selected individual tokens for consuming assigned to the places in the pre-domain and the corresponding terms (at the arcs), which will be evaluated with the token selection. After firing the transition, tokens in the post domain will be produced under consideration of the corresponding terms.

Algebraic Higher Order Nets with Individual Tokens (AHOI nets) are a restriction of AHLI nets. They are restricted to contain only nets and transformation rules as individual tokens, instead of arbitrary data types.

In our model, the system level (c.f. Fig. 2) consists of AHOI nets, because each individual token is either an AHLI net or an AHLI transformation rule. (The place *Devices* is typed over a tuple, where the first element of the tuple is an AHLI net.)

The set of individual tokens form a new level of modelling, the object level. So, in using AHLI nets and AHOI nets, we achieve a hierarchical model. Consequently, we are able to separate the system managing the communication from the devices connected to the system. The system level is able to manipulate object nets in applying given rules on those nets, but, as desired, not vice versa. Furthermore, only the system level is aware of all devices connected to the UCS, but not the devices of each other, because they are kept separately in different individual tokens. Thus, requirement (1) is fulfiled by our model.

Assigned as tokens on the system level are also *AHLI transformation rules*. Each rule consists of a span: $L \xleftarrow{l} I \xrightarrow{r} R$ with L being the left-hand side, I the interface, and R the right-hand side of the rule. The rule can be applied to an AHLI net N if a match from L to N exists, i.e., if the structure given in L can be found in N. In executing the rule, net N is transformed to net N', while the part given in I will be preserved, the part given in L but not in I will be deleted and the part given in R and not in I will be created. AHLI transformation rules can be extended by negative application conditions (NACs). A special type of transformation rules are *amalgamated rules*. They can be intuitively described as one rule consisting of isomorphic parts that can be applied to a set of isomorphic parts in one AHLI net, which are affected by the mapping of the rule. An *interaction scheme* consist of a kernel and a multi rule and serves as scheme for creating a concrete amalgamated rule. An example of an interaction scheme is given in Fig. 9, which is used for copying data from one sending device net to a set of internal global message queues at the same time. In using interaction schemes, e.g., the modelling of one-to-many communication is possible, as it occurs in our topic-based asynchronous communication.

In general, the set of transition conditions on AHLI and AHOI nets differ from transition to transition. In our system level net illustrated in Fig. 2, e.g., the transitions *execute step in device* and *subscribe topic* have a different set of firing conditions.

Several analysis techniques are available for Petri nets and graph transformation systems, e.g., analysis of functional behaviour (local confluence and termination) and conflicts of graph transformation systems [EEPT06] and analysis of conflicts in Petri nets [BM08]. Analysis of conflicts

between Petri nets and graph transformation systems, e.g., conflicts between the firing of transitions and conflicts between the application of graph transformation rules, can be used to ensure specific properties concerning the efficiency of communication. AHOI nets support the analysis of conflicts between the firing of a transition and the application of a rule due to the fact that the firing of a transition can be simulated by a rule application [MGE+10] so that the well known notion of parallel independence of graph transformation rules can be used. Two rules that are applicable to the same graph are parallel independent (can be applied in parallel), if each rule preserves that part of the graph that is used by the other rule, respectively. This technique allows to analyse properties concerning the efficiency of communication, e.g., the analysis of which parts of the specific behaviour of a synchronously communicating UCS device Fig. 3 can proceed in parallel to (which transitions can fire in parallel to) its sending of information (the application of rule in Fig. 4).

5 Related Work

The modelling, simulation and analysis of synchronous and asynchronous communication through shared named channels is extensively studied in process algebras [Bro05, AJS05]. Complex process algebra models seem to be less readable than models based on Petri nets and graph transformation due to their lack of graphical visualisation.

In [CD94] coloured Petri nets are extended by transition inscriptions for synchronous communication through shared named channels. The inscription syntax is similar to the notation that is used in process algebras. This extension allows the definition of synchronously firing transitions where tokens from the pre-domain of one transition are moved to the post-domain of another transition in the firing step. This approach does not provide the same hierarchical modelling capabilities as AHOI nets, i.e., nets and rules as tokens, which makes it difficult to model asynchronous publish/subscribe based communication with the same theory. Like process algebras, the extension only considers one-to-one communication in its basic notation without support for the modelling of one-to-many communication.

Different types of state machines are suitable to model communication. In [Sha92] solutions are given for the modelling of synchronous, asynchronous and one-to-many communication. State machines also provide visualisation, hierarchies and simulation. But in using transformation rules and nets as tokens in AHOI nets, we are able to depict dynamic changes of an UCS, e.g., changes induced by (un)subscribe operations of the publish/subscribe communication scheme. Modelling this aspect is difficult in using state machines.

Algebraic nets with flexible arcs are introduced in [KV98] that provide the possibility to model one-to-many communication. But they seem not to be appropriate, since all device nets with their initial states, all possible proceeding states and combinations must be considered in the definition of algebraic operations for arc inscriptions. This property is a contradiction to the highly dynamic and constantly changing behaviour of UCS. The approach of using AHOI nets with transformation rules that match the corresponding net structures for communication without considering the remaining net parts seems to be more appropriate.

The paradigm *nets as tokens* has been introduced by Valk [Val01] and several extension to generalise hierarchy and support specific application areas are made in the last decade (e.g.

[FK04]). But, in contrast to AHOI nets, objects nets lacks the possibility of reconfiguration to change the net structure as well as the data structure.

In [HM10] an approach similar to the one presented in this paper is given. Synchronous, asynchronous, one-to-one and one-to-many communication is modelled by using AHOI nets but the approach is suited to the scenario of the Skype communication platform and is not intended to cover general aspects of communication.

6 Conclusion

This paper presents an approach for modelling core aspects of the communication in UCSs. AHOI nets and graph transformation are used for modelling asynchronous communication following the publish/subscribe scheme and synchronous communication based on named shared channels. AHOI nets allow the modelling of UCS devices and the exchange of information between devices at different meta levels so that cleaner device models can be obtained that do not contain connecting structures for the exchange. Synchronous, asynchronous, shared channel- and public/subscribe based, one-to-one and one-to-many communication can be modelled and simulated in an integrative manner by using the same theoretical concepts, respectively. This enables the combination of all aspects in one model. Both techniques are visual and theoretically well defined. Several analysis techniques are available and the approach can be embedded in a broader context of modelling. Considering these facts, the approach seems to be suitable for modelling the presented aspects of communication in UCSs. The high complexity and diversity of UCSs make a general modelling approach, which covers all aspects of communication in UCSs, difficult to achieve. However, the presented aspects are applicable to a variety of systems, i.e., the modelling approach not only covers core aspects of communication in UCSs, but may also be applied to systems of other types.

Several tools exist for the modelling, simulation and analysis of Petri nets and graph transformation systems, e.g., the RON editor [BM08] for a subset of Algebraic Higher Order Nets with Collective Tokens called reconfigurable object nets (RONs) and the attributed graph grammar system (AGG) [AGG] and Henshin [Hen12] for graph transformation systems, but none of them support AHOI nets in full. Tool-support for AHOI nets is a necessary task for future work to support the modelling of complex systems. Using AHOI nets as modelling technique is a useful basis for the development of tool support as well as performing formal analysis. But nevertheless, analysis techniques for AHOI nets need to be developed further. The visual notation of AHOI nets is rather complex, so that the development of an intuitive concrete syntax will be an enhancement for developers of UCS, especially in abstracting from technical details, e.g., like creating a special type of transition for transitions with recurring sets of transition conditions, like “fire” transitions or “apply rule” transitions. For the transformation from abstract to concrete syntax, graph transformation can be used (c.f. [Erm06]).

Bibliography

[AGG] AGG, TU Berlin. 2012. http://tfs.cs.tu-berlin.de/agg.

[AJS05] A. E. Abdallah, C. B. Jones, J. W. Sanders. *Communicating Sequential Processes. The First 25 Years*. Springer, 2005.

[BM08] E. Biermann, T. Modica. Independence Analysis of Firing and Rule-based Net Transformations in Reconfigurable Object Nets. In *GT-VMT'08*. Volume 10. Electronic Communications of the European Association of Software Science and Technology, 2008.

[Bro05] S. Brookes. Retracing the Semantics of CSP. In *Communicating Sequential Processes. The First 25 Years*. LNCS 3525, pp. 697–698. Springer, 2005.

[CD94] S. Christensen, N. Damgaard Hansen. Coloured Petri Nets extended with channels for synchronous communication. In *Application and Theory of Petri Nets 1994*. LNCS 815, pp. 159–178. Springer, 1994.

[CMT96] B. Charron-Bost, F. Mattern, G. Tel. Synchronous, asynchronous, and causally ordered communication. *Distributed Computing* 9:173–191, 1996.

[CRL12] J. Caldeira, J. Rodrigues, P. Lorenz. Toward ubiquitous mobility solutions for body sensor networks on healthcare. *Communications Magazine, IEEE* 50(5):108 –115, 2012.

[DV09] N. K. Dave, V. B. Vaghela. Vehicular Traffic Control: A Ubiquitous Computing Approach. In *Contemporary Computing*. Communications in Computer and Information Science 40, pp. 336–348. Springer, 2009.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Volume EATCS Monographs in Theoretical Computer Science. Springer, 2006.

[EFGK03] P. T. Eugster, P. A. Felber, R. Guerraoui, A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.* 35(2):114–131, June 2003.

[Erm06] C. Ermel. *Simulation and Animation of Visual Languages based on Typed Algebraic Graph Transformation*. PhD thesis, Technische Universität Berlin, Fak. IV, Books on Demand, Norderstedt, 2006.

[FA12] J. Favela, X. Alamn. Special theme: ambient assisted living for mobility: safety, well-being and inclusion. *Personal and Ubiquitous Computing*, pp. 1–2, 2012.

[FK04] B. Farwer, M. Köhler. Mobile Object-Net Systems and their Processes. *Fundam. Inform.* 60(1-4):113–129, 2004.

[GN12] S. Gottmann, N. Nachtigall. Modelling the Living Place Project using Algebraic Higher Order Nets. TR 2011/06, Technische Universität Berlin, 2012.

[Hen12] Henshin. 2012. http://www.eclipse.org/modeling/emft/henshin/.

[HM10] K. Hoffmann, T. Modica. Formal Modeling of Communication Platforms using Reconfigurable Algebraic High-Level Nets. *ECEASST* 30:1–25, 2010.

[JH12] H.-Y. Jeong, B.-H. Hong. A practical use of learning system using user preference in ubiquitous computing environment. *Multimedia Tools and Applications*, pp. 1–14, 2012.

[KV98] E. Kindler, H. Vlzer. Flexibility in Algebraic Nets. In *Application and Theory of Petri Nets 1998*. LNCS 1420, pp. 345–364. Springer, 1998.

[LP12] Living Place Hamburg. 2012. http://livingplace.informatik.haw-hamburg.de/blog/.

[MGE+10] T. Modica, K. Gabriel, H. Ehrig, K. Hoffmann, S. Shareef, C. Ermel, U. Golas, F. Hermann, E. Biermann. Low- and High-Level Petri Nets with Individual Tokens. TR 2009/13, Tech-nische Universität Berlin, 2010.

[Mil06] R. Milner. Ubiquitous Computing: Shall we Understand It? *The Computer Journal* 49(4):383–389, 2006.

[RY09] M. Raad, L. Yang. A ubiquitous smart home for elderly. *Information Systems Frontiers* 11:529–536, 2009.

[Sha92] A. C. Shaw. Communicating Real-Time State Machines. *IEEE Trans. Softw. Eng.* 18(9):805–816, Sept. 1992.

[Val01] R. Valk. Concurrency in Communicating Object Petri Nets. In Agha et al. (eds.), *Concurrent Object-Oriented Programming and Petri Nets*. Lecture Notes in Computer Science 2001, pp. 164–195. Springer, 2001.

[Wei99] M. Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.* 3(3):3–11, July 1999.