[SRMM18] V. A. Shekhovtsov, S. Ranasinghe, H. C. Mayr, J. Michael:
Domain Specific Models as System Links.
In: Advances in Conceptual Modeling Workshops (ER'18), Springer International Publishing, Nov. 2018.
www.se-rwth.de/publications/

Domain Specific Models as System Links

Vladimir A. Shekhovtsov[1], Suneth Ranasinghe[1],
Heinrich C. Mayr[1(✉)], and Judith Michael[2]

[1] Alpen-Adria-Universität, Klagenfurt, Austria
{volodymyr.shekhovtsov, suneth.ranasinghe,
heinrich.mayr}@aau.at
[2] Software Engineering, RWTH Aachen University, Aachen, Germany
michael@se-rwth.de

Abstract. Digital Ecosystems consist of a variety of interlinked subsystems. This paper presents a flexible approach to define the links between such subsystems. The idea is to exploit the paradigm of Model Centered Architecture (MCA) and to specify all links/interfaces by means of appropriate Domain Specific Modeling Languages. The approach has been successfully applied and evaluated in several projects. As a proof of concept, we present the model-based interfacing between assistive systems and human activity recognition systems, which showed good performance as needed in real-world applications.

Keywords: Model Centered Architecture · Domain Specific Modeling
Digital ecosystem · Language hierarchies · Interfacing

1 Introduction

From a model centered perspective, any kind of information managed and/or processed by a part of a digital ecosystem (DEC) is an instance of an explicitly specified or implicitly underlying model. The same is true for the processes as well as for the models themselves [6], the latter being instances of metamodels. Consequently, we may see any DEC as a construct consisting of model handlers (consumers and/or producers). This leads to the paradigm of "*Model Centered Architecture (MCA)*" [13], independently from the nature of the handlers that may be digital or living ones.

MCA is a generalization of *Model Driven Architecture (MDA)* and *Model Driven* Software *Development (MDSD*) [11], as well as *models@runtime* [2]. Like multilevel modeling, MCA advocates, for any system aspect, the use of (possibly recursive) hierarchies of *Domain Specific Modeling Languages (DSML)*, each embedded into a *Domain Specific Modeling Method (DSMM)*. I.e., MCA focuses on models (and their metamodels) in any design and development step up to the running system.

This paper deals with an important aspect of MCA, namely the model-based linking of digital ecosystem parts. We illustrate this by an example taken from the domain of supportive systems in Active and Assisted Living (AAL) [22]. Such systems need as complete information as possible about a person's activities (in the sense of sequences of actions) and the context, in which these activities take place. Humans and so-called Human Activity Recognition (HAR) systems may provide such information. However,

© Springer Nature Switzerland AG 2018
C. Woo et al. (Eds.): ER 2018 Workshops, LNCS 11158, pp. 330–340, 2018.
https://doi.org/10.1007/978-3-030-01391-2_37

although the effectiveness of the latter has significantly improved in recent years [20], most of the accessible HAR systems are either still in project state or have a rather restricted sphere of recognition like fall incidents, anomaly behavior detection in crowded places, traffic monitoring etc. I.e., the stand-alone use of such systems does not meet the needs of comprehensive assistive systems.

Combining several (specialized) HAR systems, each of them covering a particular sphere, seems to be promising, as in combination the HARs could deliver the necessary recognition results. Such a combination requires a smooth coupling with the AAL component to manage whatsoever any output structure and integrate new and upcoming HAR systems. We show that the MCA paradigm provides the means for such a flexible, scalable and transparent integration.

The paper's organization is as follows: Sect. 2 outlines the main MCA concepts and discusses the variety of domain specific modeling and representation languages coming into play. Section 3 discusses the general aspects of model-based component linking. In Sect. 4 we present the proof of concept by means of the integration of AAL ecosystem components. Section 5 sketches related work. The paper concludes with an outlook on future research.

2 MCA: Concepts and Language Hierarchies

MCA treats all processes as well as the data they process (MOF[1] level 0) as instances of models (MOF level 1). These models in turn are instances of metamodels (MOF level 2), described using particular DSMLs, and represented using corresponding domain specific representation languages. Consequently, all system interfaces are defined through models as well, and the system components have the role of model handlers. Figure 1 sketches the related MOF hierarchy (for simplicity we omit here the more complex view when using a multilevel modeling approach). Note that in the case of DECs there might be not only various interface metamodels but also several application metamodels, and consequently several DSMLs.

The M2 interfaces support the management of metamodels and the integration of external metamodels (metamodel exchange). On M1 (model level) the M2 metamodels are instantiated for a concrete application situation. On M0 (instance level), the application itself results from creating extensions of the M1 models.

If the application domain metamodels are comprehensive in the sense of providing concepts for structure, dynamics and functionality, the M0 extensions form the models@runtime, which might be handled by an interpreter (orchestrated by M2, visualized by the arrows from M2 to M0). Alternatively, a MDA tool could generate code from the M1 models (static, dynamic and functional).

In parallel to the model hierarchy, we have to provide languages for representing the semantic artifacts (the models) by appropriate syntactic artifacts. These representation languages again form a hierarchy of three levels: (1) *Grammar definition level* (top level): contains the means of defining the language grammars. In our research, we

[1] Meta-Object Facility™ http://www.omg.org/mof/.

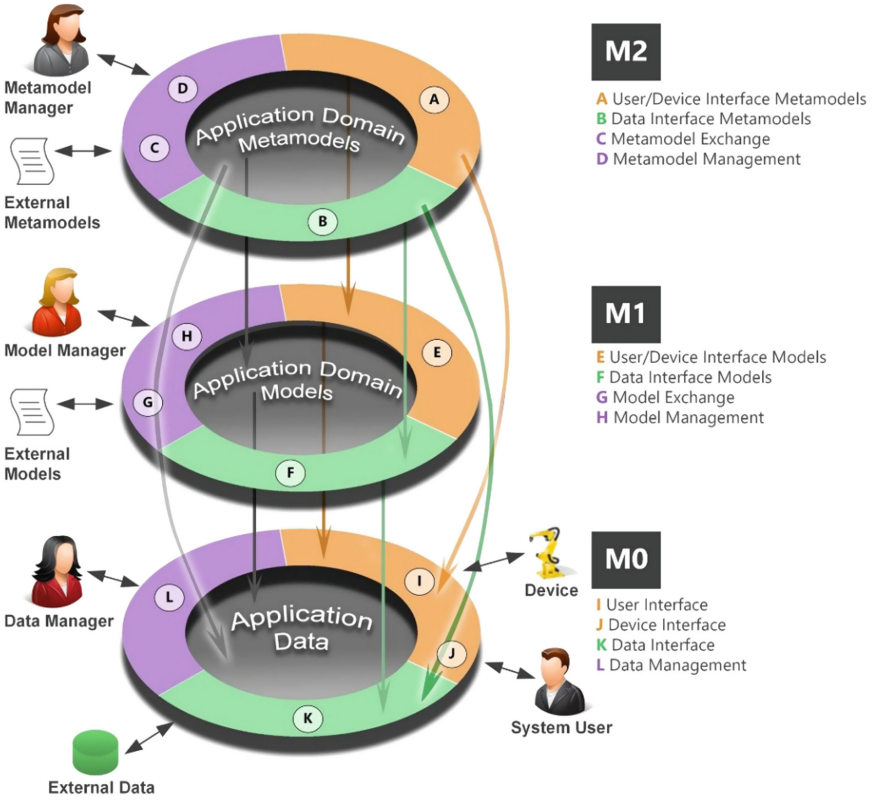

Fig. 1. MCA model hierarchy

use a specific version of EBNF, compatible with the ANTLR grammar definition language. (2) *Language definition level:* defines grammars for the representation languages (RL) related to the defined DSMLs: meta-metamodel RLs, metamodel RLs, model RLs and instance/data RLs. (3) *Language usage level:* representations of the models of all levels.

3 Linking Model-Centered Architectures

MCA proposes the explicit model-based definition for both, user interfaces and interfaces between technical parts of a DEC. This means that appropriate DSMLs and representation languages (mostly subsets of natural language in the case of user interfaces) have to be specified.

There are three general types for linking two systems:

A: Two non-MCA systems ("black boxes"): In this case, no model definitions are known or accessible. Therefore, we have to define (1) a DSML such that the outputs of both systems can be described as instances of models of that DSML, and (2) mappings between the particular models driving the conversion on M0 (if the models are not identical).

B: Black box system and MCA system: If the outputs of the former can be modeled using the DSML of the latter, we could proceed according to A) for the black box system and provide the appropriate model mapping.
C: Two MCA systems: In this case, we only would have to provide the model mapping if the models are not identical.

These types also apply for the other links appearing in a digital ecosystem: E.g., user interfaces and device interfaces are of type B, where the users and devices are seen as black boxes. External data and model interfaces are of type C, as in this case the particular (data) models are expected to be accessible.

The example given in Fig. 2 sketches such links between Systems 1, 2 and 3, all being of type C. The other links (to users and managers) are of type B, as well as the link between System 1 and a device (robot).

Linking may materialize on all MOF levels. We distinguish from top to down:

- M2 level (*metamodel link*): uses the same metamodel but different models. This implies the need of defining appropriate mappings that drive the translation between the particular representation languages. Note that linking based on different metamodels using different DSMLs for one link might be possible, but will not be discussed here.
- M1 level (*model link*): uses the same models.
- M0 level (*data link*): uses the same model instances (possibly represented using different representation languages, which makes conversion necessary).

4 Example: Integrating the Components of an AAL Ecosystem

4.1 Metamodel and Languages

We introduce a metamodel-based HAR (Human Activity Recognition) interface for connecting arbitrary heterogeneous HAR systems in an AAL ecosystem. Such interface has to interpret HAR systems' outputs and deliver appropriately converted data to a given AAL system. Moreover, it has to be adaptable to new HAR semantics in order to be sustainable. As pointed out in the introduction, such 'multi HAR system' approach might increase the recognition realm as needed by an AAL support system.

Our approach is to link a *consumer MCA AAL system* with several *blackbox HAR systems* on M2 (type B). For that purpose, we introduce two link DSMLs that cover the recognition outputs of at least the HAR systems known to us:

- Activity Recognition Environment Modeling Language (AREM-L, metamodel: Fig. 3), a visual conceptual modeling language for describing recognition structures as conceptual models; this language covers both, basic and instrumental (complex) human activities including their context [10, 15].
- Activity Recognition Instance Specification Language (ARIS-L), a textual language for M0 level representations of concrete recognition objects.

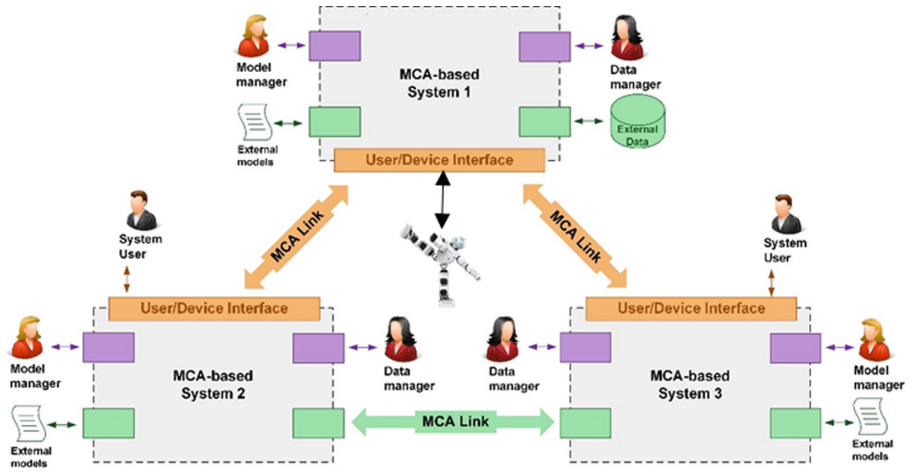

Fig. 2. MCA ecosystem

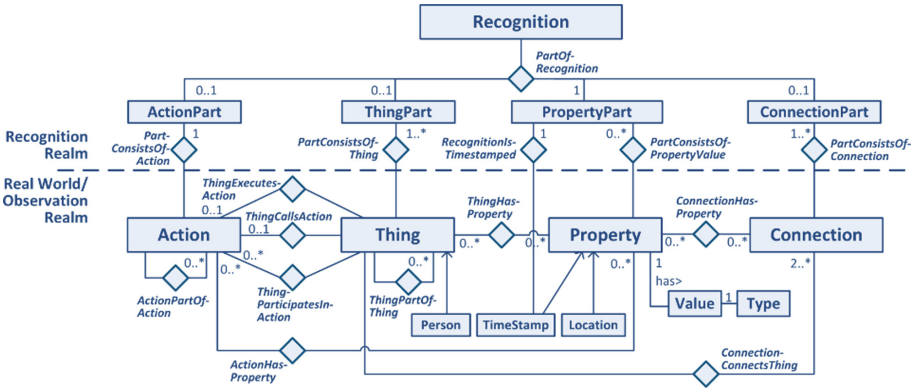

Fig. 3. HAR interface metamodel concepts

The main concept of the AREM-L metamodel is *Recognition* being an aggregate of four sub-concepts: (1) *ActionPart*: contains *Actions* which conceptualize simple or complex activities; (2) *ThingPart*: contains recognized *Things*, i.e. persons or contextual objects; *Person* is defined as a specialization of Thing; (3) *PropertyPart*: contains *Properties*, each having a *Value* of a specific *Type*; (4) *ConnectionPart*: contains *Connections* between Things.

Except from the PropertyPart (containing a mandatory *TimeStamp* Property), the other sub-concepts are optional, i.e., a Recognition may consist of the PropertyPart and none, one, or more of the others.

Things are involved in Actions: they may (1) execute an action, e.g. a coffee machine brews a coffee; this is reflected by the relationship *ThingExecutesAction*, (2) call an action (e.g. a Person pushing the 'brew' button); this is reflected by the relationship *ThingCallsAction*, (3) passively participate in an action (e.g., the coffee in the brewing action); this is reflected by the relationship *ThingParticipatesInAction*.

Actions, Things, and Connections may have Properties. To cover complex situations, Things and Actions can hierarchically be decomposed into structures of Things and Actions, respectively. This is reflected by the corresponding part-of relationships. Location is a specific Property to cover the information necessary for locating Things.

Summing up, an instance of this metamodel, i.e. a model, will describe the concepts underlying the output of a specific HAR system.

AREM-L is a visual modeling language; Fig. 4 shows its basic elements. We waive here a complete specification of the AREM-L syntax, as it should be intuitively understandable based on the example model in Fig. 5a. It represents a type of observation that a HAR system in a smart home environment might make: A person who put objects (e.g. keys) into a container (e.g. a handbag), as step in a "go to shopping" activity.

ARIS-L serves to represent the instances of AREM-L recognition models, i.e., such instances are models of concrete recognitions. Consequently, our goal was to allow for machine-readable representations of recognition data, that are easy to parse and efficient to process. The ARIS-L syntax is inspired by JSON, but includes specific constructs corresponding to the elements of AREM-L. The main rule of ARIS-L is as follows: all data belonging to a specific recognition is encapsulated inside the corresponding rRecognition construct.

Figure 5b shows an ARIS-L fragment for an instance of the recognition model depicted in Fig. 5a. The names of AREM-L model elements precede the representation of their instances' data. These names, in turn, can be preceded optionally by the corresponding concept names (Action, Thing, Property, Connection). Figure 5b illustrates these options by omitting the keyword Thing in the container section, adding this keyword in the object section and providing the keyword Person in the "Observed Person" section. The ConnectionPart shows, as an example, the observed connection "in" between the things "main keys" and "red handbag", which is instance of the model element "in". The links of the ThingPart components to the Action put (i.e., ParticipatesIn, Executes) are instances of the corresponding model element links.

Again, we have to waive presenting the entire grammar.

4.2 Implementation Approach

Our AAL ecosystem consists of (1) a set of HAR systems capturing user's activities (through sensors or by other means), (2) the AAL consumer system providing service to the user based on the captured activities; (3) the HAR interface between them.

According to the classification provided in Sect. 3, all links are of type B, the HAR systems are black boxes and the AAL system is a MCA system. The AREM-L models are used as semantic references for the conversion of HAR output to ARIS-L in the interface, and for parsing ARIS-L in the consumer system. On the data link level, HAR systems' output data is converted into ARIS-L representation.

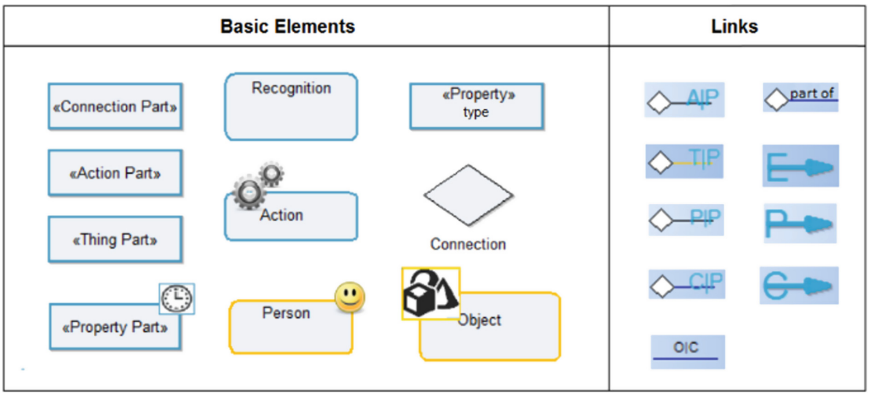

Fig. 4. AREM-L visual modeling elements

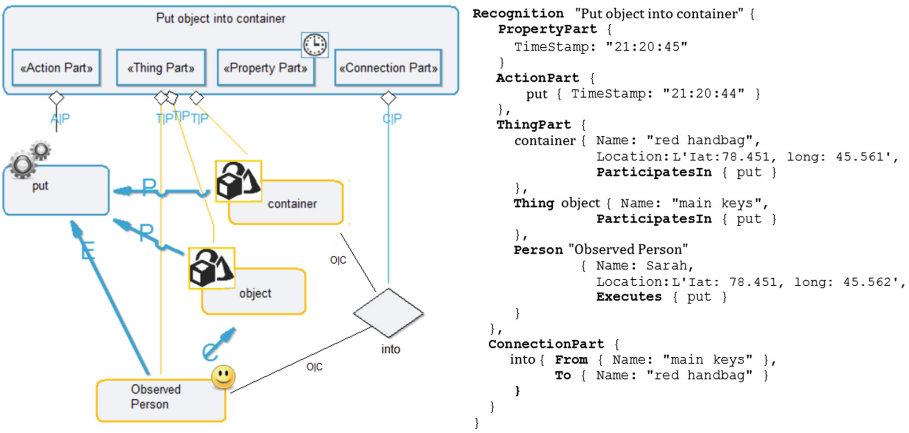

Fig. 5. AREM-L (a) and ARIS-L (b) models for the sample recognition

The steps for implementing the HAR interface were as follows:

- *Implementing AREM-L* by deploying the ADOxx meta modeling framework [7] and for generating the AREM-L modeling tool.
- *Implementing the AREM-L parser,* which serves for converting AREM-L models into the internal HAR interface representation.
- *Implementing ARIS-L* by defining the ARIS-L grammar using the ANTLR framework.
- *Implementing the ARIS-L parser* to be included into the consumer system.
- *Implementing the MCA-Links* (see Fig. 2). They consist of (a) a common core, which uses the AREM-L model and the ARIS-L grammar to convert data into ARIS-L representations to be further processed by the model handlers of the consumer system, and (b) a HAR-specific (black box) plug-in converter for each HAR

to be integrated. Such converter transforms HAR output data for being processible in the common core.

- *Configuring the interface.* Creating AREM-L models, making components accessible to each other via the network etc.

The most technically challenging and time-consuming tasks are implementing the AREM-L and the ARIS-L parser, and the MCA-Link common core. Note, that these components are a kind of standard in the sense that they have to be implemented only once and may be reused for any other HAR and consumer system combination. In contrast to that, the problem-specific tasks (e.g., creating the AREM-L model for the given combination) are straightforward and require less technical knowledge except from the concrete HAR setting. I.e., our approach substantially reduces the effort of linking HAR systems.

4.3 Proof of Concept

For a proof of concept, we used an experimental lab, established in the context of a funded research project. As at that time there was no comprehensive HAR system in product stage available, we developed some experimental systems ourselves, namely (1) a sensor-based system using a Nimbits[2] server as a central point for sensor communication, (2) a video-based HAR (V-HAR) system, which uses a technique from [9] to generate semantic descriptions of video frames in text form, and (3) a HAR simulator for providing environmental data.

As a consumer system, we used HBMS [16], an ambient assistance system supporting daily life activities. The HBMS system abstracts, aggregates and integrates the observed behavior data into an individual Human Cognitive Model (HCM) [12], and assists the supported person via a multimodal interface by retrieving knowledge from his/her HCM. The HBMS input consists of sequences of recognized behavioral actions: in our setup, they are coming from the HAR interface. The "observation engine" of the HBMS system integrates these sequences into HCM as behavioral clusters based on reasoning algorithms that deduce the goal of a sequence of actions.

The MCA-Links to the experimental HARs were developed following the approach described in Sect. 4.2. The resulting ecosystem has been tested against various criteria, involving several user groups in different settings. It showed good performance regarding both, response time and recognition accuracy, as needed in real-world applications.

To sum up, following our approach one has to spend significant effort only for implementing standard components like common core and parser parts. Adding new HAR systems causes no additional effort of this kind: only a simple converter has to be implemented for each system. Providing AREM-L models and configuring the link are high-level activities requiring mainly domain knowledge. Generating the parsers' core based on the language grammar specification by means of the ANTLR environment brings further effort reduction. In contrast to that, following the classical approach would require implementing a separate low-level data converter for each new HAR and

[2] https://www.nimbits.com/.

embedding it into the consumer system. No semantics would then be intrinsically associated with the incoming data and the conversion process.

5 Related Work

Related Work on Model-Based System Linking. The current body of research on model-based system linking deals with applications of model@runtime and MDA paradigms to this problem. In particular, on-the-fly interoperability for the systems based on the models@runtime paradigm is a topic of [1]. [3] provides a solution for automated synthesis of mediators to support component interoperability. An aspect of the model exchange while coupling systems is treated in [8]. An example of using MDA for establishing system links is [5], which introduces the concept of model-driven domain-specific middleware. The difference to our approach is that the above techniques apply models in a limited way (e.g. only at development time for MDA) without providing an integrated model-centered solution.

Related Work on Universal HAR Interfaces. The need of a generic interface to handle heterogeneous data coming from HAR [4] has been discussed since long. [21] presents a survey of ontological approaches to this problem. [17] shows a specific example of using foundational ontologies underlying the middleware for smart homes. Generic knowledge-based approaches are presented in [14], proposing a knowledge-driven framework for context-aware activity recognition, and in [23], introducing an architectural solution for cognitive sensing of activities. Some research deals with the specific categories of inputs, e.g. [18] describes how domain and contextual knowledge can be utilized for human activity recognition in video streams. Despite the above research, a commonly recognized standard or language for HAR interoperability has not been proposed to date [19] and few practical solutions exist.

6 Future Research

There is still a couple of research questions to answer and challenges to meet. For example, a comprehensive semantic grounding of the languages and models requires an appropriate hierarchy of ontologies. For practical purposes in situations where different kinds of components are to be linked in a digital ecosystem, a modeling/development framework would be advantageous, that allows for the management of several meta-models, their instance models and languages, and provides means for an easy and efficient generation of the necessary parsers and converters. In addition to that, integrating the models@runtime approach or MDA code generation would be a further step to a comprehensive MCA development environment.

References

1. Bencomo, N., Bennaceur, A., Grace, P., Blair, G., Issarny, V.: The role of models@ run. time in supporting on-the-fly interoperability. Computing **95**, 167–190 (2013)
2. Bencomo, N., France, R., Cheng, B.H.C., Aßmann, U. (eds.): Models@run.time: Foundations, Applications, and Roadmaps. LNCS, vol. 8378. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08915-7
3. Bennaceur, A., Issarny, V.: Automated synthesis of mediators to support component interoperability. IEEE Trans. Softw. Eng. **41**, 221–240 (2015)
4. Chen, L., Khalil, I.: Activity recognition: approaches, practices and trends. In: Chen, L., Nugent, C., Biswas, J., Hoey, J. (eds.) Activity Recognition in Pervasive Intelligent Environments, pp. 1–31. Springer, Paris (2011). https://doi.org/10.2991/978-94-91216-05-3_1
5. Costa, F.M., Morris, K.A., Kon, F., Clarke, P.J.: Model-driven domain-specific middleware. In: Proceedings of ICDCS 2017, pp. 1961–1971. IEEE (2017)
6. Embley, D.W., Liddle, S.W., Pastor, O.: Conceptual-model programming: a manifesto. In: Embley, D., Thalheim, B. (eds.) Handbook of Conceptual Modeling, pp. 3–16. Springer, Heidelberg (2011)
7. Fill, H.-G., Karagiannis, D.: On the conceptualisation of modelling methods using the ADOxx meta modelling platform. EMISA J. **8**, 4–25 (2013)
8. Götz, S., et al.: Adaptive exchange of distributed partial models@run.time for highly dynamic systems. In: Proceedings of SEAMS 2015, pp. 64–70. IEEE Press (2015)
9. Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 3128–3137 (2015)
10. Kofod-Petersen, A., Cassens, J.: Using activity theory to model context awareness. In: Roth-Berghofer, T.R., Schulz, S., Leake, D.B. (eds.) MRC 2005. LNCS (LNAI), vol. 3946, pp. 1–17. Springer, Heidelberg (2006). https://doi.org/10.1007/11740674_1
11. Liddle, S.W.: Model-driven software development. In: Embley, D.W., Thalheim, B. (eds.) Handbook of Conceptual Modeling, pp. 17–54. Springer, Heidelberg (2011)
12. Mayr, H.C., et al.: HCM-L: domain-specific modeling for active and assisted living. In: Karagiannis, D., Mayr, H., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling, pp. 527–552. Springer, Cham (2016)
13. Mayr, H.C., et al.: Model centered architecture. In: Cabot, J., Gómez, C., Pastor, O., Sancho, M., Teniente, E. (eds.) Conceptual Modeling Perspectives, pp. 85–104. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67271-7_7
14. Meditskos, G., et al.: MetaQ. Perv. Mob. Comput. **25**, 104–124 (2016)
15. Michael, J., Steinberger, C.: Context Modeling for Active Assistance. In: Proceedings of ER Forum 2017. CEUR Workshop Proceedings 1979, CEUR-WS.org, pp. 207–220 (2017)
16. Michael, J., et al.: The HBMS Story. Enterp. Model. Inf. Syst. Arch. **13**, 345–370 (2018)
17. Ni, Q., et al.: A foundational ontology-based model for human activity representation in smart homes. J. Ambient. Intell. Smart Environ. **8**, 47–61 (2016)
18. Onofri, L., et al.: A survey on using domain and contextual knowledge for human activity recognition in video streams. Expert Syst. Appl. **63**, 97–111 (2016)
19. Peláez, M.D., López-Medina, M., Espinilla, M., Medina-Quero, J.: Key factors for innovative developments on health sensor-based system. In: Rojas, I., Ortuño, F. (eds.) IWBBIO 2017,Part II. LNCS, vol. 10209, pp. 665–675. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56154-7_59

20. Ranasinghe, S., et al.: A review on applications of activity recognition systems with regard to performance and evaluation. Int. J. Distrib. Sens. Netw. **12**, 1550147716665520 (2016)
21. Rodríguez, N.D., Cuéllar, M.P., Lilius, J., Calvo-Flores, M.D.: A survey on ontologies for human behavior recognition (CSUR). ACM Comput. Surv. **46**, 43 (2014)
22. Siegel, C., Dorner, T.E.: Information technologies for active and assisted living—Influences to the quality of life of an ageing society. Int. J. of Med. Inform. **100**, 32–45 (2017)
23. Zgheib, R., et al.: A flexible architecture for cognitive sensing of activities in ambient assisted living. In: Proceedings of WETICE 2017, pp. 284–289. IEEE (2017)