Synthesizing Multi-agent System Organization from Engineering Descriptions

Ganesh Ramanathan

Siemens AG and University of St. Gallen, Switzerland
ganesh.ramanathan@siemens.com

Abstract. Automation of electro-mechanical systems, such as the ones deployed in a building or a factory, is engineered based on the *design-time knowledge* of requirements, system configuration, physical processes, and control and coordination strategies. However, any change in these aspects during the system's operation requires manually adapting the affected automation programs. Multi-agent systems (MAS) offer the potential to tackle dynamic changes in the system by letting the software agents autonomously reason about the means of achieving their goals at run-time while collaborating socially and being aware of the environment in which they operate. Nevertheless, designing a MAS-based solution for engineering applications is challenging because decomposing engineering system descriptions into MAS abstractions is a manual process and requires knowledge of the design and programming paradigm. This paper shows that the MAS organization dimension, which serves as the top-down specification of agent behavior, can be automatically decomposed from engineering system descriptions. The system descriptions, which are fragmented, are interlinked using an integration ontology developed for the purpose. Evaluation of the approach in a real-life deployment of a building automation system showed reduced engineering effort to deploy the MAS, and the resulting runtime was adaptive to changes.

Keywords: Multi-agent Systems · Automation Systems · Engineering.

1 Introduction

Electro-mechanical systems, such as the ones in a building or a factory, are complex compositions of subsystems and components that carry out the desired transformation of states of substances through physical processes such as thermal, electrical, or chemical reactions. Since such processes invariably involve some form of controlled energy or mass transfer (e.g., exchange of thermal energy from hot water to air when it comes to heating), automation systems regulate the physical processes and establish the coordinated operation of the interdependent subsystems.

System Descriptions (SDs) is a broad term used in factory, process, and building automation (BA) to describe knowledge that is contained in engineering artifacts, such as the documentation of the requirements, the description of the subsystems and their physical processes, and the regulations and norms that govern their operation. Automation system implementation is generally based on the SDs available at design time. Consequently, they face the challenge that changes in requirements, system capabilities, or regulations and norms during the lifetime of the system require manual (and, often costly) re-engineering of the control and coordination programs [29, 38].

To support use cases such as automated fault detection [27], SDs are becoming increasingly available in machine-readable and machine-understandable forms [25, 31, 28, 29]. Though such machine-understandable SDs could also help us tackle the challenge of adaptivity, it is yet to be seen in the practice in the automation industry because it also requires an architectural paradigm that supports *knowledge-driven run time behavior*. Current approaches in automation only address methods to use system knowledge to design procedures for control and coordination as purely reactive programs.

Rational agents in Multi-agent systems (MAS) are conceptually grounded to use system knowledge at run time to proactively deliberate about local behavior and social collaboration in their pursuit of fulfilling the requirements. The benefit of using MAS to tackle dynamic environments has been demonstrated in domains such as collaborative robotics [6], power engineering [19], and in some particular cases, in factory automation [8], and BA [40].

Although the architectural properties of MAS are well suited to building adaptive systems, the widespread adoption of MAS in engineering applications is yet to be seen [23, 13, 7, 17, 24]. In my study on the feasibility of implementing a MAS-based building automation system, the primary challenge lies in deriving the design from SDs in an *automated* manner. Automated design is vital because relying on a developer to manually carry out the design and maintain it during the system's lifetime is cumbersome (and costly) for real-life applications. Also, a MAS developer cannot be expected to possess the domain expertise of an automation engineer to understand the SDs (and vice-versa, an automation engineer is not likely to be well-versed in the principles of MAS design).

The challenge of automatically synthesizing MAS design from SDs raises the question of the relevant design abstractions. In my approach, I show that the dimension of MAS design that most *naturally* captures automation system design is the notion of the agent organization. The organization is a top-down design specification that mandates the agents to jointly consider (in a global manner) the structural contexts in which they operate (i.e., the parts of the system and the physical processes involved) and thereby adopt appropriate local control and coordination functions.

Though there are existing approaches to express individual aspects of the SDs in a machine-understandable form, an *integrated view* with semantic relationships between the aspects is missing. Such a cohesive SD is essential for the automated synthesis of MAS organization specifications. For this purpose, I have developed an *integrating* ontology that links concepts in the existing engineering ontologies, allowing us to express a *unified* SD.

Therefore, the twofold contributions of this paper are to show that MAS organization specification is an essential top-down design abstraction that is valid for automation systems, and its automated synthesis can be enabled by integrating the hitherto fragmented SDs.

I tested my approach on a real-life setup of automation for heating, ventilation, and lighting systems in a room. This scenario is representative of a complex composition of engineering subsystems and their interdependencies. The evaluation demonstrated that the organization specification, which could be automatically synthesized from the *unified* SD, was adequate for the automation agents to know about their local control goals and global coordination tasks. Changes in the SD were reflected in the organization specification, causing the agents to adapt their behavior.

2 Related Work

2.1 MAS Organization and Its Relevance to Engineering Systems

MAS-based solutions for large systems require *social organization* to direct the local autonomy of the agents towards global goals [16]. Organization specification in the form of structures, roles, and functions also reflects the top-down system design [10]. Organization as means of *orchestrated autonomy* led to research on conceptual models of MAS organization along with its formalization (see [9], and [1] for details). The model of an organization depends on the desired runtime characteristics and formation methods. Horling and Lesser have described paradigms such as hierarchies, holarchies, teams, etc. [14] and their relevance to different systems. In this regard, most automation systems can be viewed as hierarchies of software agents based on a *physical decomposition* [33] of the system. Other paradigms like holarchies, coalitions, teams and markets are also seen in applications like distributed sensing [35]. A model of the organization with hierarchies of groups with one or more functional roles [11] matches well with the design approach in the automation system. MOISE+ [15] goes beyond the focus on structures by adding the abstractions of functional schemes and norms that bind roles to the goals. As we shall see later, the abstraction of functional schemes in MOISE+ supports the modeling of automation strategies, which is also a hierarchical composition of functions.

My investigation of design methodologies in engineering domains showed that top-down physical decomposition involving structural abstractions (systems, subsystems, aggregates, and components) and its co-relation to functional abstractions (composition of process functions) is well known and is practiced in process engineering [21].

Similarly, a study of design practices prevalent in automation engineering [26, 20] revealed that the deployment and behavior of the automation programs are decided based on the structure of the electro-mechanical systems and the process functions which they expected to fulfill. Figure 1 summarizes the parts of the SD and the role they play in the design of the automation system. Juxtaposed to

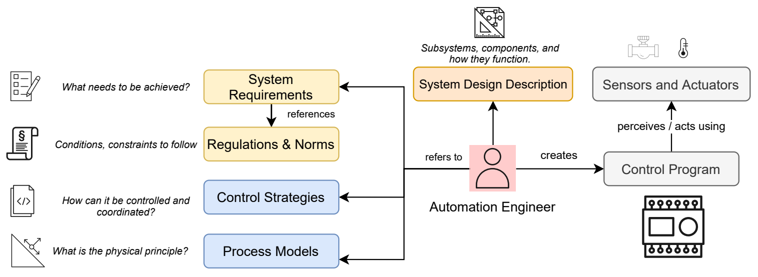
Subsystems, components, and how they function.
System Design Description
Sensors and Actuators
What needs to be achieved?
System Requirements
references
Regulations & Norms
Conditions, constraints to follow
perceives / acts using
refers to
creates
Control Program
Automation Engineer
How can it be controlled and coordinated?
Control Strategies
What is the physical principle?
Process Models

Fig. 1. Aspects of the SDs which are used for automation engineering.

top-down approaches, automation in domains like power engineering [19] (e.g., smart grids) and collaborative robotics (e.g., autonomous ground vehicles) deal with an environment that cannot be determined or designed upfront [6]. In such cases, the autonomous agents primarily rely on self-organization while using pre-defined rules regarding forming groups and adopting roles [18, 30]. While agent-centric architecture can cater to highly dynamic environments, industrial systems lay greater emphasis on having a clear definition and an understanding of the responsibilities of the agents for the sake of operational overview, explainability, and establishing rules for conflict avoidance – and this is the principal argument for an organization-centered design.

2.2 The Challenge of Synthesising MAS Organisation

Though there is a conceptual match between the engineering design of systems and the model of MAS organization, to the best of my knowledge, an automated synthesis of MAS organization specification from SDs is yet to be explored. Bastos and Castro have hinted at the possibility [3], and Freitas [12] has shown the potential of using ontology-based design.

The key aspects in SDs that are essential for decomposing the organization specification of a MAS are the description of the requirements, system design, model of the physical processes, and automation strategies. Engineering ontologies based on Semantic Web technologies have enabled machine-understandable descriptions of these aspects. Methods such as goal-oriented requirements engineering [37], which advocate the formulation of requirements in such a manner that software programs can use them to reason about the goals [4] are being used in practice [34]. Similarly methods to express machine-understandable system design [5] (for e.g., BRICK [2]), physical processes (for e.g., OntoCape [22]), and automation functions [32] are also available being put to use.

However, understanding the construction and functioning of a system, which is the basis for automation system design, requires an *integrated view* of the SDs. A major shortcoming in the current state of machine-understandable SDs is that the concepts in the individual descriptions are not interlinked. For example, a method to co-relate requirements, elements in the system design, physical processes, and automation functions is missing.

Therefore, to address the challenge of automated synthesis of organization specification of MAS, we need to identify the relevant entities and relationships in the SDs and enable its expression by integrating the fragmented knowledge.

3 Approach

3.1 Finding Organization Abstractions in System Descriptions

Amongst the abstractions in the MAS organization specification, the hierarchical group structure can be obtained rather directly from the hierarchies of subsystems and their technical equipment. In the next step, we need to define roles and assign them to groups in the structural hierarchy. For example, given that the heating system in a room needs to be automated, the question that comes up when deciding the deployment of agent(s) is *what* (in broad sense) is expected of the agent(s)? However, the notion of a role is not directly expressed in SDs, and the closest that appears as a role is the abstract conception of tasks that an automation program needs to carry out. For example, if the program for the heating system automation needs to measure air temperature and modulate a heating valve based on some control logic, we can envision its *automation role* as being the *temperature controller*. To understand how such *automation roles* are determined during the design of the (traditional) automation systems, consider the following deliberations that occur:

1. Co-relation of requirements to states in the physical processes.
2. Identifying the system parts which play a role in the physical process and the available means of sensing and actuation.
3. Programming (or choosing) an appropriate control strategy for automating the physical process using the identified system parts.
4. Identifying inter-dependent system parts and determining the coordination strategy.

Therefore, if requirements can be linked to respective subsystems and states of the physical processes, we first can infer the *physical effect* that the automation agent needs to achieve using the designated system components. For example, in the case of a heating system, the requirement of maintaining thermal comfort in the room is expected to be achieved through controlling the heat-exchange process conducted by the radiator. This indicates the *automation role* the agent plays (i.e., *temperature controller* using a radiator that conducts heat exchange).

Similarly, dependencies between the system parts, or dependencies between physical processes, should result in the linking of the respective automation roles. For example, if the room's heating system depends on the central boiler's functioning, then roles in the respective groups should also be linked. The semantics of the relationship between the roles captures the coordination foreseen in the system design.

Once we have the definition of a role (in terms of what it is meant to achieve), we need to describe *how* this role can be fulfilled. In other words, the control and coordination tasks that must be carried out by an agent adopting the role.

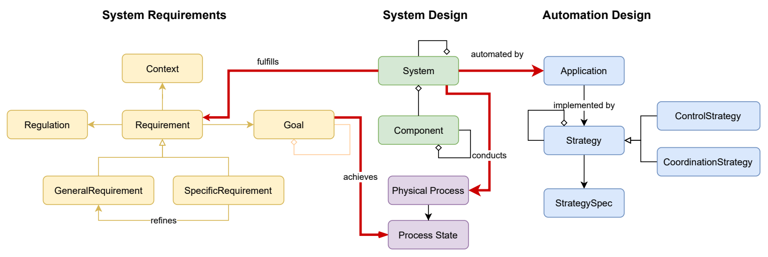

Fig. 2. The integration of engineering ontologies is achieved by establishing relationships (shown as bold red lines) between the aspects

Regarding agents adopting roles, there are two *agentification* scenarios to consider. The automation system could contain *idle* agents that are looking to adopt roles, or a *management program* recognizes unfulfilled roles and deploys agents to take up those roles. In either case, the implication of adopting a role, i.e., the control and coordination tasks, must be considered to verify whether an agent can execute those functions. For example, a control function may require access to sensors, actuators, or specific computational resources.

In domains such as BA and factory automation, SDs include descriptions of *automation applications* (for e.g., see [39]). At an abstract level, an *automation application* represents a collection of control and coordination strategies suitable for a subsystem-process combination. At design-time, a role is linked to the *automation application*, and at run time the agents need to adopt concrete control and coordination strategies depending upon the state of the system and the processes. Therefore, the abstract *application* can have one or more *functional schemes* containing the control and control strategies that an agent can follow. Since MOISE+ supports this concept by decomposing goals and plans, I have used it to model the automation applications.

3.2 Integrating the System Descriptions

Having identified the entities and relationships that need to be visible (and linked) in the SDs, the challenge was then to bridge the concepts in the existing engineering ontologies such that the structural and functional abstractions of the organization can be synthesized from it. The existing ontologies are based on Resource Description Framework (RDF), which is a W3C standard as a part of the Semantic Web Technologies for expressing knowledge as interlinked resources. The ontologies use the Ontology Web Language (OWL), which is grounded in Description Logics, to model concepts as classes and relationships formally. I developed a bridging or *integrating* ontology[1] which allows linking of requirement goals to system components and process goals. System components that need to be automated are linked to abstract *automation application*, which

[1] Can be accessed here: https://github.com/codepasta/autonomous-buildings.git

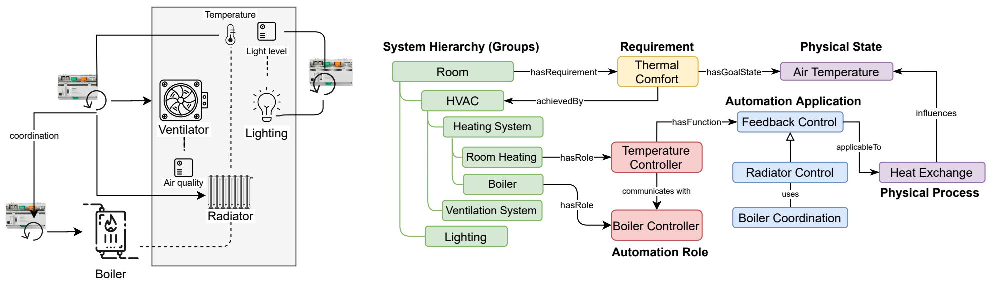

Fig. 3. The scenario for room automation (left) showing the subsystems, the process relationships (in dotted lines), and the roles designed for the automation devices. On the right we see how automation roles are recognized and correlated to functions.

captures the high-level intent of the required automation. A high-level overview of the required integration of the concepts is shown in Figure 2.

3.3 Automated Synthesis of Organisation Specification

The concepts and relationships in an *unified* SD[2] facilitate the automated synthesis of the organization specification. The SD and the ontologies are stored in a Knowledge Graph (KG), which can be queried using SPARQL[3] statements. The automated synthesis of the specification as MOISE+ model (serialized as XML) is done by a software program that interfaces with the KG. Obtaining the structural abstraction is relatively straightforward as it queries for the subsystem hierarchy and the inter-subsystem relationships, if any.

On the other hand, roles need to be identified based on the kind of physical process, the desired goal state, and the subsystem that can be used for the purpose. For example, the process *HeatExchange* conducted by a *Radiator* to maintain *Temperature* is construed as a role definition. For each such role, we need to tell (the role-playing agent) what functions are expected to be carried out – in other words, the *norms* that required to be respected. It is important to note that such functions are seen from system-level perspective, and not meant to tell *exactly* what the agent program should be doing. For example, in the role of a TemperatureController, the agent is expected to execute a suitable control logic for maintaining temperature (about which it autonomously deliberates) while coordinating with the central energy supplier (which it is constrained to do).

Roles are linked to each other if either the system components or the process functions are interdependent. If the semantics of the link requires the agents to communicate, then the link is annotated with reference to a coordination

[2] System Description as a singular is used here to emphasise that it now appears as cohesive knowledge.

[3] https://www.w3.org/TR/rdf-sparql-query/

strategy (in form of a protocol) which the agents need to use to interact with each other.

The organization specification synthesized as MOISE+ XML is then made available to the MAS runtime was implemented using the JaCaMo framework.

4 Evaluation Setup

A BA engineering tool from Siemens AG was used to engineer the automation of an office room containing subsystems for heating, ventilation, air-conditioning (HVAC), and lighting. The automation system was required to maintain temperature, humidity, air quality, and light level in an energy-efficient manner. The engineering tool exports the KG containing the SD, which is then stored in a graph database.

The JaCaMo framework was used to implement the MAS. The agents, each dedicated for the individual subsystems, were deployed in three automation hardware nodes on the network. A bootstrapping code accesses the knowledge graph containing the SD to create the organization specification as MOISE XML, along with the organizational entities representing the groups (i.e., the subsystems) and corresponding automation roles. Figure 3.2 shows a simplified representation of the room's heating system for which the role of the *temperature controller* has been inferred. Similarly, roles for the *ventilation controller*, *boiler controller*, and *lighting controller* are created and assigned to the respective subsystem groups. On initialization, the BDI agents in the controllers accessed the organization definition and evaluated their ability to play one or more automation roles. After adopting a role, they initialized the required control program, and if the role was linked to another role, then a suitable coordination program was also initialized.

5 Results and Discussions

A manual verification of the organization specification by an automation engineer confirmed that it contained the required subsystems and that the automation roles assigned to them were correct. Similarly, the choice of control and coordination programs made by the agents at runtime was confirmed to be correct. In addition, functional tests of the system functions confirmed that the specified requirements were met. Changes in the SD resulted in an update of the organization specification and the agents adapting their plans - this was tested for some sample cases involving changes in requirements and system components.

Though the current state of my evaluation shows encouraging results about the possibility of synthesizing organization specification from SD in the case of BA, this needs to be validated against design descriptions in more diverse domains. Similarly, aspects such as defining the semantic relationships between the roles (to recognize coordination), modeling regulations and norms and ensuring their compliance at runtime, and agents discovering features in the system [36] that may not be explicitly captured in the system design are planned to be researched in future steps.

Acknowledgements

I am grateful to Simon Mayer, Andrei Ciortea, and Danai Vachtsevanou, for the many discussions and inputs which has helped shape my approach to the problem. I also thank my employers Siemens AG, Smart Infrastructure Division, who have actively supported my research and provided the opportunity to conduct evaluations in real-life setups.

References

1. Abbas, H.A., Shaheen, S.I., Amin, M.H.: Organization of multi-agent systems: An overview. Journal of Intelligent Information Systems **Vol. 4, No. 3** (2015)
2. Balaji, B., Bhattacharya, A., Fierro, G., Gao, J., Gluck, J., Hong, D., Johansen, A., Koh, J., Ploennigs, J., Agarwal, Y., et al.: Brick: Towards a unified metadata schema for buildings. In: Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments. pp. 41–50 (2016)
3. Bastos, L.R., Castro, J.F.: From requirements to multi-agent architecture using organisational concepts. In: Proceedings of the fourth international workshop on Software engineering for large-scale multi-agent systems. pp. 1–7 (2005)
4. Bencomo, N., Whittle, J., Sawyer, P., Finkelstein, A., Letier, E.: Requirements reflection: requirements as runtime entities. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2. pp. 199–202 (2010)
5. Butzin, B., Golatowski, F., Timmermann, D.: A survey on information modeling and ontologies in building automation. In: 43rd Annual Conference of the IEEE Industrial Electronics Society. pp. 8615–8621. IEEE (2017)
6. Cena, C.G., Cardenas, P.F., Pazmino, R.S., Puglisi, L., Santonja, R.A.: A cooperative multi-agent robotics system: Design and modelling. Expert Systems with Applications **40**(12), 4737–4748 (2013)
7. Ciortea, A., Mayer, S., Gandon, F., Boissier, O., Ricci, A., Zimmermann, A.: A decade in hindsight: the missing bridge between multi-agent systems and the world wide web. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (2019)
8. Ciortea, A., Mayer, S., Michahelles, F.: Repurposing manufacturing lines on the fly with multi-agent systems for the web of things. In: Proceedings of the 17th international conference on autonomous agents and multiagent systems. pp. 813–822 (2018)
9. Dorri, A., Kanhere, S.S., Jurdak, R.: Multi-agent systems: A survey. IEEE Access **6**, 28573–28593 (2018)
10. Ferber, J., Gutknecht, O., Michel, F.: From Agents to Organizations: An Organizational View of Multi-agent Systems. In: Agent-Oriented Software Engineering IV, vol. 2935, pp. 214–230. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
11. Ferber, J., Michel, F., Báez, J.: Agre: Integrating environments with organizations. In: Environments for Multi-Agent Systems: First International Workshop, E4MAS 2004, New York, NY, July 19, 2004, Revised Selected Papers 1. pp. 48–56. Springer (2005)
12. Freitas, A., Bordini, R.H., Vieira, R.: Designing multi-agent systems from ontology models. In: International Workshop on Engineering Multi-Agent Systems. pp. 76–95. Springer (2018)

13. Hendler, J.: Where are all the intelligent agents? IEEE Intelligent Systems **22**(03), 2–3 (2007)
14. Horling, B., Lesser, V.: A survey of multi-agent organizational paradigms. The Knowledge engineering review **19**(4), 281–316 (2004)
15. Hübner, J.F., Sichman, J.S., Boissier, O.: Moise+ towards a structural, functional, and deontic model for mas organization. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1. pp. 501–502 (2002)
16. Jennings, N.R.: On agent-based software engineering. Artificial intelligence **117**(2), 277–296 (2000)
17. Mascardi, V., Weyns, D., Ricci, A., Earle, C.B., Casals, A., Challenger, M., Chopra, A., Ciortea, A., Dennis, L.A., Díaz, Á.F., et al.: Engineering multi-agent systems: State of affairs and the road ahead. ACM SIGSOFT Software Engineering Notes **44**(1), 18–28 (2019)
18. Mathieu, P., Routier, J.C., Secq, Y.: Dynamic organization of multi-agent systems. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1. pp. 451–452 (2002)
19. McArthur, S.D., Davidson, E.M., Catterson, V.M., Dimeas, A.L., Hatziargyriou, N.D., Ponci, F., Funabashi, T.: Multi-agent systems for power engineering applications—part i: Concepts, approaches, and technical challenges. IEEE Transactions on Power systems **22**(4), 1743–1752 (2007)
20. Mitzutani, I., Ramanathan, G., Mayer, S.: Semantic data integration with devops to support engineering process of intelligent building automation systems. In: Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation. pp. 294–297 (2021)
21. Moran, S.: An applied guide to process and plant design. Elsevier (2019)
22. Morbach, J., Wiesner, A., Marquardt, W.: Ontocape—a (re) usable ontology for computer-aided process engineering. Computers & Chemical Engineering **33**(10), 1546–1556 (2009)
23. Müller, J.P., Fischer, K.: Application impact of multi-agent systems and technologies: A survey. Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks pp. 27–53 (2014)
24. Pechoucek, M., Thompson, S., Baxter, J., Horn, G., Kok, K., Warmer, C., Kamphuis, R., Maric, V., Vrba, P., Hall, K., Maturana, F., Dorer, K., Calisti, M.: Agents in industry: the best from the aamas 2005 industry track. IEEE Intelligent Systems **21**(2), 86–95 (2006)
25. Ploennigs, J., Hensel, B., Dibowski, H., Kabitzsch, K.: Basont-a modular, adaptive building automation system ontology. In: IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society. pp. 4827–4833. IEEE (2012)
26. Ramanathan, G., Husmann, M.: Semantic description of equipment and its controls in building automation systems. In: The Semantic Web: ESWC 2022 Satellite Events: Hersonissos, Crete, Greece, May 29–June 2, 2022, Proceedings, pp. 307–310. Springer (2022)
27. Ramanathan, G., Husmann, M., Niedermeier, C., Vicari, N., Garcia, K., Mayer, S.: Assisting automated fault detection and diagnostics in building automation through semantic description of functions and process data. In: Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation. pp. 228–229 (2021)
28. Runde, S., Dibowski, H., Fay, A., Kabitzsch, K.: Integrated automated design approach for building automation systems. In: IEEE International Conference on Emerging Technologies and Factory Automation. pp. 1488–1495 (2008)

29. Runde, S., Heidemann, A., Fay, A., Schmidt, P.: Engineering of building automation systems—state-of-the-art, deficits, approaches. In: IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010). pp. 1–8. IEEE (2010)
30. Ruta, M., Scioscia, F., Loseto, G., Di Sciascio, E.: Semantic-based resource discovery and orchestration in home and building automation: A multi-agent approach. IEEE Transactions on Industrial Informatics **10**(1), 730–741 (2013)
31. Schneider, F., Berenbach, B.: A literature survey on international standards for systems requirements engineering. Procedia Computer Science **16**, 796–805 (2013)
32. Schneider, G.F., Pauwels, P., Steiger, S.: Ontology-based modeling of control logic in building automation systems. IEEE Transactions on Industrial Informatics **13**(6), 3350–3360 (2017)
33. Shen, W., Hao, Q., Yoon, H.J., Norrie, D.H.: Applications of agent-based systems in intelligent manufacturing: An updated review. Advanced engineering INFORMATICS **20**(4), 415–431 (2006)
34. Siegemund, K., Thomas, E.J., Zhao, Y., Pan, J., Assmann, U.: Towards ontology-driven requirements engineering. In: Workshop semantic web enabled software engineering at 10th international semantic web conference (ISWC) (2011)
35. Sims, M., Corkill, D., Lesser, V.: Automated organization design for multi-agent systems. Autonomous agents and multi-agent systems **16**, 151–185 (2008)
36. Vachtsevanou, D., Ciortea, A., Mayer, S., Lemée, J.: Signifiers as a first-class abstraction in hypermedia multi-agent systems (2023). https://doi.org/10.48550/ARXIV.2302.06970, https://arxiv.org/abs/2302.06970
37. Van Lamsweerde, A.: Goal-oriented requirements engineering: A guided tour. In: Proceedings fifth ieee international symposium on requirements engineering. pp. 249–262. IEEE (2001)
38. Vogel-Heuser, B., Diedrich, C., Fay, A., Jeschke, S., Kowalewski, S., Wollschlaeger, M., et al.: Challenges for software engineering in automation. Journal of Software Engineering and Applications (2014)
39. Wetter, M., Grahovac, M., Hu, J.: Control description language. In: Proceedings of The American Modelica Conference 2018. pp. 17–26. Linköping University Electronic Press (2019)
40. Zia, T., Lang, R., Boley, H., Bruckner, D., Zucker, G.: An autonomous adaptive multiagent model for building automation. IFAC Proceedings Volumes **42**(3), 250–254 (2009)