






Extending DEMO Action Rule Specifications' Syntax in a Low Code Platform Based Municipality Hearing System Implementation

David Aveiro^{1,2,3} ^a, Vítor Freitas^{1,3} ^b, Duarte Pinto^{1,2} ^c, Valentim Caires^{1,2} ^d and Dulce Pacheco^{1,2} ^e

¹ARDITI - Regional Agency for the Development of Research, Technology and Innovation, 9020-105 Funchal, Portugal

²NOVA-LINCS, Universidade NOVA de Lisboa, Campus da Caparica, 2829-516 Caparica, Portugal

³Faculty of Exact Sciences and Engineering, University of Madeira, Caminho da Penteada 9020-105 Funchal, Portugal


Keywords: DEMO, Action Rules, Business Rules, Low-Code Platform.


Abstract: The current official Design and Engineering Methodology for Organizations (DEMO) Action Rules Specification are unnecessarily complex and ambiguous. These specifications are also incomplete, lacking sufficient ontological details required to derive a fully functional implementation; and complex by containing mostly unneeded specifications. Additionally, this paper details our progress in developing a metamodel for DEMO's Action Model, using an Extended Backus-Naur Form (EBNF) syntax. These advancements were driven by our experience in implementing a system for the case study on a no-code/low-code platform to support a local Municipality Hearings Process. This implementation was done on a Low-Code platform supporting the direct execution of DEMO Models that is being developed by our team. Among our contributions are the models and patterns generated from this implementation, which provide reusable solutions that can be adopted by other low-code platforms using a similar approach.


1 INTRODUCTION


A clear specification and implementation of business rules is essential for effective processes within enterprises. The Design and Engineering Methodology for Organizations (DEMO) claims to have a thorough and structured approach to modeling organizations, capturing their essence (Dietz, 2022). However, The official DEMO Action Rules Syntax has been criticized for its complexity and unclear syntax, which complicate understanding and hinder smooth implementation (Aveiro and Freitas, 2023b). In response to these issues, this paper introduces enhancements to DEMO's Action Meta-Model, aiming to provide a more robust framework for representing action rules and supporting their implementation in large-scale systems in a concrete low-code platform being developed by our team. We iteratively tested and validated our proposal within a real case of a Hearings Pro-


cess of a local municipality. The proposed improvements to the DEMO Action Rules syntax are articulated using Extended Backus-Naur Form (EBNF) notation (ISO/IEC 14977:1996, 1996) and were guided by practical requirements encountered during the development of a low-code platform based on DEMO. Through iterative design, implementation, and rigorous testing, we refined the syntax in an attempt to improve the ontological accuracy and implementation comprehensiveness of the DEMO Action Rules. In Section 2 we outline the Research Context of the paper focused on relevant theories, tools, related work and research methodology. In Section 3 we explain the real use case of the Municipality Hearing System and present the relevant process and fact models. Section 4 details our proposal for extending DEMO's Action Rule Specification Syntax, illustrating it with concrete cases. Finally, in Section 5 we discuss our conclusions and future work.

^a  <https://orcid.org/0000-0001-6453-3648>

^b  <https://orcid.org/0009-0002-0667-5749>

^c  <https://orcid.org/0000-0002-8451-5727>

^d  <https://orcid.org/0000-0002-0871-7212>

^e  <https://orcid.org/0000-0002-3983-434X>

2 RESEARCH CONTEXT

In this section we first present the DEMO Theories and the DEMO Models, followed by a more detailed focus on the Action Model, which is the focus of this paper. We then proceed to the DEMO based Low-Code Platform being developed, from which the improvements proposed to the Action Rules derive. Finally, we present some relevant Related Work followed by the Research Method used.

2.1 DEMO Theories and Models

According to Jan Dietz in his Enterprise Ontology work, organizations can be seen as a network of actors (people or roles) who engage in commitments and follow a structured series of actions known as the Complete Transaction Pattern (CTP). This pattern includes a collection of actions like requesting, promising, executing, declaring, and accepting and also handles the exceptions to the general flow as in rejections or declines or steps for cancelling commitments if needed. Actors keep track of their responsibilities through an “actor cycle”, where they constantly check their “agenda” or list of tasks (Dietz and Mulder, 2020). With Design and Engineering Methodology for Organizations (DEMO), Dietz frames organizations in four key perspectives in the so-called aspect models and proceeds to detail how each can be represented. The Cooperation Model (CM) illustrates the coordination within the organization, detailing the roles involved in transactions and the actions they initiate or perform, the facts they handle, and how different roles wait or depend on each other. The Process Model (PM) describes the workflows and interactions that are triggered by the various actions, showing how different actors or roles coordinate. The Fact Model (FM) serves as the semantic model where the types of facts are defined, such as different types of products, properties, and attributes, along with the rules to governing their existence. Lastly, the Action Model (AM) provides operational guidance by outlining the rules for task execution, including triggers, conditions, and how to respond in each step (Dietz and Mulder, 2020).

2.2 Action Model

Since the focus of this paper is on DEMO’s Action Model, we now present its fundamentals in more detail. DEMO Action Rules Specification (ARS) are the representation of the Action Model, and sets the guidelines for managing events to which actors must react, or business rules as well as work instructions regarding the execution of production acts (Dietz,

2022). The standard syntax of ARS has evolved over time, originally starting with a pseudo-algorithmic language and currently formalized in EBNF. The representation of an action rule follows the format `<event part> <assess part> <response part>`. The event part (event or collection of concurrent events) is what triggers the rule. The assess part defines the validity claims that are being ascertained, and can be divided into three kinds: the claims to rightness, sincerity, and truth. The response part defines how to react or comply to the event if the conditions (defined in the “assess” part) are met or not. Actors can stray from the rule if they believe it is acceptable, but they are also held accountable for it (Dietz, 2022).

These ARS specifications have been argued to be somewhat ambiguous in (Aveiro and Freitas, 2023b) because, despite using a structured English syntax, it does so in a way that lacks some of the necessary ontological details needed as the basis for the implementation of an information system, e.g., the lack of a method to deal with sets of actions or operators. In (Aveiro and Freitas, 2023b) it is also argued that the ARS bring about an unneeded complexity in the assess part of the rule by including many extraneous details about three different forms of evaluation: fairness, sincerity, and truth. It is mentioned in (Perin-forma, 2015), on one hand, that action rules written with the grammar of “structured English” are incredibly simple, but, on the other hand, it is also stated that some board members appeared perplexed when an action rule with this grammar was presented to them. This was our situation too in real-life projects we have been conducting, thus the new alternative of ARS syntax we have been building up. In (Aveiro and Freitas, 2023b), many more detailed arguments can be found that reinforce our stance.

2.3 DISME Low-Code Platform

The Dynamic Information Systems Modeller and Executer (DISME) is an open-source no-code/low-code platform, specifically designed using the DEMO methodology (Freitas et al., 2022). A distinct advantage of the DISME, besides being based on DEMO, is its adherence to the principles of the Adaptive Object Model (AOM) and the Type Square pattern (Yoder et al., 2001) (Yoder and Johnson, 2002) (Aveiro and Pinto, 2015). This design allows DISME to immediately implement changes in real-time in the production environment, reflecting them in the operational system without the typical need for generating static code, compiling, or deployment, which many conventional low-code platforms require (Freitas et al., 2022).

The DISME platform is composed of two main functional interfaces, 1) the system modeler, where users can define and parameterize the system using diagrams, models forms and tables and 2) the system executor that processes the specifications created by the users at the modeller and runs it (Freitas et al., 2022).

The system modeller features several key components: 1) diagram editor that allows the diagrammatic specification of DEMO's PM and FM; 2) the system parametrization that enables the definition of users, roles, and other related elements alongside the PM and FM; 3) the action rules manager where users can specify, in a graphical way, using an interface based in Blockly¹, all details of our variant of ARS of DEMO's AM 4) the form manager to design layout and other details needed for all user input actions; and 5) the dynamic query manager which supports the creation of queries using filters and operators to meet data processing needs (Freitas et al., 2022).

The system executor has two main functions: a dashboard that allows users to interact with the system and execute processes according to their roles and permissions, and the execution engine which controls the flow of the processes according to the current state of the system's world, user interactions and runtime interpretation of the ARS (Freitas et al., 2022).

2.4 Related Work

As far as we know, besides our approach, there's only another research line ((Krouwel et al., 2024)) exploring the bridge between thorough enterprise models and implementation of low-code platform-based systems, where multiple enterprise modeling languages are referenced as possible source such as: 4EM (Lantow et al., 2022), ArchiMate², BPMN³, MEMO (Frank, 1999), SBVR⁴, UML⁵, among others. But all these alternatives either only describe a part of the enterprise or they lack formal semantics (Krouwel, 2023). Furthermore, in (Krouwel et al., 2024), it is claimed that, in order to be able to transform a model to an application, the input model must 1) be comprehensive, 2) describe all business requirements, 3) be consistent, and 4) have its semantics fully specified. For this to be achieved (Krouwel et al., 2024) states that the source model must include actor roles, products, processes, information items, and business rules effectively, and that DEMO ontological models

are particularly effective for this purpose as they provide coherent, comprehensive, consistent, and concise models of the organizational essence (Dietz and Mulder, 2020). (Krouwel et al., 2024) also claims that because DEMO models are both semantically rich and well-defined, that makes them a good fit for the needed transformation.

Regarding source models, we agree and follow the same principles as (Krouwel et al., 2024) and, thus, also adopt DEMO as the base language for our bridge with low-code systems. However, we differ strongly in the low-code implementation part. (Krouwel et al., 2024) chose Mendix⁶ as their best option and present a proposal of a mapping from DEMO models to Mendix elements, for the (automated) creation of a low-code application that also intrinsically accommodates run-time implementation design decisions. While in (Krouwel et al., 2024) a mapping approach to the Mendix metamodel is followed, we opt to use the direct execution of DEMO models, including run-time interpretation of enriched action rule specification by the execution engine of our platform.

Furthermore, some limitations recognized by our fellow authors in their work and of the current official DEMO ARS are quite alarming. Namely, researchers in (Krouwel et al., 2024), while developing their mapping, identified some significant limitations with the DEMO metamodel as it is defined in DEMO-SL (Dietz, 2022). As it is designed from a modeling perspective rather than with an operational or software development focus, while doing their manual conversion from the models to JSON code needed for Mendix, some elements that only served visualization purposes were omitted, and it became necessary to include other crucial aspects for software generation such as application names, mappings and software primitives. Those elements along with the Operational Independent Variables (OIVs) were placed separate from the DEMO conversion in a specific descriptor file. The mapping process also did not include rules for generalization and specialization of entity types from the DEMO FM. While the conversion referred to in (Krouwel et al., 2024) was taking place, researchers also found that the mapping of Derived Fact Specifications and ARS into Mendix was rather complex and that the implementation of OIVs within Mendix or even software in general would translate into a challenging labor-intensive detailing task that had to be done, not only for each OIV independently, but also in combinations and that, when combining several OIVs, the problem increases in its complexity because it was not clear whether all OIV's could be implemented completely independent of others.

¹<https://github.com/google/blockly>

²<https://www.archimatetool.com/>

³<https://www.omg.org/spec/BPMN/>

⁴<https://www.omg.org/spec/SBVR/>

⁵<http://www.uml.org/>

⁶<https://www.mendix.com/>

Summarizing, many complexities and manual steps arise that seem to make the proposed approach in (Krouwel et al., 2024) unfeasible for large-scale systems. One of the problems we identify is that, although our colleagues argue that developers should not be taking implementation decisions, this ends up happening in their proposed approach, in the many manual steps that still need to be taken and code involved in the Mendix side, and also many combinations of possible OIVs that need to be configured in runtime, occurring, partly, due to the unneeded complexities we pointed out that exist in the current official DEMO ARS. The approach we follow, which is presented next, overcomes these limitations.

2.5 Research Method

In (Hevner et al., 2004; Hevner, 2007) A. R. Hevner presents the Design Science Research. This Information Systems Research paradigm is composed by a collection of three closely related cycles of activities and regards how a study should be viewed.

Figure 1, illustrates these three cycles. Hevner argues that these activities should always be used together in order to provide a solid design of science research with a reliable result. In regard to the Relevance Cycle in Figure 1, our research has revealed a lack of conciseness and crucial information in the current official syntax of Action Rules in DEMO, as well as some degree of ambiguity (Andrade et al., 2020) (Aveiro and Freitas, 2023b).

As a result, in the Design Cycle, in the efforts reported in this paper, we continued to design and build upon previous work of what we consider to be a more comprehensive and complete syntax for the DEMO action rules (Andrade et al., 2020) (Aveiro and Freitas, 2023b). New improvements in the grammar took place during the last couple of years after multiple iterations and thorough design, testing, and validation of the use and comprehensiveness of the new language elements while applying them in the DISME's execution engine, with the EU-Rent case, as well as another more recent case of a neurorehabilitation information system (Aveiro et al., 2023) and the case now presented in this paper, from a local municipality. Thus, the latest version of the ARS hereby presented, allows for an increased coverage of DEMO's AM in specifying thoroughly low-code based systems for large scale and complex cases. Finally, in regard to the Rigor Cycle, the DEMO theoretical grounding, in itself, provides the needed support for the designed artifact that is our latest grammar proposal.

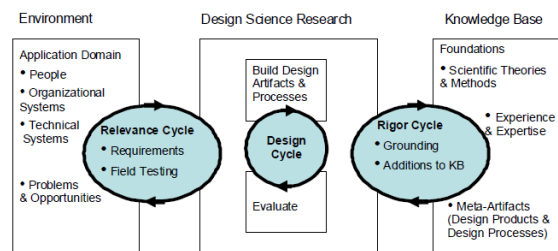


Figure 1: Design science research cycles (Hevner, 2007).

3 THE MUNICIPALITY HEARINGS PROCESS

The Municipality Hearing Process (MHP) involves several steps to ensure efficient citizen engagement. Before a hearing takes place, each Hearing Officer (an individual or group of individuals with a position of responsibility in specific domains inside the municipality, for example, construction licensing or water distribution) needs to have their schedule defined.

Traditionally, each Hearing Officer has a specific day in the week when they provide a certain number of hearings to the Citizen for them to expose their claims. Once the scheduling is settled, the Hearing Officer can be subject to a Hearing Request by the Citizens.

The traditional way for this to happen is for the Citizen to go to the municipality's service desk and request a hearing. The clerk will then ascertain the need for said hearing and collect all the relevant information such as citizen identification, the process, the subject or relevant observations. These clerks are trained and follow a set of guidelines allowing them to determine if a hearing should take place, or not, at a given time for multiple reasons. For example, if a previous hearing with the same citizen/theme took place a short time before, or that the process it pertains to is too recent.

If a hearing request reason is deemed justifiable to happen, the clerk fills the request form and then proceeds to print a PDF file of the accepted request to give as proof to the citizen. The next step is to define who (Hearing Officer) is the most relevant to attend the citizen's claim in a hearing. In some cases, the claim may involve multiple departments of the municipality and the clerk may not be able to ascertain the correct individual on the spot, so they can ask the Hearing Officers themselves or their assistants if they are indeed the most qualified for that hearing before proceeding to its scheduling.

Once the correct Hearing Officer has been identified and selected, a hearing can be scheduled, traditionally by the clerks, that will then proceed to print

another PDF file with the Hearing details and send it to the Citizen by email or postal mail.

Between this time and the time of the hearing both the Citizen and the Hearing Officers can request the cancellation or rescheduling of the hearing for multiple reasons, like that their schedule has changed, and they cannot be present at the defined time slot or simply that the reason for the hearing itself is no longer valid. Because the Hearings might take some time to happen, it is often the case that the claim from the citizen has already been resolved and as such, it is normal for the Hearing Officers to check their future scheduled hearings and identify those that no longer are needed. If that is not the case, the hearing meeting eventually takes place on the defined or rescheduled date. After the meeting, the conclusions and observations are added to the process, and it's concluded by the Hearing Official. Besides the main process, there are additional tasks that need to be performed, like being able to print the Hearing Officials schedule each day as well as access the history of all hearings from a specific citizen.

We will now present the models of the MHP using an evolved notation of DEMO's representations proposed in (Pinto et al., 2021) and (Gouveia et al., 2021). An explanation of the notation follows suit. These proposals are being used in DISME, and will also be used in this paper due to the benefits mentioned hereafter. Due to space limitations, a more detailed version of the New Process Model Representation and New Fact Model Representation can be accessed as an online annex (Aveiro et al., 2024), alongside some illustrative figures from the MHP case study.

3.1 New Process Model Representation

In (Pinto et al., 2021) an alternative Process Structure Diagram (PSD) representation for DEMO's PM is proposed which integrates some contents of the CM, in order to improve the clarity, reduce the redundancies and complexity and increase the transparency in the representations allowing for a more comprehensive representation of the operational flow of the organizational processes. These goals are achieved by: 1) simplifying the nomenclature, 2) enhancing the process model diagram and 3) moving the fine details of the process model to a transaction description table but allowing it all to still be easily accessible and manageable, using the process model diagram as reference (Pinto et al., 2021). These improvements were positively validated in (Pacheco et al., 2022b). In the online annex (Aveiro et al., 2024), there is an example of the Process Model Diagram, as well as a more

detailed of explanation of the goals achieved with it.

3.2 New Fact Model Representation

In (Gouveia et al., 2021), the authors address what they consider to be a set of issues in the current official DEMO's FM, namely the complexity and poor usability caused by the over-cluttering of shapes that are hard to understand by those not specialized in DEMO, the lack of flexibility to accommodate changes and updates, and the inadequate visual representations with multiple symbols that are not intuitive and can lead to misinterpretation and errors. To address these problems, in (Gouveia et al., 2021) an alternative notation is proposed, with what are considered simpler and more intuitive diagrams and tables, readable by any stakeholders that have no knowledge of DEMO, just business know-how. These improvements were validated in (Pacheco et al., 2022a).

The main artifact of this new Fact Model is the Concepts and Relationships Diagram (CRD), a generic, global, and synthetic view of an entire domain's concepts while abstracting from their attributes (Pacheco et al., 2022b). In the online annex (Aveiro et al., 2024), there is an example of the CRD, including a detailed explanation of each concept.

4 EXTENDING ACTION RULE SPECIFICATIONS SYNTAX

Here, we introduce the key advancements in our alternative DEMO ARS, as originally outlined in (Andrade et al., 2020) and (Aveiro and Freitas, 2023b). Because of space limitations, a complete version of the EBNF grammar that formalizes our refined DEMO ARS syntax can also be accessed in the online annex mentioned in the previous section, alongside four intricate action rules from the MHP case study, showcasing the recent enhancements. It is important to note that, for historical reasons, the DISME's database (DB) tables and concepts employ slightly different terminology in the grammar. Specifically, concepts and attributes from the FM are saved as *entity_types* and *properties*, respectively, while transactions and tasks are recorded under the *transaction_type* table. This section summarizes the key innovations along with selected entries from the grammar where necessary. Figure 3 of the annex (Aveiro et al., 2024) presents the first example of a complex action rule derived from the MHP case.

These innovations were introduced to address the complexities encountered in the MHP, which our previous version of the ARS grammar could not fully

handle. While earlier iterations were enough for simpler applications like the neurorehabilitation case managed within DISME. This case was presented in (Aveiro et al., 2023) where the development efforts of developing the needed system in a low-code and traditional way were compared. The formal evaluation of the usability of both systems was published in (Aveiro and Freitas, 2023a) which was quite similar for both systems, proving the capability of our ARS syntax and of DISME in this already complex scenario. However, in response to the unique challenges presented by the MHP, we introduced innovative language elements designed to address specific gaps in the ARS syntax, allowing for more seamless handling of complex scenarios.

The Figure 3 of the annex (Aveiro et al., 2024) demonstrates the action rule associated with the execution phase of the 'Placing Hearing Request' transaction. In this scenario, a municipality employee is tasked with selecting a citizen from the database, reviewing prior hearing records, and completing a request form based on relevant details such as the associated process or subject. Once the form is properly filled out, the municipality clerk will then generate a PDF with the request information, print it and deliver it to the citizen. After that, a Hearing Officer must be assigned to the request, although the assigning might be either immediately done by the clerk or delayed to a posterior moment and decided by someone else, when the best option of hearing officer is decided for that particular hearing. This last step is achieved in the action rule using the user *evaluated expression* element, where the clerk can decide if he wants to assign the hearing officer right away, or wants to request a transaction for this purpose to be handled later.

One of the most relevant changes in the new version of the grammar was renaming/adapting the element previously known as *user_input* to the new *create_instance*. This happened because a created instance of a certain entity type might obtain their fact values from things other than inputs from a form (the only possibility in the previous version). Now, in this action, the system will prompt the user for input through a form, that is, for the user to input some data, if there are specified *form facts* (previously known as properties) for this action. The new concept of *derived facts* introduced in this work allows for automatic value assignments, effectively incorporating DEMO's framework of derived fact specifications (Dietz, 2022), which had not been included in earlier versions of DISME and something our colleagues in (Krouwel et al., 2024) were struggling with, as previously mentioned. In Figure 4 of the annex we can find a more complex *derived fact* including a *compute ex-*

pression. If there are no *form facts* specified, but there are *derived facts*, the action will run automatically by the system's engine without user intervention.

Other very relevant additions to the grammar, which can be seen right at the beginning of Figure 3 of the annex (Aveiro et al., 2024), are the elements: *context_variable*, which serves a similar function as local variables in a function in a programming language; and the possibility of specifying that the selectable options to be made available in a certain form fact input must come from a *query* executed at runtime (which might accept parameters or not, e.g., a *context variable*). In the particular case of this example, the citizen is set as a context variable at the start, and can then be used, not only to query previous hearing requests that he has made in the past, but also to automatically derive its value into the respective form fact needed in the *Placing Hearing Request* transaction.

In some situations one will want to save in a certain generated instance, a reference to the person who executed a certain action/created a certain instance, e.g., the concrete admin staff processing the hearing request. This is the reason for the new element *executing_user*, which can also be seen in Figure 3 of the annex (Aveiro et al., 2024).

The *term* element was also the target of a relevant change: *term* = constant — **value** — **property** — *query* — *compute_expression* — **context_variable** — **executing_role** — **executing_user**. Namely, the last three elements are new. This allows even greater reusability and componentization of the different elements of the syntax especially thanks to the introduction of the element *context_variable*. In fact, this is the key new element in our grammar that, together with the use of dynamic queries with parameters, allows solving, at design time, many types of complexities our colleagues faced in (Krouwel et al., 2024). This is a quite traditional and important construct needed in specifying business logic, most of the time buried in the code of some common programming language. As presented in (Aveiro and Freitas, 2023b) the term element is one of the key elements in our grammar that allow modularization of and integration of different system elements and logic. We bring to attention the fact that following the AOM principle (Yoder and Johnson, 2002) all elements of our "programs" specified in DEMO ARS are objects stored in the DISME's DB. This allows a huge capacity of reuse and also dependency detection and dynamic impact analysis of potential changes in process logic or data structure or value types. The introduction of the *context_variable* element brings a positive increase in these capacities.

In conclusion, we developed a new set of elements to address a common requirement in many en-

terprises - the ability to create and manage service slots, whether scheduled by clients or internal collaborators.

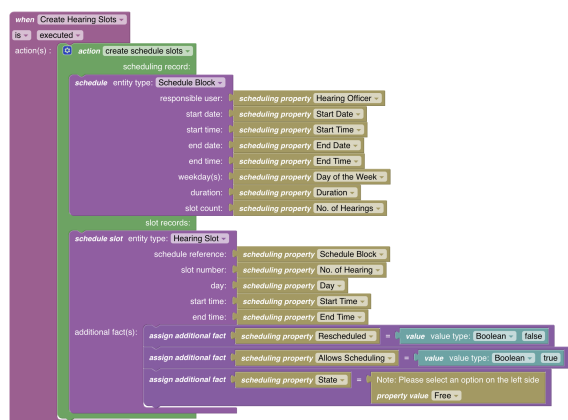


Figure 2: Action Rule - Create Hearing Slots.

This new set of elements composes a new action type: *create_schedule_slots* = *scheduling_record* *slot_records*, that will serve to automatically create time slots based on a *scheduling_record* entity type, as can be seen in the new element *scheduling_record* = **entity_type** **responsible_user** **start_date** **start_time** **end_date** **end_time** **weekday** **duration** **slot_count**. This entity type must be specified in the system and will be used here to set which of its properties holds the information about specific scheduling properties that are needed to create time slots. The *slot_records* entity type must also be specified in the system and will be used to set in which properties the information about the specific time slots created should be stored, as can be seen in the new element *slot_records* = **entity_type** **schedule_reference** **slot_number** **day** **start.time** **end.time** {*additional_fact*}. In case we have other properties belonging to the specified slot records entity type that aren't specific to the slot records fundamental elements, that we want to specify a value for, we also have a new element *additional_fact* = **property** "=" term — **property_value**. As scheduling functionalities are crucial in many business types, this kind of action and the elements following below were devised to facilitate the specification of scheduling properties and execution of scheduling mechanisms. Several of these properties were inspired by the proficient online scheduling service Appointlet⁷.

In Figure 2 we can see the usage of this new action type in the context of the MHP. Here we can see the matching of the scheduling properties in full effect. In the *scheduling_record* input of the action block, af-

ter selecting the **Schedule Block** internal *entity type* that contains the information about the scheduling that needs to be made, previously inserted by the user in a *create_instance* action, the matching is made individually for each scheduling property. This way, we now know in which *entity type* and in which properties we can get the information needed for generating the scheduling's slots, that is, the *start date*, *end date*, *duration* and so on. Next, in order to generate the schedule slot instances, one needs to know the type of these entities. That, alongside other important information, is specified in the *schedule slot* input of the action block, as can be seen in Figure 2. After selecting the **Hearing Slot** entity type of which instances will be created to store the generated scheduling slots, matching is made again for the fundamental properties of a schedule slot, which include, for example, the *day*, *start time* and *end time*. By matching these fundamental properties with the chosen entity type's properties, the system is able to automatically generate the schedule slots and store each of these fundamental properties on the system's entity instances. Furthermore, as the **Hearing Slot** entity type contains more properties that need their value set other than the fundamental properties of a scheduling slot, we define how those *additional facts* should have their value defined in the last input of this *schedule slot* block.

5 CONCLUSIONS AND FUTURE WORK

This paper presents key findings from our collaboration with a local municipality, where we developed a system to streamline the Hearings Process using DISME. This applied research project led to the realization that we needed to improve and extend our already evolved grammar of DEMO's Action Model, one of the main contributions of the reported work. Our implementation of DEMO's Action Model for the Hearings Process brought new elements that directly addressed the complexities unique to managing citizen requests and officer availability. This specific approach allowed us to translate intricate workflows into a richer Action Rule Syntax (ARS), which could be validated and adjusted by municipal staff without the need for deep technical knowledge. The project's success highlights the potential for DEMO's framework to significantly improve productivity and usability in local government systems. In response to the unique needs of the municipal Hearings Process, we incorporated advanced programming constructs such as conditional logic (if-then-else), complex arithmetic operations, and dynamic queries with parameters.

⁷<https://www.appointlet.com>

These additions enabled the system to handle flexible scheduling requests and context-sensitive data management, making it adaptable to the varying demands of the local administration. One could argue that what we are doing with DEMO's ARS is complex and almost amounts to programming at a similar level as other languages. However, what we are aiming to offer is different and at a much higher level: a way to visually specify complex processes and flow logic in the most technologically neutral and user-friendly way possible. In fact, if one looks at life before computers, human and paper based information systems were already performing all these complex process logic and flow operations, albeit in a very slow way, and the "programming logic" was imbued in the processes. With our approach, by using a visual and block-based interface that eliminates the risk of syntactic errors and allows dynamic semantic, and even pragmatic validations with runtime and production data, thanks to the flexibility provided by Blockly, we expect that we will be able to bring a lot of power back to the business users, as citizen developers, if not for all kinds of software systems (not reasonable or feasible), at least for a good amount of typical information systems.

The other contribution is the conceptual models of the Process and Facts of the MHP process, which constitutes a generic pattern that might be used for other similar ends in other public institutions, with a similar process and information to manage. Some relevant aspects of this model are the nuances of: splitting the request for the hearing from the scheduling of the hearing itself, while taking into account the historical record of hearings of requesting citizens; and flexible functionalities for rescheduling hearings and managing officers' hearing slots. The municipality currently uses a Commercial-Off-The-Shelf software solution for managing scheduling, which is expensive and not flexible to accommodate this and other nuances of the process that need to be supported in order to achieve higher operational efficiency in the use of the officer's time. When they saw the potentialities of DISME in a public presentation, they immediately approached us to test our platform to implement a better system for their purposes. This implementation of the low-code based system to support the MHP is not yet finished, as some use cases are still missing to be specified. Completing this specification and making it available to the academic, industrial and scientific communities is one of the other lines of future work. Looking ahead, we plan to refine the DEMO Action Model further by collaborating with municipal staff and public administration experts. Our focus will be on enhancing the usability of our Blockly-based interface, mak-

ing it more intuitive for non-technical users in government settings. Usability studies within this public administration context will guide the future iterations of the platform, ensuring it meets the operational needs of diverse public institutions. So, detailed usability studies on these more recent implementations will also be done and reported in the near term.

REFERENCES

- Andrade, M., Aveiro, D., and Pinto, D. (2020). Bridging Ontology and Implementation with a New DEMO Action Meta-model and Engine. In Aveiro, D., Guizzardi, G., and Borbinha, J., editors, *Advances in Enterprise Engineering XIII*, Lecture Notes in Business Information Processing, pages 66–82, Cham. Springer International Publishing.
- Aveiro, D. and Freitas, V. (2023a). Evaluating the usability of a system implemented on a DEMO based low-code platform. In Sales, T. P., Aveiro, D., Proper, H. A., Asprion, P. M., Marcelletti, A., Morichetta, A., Schneider, B., Zech, P., Kulkarni, V., Breu, R., Barat, S., Poels, G., Riel, J. V., Calhau, R. F., Bork, D., Mulder, M., de Kinderen, S., Guerreiro, S., and Griffo, C., editors, *Companion Proceedings of the 16th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling and the 13th Enterprise Design and Engineering Working Conference: BES, DTE, FACETE, Tools & Demos, Forum, EDEN Doctoral Consortium co-located with PoEM 2023, Vienna, Austria, November 28 - December 01, 2023*, volume 3645 of *CEUR Workshop Proceedings*. CEUR-WS.org.
- Aveiro, D. and Freitas, V. (2023b). A new action meta-model and grammar for a demo based low-code platform rules processing engine. In *Advances in Enterprise Engineering XVI*, pages 33–52, Cham. Springer Nature Switzerland.
- Aveiro, D., Freitas, V., Cunha, E., Quintal, F., and Almeida, Y. (2023). Traditional vs. low-code development: comparing needed effort and system complexity in the nexusbrant experiment. In *2023 IEEE 25th Conference on Business Informatics (CBI)*, pages 1–10, Los Alamitos, CA, USA. IEEE Computer Society.
- Aveiro, D., Freitas, V., Pinto, D., Caires, V., and Pacheco, D. (2024). EBNF Grammar of DEMO Action Rules Specification and Process and Fact Models Representations for a Municipal Hearings Process. <https://github.com/EnterpriseEngineeringLab/KEOD-2024-annex/blob/main/KEOD-2024-annex.pdf>
- Aveiro, D. and Pinto, D. (2015). Universal Enterprise Adaptive Object Model: A Semantic Web-Based Implementation of Organizational Self-Awareness. *Intelligent Systems in Accounting, Finance and Management*, 22(1):3–28. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/isaf.1363>.
- Dietz, J. L. G. (2022). DEMO Specification Language 4.7.2 – Enterprise Engineering Institute.

- Dietz, J. L. G. and Mulder, H. B. F. (2020). *Enterprise Ontology: A Human-Centric Approach to Understanding the Essence of Organisation*. The Enterprise Engineering Series. Springer International Publishing.
- Frank, U. (1999). The MEMO META-METAMODEL.
- Freitas, V., Pinto, D., Caires, V., Tadeu, L., and Aveiro, D. (2022). The DISME low-code platform - from simple diagram creation to system execution. In Guerreiro, S., Griffo, C., and Jacob, M., editors, *Proceedings of the 22nd CIAO! Doctoral Consortium, and Enterprise Engineering Working Conference Forum 2022 co-located with 12th Enterprise Engineering Working Conference (EEWC 2022), November 2-3, 2022, Leusden, the Netherlands*, volume 3388 of *CEUR Workshop Proceedings*. CEUR-WS.org.
- Gouveia, B., Aveiro, D., Pacheco, D., Pinto, D., and Gouveia, D. (2021). Fact model in demo - urban law case and proposal of representation improvements. In Aveiro, D., Guizzardi, G., Pergl, R., and Proper, H. A., editors, *Advances in Enterprise Engineering XIV*, pages 173–190, Cham. Springer International Publishing.
- Hevner, A. (2007). A three cycle view of design science research. *Scandinavian Journal of Information Systems*, 19.
- Hevner, A., R. A., March, S., T. S., Park, J., Ram, and Sudha (2004). Design science in information systems research. *Management Information Systems Quarterly*, 28:75–.
- ISO/IEC 14977:1996 (1996). ISO/IEC 14977 - Information technology — Syntactic metalanguage — Extended BNF. Standard, International Organization for Standardization, Geneva, CH.
- Krouwel, M., Op 't Land, M., and Proper, H. (2024). From enterprise models to low-code applications: mapping demo to mendix; illustrated in the social housing domain. *Software and Systems Modeling*, pages 1–28.
- Krouwel, M. R. (2023). *On the design of enterprise ontology-driven software development*. Maastricht University, Maastricht.
- Lantow, B., Sandkuhl, K., and Stirna, J. (2022). Enterprise Modeling with 4EM: Perspectives and Method. In Karagiannis, D., Lee, M., Hinkelmann, K., and Utz, W., editors, *Domain-Specific Conceptual Modeling: Concepts, Methods and ADOxx Tools*, pages 95–120. Springer International Publishing, Cham.
- Pacheco, D., Aveiro, D., Gouveia, B., and Pinto, D. (2022a). Evaluation of the perceived quality and functionality of fact model diagrams in demo. In *Advances in Enterprise Engineering XV*, pages 114–128, Cham. Springer International Publishing.
- Pacheco, D., Aveiro, D., Pinto, D., and Gouveia, B. (2022b). Towards the x-theory: An evaluation of the perceived quality and functionality of demo's process model. In Aveiro, D., Proper, H. A., Guerreiro, S., and de Vries, M., editors, *Advances in Enterprise Engineering XV*, pages 129–148, Cham. Springer International Publishing.
- Perinforma, A. (2015). *The Essence of Organisation: An Introduction to Enterprise Engineering*. Sapio Enterprise Engineering.
- Pinto, D., Aveiro, D., Pacheco, D., Gouveia, B., and Gouveia, D. (2021). *Validation of DEMO's Conciseness Quality and Proposal of Improvements to the Process Model*, pages 133–152.
- Yoder, J. W., Balaguer, F., and Johnson, R. (2001). Architecture and design of adaptive object-models. *SIGPLAN Not.*, 36(12):50–60.
- Yoder, J. W. and Johnson, R. (2002). The Adaptive Object-Model Architectural Style. In Bosch, J., Gentleman, M., Hofmeister, C., and Kuusela, J., editors, *Software Architecture: System Design, Development and Maintenance*, IFIP — The International Federation for Information Processing, pages 3–27. Springer US, Boston, MA.