Bridging Models and Practice: Action Rules in a DEMO-Based Low-Code Platform

Vítor Freitas[1,3,*,†], David Aveiro[1,2,3,*,†] and Duarte Pinto[1,2,†]

[1]ARDITI - Regional Agency for the Development of Research, Technology and Innovation, 9020-105 Funchal, Portugal
[2]NOVA-LINCS, Universidade NOVA de Lisboa, Campus da Caparica, 2829-516 Caparica, Portugal
[3]Faculty of Exact Sciences and Engineering, University of Madeira, Caminho da Penteada 9020-105 Funchal, Portugal

Abstract

Municipalities face rising demands for transparency and efficiency while still relying on legacy systems that are hard to adapt to new regulations. Low-code/no-code platforms promise relief but often lack theoretical grounding, leading to fragmented solutions. This paper presents the *Dynamic Information System Modeller and Executor (DISME)*, a DEMO-based low-code/no-code platform that executes organizational models directly at runtime. DISME is being applied in the Smart Islands Hub project's *Test-Before-Invest* services, with a pilot implementation of the Municipality Hearings Process (MHP). We show how DISME parametrizes roles, transactions, entities, and user interactions, and highlight the role of Action Rules in defining executable business logic such as agenda slot generation and hearing rescheduling. A usability study with municipal staff confirmed the value of adaptive user interfaces designed with the GenderMag framework. The case study demonstrates three main contributions: (i) the feasibility of executable DEMO models in practice, (ii) the expressiveness of extended Action Rules, and (iii) the added value of adaptive UIs. Together, these results advance executable enterprise modelling and provide practical insights for digital transformation in public administration.

Keywords

low-code, no-code, enterprise engineering, DEMO, action rules, information systems, modelling, UX, gendermag

1. Introduction

Municipalities and other public organizations are under constant pressure to be more transparent, more efficient, and to keep citizens satisfied, while at the same time dealing with legacy systems and complicated regulations. One of the recurring problems is how to turn the way organizations actually work into digital systems that support them. Traditional software development has a hard time with this: systems often end up too rigid, and every time a law or policy changes, a round of expensive adjustments follows.

Low-code/no-code platforms (LCPs) have been introduced as one way to deal with this. They offer visual environments where domain experts can create functional applications with minimal programming knowledge. The promise is faster results, less dependency on IT specialists, and more flexibility when things change. But most platforms still build on simplified workflows or ad-hoc models. Without a solid foundation, they tend to create fragmented solutions that do not really capture how an organization works as a whole [1].

The **Design and Engineering Methodology for Organizations (DEMO)** [2] tries to solve this by giving a way to describe organizations in terms of actors, transactions, facts, and rules. The method is clear and formal, which makes it useful as a basis for executable models. However, at the same time, DEMO's official Action Rule Specification is seen as unnecessarily complex and difficult to apply in practice, which makes it hard to use effectively.

PoEM2025: Companion Proceedings of the 18th IFIP Working Conference on the Practice of Enterprise Modeling: PoEM Forum, Doctoral Consortium, Business Case and Tool Forum, Workshops, December 3-5, 2025, Geneva, Switzerland
*Corresponding author.
†These authors contributed equally.
vitor.freitas@staff.uma.pt (V. Freitas); daveiro@staff.uma.pt (D. Aveiro); duarte.nuno@arditi.pt (D. Pinto)
0009-0002-0667-5749 (V. Freitas); 0000-0001-6453-3648 (D. Aveiro); 0000-0002-8451-5727 (D. Pinto)

In this paper we present the **Dynamic Information System Modeller and Executor (DISME)** [3, 4], a DEMO-based low-code/no-code platform that can run DEMO models directly. We use it on a concrete case, the **Hearings Management Process of a municipality**, to show how models can be turned into a working system without generating code. The novelty of our contribution lies in three aspects: first, we extend the DEMO Action Rule Syntax into a form that is both easier to use and more expressive, implemented through a block-based interface and a formal EBNF grammar; second, we demonstrate the parametrization of a complex public administration process into a running system that operates via direct runtime interpretation, thereby eliminating the need for code generation; and third, we evaluate the resulting system, including adaptive user interfaces designed with the GenderMag framework [5] to support different cognitive styles.

By combining the DEMO theory with practical tools, our work contributes to advancing the state of the art in executable modelling and low-code development, while also supporting digital transformation in the public sector. The rest of the paper is structured as follows: Section 2 provides background on DEMO and the DISME platform; Section 3 outlines the research context and methodology; Section 4 details the case study, followed by a deep dive into its Action Rule design in Section 5; Section 6 presents the user experience evaluation; Section 7 discusses results; and Section 8 concludes the paper.

2. Background and Related Work

2.1. Enterprise Modelling and DEMO

Enterprise modelling has always tried to close the gap between business requirements and information systems. Different notations have been used for this, such as BPMN [1], UML[2], and ArchiMate [3]. They help describe processes and rules, but often fall short: some lack ontological precision, others fail to capture the full organizational perspective [1].

DEMO [2] takes a different approach by offering a formal, ontologically grounded framework that models the essence of an organization through four interconnected models [2]. The Construction Model (CM) defines actor roles and the transactions they carry out, while the Process Model (PM) captures the conditions and causal relations that guide how these transactions unfold. The Fact Model (FM) serves as the semantic layer, specifying object classes and the facts that describe the state of the organization. Finally, the Action Model (AM) specifies the business rules that shape how actors behave at each step of a transaction [2].

This layered view highlights the essence of organizations while staying independent of technology [6]. It also guarantees consistency across models. The main difficulty lies in the Action Rule Specifications (ARS). They have long been seen as too complex and sometimes ambiguous, which discourages practitioners from using them in real projects [7, 8, 9].

2.2. DISME: A DEMO-based Low-Code Platform

DISME (Dynamic Information System Modeller and Executor) [3] was built to tackle that problem. It is an open-source, web-based low-code/no-code platform that makes DEMO models executable by combining the formal rigor of DEMO with the usability of drag-and-drop environments. Its architecture has two main parts. The first is the **System Modeller**, a design-time environment where domain experts can define and configure processes, transactions, roles, and entities. It includes a block-based Action Rules editor built on Google Blockly[4], a form manager (Form.io[5]) for user interfaces, and modules for REST API integration [10] and dynamic queries. The goal is to make complex system design accessible without programming [3]. The second part is the **System Executor**, a runtime engine that interprets

[1] https://www.omg.org/spec/BPMN/
[2] http://www.uml.org/
[3] https://www.archimatetool.com/
[4] https://github.com/google/blockly
[5] https://form.io/

models directly without code generation. Any change in business logic or data structures appears immediately, following the Adaptive Object Model (AOM) pattern [11, 12], which supports continuous and dynamic adaptation of the system [3].

2.3. Limitations of Current Approaches

Well-known low-code/no-code platforms include Mendix [6], OutSystems [7], and PowerApps [8]. They are powerful, but most depend on code generation pipelines and lack the semantic foundation of enterprise modelling. When enterprise models are mapped into such tools, manual adjustments are usually required. This often leads to semantic loss or repetitive detailing work [6].

Our approach differs fundamentally by allowing direct execution of ontologically-grounded DEMO models with an extended, executable Action Rule grammar. The result is a system where the organizational model remains the single authoritative specification. Additionally, usability enhancements such as adaptive interfaces (informed by the GenderMag framework) [5] address another common limitation of LCPs: the lack of cognitive adaptability and inclusivity for non-technical users [13, 14].

While other process-modelling standards such as BPMN or DMN focus on procedural and decision logic, DISME—grounded in DEMO—captures the underlying actor commitments and organizational transactions that define business reality. This distinction positions DISME as a complementary rather than competing approach: it emphasizes semantic integrity and direct executability over syntactic workflow representation. In the future, lightweight interoperability through data exchange or semantic mappings could allow DEMO-based executable models to coexist with mainstream process-management environments while preserving their ontological foundation.

3. Case Study: Municipality Hearings Process

This platform is being used in the context of the Smart Islands Hub's *Test-Before-Invest* services. To illustrate the paper, we use the example of the Municipality Hearings Process (MHP), a pilot project currently underway in this context. The MHP represents a typical citizen-facing workflow with multiple actors and complex scheduling, and in many municipalities such processes are still managed manually or through outdated systems. This situation often results in inefficiencies, duplicated effort, and a lack of transparency for the people involved. As part of the pilot, we have already conducted a user study on the MHP to evaluate the platform in practice.

The MHP's main Hearings Management process can be described in five main stages:

1. **Citizen Request**: A citizen asks for a hearing at the service desk. The clerk checks the request against the rules (for example, making sure it is not a duplicate or filed too recently).
2. **Assignment of Officer**: The clerk identifies the officer best suited for the case. In some cases the officer, or their assistant, must give explicit approval before a hearing can be scheduled.
3. **Scheduling**: Once approved, the hearing is placed in the officer's agenda. The citizen then receives an official confirmation by their chosen notification method.
4. **Rescheduling or Cancellation**: Either the citizen or the officer may request changes.
5. **Execution and Completion**: The hearing takes place, notes are recorded, and the case is closed.

In addition to these steps, clerks often need to print daily agendas, check a citizen's hearing history, or generate various notifications.

We modelled the MHP using DISME's adapted versions of the DEMO Process and Fact Model representations [15, 16]. These notations aim to keep the formal semantics but present them in a way that is easier for non-specialists to follow. Two diagrams are central here [4, 17]: the **Process Structure Diagram (PSD)**, which combines aspects of the Construction and Process Models to show tasks and

[6] https://www.mendix.com/
[7] https://www.outsystems.com
[8] https://www.microsoft.com/en-us/power-platform/products/power-apps

their coordination [15], and the **Concepts and Relationships Diagram (CRD)**, which provides a clear view of entities such as *Citizen*, *Hearing Officer*, *Agenda Block*, *Agenda Slot*, and *Hearing*, along with their main attributes and relations [16]. Due to space limitations, a more detailed version of these diagrams is provided in the Zenodo annex [18] (Process_Fact_Model_Annex.pdf), with some illustrative figures from the MHP case study.

Although DISME can represent the full MHP, in this paper we focus on the Action Rules that drive system behaviour. These rules govern the creation and editing of citizens and officers, the automatic generation of officer agenda slots, the scheduling of hearings with or without explicit officer authorization, and finally, the completion and rescheduling of hearings.

Among them, the *Rescheduling Hearing* rule stands out as the most complex. It coordinates several validations and updates at once. This rule is a good example of how Action Rules capture organizational logic beyond a simple workflow, embedding business semantics directly into the executable model.

4. Parametrization in DISME

The strength of the DISME platform lies in its ability to translate organizational specifications into executable artifacts without requiring traditional programming. This is achieved through a parametrization layer where roles, entities, processes, transactions, and forms are defined in accordance with DEMO. In the case of the Municipality Hearings Process (MHP), the parametrization made sure that all relevant elements of the process were captured and could run directly in the system's executor.

4.1. Roles and Users

The first step in parametrization was defining the roles involved in the hearings process. For the MHP we identified: Clerks — frontline staff responsible for receiving requests, validating them, and interacting with citizens; Hearing Officers — councilors, directors, or designated experts who conduct hearings; Assistants — staff members who support hearing officers, often responsible for agenda management; and Citizens — requesters of hearings, represented as external actors but included in the system for completeness.

Roles were then linked to individual users within the municipality, determining permissions across the system functions. This mapping ensured that each task in the workflow was tied to the appropriate human responsibility, thus operationalizing the Construction Model (CM).

4.2. Processes and Transactions

Building on the defined roles and thinking about the latter specified entities, the next step was to parameterize the processes and transactions that constitute the MHP workflow. DISME allows each transaction type to be associated with an initiating role, an executing role, and a business process it belongs to. For the MHP we specified the following processes and transactions:

- **Citizen Management Process**: `Create Citizen` and `Edit Citizen` transactions.
- **Officer Management Process**: `Create Hearing Officer` and `Edit Hearing Officer` transactions.
- **Hearing Official's Agenda Management Process**: `Create Agenda Block` transaction, which automatically generates a series of finer-grained agenda slots.
- **Hearing Management Process**: `Create Hearing`, `Schedule Hearing with Pending Authorization`, and `Complete Hearing` transactions. These cover the initial request, conditional scheduling (where officer approval is required), and finalization.
- **Rescheduling Hearing Process**: `Reschedule Hearing` transaction, which is the most complex, managing cancellations, slot reallocations, and notifications.

By associating transactions with their corresponding processes and roles, DISME provided a mapping between the organization's procedures and the implemented workflows, thereby operationalizing the Process Model (PM).

4.3. Entities

Entities in DISME correspond to the concepts of the DEMO Fact Model, serving as the backbone of the information structure. For the MHP, we parametrized several entities. The **Citizen** entity includes attributes such as name, fiscal number, contact details, and status. The **Hearing Officer** represents municipal staff with attributes such as area of expertise, section, and hearing location. The **Agenda Block** captures a block of availability for a hearing officer, including start and end dates, weekdays, and time duration, from which the **Agenda Slot** entity is automatically generated to represent specific bookable times. The **Hearing** entity encapsulates the request, the assigned officer, the scheduled slot, the subject, and any observations. Finally, the **Rescheduling** entity records the details of a rescheduling event, including the source (citizen or officer), the previous and new agenda slots, and the means of notification.

These entities are connected in the Concepts and Relationships Diagram (CRD), ensuring that dependencies (e.g., a Hearing entity requiring both a Citizen and a Hearing Officer) are made more explicit.

4.4. Forms and User Interfaces

Forms are mostly where end-users interact with the system. In DISME the ordering is important: modellers first define processes, transactions, entities and their properties; then they create *Action Rules* that express the behaviour for those transactions. Only when an Action Rule contains an action that requires user input (for example, `Create Instance` or `Update Instance` actions) does the modeller need to design a form.

Form authoring is performed using DISME's Form.io-based editor, which provides a WYSIWYG environment for placing fields, panels, tabs and other layout elements. Field definitions derive from the Fact Model properties that the Action Rule references, but the link between a form and a transaction is established explicitly at the Action Rule level: a form becomes part of a task when the rule references it. Validation and presentation logic can be split between the form editor (such as additional tooltips) and the Action Rule (conditional checks, cross-field validation, and other options implemented via Blockly).

This explicit, rule-driven form association ensures that the UI is only presented when behaviour requires it, and that validations and business logic remain expressed in the Action Rules rather than implicitly assumed by the parametrization.

After defining each task's action and business rules—which may themselves include the display of forms for user input—we used DISME's Form.io-based editor to create a set of customized forms. These included forms for citizen registration and editing, with field validations for mandatory attributes, and for hearing officer registration and editing, capturing both personal and organizational details. We also designed forms for agenda definition, where officers specify availability periods, and for hearing requests, where clerks can indicate the responsible officer and other case-specific data. In addition, hearings that require explicit officer approval are supported through appointment forms with pending authorisation. To complete the workflow, we developed forms for hearing completion reports, enabling officers to record outcomes, and for hearing rescheduling, allowing either the citizen or the officer involved to request and manage a change to a scheduled hearing.

These forms were dynamically linked to the entities and transactions defined earlier, ensuring that data integrity and business validations (expressed in the Action Rules) stayed consistent.

4.5. Runtime Interpretation

DISME runs everything at runtime — no compilation, no code generation. When an Action Rule points to a form, the system shows it immediately in the runtime dashboard. Field-level validators handle initial checks, while more elaborate business validations and conditional flows are evaluated by the interpreter.

From an execution-semantics perspective, each Action Rule is parsed into an internal execution tree that the runtime engine evaluates step by step. The interpreter resolves context variables, evaluates

conditional branches, and triggers entity or form actions as defined by the rule. Unlike typical low-code platforms that compile models into generated source code, DISME keeps the organizational model as the single executable specification, ensuring that every model change is immediately reflected in the running system and keeping implementation perfectly synchronized with the organizational specification, which enables fast feedback loops and continuous alignment between design and operation.

For the municipality case, this approach allowed clerks and officers to test a working version of the hearings system as soon as parametrization was done, accelerating the feedback cycle.

5. Action Rules: Defining System Behavior

The core of the MHP in DISME is the set of Action Rules (ARs). These rules define what the system does at each step of a transaction. They are created using a visual, Blockly-based editor, so the organization's business logic can be translated directly into something the system can run. This approach removes the need for coding and makes the system behavior clear and changeable by domain experts. Below, we walk through the key ARs that make the MHP work, showing how DISME's extended AR syntax [17] handles everything from simple entity creation to complex rescheduling.

5.1. Creating and Editing Citizens and Hearing Officers

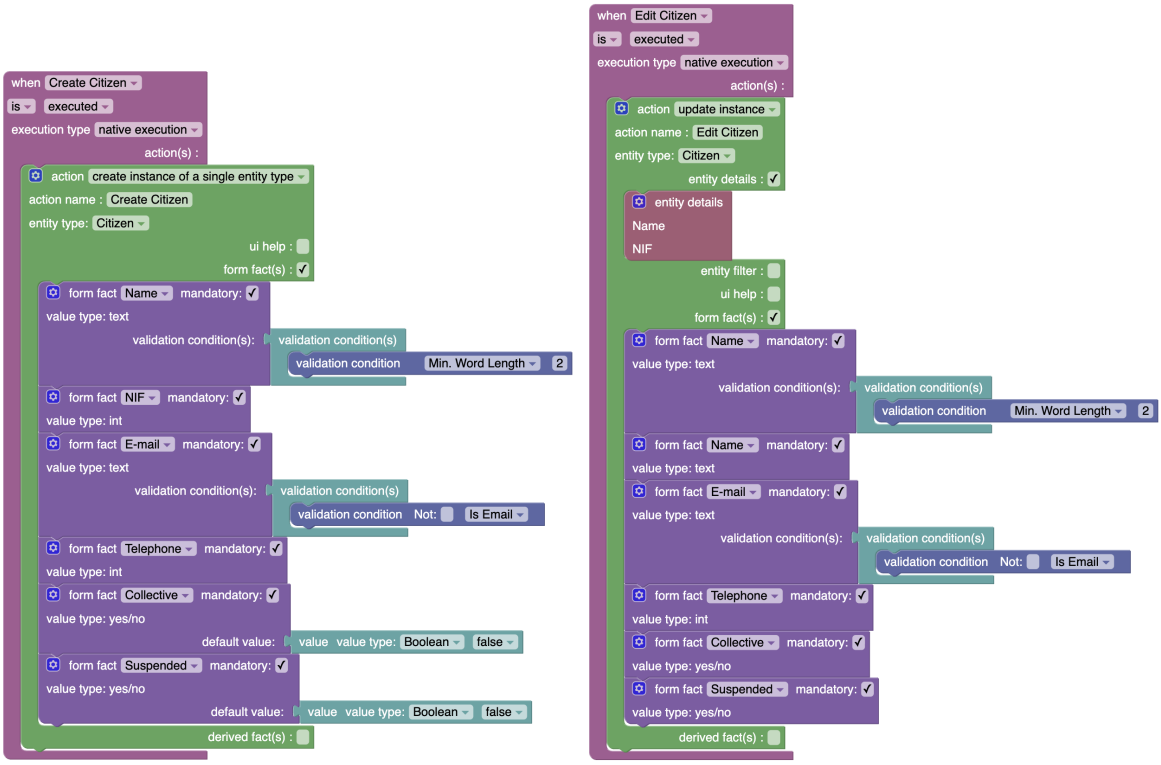

Figure 1: Action Rules for Hearing Officer Creation (left) and Editing (right)

The foundation of the MHP involves managing the main entities: Citizens and Hearing Officers. The **Create Citizen** and **Create Hearing Official** ARs (Figure 1) are triggered when these transactions are executed. They use the *create instance* action. The *form fact* blocks inside this action define the fields that will appear in the user-facing form, with each field typed and optionally marked as mandatory. Validation conditions enforce business constraints like minimum name length, proper email formatting, or other entity-specific rules. At runtime, when the form is submitted, a new entity instance is created in the system's database.

Editing existing entities is similar. The **Edit Citizen** and **Edit Hearing Official** rules use the *update instance* action. The system presents a form pre-filled with the entity's current data, lets the user make changes, and saves them back into the system. These editing actions ensure that firstly the correct entity is selected based on the process and transaction context, preserving data consistency.

5.2. Hearing Officers' Agenda Block and Slot Creation

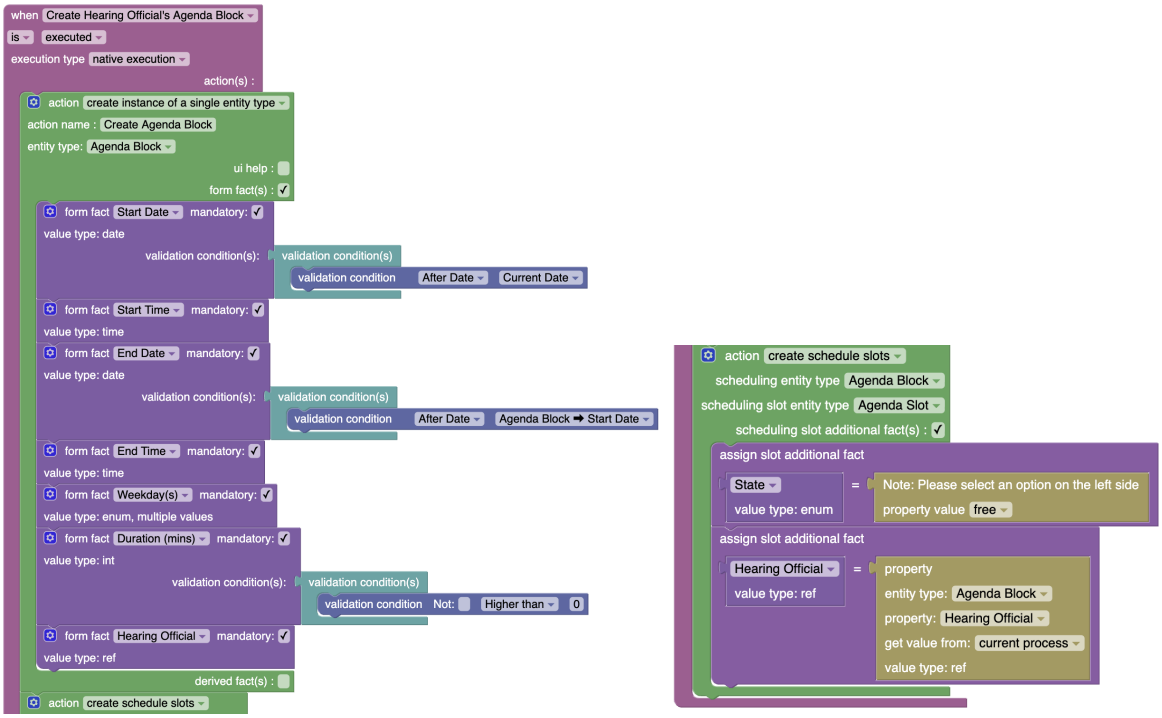

Figure 2: Action Rule for *Create Hearing Official's Agenda Block*

Scheduling hearings requires officers to define blocks of availability, from which individual slots are generated automatically. The **Create Hearing Officer's Agenda Block** AR (Figure 2) handles this in two stages. First, a *create instance* action captures the parameters of the Agenda Block, including start and end dates, recurring weekdays, and the duration of each hearing. Second, the *create schedule slots* action, an innovative extension in our AR grammar, automatically generates individual *Agenda Slot* instances based on the block parameters. An additional feature, *assign slot additional fact*, allows setting default values for each slot — such as *State = free* — along with other optional - non-essential for the agenda slots creation - properties.

5.3. Creating - and Scheduling - a Hearing

The **Create Hearing** AR (Figure 3) is more complex, as it combines context variables, queries, conditional flows, and state updates across multiple entities. A *set context variable* block stores the selected Hearing Officer for reuse throughout the AR. A subsequent *query* populates the Agenda Slot form field with only free slots for that officer, provided that the officer does not need to explicitly authorise these appointments in advance. Finally, if a slot is selected, an *update instance* action changes its state from *free* to *busy*, but only when the hearing has been directly scheduled — meaning no explicit officer approval was required and an Agenda Slot was actually chosen.

The **Hearing Appointment with Pending Authorisation** rule (Figure 4) handles cases where explicit approval is needed. The AR first selects a pending hearing, then updates the selected officer's free agenda slot and sets the Hearing's *State* to *Scheduled*. A second *update instance* action sets the Agenda Slot's state to *busy*. This shows how multiple dependent actions can be chained in a single AR

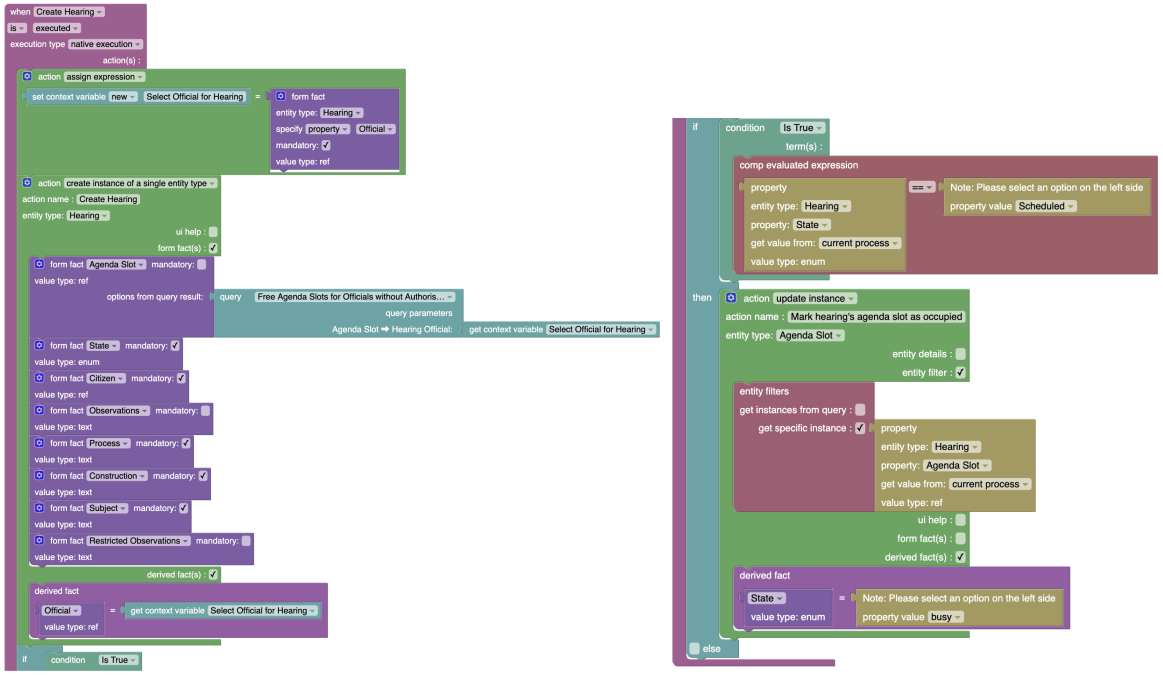

Figure 3: Action Rule for *Create Hearing*

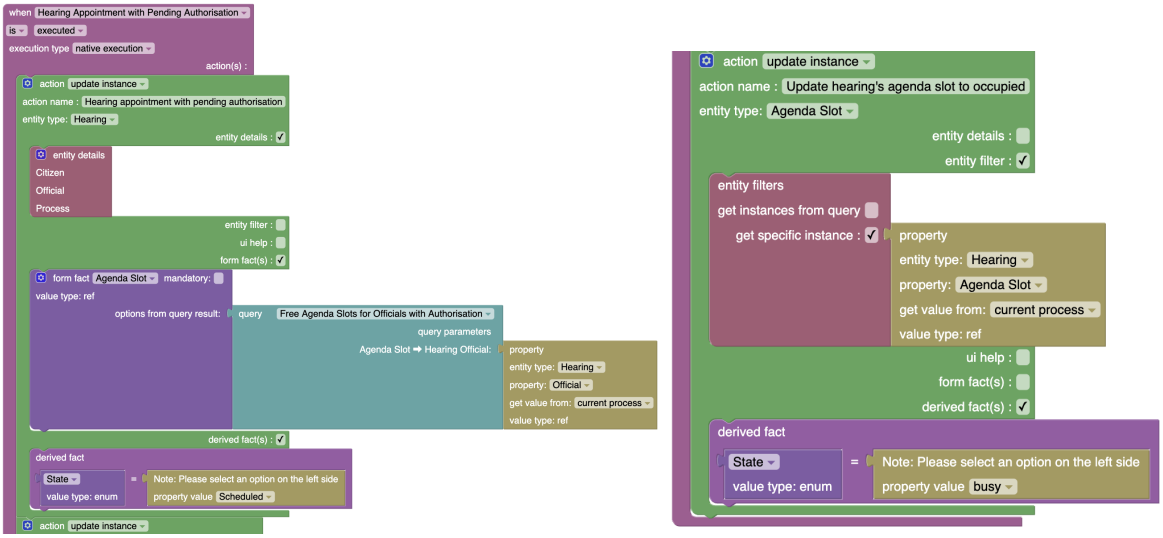

Figure 4: Action Rule for *Schedule Hearing with pending Authorisation*

to reflect organizational rules. The **Hearing Completion** rule (not shown) updates the Hearing's *State* to *Performed* and allows the clerk to add final general and restricted observations, officially finishing the process.

5.4. Rescheduling a Hearing

The **Reschedule Hearing** AR (see Zenodo annex [18], Figures S1, S2 and S3) is the most complex in the MHP, handling the full logic of moving an appointment. It begins by gathering context, storing the source of the rescheduling request — whether from a citizen or an officer — in a context variable (Figure S1 in Zenodo annex [18]). Next, an *if/then/else* block guides the system along different paths depending on the source. The system prompts the user to select the relevant hearing, using a query filtered by citizen or officer to ensure that only valid appointments are shown (Figure S1 in Zenodo

annex [18]). Once the hearing is selected, a new *Hearing Rescheduling* instance is created to log the details of the event, including the reason for the change, the previous slot, and the newly chosen slot. Some of this data is taken from the context variables set previously, to avoid definition repetition, while the rest is provided by the user (Figure S2 in Zenodo annex [18]). Finally, three *update instance* actions manage the states: the original Agenda Slot is reset to *free*, the new Agenda Slot is set to *busy*, and the Hearing is linked to the newly selected *Agenda Slot* (Figures S2 and S3 in Zenodo annex [18]).

This AR illustrates how DISME can handle complex transactions involving multiple entities (*Hearing*, *Agenda Slot*, *Hearing Rescheduling*), conditional logic, and coordinated state changes, all within a single visual, directly executable model.

6. User Interface and Usability

Besides running models, DISME also deals with a common problem in low-code platforms: making the system usable for people who are not technical [19, 20]. To do this, it includes adaptive user interfaces based on the **GenderMag framework**. The GenderMag framework provides a systematic method to assess and design software for inclusivity by analyzing cognitive styles represented through personas. These personas differ in motivation, information-processing style, and risk aversion, helping identify where interfaces may inadvertently favor or hinder certain user approaches [21]. For instance, the **Abi** persona is careful, detail-focused, and risk-averse, while the **Tim** persona is more exploratory and goal-driven, with less need for guidance. In DISME, this framework informed the design of adaptive interface variants tailored to different working styles by adjusting things like guidance, form layout, and help text depending on these styles. Due to space limitations, the Zenodo annex [18] (Forms_Annex.pdf) provides a detailed set of forms, comprising illustrative examples and persona-dependent variants.

6.1. Adaptive User Interfaces

Instead of offering a one-size-fits-all solution, DISME supports the selection of different interface variants. For hearings scheduling, agenda management, or rescheduling, one user might prefer a wizard-like step-by-step guide, while another prefers a compact form with an overview. By linking these options to GenderMag [5] personas, the platform supports different working styles in a way that feels more natural and inclusive.

6.2. Usability Evaluation

We conducted a usability study with 23 municipal employees to evaluate how well the adaptive interfaces supported realistic hearing scheduling tasks. The study compared two adaptive versions of DISME, labelled Version A and Version B.

Both versions achieved excellent usability, with mean System Usability Scale (SUS) scores above 85%, confirming that participants found the system highly usable. Workload ratings on the NASA-TLX scale averaged around 34%, indicating a low perceived effort during task execution. Task-success rates were moderate, at roughly 60%, which is consistent with expectations for complex administrative processes and shows that most participants could complete the main workflows independently. Although Version B scored slightly higher, the difference was not statistically significant, suggesting both variants performed consistently well..

Employees who used an interface aligned with their preferred cognitive style reported lower perceived cognitive workload, suggesting that tailoring the UI to individual thinking patterns can improve efficiency and reduce frustration. In addition, qualitative feedback highlighted the clarity of the forms, the logical flow of the tasks, and the simplicity of the overall user experience, demonstrating that the underlying complexity of the DEMO model was effectively hidden from end-users. The full results of this study are still being analyzed and will be submitted to a journal for publication.

These results suggest that human-centered design makes a real difference in DISME. Adaptive user interfaces are not just cosmetic; they help people work more efficiently and feel included. For

municipalities, this increases the chance that a low-code system, like DISME, will be accepted and used in daily practice in public administration.

7. Discussion, Conclusion and Future Work

The case study shows that the Municipality Hearings Process can be captured and run in a DEMO-based low-code platform. A few points stood out.

We were able to model the entire process — registration, scheduling, cancellations, and rescheduling — inside DISME without writing any code. Even though the process involves several functions and complex logic and conditional steps, it could all be represented directly. This suggests that processes with multiple actors and intricate coordination rules can work in a low-code/no-code setting if the platform is built on a solid methodology like DEMO.

Action Rules were really at the heart of the modelling of the hearings process. They handle validations, conditional flows, slot generation, and updates across multiple entities. This shows that the DEMO Action Model can be turned into something practically usable and expressive. Compared to typical low-code workflow languages, Action Rules let one spell out organizational logic directly, without needing technically complex and additional implementation details.

The usability study confirms that adaptive interfaces help municipal staff, with different ways of thinking, interact with the system. Many low-code platforms focus only on modeling efficiency, but the user side is often ignored. Considering cognitive diversity in DISME helps connect the model to everyday practice and makes the system more approachable for all users.

There are still some limitations to keep in mind. While the Action Rule editor is powerful, new users need training if they aren't familiar with model-driven concepts. Scaling up to larger departments with hundreds of officers and thousands of citizens has not yet been tested. Additionally, the evaluation so far has focused only on short-term usability, so it remains unknown how DISME will perform in daily, long-term use.

In this paper, we showed how the *Dynamic Information System Modeller and Executor (DISME)* can be used to model and run a Municipality Hearings Process. Built on DEMO, DISME allows organizational models to be executed directly, reducing the gap between design and implementation. The case study covered the full parametrization of a citizen-facing process and highlighted how Action Rules serve as executable business logic.

Our contributions can be summarized in three main points. First, we demonstrated that a DEMO-based low-code/no-code platform can fully operationalize a real-world complex public administration process. Second, we applied the extended DEMO Action Rules Specifications into an executable and expressive form, enabling advanced behaviors such as automatic agenda slot creation and hearing rescheduling. Third, we confirm that adaptive user interfaces can improve usability and inclusiveness for municipal staff, helping a diverse range of users interact effectively with the system.

Looking ahead, there are several directions for future work. One is to apply DISME in other domains, such as healthcare, licensing, or education, to test how well it generalizes. Another is to extend the Action Rule editor with semantic validation and recommendation features, potentially leveraging artificial intelligence to suggest rules or detect anomalies. Finally, we plan to explore collaborative modelling features, allowing multiple users to co-design and refine organizational models in real time.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT and Grammarly for grammar and spelling check, paraphrasing and rewording. After using these tools/services, author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] M. R. Krouwel, On the design of enterprise ontology-driven software development, Maastricht University, Maastricht, 2023. doi:10.26481/dis.20231103mk.

[2] J. L. G. Dietz, H. B. F. Mulder, The DEMO Methodology, in: J. L. G. Dietz, H. B. F. Mulder (Eds.), Enterprise Ontology: A Human-Centric Approach to Understanding the Essence of Organisation, Springer International Publishing, Cham, 2024, pp. 267–306. URL: https://doi.org/10.1007/978-3-031-53361-7. doi:10.1007/978-3-031-53361-7_12.

[3] V. Freitas, D. Pinto, V. Caires, L. Tadeu, D. Aveiro, The DISME low-code platform - from simple diagram creation to system execution, in: S. Guerreiro, C. Griffo, M. Jacob (Eds.), Proceedings of the 22nd CIAO! Doctoral Consortium, and Enterprise Engineering Working Conference Forum 2022, volume 3388 of *CEUR Workshop Proceedings*, CEUR, November, 2022. URL: https://ceur-ws.org/Vol-3388/paper4.pdf, iSSN: 1613-0073.

[4] D. Aveiro, V. Freitas, D. Pinto, Disme: A demo based model-driven low-code/no-code platform, in: 27th International Conference on Business Informatics - CBI Paralell Tracks (workshops, Forum, Tools Demo, etc.), LNBIP Series Springer International Publishing, Forthcoming, 2025.

[5] M. Burnett, S. Stumpf, J. Macbeth, S. Makri, L. Beckwith, I. Kwan, A. Peters, W. Jernigan, GenderMag: A Method for Evaluating Software's Gender Inclusiveness, Interacting with Computers 28 (2016) 760–787. URL: https://doi.org/10.1093/iwc/iwv046. doi:10.1093/iwc/iwv046.

[6] M. Krouwel, M. Op 't Land, H. Proper, From enterprise models to low-code applications: mapping demo to mendix; illustrated in the social housing domain, Software and Systems Modeling (2024) 1–28. doi:10.1007/s10270-024-01156-2.

[7] J. L. G. Dietz, DEMO Specification Language 4.7.2 – Enterprise Engineering Institute, 2022. URL: https://ee-institute.org/download/demo-specification-language/.

[8] D. Aveiro, V. Freitas, A new action meta-model and grammar for a demo based low-code platform rules processing engine, in: Advances in Enterprise Engineering XVI, Springer Nature Switzerland, Cham, 2023, pp. 33–52.

[9] A. Perinforma, The Essence of Organisation: An Introduction to Enterprise Engineering, Sapio Enterprise Engineering, 2015. URL: https://books.google.pt/books?id=XtyEAQAACAAJ.

[10] D. Aveiro, V. Caires, DEMO Model based Rapid REST API Management in a low code platform, Proceedings of the 22nd CIAO! Doctoral Consortium, and Enterprise Engineering Working Conference Forum 2022 co-located with 12th Enterprise Engineering Working Conference (EEWC 2022) 3388 (2022). URL: https://ceur-ws.org/Vol-3388/paper2.pdf.

[11] J. W. Yoder, F. Balaguer, R. Johnson, Architecture and design of adaptive object-models, ACM SIGPLAN Notices 36 (2001) 50–60. URL: https://doi.org/10.1145/583960.583966. doi:10.1145/583960.583966.

[12] J. W. Yoder, R. Johnson, The adaptive object-model architectural style, in: J. Bosch, M. Gentleman, C. Hofmeister, J. Kuusela (Eds.), Software Architecture: System Design, Development and Maintenance, IFIP – The International Federation for Information Processing, Springer US, Boston, MA, USA, 2002, pp. 3–27. URL: https://link.springer.com/chapter/10.1007/978-0-387-35607-5_1. doi:10.1007/978-0-387-35607-5_1.

[13] S. Käss, S. Strahringer, M. Westner, Drivers and inhibitors of low code development platform adoption, in: 2022 IEEE 24th Conference on Business Informatics (CBI), volume 01, 2022, pp. 196–205. doi:10.1109/CBI54897.2022.00028.

[14] D. Pinho, A. Aguiar, V. Amaral, What about the usability in low-code platforms? A systematic literature review, Journal of Computer Languages 74 (2023) 101185. URL: https://www.sciencedirect.com/science/article/pii/S259011842200082X. doi:10.1016/j.cola.2022.101185.

[15] D. Pinto, D. Aveiro, D. Pacheco, B. Gouveia, D. Gouveia, Validation of DEMO's Conciseness Quality and Proposal of Improvements to the Process Model, in: D. Aveiro, G. Guizzardi, R. Pergl, H. A. Proper (Eds.), Advances in Enterprise Engineering XIV, Lecture Notes in Business Information Processing, Springer International Publishing, Cham, 2021, pp. 133–152. doi:10.1007/978-3-030-74196-9_8.

[16] B. Gouveia, D. Aveiro, D. Pacheco, D. Pinto, D. Gouveia, Fact Model in DEMO - Urban Law Case and Proposal of Representation Improvements, in: D. Aveiro, G. Guizzardi, R. Pergl, H. A. Proper (Eds.), Advances in Enterprise Engineering XIV, Lecture Notes in Business Information Processing, Springer International Publishing, Cham, 2021, pp. 173–190. doi:10.1007/978-3-030-74196-9_10.

[17] D. Aveiro, V. Freitas, D. Pinto, V. Caires, D. Pacheco, Extending DEMO Action Rule Specifications' Syntax in a Low Code Platform Based Municipality Hearing System Implementation:, in: Proceedings of the 16th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, SCITEPRESS - Science and Technology Publications, Porto, Portugal, 2024, pp. 243–251. URL: https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0013068800003838. doi:10.5220/0013068800003838.

[18] V. Freitas, D. Aveiro, D. Pinto, Annex to: Bridging models and practice: Action rules in a demo-based low-code platform, 2025. URL: https://doi.org/10.5281/zenodo.17591315. doi:10.5281/zenodo.17591315.

[19] M. Bexiga, S. Garbatov, J. C. Seco, Closing the gap between designers and developers in a low code ecosystem, in: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 1–10. URL: https://dl.acm.org/doi/10.1145/3417990.3420195. doi:10.1145/3417990.3420195.

[20] J. Pacheco, S. Garbatov, M. Goulão, Improving Collaboration Efficiency Between UX/UI Designers and Developers in a Low-Code Platform, in: 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), 2021, pp. 138–147. URL: https://ieeexplore.ieee.org/document/9643662. doi:10.1109/MODELS-C53483.2021.00025.

[21] E. Murphy-Hill, A. Elizondo, A. Murillo, M. Harbach, B. Vasilescu, D. Carlson, F. Dessloch, Gendermag improves discoverability in the field, especially for women: An multi-year case study of suggest edit, a code review feature, in: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, 2024, pp. 1–12.