

Spatial Conceptual Modeling of Robotic Tasks in Virtual Reality Using Petri Nets

Naomi Feldmann^{1,*}, Daniel Borcard^{1,*}, Hans-Georg Fill^{1,*} and Simon Curty^{1,*}

¹University of Fribourg, Fribourg, Switzerland

Abstract

Conceptual models are widely used to design and control robotic tasks. Existing modeling environments in robotics are usually two-dimensional and separated from robot avatar simulations. This makes it difficult to represent spatial aspects of robotic behavior such as the robot's position and movement. We introduce a browser-based virtual reality environment that combines immersive Petri net modeling, execution, and synchronized robot behavior visualization. Users can create, modify, and execute Petri net models directly in virtual reality. The execution of the Petri net is connected to a robot avatar, which performs corresponding actions such as movement, pick, and place operations in real time. This way users can directly observe how the execution of a conceptual model translates into robot behavior.

Keywords

Conceptual Modeling, Robotics, Petri Net, Virtual Reality, Model Execution

1. Introduction

Programming robots to execute tasks can become challenging for non-experts due to the complexity of the robotic domain. To manage this complexity, model-driven approaches are increasingly used in robotics for task planning and execution, e.g [1, 2, 3]. Robot actions and their dependencies are then represented using formal or semi-formal models, which make the tasks easier to design, analyze and execute [4]. Similarly, virtual reality (VR) has also become a tool for robot programming [5]. Furthermore, modern web technologies such as Three.js and WebXR make it possible to create interactive 3D and VR applications directly in the browser without the need for a standalone application. Approaches such as spatial conceptual modeling [6] and the MM-AR metamodeling platform [7, 8] demonstrate how these technologies can be used to build accessible, immersive, three-dimensional modeling environments.

However, these developments have taken place rather independently of one another. Current approaches for designing robotic tasks interactively have several limitations: a. existing visual modeling approaches in robotics rarely use 3D representations, so users cannot create and interact with models in a three-dimensional or immersive environment [3]; b. VR-based robot programming systems typically have the user control the robot avatar directly, for example by manipulating a digital twin of the robot, rather than executing a conceptual model that drives the avatar's behavior; c. technologies such as WebXR and Three.js enable the creation of browser-based immersive environments based on open standards without commercial engines, but have rarely been applied to conceptual modeling and execution scenarios in robotics.

We present a modeling approach based on Petri nets. Although alternatives, such as BPMN, have been explored in previous works, we chose Petri nets because they are an established option for such use cases and have well-defined execution semantics and a simple notation. To the best of our knowledge, this paper presents the first web-based virtual reality environment that integrates immersive Petri net modeling, model execution, and synchronized robot behavior visualization.

The remainder of this paper is structured as follows. Section 2 reviews related work on Petri nets for

ER2026: Companion Proceedings of the 45th International Conference on Conceptual Modeling: Doctoral Consortium, Tutorials, Posters and Demos, October 5-8, 2026, St. John's, NL, Canada.

*Corresponding author.

✉ naomi.feldmann@unifr.ch (N. Feldmann); daniel.borcard@unifr.ch (D. Borcard); hans-georg.fill@unifr.ch (H. Fill); simon.curty@unifr.ch (S. Curty)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

robotic tasks, immersive conceptual modeling, and VR-based robot programming, and introduces the foundations our approach builds on. Section 3 presents the spatial extension of Petri nets and the design and implementation of the browser-based VR environment. Section 4 describes the pick-and-place scenario used for the demonstration. Section 5 discusses the results and outlines future work.

2. Related Work & Foundations

Petri nets are an established formalism for specifying and executing robotic tasks. For example, Costelha and Lima use Petri nets to model, analyze, and execute task plans based on the current state of the Petri net and sensor data [4]. Ebert et al. propose distributed runtime Petri nets for model-driven robotic applications in ROS [9]. Other approaches use Petri nets to generate robot control code [10] or to optimize robot motion plans [11]. However, these approaches rely on traditional 2D modeling tools or execute models without immersive visualization. More advanced techniques, such as 3D visualizations of models or explicit representations of orientation and direction, are rarely used in robotics [3].

Although not specifically applied to robotics, several approaches investigated three-dimensional or immersive conceptual modeling. Oberhauser et al. propose a concept for the visualization of and interaction with BPMN models in VR [12]. Abdul et al. combine 3D visualization of BPMN models with token-based execution [13]. Other systems allow users to create or manipulate UML models within virtual environments [14] or enterprise models in immersive settings [15]. VR has also been explored for robot programming. For example, Gorzelak and Grzejszczak present a VR environment for programming robot motions through direct manipulation of a robot arm [5]. Douthwaite et al. propose a interactive virtual representation of a robot through VR for safety analysis and control [16].

Our approach builds on *spatial conceptual modeling*, which integrates spatial computing concepts into conceptual modeling at the metamodeling level, so that coordinates, transforms, and 3D representations become part of the modeling language rather than of a particular tool [6]. Borcard and Fill apply this to robotics through a metamodel for the Unified Robot Description Format on the MM-AR platform, linking a BPMN pick-and-place process to the structure of a 6-DoF arm [17]. They note that the languages commonly used for robot behavior, including Petri nets, lack inherent concepts for three-dimensional space, which is the gap our prototype addresses.

3. Design and Implementation of Our Approach

Our approach supports the creation of Petri Net models in 3D, consisting of places, transitions, arcs, and tokens. Places represent system states or conditions and contain tokens that indicate the current state of the system. Transitions define actions that change the system state when they are executed. Arcs connect places and transitions and define the flow of tokens through the system. Places are represented as spheres, transitions as rectangular boxes, and arcs as directed arrows. Tokens are displayed inside places, and labels and arc weights are shown next to the corresponding elements. Places with different semantic roles are colored differently to distinguish them. Following standard Petri nets semantics, a transition is enabled if all its input places contain at least the number of tokens required by the corresponding arc weights. Firing an enabled transition consumes tokens from input places and produces tokens in output places. Our approach extends standard Petri nets with:

- A *role* assigned to each place (end-effector position, end-effector state, or object state).
- A *capability* assigned to each transition (move, pick, place).
- A *3D position* assigned to places and transitions.

The 3D position serves two purposes. It determines for each element where the element is placed in the virtual environment. For places with the end-effector position role, it is in addition interpreted as the target position of the robot: firing a move transition moves the end-effector to the position of the transition's output place. Repositioning such a place by drag-and-drop therefore changes the executed robot trajectory and not only the layout of the model. The robot can thus be seen as interacting

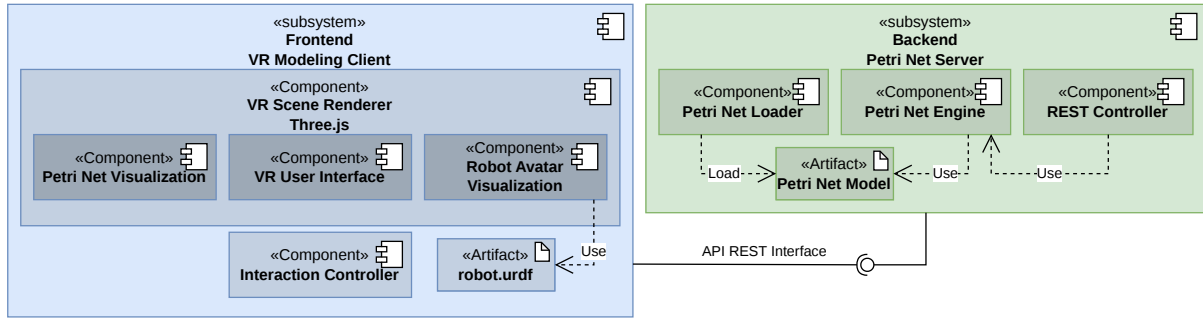


Figure 1: Architecture of the web-based VR environment.

with the model itself by moving between the positions defined by the Petri net, which improves the understanding of both the model structure and its execution. The prototype has been implemented as a web-based client-server application following a distributed MVC architecture. Figure 1 shows an overview of its main components.

The backend is implemented in Node.js with the Express framework and holds the model. The *Petri net loader* parses a JSON-based Petri net into the internal data structure. The *Petri net engine* determines the enabled transitions, executes them according to the firing semantics, and applies the operations for creating, deleting, and editing elements. The *REST controller* exposes both through an HTTP API.

The frontend is implemented with Angular and holds the view and the controller. The *Interaction controller* translates inputs from the VR controllers and the mouse into API calls and updates the scene with the state returned by the backend. The *VR scene renderer* builds and manages the 3D scene with Three.js and uses WebXR for the VR capabilities in the browser. Three components are rendered inside this scene: the *Petri net visualization*, the *robot avatar visualization*, which loads the robot geometry from a URDF file, and the *VR user interface* for the control and property panels.

3.1. Model Interaction

A virtual control panel is positioned next to the model. It dynamically displays the transitions that are enabled in the current state of the system, so users can fire transitions by clicking the transition buttons. Users can interact through both a mouse and VR controllers. In immersive mode, a ray is projected from the controller into the scene. Raycasting is used to determine which UI button or model element is selected. The control panel provides additional buttons for the following modeling operations: (i) creating places and transitions; (ii) connecting places and transitions through arcs; (iii) repositioning model elements through drag-and-drop; (iv) deleting places, transitions, and arcs; (v) editing labels, token counts, roles, capabilities, and arc weights; (vi) resetting the simulation to the originally loaded model; (vii) saving the current model as a JSON file.

When editing properties of the model elements, a second virtual panel is displayed. Text attributes are modified through a virtual keyboard, while token counts and arc weights are entered with a numeric keypad. An overview of the environment in a browser-based VR emulator is shown in Figure 2.

3.2. Robot Avatar and Model Execution

A six-degree-of-freedom robot arm defined using the Unified Robot Description Format (URDF) is integrated into the virtual environment. Its behavior is triggered by the capabilities of the executed Petri net transitions. For a move transition, the frontend takes the corresponding output place as the robot target position. The required joint configuration is computed using the Cyclic Coordinate Descent inverse kinematics method [18] so that the robot movements are smoothly animated. The pick and place capabilities currently provide visual feedback by temporarily changing the color of the robot's final joint, since the prototype does not yet include a representation of the gripper. Each fired transition therefore updates both the Petri net state/marking and the corresponding robot behavior.

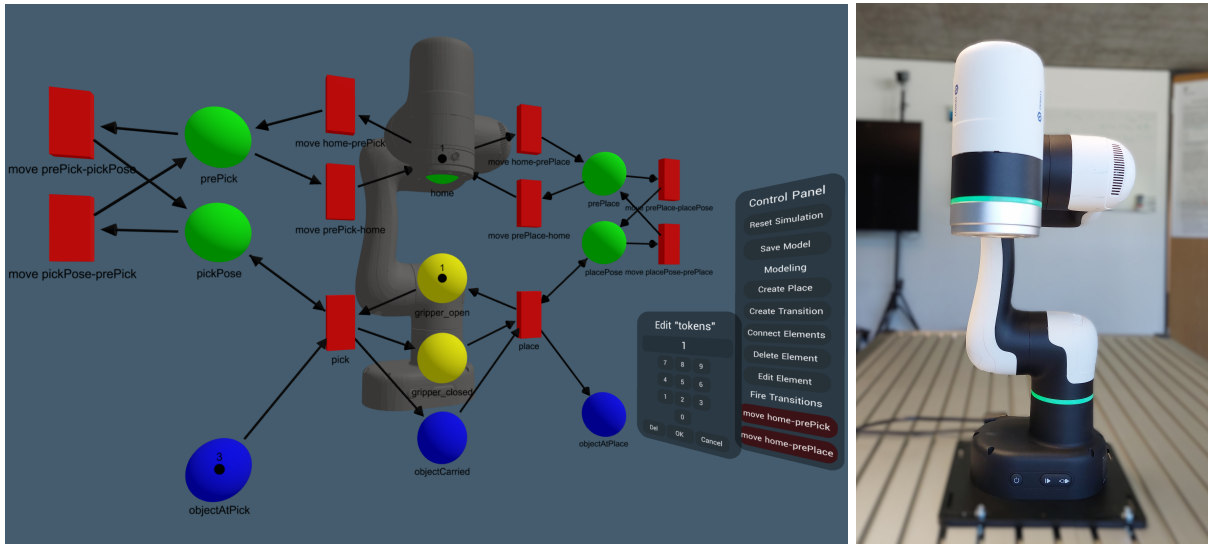


Figure 2: Virtual environment and corresponding physical robot. **Left:** Browser-based VR user interface with a Petri net, a robot avatar, and control panels. The colors of places indicate their role: green for end-effector position, blue for object state, and yellow for end-effector state. **Right:** Physical robot with matching pose.

4. Demonstration

The prototype is demonstrated using a robotic pick-and-place scenario. A predefined Petri net controls a robot avatar that moves towards an object, picks the object, moves to a destination, and places the object. The demonstration includes: Execution of transitions and token flow; synchronized robot movements; execution of the pick and place capabilities; drag-and-drop repositioning of a place; execution of the modified model; creation, and editing of a model element. The prototype is demonstrated using a Meta Quest 3 headset and can also be executed in desktop mode or through the browser-based VR emulator. The screencast with English voice-over, source code, and archived prototype are available online¹.

5. Conclusion

In this paper, we presented a browser-based virtual reality environment for immersive Petri net modeling, model execution, and synchronized robot behavior visualization. The prototype demonstrates how executable conceptual models can be directly linked to observable robot behavior within a single immersive environment. Future work includes improved VR interaction and connecting the virtual robot avatar to a physical robot.

Declaration on Generative AI

ChatGPT and DeepL were used to assist with text condensation, spelling, grammar, and language refinement. These tools did not influence the content, the design, or the results of this work.

References

- [1] F. Corradini, S. Pettinari, B. Re, L. Rossi, F. Tiezzi, A BPMN-driven framework for multi-robot system development, *Robotics and Autonomous Systems* 160 (2023) 104322. doi:10.1016/j.robot.2022.104322.

¹Screencast: https://youtu.be/SKl8_BAcLF0; GitHub Repository: <https://github.com/naomifeldmann777/my-app-bachelor>; Zenodo Repository: <https://doi.org/10.5281/zenodo.20121666>

- [2] G. L. Casalaro, G. Cattivera, F. Ciccozzi, I. Malavolta, A. Wortmann, P. Pelliccione, Model-driven engineering for mobile robotic systems: a systematic mapping study, *Software and Systems Modeling* 21 (2022) 19–49. doi:10.1007/s10270-021-00908-8.
- [3] D. Borcard, H.-G. Fill, State of the art and research directions for visual conceptual modeling in robotics, in: *Enterprise, Business-Process and Information Systems Modeling*, Springer, Cham, 2025, pp. 415–430. doi:10.1007/978-3-031-95397-2_26.
- [4] H. Costelha, P. Lima, Robot task plan representation by petri nets: modelling, identification, analysis and execution, *Autonomous Robots* 33 (2012) 337–360. doi:10.1007/s10514-012-9288-x.
- [5] M. Gorzelak, T. Grzejszczak, Controlling and programming a robot using virtual reality, in: *2024 28th International Conference on Methods and Models in Automation and Robotics (MMAR)*, IEEE, Poland, 2024, pp. 241–246. doi:10.1109/MMAR62187.2024.10680817.
- [6] H.-G. Fill, *Spatial Conceptual Modeling: Anchoring Knowledge in the Real World*, Springer, 2024, p. 35–50. doi:10.1007/978-3-031-56862-6_3.
- [7] F. Muff, H.-G. Fill, M2ar: A web-based modeling environment for the augmented reality workflow modeling language, in: *ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, ACM, 2024, pp. 1–5. doi:10.1145/3652620.3687779.
- [8] F. Muff, D. Borcard, H.-G. Fill, MM-AR: a web-based open-source metamodeling platform for spatial conceptual modeling, *Software and Systems Modeling* (2026). doi:10.1007/s10270-025-01351-9.
- [9] S. Ebert, J. Mey, R. Schöne, S. Götz, U. Aßmann, Distributed petri nets for model-driven verifiable robotic applications in ros, *Innovations in Systems and Software Engineering* 20 (2024) 531–557. doi:10.1007/s11334-024-00570-5.
- [10] G. Gelen, Y. İçmez, Task planning and formal control of robotic assembly systems: A petri net-based approach, *Ain Shams Engineering Journal* 15 (2024) 102804. doi:10.1016/j.asej.2024.102804.
- [11] T. Bando, T. Nishi, M. M. Alam, Z. Liu, T. Fujiwara, Automatic generation of optimization model using process mining and petri nets for optimal motion planning of 6-dof manipulators, in: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Kyoto, Japan, 2022, pp. 11767–11772. doi:10.1109/IROS47612.2022.9982201.
- [12] R. Oberhauser, C. Pogolski, A. Matic, Vr-bpmn: Visualizing bpmn models in virtual reality, in: *Business Modeling and Software Design*, volume 319, Springer, Cham, 2018, pp. 83–97. doi:10.1007/978-3-319-94214-8_6.
- [13] B. Abdul, F. Corradini, B. Re, L. Rossi, F. Tiezzi, Ubba: Unity based bpmn animator, in: *Information Systems Engineering in Responsible Information Systems*, 2019, pp. 1–9. doi:10.1007/978-3-030-21297-1_1.
- [14] M. Stancek, I. Polasek, T. Zalabai, J. Vincur, R. Jolak, M. Chaudron, Collaborative software design and modeling in virtual reality, *Information and Software Technology* 166 (2024) 107369. doi:10.1016/j.infsof.2023.107369.
- [15] F. Muff, H.-G. Fill, Initial Concepts for Augmented and Virtual Reality-based Enterprise Modeling, in: *Proceedings of the ER Demos and Posters 2021*, volume 2958 of *CEUR Workshop Proceedings*, CEUR, St. John's, NL, 2021, pp. 49–54. URL: <https://ceur-ws.org/Vol-2958/#paper9>.
- [16] J. Douthwaite, B. Lesage, M. Gleirscher, R. Calinescu, J. M. Aitken, R. Alexander, J. Law, A Modular Digital Twinning Framework for Safety Assurance of Collaborative Robotics, *Frontiers in Robotics and AI* 8 (2021) 758099. doi:10.3389/frobt.2021.758099.
- [17] D. Borcard, H.-G. Fill, Towards spatial conceptual modeling for robotic digital twins based on URDF, in: *PoEM Companion Proceedings 2025*, volume 4171 of *CEUR Workshop Proceedings*, CEUR, Geneva, Switzerland, 2025. URL: https://ceur-ws.org/Vol-4171/paper_59.pdf.
- [18] L.-C. Wang, C. Chen, A combined optimization method for solving the inverse kinematics problem of mechanical manipulators, *IEEE Transactions on Robotics and Automation* 7 (1991) 489–499. doi:10.1109/70.86079.