*Bull. Nov. Comp. Center*, Comp. Science, 48 (2024), 33–46


# Methodology for using Cellular Automata simulators in engineering-physics modeling systems: converting COMSOL multiphysics model data representations into CATLIB formats

A.A. Korolev

**Abstract.** Computer-Aided Engineering (CAE) systems face significant challenges in modeling complex, nonlinear, and discrete engineering phenomena. Traditional numerical methods based on partial differential equations struggle to accurately simulate systems with intricate spatial dynamics and stochastic behaviors.

This research introduces a novel methodology for integrating Cellular Automata (CA) models with existing CAE platforms, specifically focusing on converting data representations between COMSOL Multiphysics and the CATLIB cellular automata library. The study addresses critical interoperability limitations by developing a specialized converter tool that enables seamless data translation between different computational paradigms. Utilizing COMSOL's Java API and the Python MPh library, the research demonstrates a robust approach to extracting and transforming complex geometrical and physical simulation data. The proposed solution aims to enhance the accessibility of cellular automata modeling techniques for engineers and researchers with varying levels of technical expertise.

Key contributions include a framework for comparative analysis between traditional CAE solutions and cellular automata models, a methodology for bridging rule-based and equation-based solvers, and a proof-of-concept implementation that showcases the potential of integrating alternative modeling approaches. The research successfully extracts critical simulation parameters, including geometric boundaries, mesh configurations, and initial conditions, from COMSOL models for subsequent CA processing. Experimental validation using a turbulence formation simulation demonstrated the converter's capability to handle complex engineering scenarios. The methodology not only addresses current limitations in computational modeling but also provides a foundation for more flexible and comprehensive approaches to engineering-physics simulations. This work represents a significant advancement in computational modeling techniques, offering a standardized approach to integrating cellular automata models into mainstream engineering simulation platforms. The research has broader implications for interdisciplinary innovation, potentially transforming how engineers and researchers approach the modeling of complex, dynamic systems.

**Keywords:** Cellular Automata, Computer-Aided Engineering, COMSOL Multiphysics, Simulation Modeling, Data Conversion

## Introduction

CAE systems have become indispensable tools for modeling complex engineering and physical processes. However, these systems face significant limitations when addressing highly nonlinear, discrete, and stochastic phenomena. Traditional numerical methods, such as Finite Element Methods (FEM) and Finite Volume Methods (FVM), struggle to accurately simulate dynamic systems characterized by intricate spatial interactions and complex behavioral patterns.

CA models offer a promising alternative approach to these computational challenges. With their ability to represent localized interactions and model emergent behaviors, CA models can provide insights into systems that conventional CAE tools find intractable. Despite their potential, the integration of CA models into mainstream engineering simulation platforms remains a critical technological barrier.

This research addresses this gap by developing a novel converter and simulation tool designed to facilitate seamless interoperability between COMSOL Multiphysics and CATLIB, a cellular automata library. The primary objectives are to:

1. Enable smooth data translation between different modeling paradigms.
2. Enhance the accessibility of CA modeling techniques for engineers with limited technical expertise.
3. Provide a framework for comparative analysis between traditional and cellular automata-based simulation approaches.

## 1. Overview of CAE Systems for Engineering and Physics

The integration of CA models with CAE systems presents both an opportunity and a challenge in modern engineering-physics simulations. To contextualize the proposed research, this section reviews the principles, functions, and limitations of current CAE systems, as well as the applications and challenges associated with CA models. This dual focus highlights the technological gap this research aims to address.

**1.1. Key functions of CAE systems based on differential equation solvers.** Modern CAE systems offer a broad range of functionalities for modeling engineering and physical processes. At their core, these systems rely on numerical methods for solving partial differential equations (PDEs), which describe the dynamics and behavior of various systems in space and time. The primary functions of such systems include:

1. Geometry construction and processing.
2. Mesh generation.

3. Defining physical parameters and boundary conditions.
4. Numerical solvers for analysis.
5. Post-processing and visualization of results.
6. Support for multi-physics and multi-tasking models.
7. Integration with High-Performance Computing (HPC).

These functionalities make CAE systems versatile tools for addressing complex engineering and scientific problems. However, as studies (e.g., Bandman's paper [1]) suggest, many challenges related to nonlinear or highly discrete processes remain difficult for classical numerical methods. This highlights the necessity for alternative approaches, such as cellular automata modeling, to complement traditional CAE solutions.

**1.2. Examples and Analysis of Popular CAE Systems.** Here, we examine several leading CAE platforms, their distinctive features, supported formats, and areas of application.

**MFEM** is a lightweight, scalable C++ library tailored for finite element methods. Its modular design supports integration with a variety of mesh formats such as VTK, Gmsh, and CUBIT, making it versatile for mesh generation and visualization. With output options like ParaView (VTU), VisIt, and GLVis, MFEM excels in offering seamless visualization pathways for high-performance simulations in structural mechanics and thermal analysis.

**GetFEM** stands out for its emphasis on versatility in finite element analysis. Its compatibility with widely used formats like Gmsh and Ansys for mesh input, combined with output options such as VTK and OpenDX, ensures smooth integration into diverse workflows. This flexibility makes GetFEM particularly suitable for research-driven projects requiring customized solutions.

The **MOOSE** framework is designed for multiphysics simulations, offering extensive support for mesh formats like ExodusII, Abaqus, and Tecplot, among others. Its output capabilities, including VTK (.pvd, .vtu) and XDMF/HDF5, facilitate robust post-processing and visualization. MOOSE's strength lies in its ability to couple various physical phenomena, making it a powerful tool for solving complex engineering problems.

**OpenModelica** specializes in dynamic system simulations, particularly for mechanical, electrical, and hydraulic systems. Using Modelica (.mo) and XML-based FMI (.fmu) formats for model representation, it offers a standardized approach to model exchange and simulation. With output options such as MATLAB (.mat), CSV, and plain text, OpenModelica integrates well with popular data analysis tools like Python and MATLAB.

**SU2** is a comprehensive open-source computational fluid dynamics (CFD) tool with strong capabilities in aerodynamic shape optimization. Supporting formats like SU2 grid (.su2), CGNS, and OpenFOAM, it facilitates efficient integration into CFD workflows. SU2's robust visualization support, including Tecplot (.dat) and ParaView (.vtk), makes it an excellent choice for fluid dynamics and multidisciplinary optimization tasks.

**FEniCS** is a flexible computing platform for solving partial differential equations. It supports numerous mesh formats, such as Gmsh, XDMF, and Triangle, and employs Python-based scripting for problem definition, enhancing accessibility for users. Its output capabilities through ParaView and VTK further extend its application in finite element analysis, making it a favorite in academic and research settings.

**Elmer** offers robust multiphysical simulation capabilities, particularly in electromagnetics, structural mechanics, and fluid dynamics. With mesh formats such as Gmsh and unstructured VTK, and output options including Tecplot and ParaView, Elmer provides flexibility and precision for advanced simulation needs.

**Deal.II** is a C++ library focused on adaptive finite element methods. Its compatibility with formats like Gmsh, Tecplot, and VTK, along with CAD format integration via IGES and STEP, makes it a powerful tool for developing scalable simulations. Its emphasis on adaptive meshing ensures efficiency in solving large-scale problems with complex geometries.

**Code-Aster**, coupled with its pre- and post-processing environment **Salome-Meca**, is a prominent open-source solution for structural mechanics and coupled physics simulations. Supporting various formats for mesh and results representation, it caters to demanding industrial applications, including thermal and fluid flow analysis.

The diversity of CAE systems reflects the varied needs of modern engineering and scientific research. Each platform offers unique capabilities and supports a range of formats, enabling users to choose the tools that best match their simulation requirements. From general-purpose solvers like MFEM and MOOSE to specialized platforms like SU2 and OpenModelica, these systems continue to drive innovation and efficiency in modeling and simulation.

**1.3. Limitations and bottlenecks in current CAE systems for modeling complex systems.** Despite the extensive capabilities of modern CAE systems, they encounter several limitations that hinder their effectiveness in modeling complex engineering and physical processes. These bottlenecks encompass methodological challenges, performance constraints, and interoperability issues, which collectively affect their utility in advanced simulations.

**Methodological challenges.** CAE systems predominantly rely on numerical solvers like FEM and FVM. While effective for problems with smooth solutions, these methods face significant hurdles in addressing:

1. Highly nonlinear systems: Turbulent flows, multiphase systems, and reaction-diffusion processes demand substantial computational resources to ensure stability and convergence. Traditional numerical approaches often struggle with inefficiencies and inaccuracies when tackling such nonlinear complexities.
2. Discrete and stochastic phenomena: Problems governed by discrete dynamics or stochastic behaviors fall outside the typical scope of numerical solvers. Cellular automata, which excel in modeling such phenomena, require fundamentally different computational frameworks that are rarely supported by conventional CAE tools [1, 2].
3. Performance issues: High computational costs and parallelization bottlenecks.
4. Diverse data formats: Proprietary data formats (e.g., CGNS, VTK) used by CAE systems create challenges for cross-platform data exchange, often leading to data loss during conversions.
5. Scalability and adaptation challenges.

**The role of cellular automata in addressing limitations.** These challenges highlight the need for alternative methods to complement traditional CAE approaches. Cellular automata offer distinct advantages in handling discrete, nonlinear, and stochastic processes. Their inherent scalability and adaptability make them well-suited for integration with modern visualization and data management tools. Moreover, CA align effectively with the demands of high-performance computing, offering viable solutions for large-scale and complex simulations [3].

While mathematical physics and partial differential equations form the backbone of traditional CAE systems, they face growing difficulties in managing the increasing complexity of engineering problems. Implicit methods, typically used for stationary problems, are stable but challenging to parallelize, whereas explicit methods for non-stationary problems are easier to parallelize but limited by strict stability constraints, such as requiring smaller time steps. These limitations align explicit methods with the behavior of continuous cellular automata, positioning CA as a robust alternative for specific applications [2].

By leveraging the unique capabilities of CA, future CAE systems can overcome existing limitations, paving the way for more efficient and accurate modeling of complex engineering phenomena.

**1.4. Advantages and disadvantages of using CA for engineering modeling.** Unlike numerical methods that approximate PDE solutions, CA models provide an entirely different paradigm. This distinction is particularly relevant for highly nonlinear systems, where traditional numerical methods struggle with complexity, and CA models can offer simpler, more effective alternatives [2,3]. However, this advantage is context-dependent, as the comparative utility of CA and PDE-based methods varies significantly by application.

Several critical advantages of CA modeling make it attractive. First, CA models eliminate rounding errors, as their state representations are based on Boolean vectors. This ensures that all computational bits are equally significant, avoiding the cumulative inaccuracies that iterative numerical methods may introduce. Additionally, CA models are memory-efficient, particularly for large datasets, as Boolean states occupy minimal storage. Their inherent parallelism also supports extensive scalability, enabling domain decomposition across any number of processors without performance degradation. Moreover, boundary conditions in CA models are easier to implement, even for complex geometries like porous media or curved structures, as they are represented by specialized boundary cells with transition rules [4].

Despite these strengths, CA modeling is not without limitations. One significant issue is automaton noise, which arises from the discrete representation of states. While this can often be mitigated with sufficient averaging radii, it remains a concern in high-resolution models. Furthermore, CA lacks formal predictive tools for designing transition functions, making it difficult to synthesize these functions to achieve specific desired outcomes. This limits the ease with which CA models can be systematically tailored to match the behavior of physical systems.

Lastly, the theoretical foundation of CA, while promising, is still in its nascent stages compared to the well-established body of knowledge supporting numerical methods. The most critical insight when comparing CA with traditional methods is that CA does not merely approximate PDEs but offers an alternative framework, better suited to certain classes of problems, especially those characterized by non-reversible, nonlinear dynamics [5].

**1.5. Related research done in Novosibirsk State University.** This work presents the development of algorithms for integrating geometric, boundary condition, and visualization data into CA framework implemented as a plugin for the FreeCAD software. These algorithms enable the seamless translation of model geometry and simulation parameters into the CA computational domain enhancing the capabilities of FreeCAD for physical process modeling [6].

The work focuses on algorithms for converting geometric data into CA models, specifically targeting integration with the LOGOS CAE system.

The developed methods facilitate the translation of raster images and finite element geometries into the FHP-MP CA framework enabling efficient simulation setup [7].

## 2. Problem statement

**2.1. Problem description.** The integration of CA models into CAE systems faces significant challenges due to a lack of dedicated tools and streamlined processes. Existing CAE systems, such as COMSOL Multiphysics, primarily rely on solvers based on differential equations, which often do not support the flexible, rule-based computation inherent to CA models. This creates a technological and operational gap, leaving engineers and researchers without user-friendly solutions to leverage the strengths of both CA and CAE systems.

This deficiency limits the practical adoption of CA modeling techniques in engineering-physics simulations, where CA models could provide innovative approaches to solving dynamic system problems. A seamless method for integrating CA with established CAE systems is urgently needed to bridge this gap, especially for users with limited technical expertise in programming or CA-specific methodologies.

**2.2. Research objectives:**

1. Design and develop a converter and simulation tool that enables smooth interoperability between COMSOL and CATLIB.
2. Ensure accessibility for beginners in CAE systems, providing them with intuitive tools to adopt CA-based modeling techniques without extensive technical overhead.
3. Facilitate comparative analysis by enabling researchers to directly evaluate CA models alongside solutions generated by conventional CAE systems, fostering deeper insights into their relative strengths and limitations.

**2.3. Significance of the research.** This research contributes to both the scientific and engineering communities by providing a standardized methodology for integrating CA with CAE systems. The proposed solution aims to:

1. Enhance the usability and flexibility of CA models for practical engineering applications, making them accessible to a broader range of users.
2. Address a critical gap in the integration of rule-based and equation-based solvers, promoting interdisciplinary innovation in engineering-physics simulations.

By improving workflows for researchers and engineers, this study paves the way for more effective utilization of CA in solving complex, dynamic problems across various scientific and engineering domains [6–8].

## 3. Integration of CA with CAE System COMSOL

**3.1. Requirements definition.** The primary objective is to develop a converter capable of translating data representations from COMSOL Multiphysics simulations into a format compatible with the CATLIB cellular automata library. Architecturally, this converter replaces COMSOL's solver with the CATLIB solver. Additionally, the converter must function bidirectionally, enabling data transformation back into COMSOL's expected format for further processing (Figure 1).

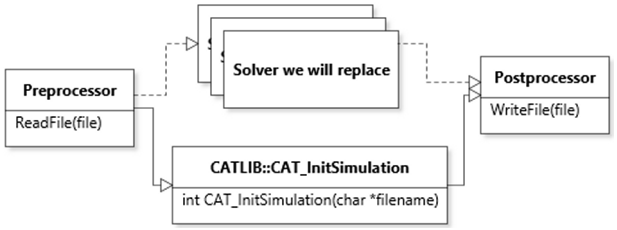


**Figure 1.** Diagram with the outline of the proposed architecture

This converter serves as both a pre- and post-processing tool for COMSOL's MPH models. Ideally, the solution should seamlessly integrate CATLIB within COMSOL, offering users an uninterrupted modeling experience while leveraging CATLIB's solver capabilities.

**3.2. Structure of MPH models.** COMSOL's MPH/MPHBIN files were analyzed to understand their structure and content. The key elements include:

- Model geometry: Definitions of geometric forms, parameters, and information on various regions and boundaries.
- Simulation settings: Selected physical phenomena, equations, mesh configurations, and numerical methods.
- Initial and boundary conditions: Specifications for initial parameter values and boundary equations.
- Simulation results: Data distributions (e.g., fields, temperatures, pressures) and visualizations.

The proprietary binary nature of these file formats complicates direct manipulation, necessitating efficient parsing techniques or API usage.

**3.3. Approaches considered:**

Manual parsing of MPH files — an initial approach involved manually loading and parsing MPH files line-by-line as text. While this avoided reliance on external APIs, it lacked flexibility and compatibility with COMSOL's binary MPHBIN files.

Explicit export by COMSOL users — exporting COMSOL data to open formats such as XML or CSV for subsequent processing. However, this added complexity and was limited in scope, as not all required data is exportable to CSV.

Using the COMSOL Java API — the final and preferred approach is the utilizing COMSOL's official Java API. Its advantages include robust documentation and flexibility. Challenges encountered during implementation are discussed in later sections.

**3.4. Selection of COMSOL version.** The integration was focused initially on COMSOL version 4.3a due to licensing considerations. However, significant compatibility issues arose between the required Java environment and the API. These included challenges with javac compiler versions and consistent library dependencies. To address these challenges, we shifted to the newer versions of COMSOL (e.g., 6.2). Major differences between API versions required adapting methods for model loading, data extraction, and content manipulation.

**3.5. Selected libraries and their benefits.** The Python library MPh was chosen as the primary tool for working with the COMSOL Java API due to several advantages:

- Simplified Java environment setup: MPh utilizes Jpype to bridge Python with Java, encapsulating the complexities of Java setup.
- Dependency management: Installation of required libraries, such as NumPy, is streamlined through Python's pip package manager.
- API reflection: MPh directly mirrors COMSOL's Java API functionality, simplifying development.
- Reproducibility: The use of Python enhances the reproducibility and portability of the solution.

**3.6. Proof-of-concept program and data extraction.** A proof-of-concept (PoC) program was tested using COMSOL's built-in model library. A specific model simulating turbulence formation around a vertical cylinder in a laminar flow was used (Figures 2 and 3). This example provides valuable insights into MPH file structure and MPh library functionality.

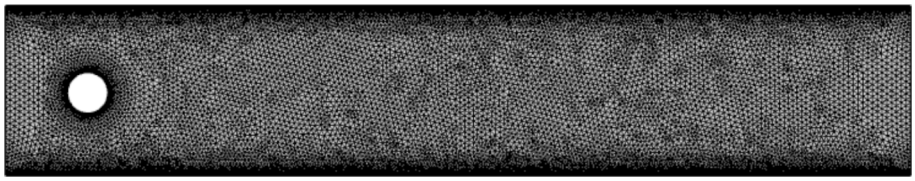

**Figure 2.** The raw model mesh displayed in COMSOL

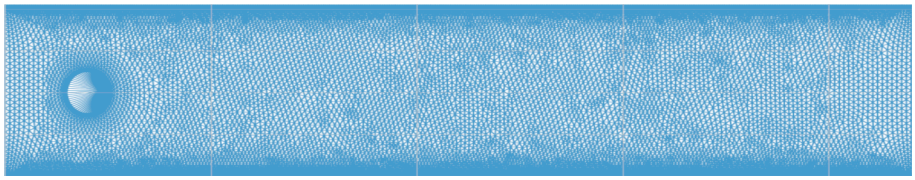

**Figure 3.** The raw mesh extracted and reconstructed with our software

Extracted data from the PoC program included:

- Geometries: Rectangles, circles, unions, and more.
- Physics: Fluid properties, initial values, and boundary conditions for laminar flow simulations.
- Meshes: Raw vertex coordinates and structural elements.
- Plots: Velocity surfaces and particle tracing.
- Boundary conditions: Inlets, outlets, and walls.

The code implements a method to extract boundary condition data (inlets, outlets, and walls) from a COMSOL MPH model:

```
def extract_boundary_conditions(self):
    """Extract boundary condition data such as inlets and outlets."""
    data = { 'inlets': [], 'outlets': [], 'walls': [] }
    edges = self.get_edges_vertices()
    for tag in self.spf.feature().tags():
        feature = self.spf.feature(tag)
        bc_type = feature.getType()
        selection = feature.selection()
        entities = selection.entities()
        # For each entity, check its boundary condition type and get
        # the corresponding coordinates for entity in entities:
            # Adjust entity index for zero-based one
            entity_index = entity - 1  # COMSOL returns 1-based indices
            if entity_index < 0 or entity_index >= len(edges):
                continue  # Skip if the entity is out of bounds
            coords = edges[entity_index]  # Fetch edge coordinates
            if bc_type == 'InletBoundary':
                data['inlets'].append({'tag':tag, 'coordinates':coords})
            elif bc_type == 'OutletBoundary':
```

```
                data['outlets'].append({'tag':tag, 'coordinates':coords})
            elif bc_type == 'WallBC':
                data['walls'].append({'tag':tag, 'coordinates':coords})
    return data
```

The function scans through boundary features defined in the simulation physics (i.e. single phase flow). For each tagged feature, it checks the boundary condition type (e.g., inlet, outlet, wall) and retrieves the geometric entities (edges) associated with it. It then maps these entities to their corresponding coordinates using precomputed edge-vertex data. Depending on the boundary type, it stores the coordinates along with the tag in categorized lists (inlets, outlets, walls). The result is a structured dictionary containing all relevant boundary condition data, ready for downstream use in cellular automata or mesh processing.

The geometric boundary data was retrieved in the following form:

```
Inlets: 1 item
        Tag: inl1, Coordinates: ((0.0, 0.0), (0.0, 0.41))
Outlets: 1 item
        Tag: out1, Coordinates: ((2.2, 0.41), (2.2, 0.0))
Walls: 6 items
        Tag: wallbc1, Coordinates: ((2.2, 0.0), (0.0, 0.0))
        Tag: wallbc1, Coordinates: ((0.0, 0.41), (2.2, 0.41))
        ...
```

As a result of research and experiments for integrating cellular automata with CAE systems, we have successfully fetched not only raw mesh data but also boundary condition information, including the locations and identifiers of inlets and outlets, from COMSOL models. This milestone enables accurate mapping of boundary-driven dynamics into Cellular Automata representations, which is crucial for modeling fluid behavior and domain transitions.

## 4. Results

**4.1. Experimental setup.** To evaluate the effectiveness of the proposed methodology for integrating CA with CAE models, a series of experiments were conducted using COMSOL Multiphysics. The selected models focused on fluid dynamics, particularly laminar flow around a cylindrical obstacle—a scenario where CA-based methods can enhance the simulation of localized effects.

The proof-of-concept software was tested against these models to assess its ability to extract and translate simulation data, especially boundary-related information, for use in CA systems.

The following criteria were used for evaluation:

- **Integration efficiency**: Time taken to parse MPH model files and generate CATLIB-compatible data.

- **Boundary fidelity**: Accuracy in identifying and categorizing simulation boundaries such as inlets, outlets, and walls.
- **Representation accuracy**: Degree to which geometric and physical data aligned between COMSOL and the CA model.
- **Solver comparison**: Behavioral similarity of results between COMSOL's PDE-based solver and CATLIB's CA-based solver.

**4.2. Findings:**

Extraction performance:

- The converter successfully parsed MPH files and extracted geometry, mesh data, and physical properties.
- Critically, the software was able to extract boundary condition entities, including precise coordinates and tags for inlets, outlets, and walls, using the implemented boundary feature parsing routine.
- The boundary extraction method demonstrated robustness across different test cases, ensuring that all relevant flow interfaces were correctly mapped to the CA domain.

Converter speed and efficiency:

- End-to-end conversion (including mesh and boundary extraction) completed in under 5 seconds per model on average.
- Minimal preprocessing was required once the MPH file was loaded, suggesting good scalability for larger model sets.

Boundary representation:

- All major boundary types — InletBoundary, OutletBoundary, and WallBC — were successfully extracted and labeled.
- Coordinate mapping was consistent with COMSOL's edge definitions, and the software handled index alignment issues (e.g., 1-based vs. 0-based indexing) effectively.
- Extracted boundaries were visualized independently to validate correctness, showing high overlap with the original model representations.

## Conclusion

This research successfully demonstrated a novel approach for bridging the gap between traditional Computer-Aided Engineering systems based on differential equations and Cellular Automata models, which use rule-based, localized computation. By developing a converter capable of extracting and transforming data from COMSOL Multiphysics to CATLIB, we tackled several critical challenges in achieving meaningful interoperability between fundamentally different simulation paradigms.

**Key contributions:**

- **Interoperability framework:** We established a robust and extensible methodology to translate CAE model data—including geometry, mesh, and physics—into a form suitable for cellular automata simulations.
- **Boundary condition extraction:** A major technical milestone was the successful extraction of inlet, outlet, and wall boundary condition data from COMSOL MPH files. This resolves a key limitation in past efforts and allows CA models to accurately reflect physical constraints at simulation interfaces [4, 6, 7].
- **Tooling and integration:** Using the COMSOL Java API and Python MPh library, we built a flexible toolchain that automates the extraction of simulation metadata and geometric detail for CA integration.
- **Accessibility:** The converter lowers the technical barriers for researchers and engineers by streamlining the process of using CA models as a complementary or alternative simulation approach.

**Practical implications.** The developed tool supports:

- Direct comparison between PDE-based solvers and CA models on shared geometries and boundary conditions.
- Integration of CA techniques into existing engineering workflows, enabling exploration of localized, nonlinear, or emergent phenomena that are difficult to capture with conventional solvers.
- A standardized path for extending CA modeling to domains traditionally dominated by finite element or finite volume methods [3, 5, 8].

**Limitations and future work.** While the current implementation provides a solid foundation, several areas remain open for future research:

- Expanding support for additional CAE platforms and export formats.
- Improving the precision of data transformation, especially for more complex 3D and multiphysics models.
- Developing higher-level tools and interfaces to streamline user interaction and enable model customization within the CA framework.

**Broader impact.** This work contributes to the evolution of simulation technology by offering a practical, flexible way to integrate discrete CA models into the established CAE ecosystem. It emphasizes the value of cross-paradigm thinking in engineering and physics modeling and opens up new possibilities for simulating systems that exhibit complex, localized, or non-continuous behavior.

By enabling more comprehensive, modular, and accessible simulation tools, this research lays important groundwork for future innovation across domains ranging from fluid dynamics and materials science to biological systems and distributed computing.

## References

[1] Bandman O. Implementation of large-scale cellular automata models on multi-core computers and clusters // Int. Conf. High Performance Computing and Simulation (HPCS), Helsinki, Finland. — 2013. — P. 304–310. DOI:10.1109/HPCSim.2013.6641431.

[2] Achasova S., Bandman O., Markova V., Piskunov S. Parallel Substitution Algorithm. Theory and Application / World Scientific Publ. — 1994. DOI:10.1142/2369.

[3] Medvedev Yu.G. Lattice gas cellular automata for a flow simulation and their parallel implementation // Parallel Programming: Practical Aspects, Models and Current Limitations. Series: Mathematics Research Developments / Ed.: Tarkov M.S. Hauppauge. — New York: Nova Science Publishers, Inc., 2014. — P. 143–158.

[4] Kireev S., Trubitsyna Yu. Software implementation of asynchronous and synchronous cellular automata with maximum domino tiles coverage // Bull. Novosibirsk Comp. Center. Ser. Computer Science. — Novosibirsk, 2022. — Iss. 46. — P. 13–25.

[5] Von Neumann J. General and Logical Theory of Automata. — New York: Wiley, 1951; J. Von Neumann. Collected works // Hixon Symposium. — Vol. V. — 1948. — P. 288–328.

[6] Belyavtsev B.V. Development of an add-on to the FreeCAD system for interfacing with the library of cellular automata topologies // Information Technology. Scientific Engineering: Proc. 62nd International Scientific Student Conference. April, 17–23, 2024 / Novosibirsk State University. — Novosibirsk: IPC NSU, 2024. — P. 151 (In Russian).

[7] Burnyshev E.K. Software system for cellular automata modeling of gas flows // Bull. Novosibirsk Comp. Center. Ser. Computer Science. — Novosibirsk, 2022. — Iss. 48.

[8] Pogudin Y., Bandman O. Simulating cellular computations with ALT: A tutorial // LNCS. — Springer, 1997. — Vol. 1277. — P. 424–435. DOI:10.1007/3-540-63371-5_52.