INTERNATIONAL SYSTEM DYNAMICS CONFERENCE
ISDC2025
Incorporating AI into Dynamic Modeling

Boston MA, USA

Integrating Artificial Intelligence Techniques into System Dynamics: Opportunities and Challenges on the Path Forward

Authors: Zhenghua Yang[1]*, David Bruce Matchar[2], Enzo Bivona[1]

Abstract

The integration of artificial intelligence (AI) techniques into system dynamics (SD) may show promising potential for enhancing the dynamic modelling process, structure and behaviour analysis, and decision-making capabilities in complex systems. This article explores the advancements and future directions in combining AI with SD modelling. Especially, we highlight specific applications of AI techniques that can be employed at different stages of the SD modelling process, including augmented SD with computer vision, dynamic model building in natural language, and AI-generated learning environment design. It is expected that AI tools like ChatGPT can play a key role of copilot to help develop model structures and understand dynamic behaviours. Furthermore, we discuss the key benefits, challenges, and opportunities that may arise from AI-enriched modelling and simulation, emphasizing the need for interdisciplinary collaboration and bold attempts in the SD society.

Key words: Artificial Intelligence, System Dynamics, Modelling and Simulation

1. Introduction

During the 1950s, artificial intelligence (AI) and system dynamics (SD) were found almost in the meantime. It's considered by many that the field of AI research was born at a workshop hosted by John McCarthy and Marvin Minsky at Dartmouth College in 1956 (Anyoha, 2017). By the same year, Professor Jay Forrester joined the MIT Sloan School of Management and began to apply engineering insights to management and business issues, which led to the creation of SD later (Lane, 2007). There is no doubt that both disciplines have made tremendous progress and contributions to social and human development over the last sixty years. Specifically, the launch

[1] University of Palermo, Italy. *Corresponding author. Email: zhenghua.yang@unipa.it.
[2] Duke-NUS Medical School, Singapore.

of ChatGPT (Open AI, 2022) has successfully attracted the wide public attention and tiggered much concern on human safety in the short future (Makridakis, 2017; Marr, 2018). However, it has been widely acknowledged that AI techniques can greatly boost the productivity of human beings, according to suggestive evidence (Brynjolfsson et al., 2023; Noy & Zhang, 2023) and to our own user experience. As a general-purpose technology, integrating AI techniques into other disciplines is expected to stimulate their development into a new era and bring many possibilities in the foreseeable future.

With regard to SD, there had already been some methodological advances and successful applications of machine learning algorithms and techniques in the domain focusing on automated parameter estimation and calibration, decision support and policy optimization, and loop dominance analysis (An et al., 2015; Schoenberg et al., 2020; Yücel & Barlas, 2011), prior to the announcement of ChatGPT. Not long after this, there has been a growing trend of integrating SD modelling workflow and advance AI techniques. For instance, a few recent studies show great promise in combining AI language models with qualitative SD research like constructing causal loop diagrams (CLD) from textual data (Hosseinichimeh et al., 2024; Veldhuis et al., 2024) or replicating interview data analysis with ChatGPT (Jalali & Akhavan, 2024). In addition, Rahmandad et al. (2025) underscored the potential of deep learning technique of Amortized Bayesian Inference for scalable likelihood-free parameter estimation in quantitative SD modelling work, and Hu (2025) talked about how to embed and simulate SD models directly in ChatGPT conversational AI.

Looking ahead, the need for significant changes in the field of SD in order to realize its full promise had been well recognized by the founder Prof. Forrester (2007). Hence, incorporating some state-of-art AI techniques into SD is expected to revolutionize the traditional dynamic modelling process, structure and behaviour analysis, group model building, and so on (Sterman, 2000, ch. 22.2). For this reason, this article is intended to present some subjective but visionary perspectives or ideas on how to advance SD with the emergence of AI by giving several possible application cases. The primary purpose of this research is to explore the frontier and to share useful insights with the wide SD modellers to work together and prepare for the challenges and opportunities lying on the path forward (Sterman, 2018).

2. The dissemination of ChatGPT

ChatGPT is an AI chatbot built upon GPT-3.5 and GPT-4, which was developed by OpenAI and launched on November 30, 2022 (Wikipedia, 2023b). In essence, it is driven by large language models (LLMs) based on a wide variety of training data. As an advance natural language processing (NLP) tool, ChatGPT can be used to compose written content, to solve math problems, to write programming codes, and much more new possibilities and functions are still under development. As human beings, we can always get instant answers, find creative inspiration, and learn something new from ChatGPT (OpenAI, 2023), which has made it extraordinarily popular among students, scholars, reporters, programmers, and others. According to the user data[3] (Cuevas, 2023), it is reported that ChatGPT reached 1 million users in just 5 days from launch. By contrast, it took approximately 75 days and 5 months for Instagram (Meta Platforms, Inc.) and Spotify (Spotify AB) respectively to achieve the same number of active users in the past. Furthermore, we build a simple SD model for the innovation diffusion process by using the classic and generic "Bass Diffusion" model structure (Sterman, 2000, Ch. 9), shown in Fig. 1 (a). In general, the adoption rate (AR) is an aggregation of the "advertising" effect and "word of mouth" effect. At the very beginning, the positive feedback loop (R) dominates the innovation diffusion process and reinforces the "word of mouth" effect as the number of adopters (A) increases. Therefore, more and more potential users start to adopt these new digital platforms and an exponential growth trend is generated. Fig. 1 (b) compares the development paths to 1 million users for each platform and illustrates how fast the adoption of ChatGPT is. However, as the number of potential adopters (P) declines and market saturates, the negative feedback loops (B1 & B2) will gradually dominate the system and the AR will decrease in the end. Although the SD model in Fig. 1 is overly compact with a number of simplifying assumptions, it helps aid understanding of the dynamic innovation adoption process and clearly indicates the potential "limits to growth" (Meadows, 2008, ch. 2) in the future by making the negative feedback loops (B1 & B2) explicit in the model.

[3] The data are very limited because there are no detailed daily user data available for reference and calibration.

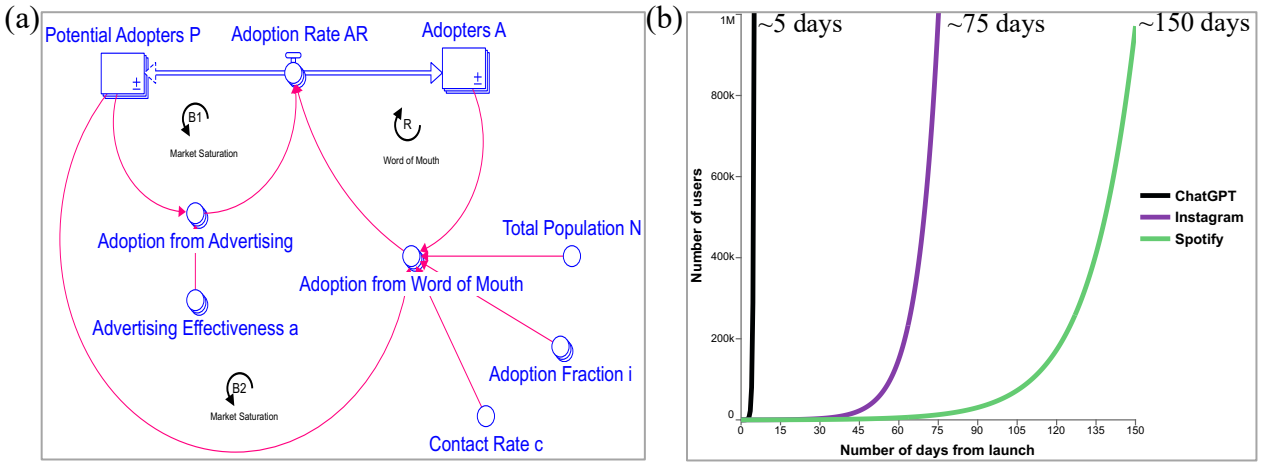

Fig. 1. Model structure and dynamic behavior for the 3 digital platforms' adoption, see APPENDIX A for detailed model documentation. In (a), the SD model structure is the same for modelling the diffusion of each platform, but the calibrated parameter (i.e., Adoption Fraction i) values are different by using the array function. In (b), the simulation runs from 0 to 150 days, but the behaviors are omitted for brevity when the number of users reach 1 million.

3. The potential of AI to improve SD modelling process

Instead of focusing on the adoption dynamics of ChatGPT, it is of great concern and interest to SD modellers or practitioners that how we can take advantage of sophisticated AI techniques to aid and improve our modelling practice. More than 2 decades ago, the well-known SD expert with great foresight - Professor John Sterman (2000, ch. 22.2) talked about automated help, like dimension consistency test, robustness test, and others by harnessing AI tools. According to his vision, "*it should soon be possible for simulation software to serve as an automated model-building tutor and guide*" (Sterman, 2000, pp. 897-898). Given the great potential of AI techniques, the integration of AI into SD is expected to offer a whole new way to build and understand SD models in the future. And the long-sought holy grail of automated dynamic model building will eventually happen. To reach this goal, there are some promising directions, as depicted in Fig. 2, where AI techniques are supposed to be applicable from a broad range of domains, including but not limited to augmented SD models with computer vision, to dynamic model building in natural language, and to AI-generated management flight simulator.

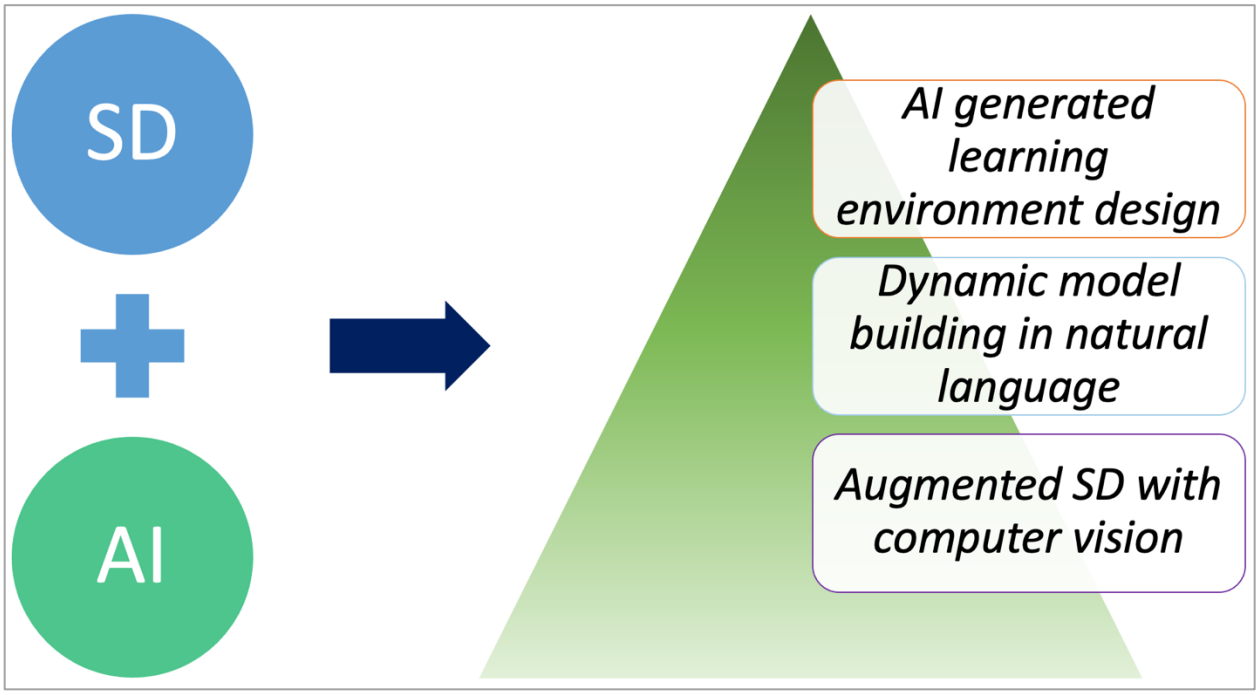

Fig. 2. Schematic diagram of possible AI-driven application cases in the field of SD.

- ***Augmented SD with computer vision***

The transformation of static stock and flow diagrams (SFD) into fully interactive, embedded SD simulation models represents a significant advancement in modelling practice. By leveraging NLP capabilities, such as those offered by ChatGPT, alongside computer vision and standardized file formats like XMILE (xmile-v1.0, 2015), modellers can now automate the extraction of structural information from hand-drawn or digitally produced SFDs and rapidly generate executable simulation models.

The first step in this integrated workflow involves the use of ChatGPT to analyze images of SFDs. Through computer vision techniques, potentially augmented by deep learning-based object detection methods, the static diagram is processed to identify the fundamental components of an SD model. ChatGPT is then queried to classify these visual elements into stocks, flows, auxiliary variables, and connectors. This initial extraction phase provides a structured representation of the diagram in JSON format, which captures not only the variable names but also the inherent causal links between them. Once the model structure is identified, the next step is to refine the model by requesting ChatGPT to suggest appropriate equations, units, and parameters for each variable.

The subsequent phase involves the conversion of the structured JSON output into a formal XMILE file. XMILE is an XML-based standard for representing SD models, which is widely adopted for its interoperability between different SD modelling tools like Vensim (ventana systems inc), Stella (isee systems inc, 2023), and Anylogic (The AnyLogic Company). By building an API interface between ChatGPT and the SD modelling software, the JSON data can be programmatically transformed into the XMILE format. This process involves mapping the JSON fields such as variable names, equations, inflows, outflows, and connector relationships into the corresponding XMILE tags. For instance, stock variables in the JSON are written as `<stock>` elements with nested `<eqn>`, `<inflow>`, and `<outflow>` tags, and flows and auxiliaries are mapped in a similar way as required by the XMILE specification. **APPENDIX B** presents the sample Python code to create interactive and embedded SD simulations directly from static SFDs by using the computer vision of ChatGPT.

This integrated approach exemplifies the concept of "augmented SD", where advanced AI tools not only facilitate model creation but also serve as a bridge between static diagrammatic representations and interactive simulations (Gunturu et al., 2024). By automating the tedious and error-prone process of manual model transcription, SD modellers and practitioners can focus on refining model logic, validating assumptions, and exploring complex system behaviours more efficiently. Furthermore, the seamless embedding of these models into SD simulation software via a standardized API can accelerate the iteration cycle and fosters greater reproducibility in modelling studies. In summary, the use of ChatGPT in conjunction with computer vision techniques to extract and annotate static SFDs and the subsequent translation of these annotations into standard XMILE files, provides a powerful framework for the creation of interactive and embedded SD simulations. This augmented modelling approach not only enhances productivity but also opens new avenues for research and application in the field of SD.

- ***Dynamic model building in natural language***

With the help of AI language models like ChatGPT, it has been demonstrated that users can explore SD models through natural language interactions (Hu, 2025). To be more exact, by translating SD models with PySD (Martin-Martinez et al., 2022), ChatGPT is capable of running simulations based on user-provided parameters and offering explanations for cause and effect relationships, and providing detailed analyses and visualization of the simulation results. However, there is a key

question related to the application case suggested by Hu (2025) – where does the fundamental SD model come from? It seems that the previous study solely deals with using ChatGPT to aid our understanding of extant SD modelling work without toughing upon the origin of SD models, namely the fundamental structure development.

As a causal modelling methodology, the model structure is essential for SD simulation work because structure determines behavior (Kampmann & Oliva, 2009; Meadows, 2008, ch. 2; Schoenenberger et al., 2021; Sterman, 2000). Choosing the appropriate and reasonable SD model structure is generally difficult and iterative, which mostly depends on the modeller's understanding of the research context at hand and their experience in SD modelling. Over the last few decades, SD modellers have learned to build dynamic models by placing stocks and flows one by one on a blank page, which is time-consuming and limited to our mental model database. Although the popular SD simulation software Stella (isee systems inc, 2023) provides a new function called "Assembly" at the moment, it still has some limitations. One of the challenges is "availability". As human beings are bounded by rationality and memory, the ideal model structures are not always available when we need. For example, the modeller has to be very familiar with all the stored assemblies first, then they can select one or more assemblies and add to the model structure. Without good understanding of all the stored model structures in the assembly, the modellers can easily get lost and do not know how to select and what to select at all.

On the contrary, LLMs like ChatGPT have been trained with massive data, so they can swiftly connect users' prompts with the most relevant information stored in their data center. Therefore, by leveraging the extensive knowledge base of LLMs, natural language-based model development could be a game changer, and will permit autonomous SD model structure suggestion, replacing the traditional modelling method. For example, instead of building a population model by placing all the stocks, flows, auxiliary variables, and connectors between them on the canvas of SD modelling software, we can simply ask ChatGPT to propose the possible model structures to us. If we want to incorporate the "aging chain" (Sterman, 2000, ch. 12.1) into the population model, we can tell ChatGPT to refine the suggested model structure with further hints or prompts. The interactions between users and AI chatbots ought to be iterative in order to achieve the desired model structure. In general, building SD models is an art depending on the modellers' understanding of the research problem of interest, and there is no absolutely standard or correct

answer to the underlying model structure indeed. Once users are satisfied with the fundamental model structure, ChatGPT can additionally suggest equations, units, and parameters at their request. By then, the natural language-based model development process is over. Next, users may ask ChatGPT to export the suggested model as the XMILE format file and simulate it on the traditional SD modelling software, or they can directly ask ChatGPT to perform the simulation and provide in-depth analysis of simulation results.

- ***AI-generated learning environment design***

Another possible integration of AI technique into SD is model-based learning environment design, also known as "management flight simulators (MFS)" or simply referring to simulation games. As emphasized by Box (1976), "*All models are wrong, but some are useful.*" Furthermore, it has been acknowledged that the usefulness of a model (i.e., any model, not limited to SD models) depends on not only the quality and credibility of the model per se, but the effectiveness of results and insights communication with the widest possible audience such as customers, decision-makers, or policy-makers. On this ground, the model-based interface can be used to increase the ease of communication and test the final users' own assumptions by providing an interactive approach to the decision-making process (Tsaples & Tarnanidis, 2020). Sterman (2014a, 2014b) introduced a wide range of interactive web-based SD simulations for strategy and sustainability. More recently, the popularity of C-ROADS (https://www.climateinteractive.org/c-roads/) and En-ROADS (https://www.climateinteractive.org/en-roads/) among the scholars, government, business units, NGOs, and so on has demonstrated the great value of interactive simulators in real world (Creutzig & Kapmeier, 2020; Kapmeier et al., 2021; Sterman et al., 2013).

Given the importance of model-based learning environment in practice, modellers should not only focus on building "good" models, but also attach importance to model-based interface design for facilitating learning. However, it usually takes a lot of time and efforts to design an interactive simulator from a blank page. Although Stella Architect from version 3 (isee systems inc, 2023) and Forio (Forio Corporation) provide several pre-built interface templates that users can select and then customize, these platforms have not been smart and intelligent enough to design the interface automatically based on the underlying model structure and research context. At this stage, the "Design" function provided by the SD simulation software - Stella is quite similar to the traditional PowerPoint (Microsoft Corporation, 2023) templates, but our goal or expectation is to

use next-generation AI techniques including ChatGPT to learn the fundamental model structure, to tell a good story about the fundamental SD model, and then create user-friendly simulators. For example, Microsoft has recently introduced "Microsoft 365 Copilot" to all the Office suites (Spataro, 2023). As is emphasized, "*Copilot in PowerPoint helps you create beautiful presentations with a simple prompt, adding relevant content from a document you made last week or last year*" (Spataro, 2023). Rather than prepare presentation slides page by page, "Microsoft 365 Copilot" can now automate the whole presentation slides preparation given some written texts and a new wave of productivity growth has been unlocked by AI. For this reason, we strongly believe that the SD model-based learning environment design can be or should be automated in the short future, and we need a revolutionary way to work.

How could the working path be? Firstly, the modeller can give some introductory information about the research background, research question, and research purpose. Based on relevant inputs, the simulation software should be able to automatically develop aesthetic pages for presentation with necessary texts, shapes, graphs, or even online pictures organized in an organic way. With respect to model structure explanation, the software shall be able to identify the key stocks and flows, and feedback loops in the fundamental model based on the calculated loop dominance information (Schoenberg et al., 2020; Schoenberg et al., 2023) and expand the model structure page by page for an excellent story-telling experience. Afterwards, the software might guide users to the control panel and let them play with controllable variables in the model and identify high-leverage policy for improvement. In the end, the simulation software is supposed to learn from the users' decision-making process and share some useful insights or suggestions with target audience.

4. Discussion

At the end of last century, Richardson (1996) called for technical support for understanding the connections between model structure and dynamic behaviour. Given the fast development in computer science over recent years, the powerful AI techniques are changing the way we work, the way we learn, the way we drive, you name it. Numerous workflows can be automated with the application of advanced AI techniques, including the development and analysis of simulation models (Widman & Loparo, 1990; Zeigler et al., 2009). Hence, a combination of AI and SD seems to hold promise in this context. In the future, it is believed that artificial general intelligence (AGI)

can accomplish any intellectual tasks that we human beings can perform or achieve even better performance (Wikipedia, 2023a). Until then, there are going to be plenty of both opportunities and challenges that the SD society has to take on.

- ***Interactions between AI and SD***

The interplay between AI and SD is creating novel opportunities to model complex systems and simultaneously advance AI research and development. On one hand, AI technologies are being integrated into SD modelling methods to automate and enhance processes such as model construction, calibration, and parameter estimation, as has been documented at the beginning. On the other hand, systems thinking and dynamic modelling offer a powerful framework to improve AI research by identifying latent unintended negative consequences resulting from the development and application of AI techniques. This perspective helps researchers and policy-makers understand and optimize the adaptability, stability, and safety of AI algorithms, leading to the design of more robust, transparent, and explainable systems (Martin et al., 2020; Moosavihaghighi, 2024; Nabavi & Browne, 2023; Prabhakaran & Martin, 2020). Together, these dual approaches promise to create a synergistic environment where AI not only accelerates SD modelling but also benefits from the holistic insights provided by SD.

- ***Aid understanding of complex model behaviours***

Considering the extraordinary NLP capability of ChatGPT, the modeller can easily communicate with the chatbot and hence ask it to help explain the complex behavior in terms of the feedback structure of the underlying SD model (Mitchell, 2023). From this perspective, AI-powered LLMs like ChatGPT play an important role of "individualized" tutor, who may provide coherent responses to any queries with regard to the interrelationship between model structure and dynamic behaviour. It is expected that there would be iterative rounds of prompting and discussion between ChatGPT and final users. We believe the interaction between people and ChatGPT in natural language can significantly ease the understanding of complex systems for individuals and provide valuable policy insights for decision-makers, as echoed by Mitchell (2023).

- ***Pay attention to possible limitations***

So far we have mainly discussed the exciting and promising applications of AI techniques to the automated SD model building and interface design, but the potential limitations of some advanced AI tools such as ChatGPT have not been investigated at all. One of the biggest challenges faced

by AI is the ability to "reason" and "think" (Boden, 2018; Ray, 2023). As the anonymous Twitter account ARC Tracker previously noted, "*AI isn't yet capable of reasoning, so anything it writes is merely a summary of information rather than novel critique or insight.*" (Wilcox, 2023). Before the introduction of OpenAI o1 model[4], the LLMs do not really understand the underlying cause and effect because they simply using sophisticated statistical methods to put words together in a reasonable sequence based on the massive training documents (Mitchell, 2023). However, several chatbots nowadays can perform complex reasoning by producing a long internal chain of thought trained with reinforcement learning before it answers. To a large degree, AI technology is accelerating unpredictably all over the world, and some limitations may automatically disappear with more synthetic data and better algorithm training in the short future. When applying AI techniques to SD modelling, we do need to be cautious because AI is not perfect currently. As emphasized on the website of OpenAI, "*ChatGPT can make mistakes. Check important info.*"

- ***Worry the diminishing modelling ability***

Like the ever-lasting concerns and debates about the impact of AI writing tools (Iskender, 2023; Johinke et al., 2023; Marzuki et al., 2023), there may be growing concern over the apparently significant impact of the powerful generative AI technologies on the development of human modelling skills. Although this article has by far expressed an optimistic attitude towards the integration of AI technologies into SD, some people may worry about the diminishing modelling ability of individuals over time. It is well known that practice makes perfect. Hence, modellers may get fewer opportunities to practice their SD modelling skills if they become overly dependent on automated modelling functions. In my opinion, such kind of worries are largely unnecessary. The overarching goal of modelling is very often to solve a problem. If the solution is found no matter by human workers or AI robots, then the goal is met. According to the latest experimental evidence on the productivity effects of generative AI, Noy and Zhang (2023) found that ChatGPT substantially raised productivity and reduced the inequality between workers. Therefore, it is better to treat AI-powered chatbots as a modelling assistant who allows the modeller to gain inspirations or brainstorm ideas and to spend more time learning the suggested SD model structure in this context. More importantly, there is no call for completely depriving the right or option of dynamic modelling in a traditional way (i.e., building models completely manually) in this article. By

[4] For more information of OpenAI o1: https://openai.com/index/learning-to-reason-with-llms/

contrast, it highlights the interaction between modellers and machine by taking advantage of advance AI technologies to build better SD models and gain better insights into the complex system under investigation. In essence, we hope that the SD society can enjoy the benefits from the integration of AI into SD methodology.

- ***Challenges on the path forward***

The mission of fully automated SD model development may look gloomy or impossible to some people including SD experts because of the tremendous workload and crazy ideas. It is true if we have to develop all the necessary techniques from scratch. Fortunately, this is not the case as lots of advance AI techniques and algorithms are open source and readily available. Therefore, software developers can easily deploy and integrate some AI tools like ChatGPT into their own SD simulation software products through API or other connecting ways. It will harness the breakthrough and advance in other scientific fields like computer science and help reduce the complexity with respect to automated SD model development to a manageable level. To achieve the target, the need for interdisciplinary collaborations between AI researchers (external knowledge) and SD modellers and practitioners (internal insights) is urgent. Most importantly, "*Scholars and practitioners in the field need to recognize the importance of these efforts and support them actively.*" (Richardson, 1996). More concerted efforts and great passion from the SD society are in need of to continuously push the boundary of SD methodology and believe the magic will become true someday.

5. Conclusion

This article provides a vision for future research and applications of AI-enriched SD modelling and simulation, underlining the transformative potential of AI-driven tools in achieving automated model building and interface design in natural language. By harnessing the potential of AI in SD modelling, AI tools like ChatGPT can play an important role of copilot to improve the dynamic modelling process and to enhance the understanding of complex systems. By that time, we may say that a truly high-level intelligence era within the SD domain has finally come. There is no doubt that the whole society has to undertake massive work and face critical challenges so as to meet the target. However, if autonomous driving is soon possible in the real world, why can't it be automated modelling? To be bold or not to be, that is a question.

6. References:

An, W., Anderson Jr., E. G., Barlas, Y., Chalise, N., Eberlein, R., Ghoddusi, H., Grassmann, W., Hovmand, P. S., Jalali, M. S., Joglekar, N., Keith, D., Liu, J., Moxnes, E., Rahmandad, H., Oliva, R., Osgood, N. D., Spiteri, R., Sterman, J., Struben, J.,…Yücel, G. (2015). *Analytical Methods for Dynamic Modelers*. The MIT Press. https://doi.org/10.7551/mitpress/9927.001.0001

Anyoha, R. (2017, August 28). The History of Artificial Intelligence. https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/

Bjørnelv, G. M. W., Halsteinli, V., Kulseng, B. E., Sonntag, D., & Ødegaard, R. A. (2020). Modeling Obesity in Norway (The MOON Study): A Decision-Analytic Approach—Prevalence, Costs, and Years of Life Lost. *Medical Decision Making*, *41*(1), 21-36. https://doi.org/10.1177/0272989x20971589

Boden, M. A. (2018). *Artificial Intelligence: A Very Short Introduction*. Oxford University Press. https://books.google.co.jp/books?id=ITTsvQEACAAJ

Box, G. E. P. (1976). Science and Statistics. *Journal of the American Statistical Association*, *71*(356). https://doi.org/10.1080/01621459.1976.10480949

Brynjolfsson, E., Li, D., & Raymond, L. (2023). Generative AI at Work [Working Paper]. *NBER*. https://doi.org/10.3386/w31161

Creutzig, F., & Kapmeier, F. (2020). Engage, don't preach: Active learning triggers climate action. *Energy Research & Social Science*, *70*. https://doi.org/10.1016/j.erss.2020.101779

Cuevas, Ó. (2023). *ChatGPT: What it is, what it's used for, and how to use it.* https://www.sacyr.com/en/-/chatgpt-que-es-para-que-sirve-y-como-usarlo

Forrester, J. W. (2007). System dynamics—the next fifty years. *System Dynamics Review*, *23*(2-3), 359-370. https://doi.org/10.1002/sdr.381

Gunturu, A., Wen, Y., Zhang, N., Thundathil, J., Kazi, R. H., & Suzuki, R. (2024). *Augmented Physics: Creating Interactive and Embedded Physics Simulations from Static Textbook Diagrams* Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology,

Hosseinichimeh, N., Majumdar, A., Williams, R., & Ghaffarzadegan, N. (2024). From text to map: a system dynamics bot for constructing causal loop diagrams. *System Dynamics Review*, *40*(3). https://doi.org/10.1002/sdr.1782

Hu, B. (2025). ChatPySD: Embedding and Simulating System Dynamics Models in ChatGPT-4. *System Dynamics Review*, *41*(1). https://doi.org/10.1002/sdr.1797

isee systems inc. (2023). *FEATURE UPDATES*. https://www.iseesystems.com/store/products/feature-updates.aspx

Iskender, A. (2023). Holy or Unholy? Interview with Open AI's ChatGPT. *European Journal of Tourism Research*, *34*. https://doi.org/10.54055/ejtr.v34i.3169

Jalali, M. S., & Akhavan, A. (2024). Integrating AI language models in qualitative research: Replicating interview data analysis with ChatGPT. *System Dynamics Review*, *40*(3). https://doi.org/10.1002/sdr.1772

Johinke, R., Cummings, R., & Di Laurao, F. (2023). Reclaiming the technology of higher education for teaching digital writing in a post—pandemic world. *Journal of University Teaching and Learning Practice*, *20*(2). https://doi.org/10.53761/1.20.02.01

Kampmann, C. E., & Oliva, R. (2009). System Dynamics, Analytical Methods for Structural Dominance Analysis in. In *Encyclopedia of Complexity and Systems Science* (pp. 8948-8967). https://doi.org/10.1007/978-0-387-30440-3_535

Kapmeier, F., Greenspan, A. S., Jones, A. P., & Sterman, J. D. (2021). Science-based analysis for climate action: how HSBC Bank uses the En-ROADS climate policy simulation. *System Dynamics Review*, *37*(4), 333-352. https://doi.org/10.1002/sdr.1697

Lane, D. C. (2007). The power of the bond between cause and effect: Jay Wright Forrester and the field of system dynamics. *System Dynamics Review*, *23*(2-3), 95-118. https://doi.org/10.1002/sdr.370

Makridakis, S. (2017). The forthcoming Artificial Intelligence (AI) revolution: Its impact on society and firms. *Futures*, *90*, 46-60.

Marr, B. (2018). Is Artificial Intelligence dangerous? 6 AI risks everyone should know about. *Forbes*.

Martin, Jr., Prabhakaran, V., Kuhlberg, J., Smart, A., & Isaac, W. (2020). *Extending the Machine Learning Abstraction Boundary: A Complex Systems Approach to Incorporate Societal Context*. https://doi.org/10.48550/arXiv.2006.09663

Martin-Martinez, E., Samsó, R., Houghton, J., & Solé, J. (2022). PySD: System Dynamics Modeling in Python. *Journal of Open Source Software*, *7*(78). https://doi.org/10.21105/joss.04329

Marzuki, Widiati, U., Rusdin, D., Darwin, & Indrawati, I. (2023). The impact of AI writing tools on the content and organization of students' writing: EFL teachers' perspective. *Cogent Education*, *10*(2). https://doi.org/10.1080/2331186x.2023.2236469

Meadows, D. H. (2008). *Thinking in systems: a primer*. Chelsea Green Pub.

Mitchell, F. H. (2023, July). *CHATGPT MEETS SYSTEM DYNAMICS* International System Dynamcis Conference, Chicago.

Moosavihaghighi, M. (2024). Analyzing the Impacts of AI Development and Adaptation on Human Life Using a System Dynamics Model. International System Dynamics Conference, Bergen, Norway.

Nabavi, E., & Browne, C. (2023). Leverage zones in Responsible AI: towards a systems thinking conceptualization. *Humanities and Social Sciences Communications*, *10*(1). https://doi.org/10.1057/s41599-023-01579-0

Noy, S., & Zhang, W. (2023). Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, *381*(6654), 187-192. https://doi.org/10.1126/science.adh2586

OpenAI. (2023). *ChatGPT*. https://openai.com/chatgpt

Prabhakaran, V., & Martin, D. (2020). Participatory Machine Learning Using Community-Based System Dynamics. *Health and human rights*, *22*, 71-74. https://pmc.ncbi.nlm.nih.gov/articles/PMC7762892/pdf/hhr-22-02-071.pdf

Rahmandad, H., Akhavan, A., & Jalali, M. S. (2025). Incorporating Deep Learning Into System Dynamics: Amortized Bayesian Inference for Scalable Likelihood-Free Parameter Estimation. *System Dynamics Review*, *41*(1). https://doi.org/10.1002/sdr.1798

Ray, P. P. (2023). ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*, *3*, 121-154. https://doi.org/10.1016/j.iotcps.2023.04.003

Richardson, G. P. (1996). Problems for the future of system dynamics. *System Dynamics Review*, *12*(2), 141-157. https://doi.org/https://doi.org/10.1002/(SICI)1099-1727(199622)12:2<141::AID-SDR101>3.0.CO;2-O

Schoenberg, W., Davidsen, P., & Eberlein, R. (2020). Understanding model behavior using the Loops that Matter method. *System Dynamics Review*, *36*(2), 158-190. https://doi.org/10.1002/sdr.1658

Schoenberg, W., Hayward, J., & Eberlein, R. (2023). Improving Loops that Matter. *System Dynamics Review*, *39*(2), 140-151. https://doi.org/10.1002/sdr.1728

Schoenenberger, L., Schmid, A., Tanase, R., Beck, M., & Schwaninger, M. (2021). Structural Analysis of System Dynamics Models. *Simulation Modelling Practice and Theory*, *110*. https://doi.org/10.1016/j.simpat.2021.102333

Spataro, J. (2023). Introducing Microsoft 365 Copilot – your copilot for work. https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/

Sterman, J. (2000). *Business dynamics: systems thinking and modeling for a complex world*. Irwin/McGraw-Hill.

Sterman, J. (2014a). Interactive web-based simulations for strategy and sustainability: The MIT Sloan LearningEdge

management flight simulators, Part I. *System Dynamics Review*, *30*(1-2), 89-121. https://doi.org/10.1002/sdr.1513

Sterman, J. (2014b). Interactive web-based simulations for strategy and sustainability: The MIT SloanLearningEdgemanagement flight simulators, Part II. *System Dynamics Review*, *30*(3), 206-231. https://doi.org/10.1002/sdr.1519

Sterman, J. (2018). System dynamics at sixty: the path forward. *System Dynamics Review*, *34*(1-2), 5-47. https://doi.org/10.1002/sdr.1601

Sterman, J. D., Fiddaman, T., Franck, T., Jones, A., McCauley, S., Rice, P., Sawin, E., & Siegel, L. (2013). Management flight simulators to support climate negotiations. *Environmental Modelling & Software*, *44*, 122-135. https://doi.org/10.1016/j.envsoft.2012.06.004

Tsaples, G., & Tarnanidis, T. (2020). A System Dynamics Model and Interface for the Simulation and Analysis of Milk Supply Chains. In *Supply Chain and Logistics Management* (pp. 108-135). https://doi.org/10.4018/978-1-7998-0945-6.ch006

Veldhuis, G. A., Blok, D., de Boer, M. H. T., Kalkman, G. J., Bakker, R. M., & van Waas, R. P. M. (2024). From text to model: Leveraging natural language processing for system dynamics model development. *System Dynamics Review*, *40*(3). https://doi.org/10.1002/sdr.1780

Widman, L. E., & Loparo, K. A. (1990). Artificial Intelligence, Simulation, and Modeling. *Interfaces*, *20*(2), 48-66. http://www.jstor.org/stable/25061332

Wikipedia. (2023a). *Artificial general intelligence*. https://en.wikipedia.org/wiki/Artificial_general_intelligence

Wikipedia. (2023b). *ChatGPT*. https://en.wikipedia.org/wiki/ChatGPT

Wilcox, C. (2023, 17 JUL 2023). ScienceAdviser: AI can't help you write reviews, funders say. *ScienceAdviser*. https://www.science.org/content/article/scienceadviser-ai-can-t-help-you-write-reviews-funders-say

xmile-v1.0. (2015). *XML Interchange Language for System Dynamics (XMILE) Version 1.0.* In (Version xmile-v1.0) OASIS Standard. http://docs.oasis-open.org/xmile/xmile/v1.0/os/xmile-v1.0-os.html

Yücel, G., & Barlas, Y. (2011). Automated parameter specification in dynamic feedback models based on behavior pattern features. *System Dynamics Review*, *27*(2), 195-215. https://doi.org/10.1002/sdr.457

Zeigler, B., Muzy, A., & Yilmaz, L. (2009). Artificial Intelligence in Modeling and Simulation. In *Encyclopedia of Complexity and Systems Science* (pp. 344-368). https://doi.org/10.1007/978-0-387-30440-3_24

APPENDIX A Model Documentation

Variable	*Equation*	*Properties*	*Units*	*Documentation*
Adopters_A[App](t)	Adopters_A[App](t - dt) + (Adoption_Rate_AR[App]) * dt	INIT Adopters_A[App] = 0	People	The number of active adopters in the system.
Potential_Adopters_P[App](t)	Potential_Adopters_P[App](t - dt) + (- Adoption_Rate_AR[App]) * dt	INIT Potential_Adopters_P[App] = Total_Population_N - Adopters_A	People	The number of potential adopters is determined by the total population size and the current number of active adopters.
Adoption_Rate_AR[App]	Adoption_from_Advertising +Adoption_from_Word_of_Mouth		People /Day	The rate at which a potential adopter becomes an active adopter. This is driven by advertising efforts and the word of mouth effect.
Adoption_Fraction_i[ChatGPT]	.036		Dimensionless	The fraction of times a contact between an active adopter and a potential adopter results in adoption.
Adoption_Fraction_i[Instagram]	.00128		Dimensionless	The fraction of times a contact between an active adopter and a potential adopter results in adoption.
Adoption_Fraction_i[Spotify]	.00058		Dimensionless	The fraction of times a contact between an active adopter and a

				potential adopter results in adoption.
Adoption_from_Advertising[App]	Advertising_Effectiveness_a*Potential_Adopters_P		People/Day	Adoption can result from advertising according to the effectiveness of the advertising effort with the pool of potential adopters.
Adoption_from_Word_of_Mouth[App]	Contact_Rate_c*Adoption_Fraction_i*Potential_Adopters_P*Adopters_A/Total_Population_N		People/Day	Adoption by word of mouth is driven by the contact rate between potential adopters and active adopters and the fraction of times these interactions will result in adoption. The word of mouth effect is small if the number of active adopters relative to the total population size is small.
Advertising_Effectiveness_a[App]	.00000001		1/Day	Advertising results in adoption according the effectiveness of the advertising.
Contact_Rate_c	100		1/Day	The rate at which active adopters come into contact with potential adopters.

Total_Population_N	1e+009		People	The size of the total population.

Run Specs	
Start Time	0
Stop Time	150
DT	1/4
Time Units	Day
Integration Method	Euler

Array Dimension	Indexed by	Elements
App	Label (3)	ChatGPT, Instagram, Spotify

APPENDIX B Static SD Model Identification in OpenAI

```
from openai import OpenAI
import base64
import json
import re
import xml.etree.ElementTree as ET

# Set your OpenAI API key (ensure this is kept secure)
openai_api_key = "YOUR_API_KEY"
client = OpenAI(
    api_key = openai_api_key,
)

def extract_components_from_image(image_path):
    """
    Extracts the components of a stock and flow diagram using the OpenAI API.
    The function reads an image file, encodes it in base64, and sends it to
the API
    with a prompt to extract stocks, flows, auxiliaries, and connectors.
    Each variable may now also include location keys "x" and "y".
    Expected output is a JSON object with keys: 'stocks', 'flows',
'auxiliaries', 'connectors'.
    """
    # Read and encode the image file
    with open(image_path, "rb") as image_file:
        image_data = image_file.read()
    encoded_image = base64.b64encode(image_data).decode("utf-8")

    # Construct the prompt
    prompt = (
            "Analyze the following image (provided as a base64 encoded
string) of a stock and flow diagram. "
            "Extract the following information in JSON format: stocks, flows,
auxiliaries, and connectors. "
            "Return a JSON object with keys: 'stocks', 'flows',
'auxiliaries', 'connectors'. "
            "Each of 'stocks', 'flows', and 'auxiliaries' should be an array
of objects with a 'name' property, and an 'eqn' property for reasonable
equations that you may suggest. Note: only constant values for 'stocks'."
            "As well as their relative location information in the given
image: 'x' and 'y' coordinates (in pixels). "
            "For stocks, also extract 'inflows' and 'outflows' as lists of
flow names."
            "Keep all the names of stocks, flows, and auxiliaries the same as
they were shown in the stock and flow diagram."
            "Each connector should be an object with properties 'src' and
'tgt' and ONLY take the arrow links between model variables into account."
            "There is no need of accounting for the causal links from 'flows'
to 'stocks' in the 'connectors'"
            "To avoid any omissions, make sure and check every object with
properties 'src' and 'tgt' in 'connectors' has been classified as either
'stocks', or 'flows', or 'auxiliaries'"
            "Add their causal relationships in 'connectors' part if any
variable is used as part of the equation of another variable"
            "The base64 image is: " + encoded_image
```

```
    )

    response = client.chat.completions.create(
        model="gpt-4o",
        messages=[
            {
                "role": "user",
                "content": [
                    {
                        "type": "text",
                        "text": prompt,
                    },
                    {
                        "type": "image_url",
                        "image_url": {"url":
f"{encoded_image}"},
                    },
                ],
            }
        ],
        response_format={"type": "json_object"},
        temperature=0.5,
        top_p=0.1
    )

    # Debug: Print the entire API response to see its structure
    print("Raw API response:")
    print(response)

    # Extract and check the response content
    structured_data = response.choices[0].message.content
    if not structured_data.strip():
        raise ValueError("API returned an empty response. Check your model
access, prompt, and image input.")

    # Debug: Print the extracted content before parsing
    # print("Extracted content:")
    # print(structured_data)

    # Parse the JSON from the API response
    try:
        model_data = json.loads(structured_data)
        with open('output.json', 'w') as outfile:
            json.dump(model_data, outfile, indent=2)
    except json.JSONDecodeError as e:
        print("Failed to parse JSON. The extracted content is:")
        print(structured_data)
        raise e

    # Extract and parse the JSON from the API response
    # structured_data = response.choices[0].message.content
    # model_data = json.loads(structured_data)

    return model_data
```

```
# NS = "http://docs.oasis-open.org/xmile/ns/XMILE/v1.0"
#
# def ns_tag(tag):
#     return f"{{{NS}}}(Bjørnelv et al.)"

def clean_eqn(eqn):
    """
    Replace spaces within multi-word variable names in an equation string
    with underscores, while preserving spaces between tokens.
    This regex finds sequences of at least two words (letters with spaces)
    and joins them with underscores.
    """
    pattern = r'\b([A-Za-z]+(?:\s+[A-Za-z]+)+)\b'
    def repl(match):
        # Join the matched group (a multi-word variable name) with
underscores.
        return '_'.join(match.group(1).split())
    return re.sub(pattern, repl, eqn)

def generate_xmile(model_data, filename):
    """
    Converts the structured model data into an XMILE file.
    """
    # Create the root XMILE element
    xmile = ET.Element("xmile", {"version": "1.0"})

    # Add header information
    header = ET.SubElement(xmile, "header")
    model_name = ET.SubElement(header, "name")
    model_name.text = "Converted System Dynamics Model"

    # Simulation specifications
    sim_specs = ET.SubElement(xmile, "sim_specs")
    start = ET.SubElement(sim_specs, "start")
    start.text = "0"
    stop = ET.SubElement(sim_specs, "stop")
    stop.text = "100"
    dt = ET.SubElement(sim_specs, "dt")
    dt.text = "1/4"

    # Model section
    model = ET.SubElement(xmile, "model")
    variables = ET.SubElement(model, "variables")

    # Add stocks
    for stock in model_data.get("stocks", []):
        stock_el = ET.SubElement(variables, "stock", {"name": stock["name"]})
        if "eqn" in stock and stock["eqn"]:
            eqn_el = ET.SubElement(stock_el, "eqn")
            eqn_el.text = stock["eqn"]
        # Add inflow tags if available
        for inflow in stock.get("inflows", []):
            ET.SubElement(stock_el, "inflow").text = inflow.replace(" ", "_")
        for outflow in stock.get("outflows", []):
            ET.SubElement(stock_el, "outflow").text = outflow.replace(" ",
```

```
"_")

    for flow in model_data.get("flows", []):
        flow_el = ET.SubElement(variables, "flow", {"name": flow["name"]})
        if "eqn" in flow and flow["eqn"]:
            eqn_el = ET.SubElement(flow_el, "eqn")
            eqn_el.text = clean_eqn(flow["eqn"])

    for aux in model_data.get("auxiliaries", []):
        aux_el = ET.SubElement(variables, "aux", {"name": aux["name"]})
        if "eqn" in aux and aux["eqn"]:
            eqn_el = ET.SubElement(aux_el, "eqn")
            eqn_el.text = clean_eqn(aux["eqn"])

    # # Structure section for connectors (causal relationships) - how to
represent the connectors in XMILE format?!
    # structure = ET.SubElement(model, "structure")
    # for conn in model_data.get("connectors", []):
    #     ET.SubElement(structure, "connector", {"from": conn["src"], "to":
conn["tgt"]})

    # Create views element and a single view element
    views = ET.SubElement(model, "views")
    view = ET.SubElement(views, "view")

    # Add connector elements in the view with sequential uid starting from 1
    connectors = model_data.get("connectors", [])
    for i, conn in enumerate(connectors, start=1):
        connector_attribs = {
            "uid": str(i),  # Sequential uid starting from 1
            "angle": str(conn.get("angle", "0"))
        }
        connector_el = ET.SubElement(view, "connector",
attrib=connector_attribs)
        from_el = ET.SubElement(connector_el, "from")
        from_el.text = conn["src"]
        to_el = ET.SubElement(connector_el, "to")
        to_el.text = conn["tgt"]

    # Output display objects for variables (if needed, we re-output them
here)
    for stock in model_data.get("stocks", []):
        attribs = {
            "x": str(stock.get("x", "0")),
            "y": str(stock.get("y", "0")),
            "name": stock["name"]
        }
        ET.SubElement(view, "stock", attrib=attribs)

    for flow in model_data.get("flows", []):
        attribs = {
            "x": str(flow.get("x", "0")),
            "y": str(flow.get("y", "0")),
            "name": flow["name"]
        }
```

```
        # ET.SubElement(view, "flow", attrib=attribs)
        flow_view = ET.SubElement(view, "flow", attrib=attribs)
        # Generate <pts> in view as well using the same rule:
        try:
            flow_x = float(flow.get("x", 0))
        except ValueError:
            flow_x = 0.0
        flow_y = flow.get("y", "0")
        pts_el = ET.SubElement(flow_view, "pts")
        ET.SubElement(pts_el, "pt", attrib={"x": str(flow_x - 60), "y":
str(flow_y)})
        ET.SubElement(pts_el, "pt", attrib={"x": str(flow_x + 60), "y":
str(flow_y)})

    for aux in model_data.get("auxiliaries", []):
        attribs = {
            "x": str(aux.get("x", "0")),
            "y": str(aux.get("y", "0")),
            "name": aux["name"]
        }
        ET.SubElement(view, "aux", attrib=attribs)

    # Write the XMILE file
    tree = ET.ElementTree(xmile)
    tree.write(filename, encoding="utf-8", xml_declaration=True)

def print_summary(model_data):
    """
    Prints a summary of the model data, including the number of stocks,
flows,
    and auxiliary variables, as well as their names.
    """
    stocks = model_data.get("stocks", [])
    flows = model_data.get("flows", [])
    auxiliaries = model_data.get("auxiliaries", [])
    connectors = model_data.get("connectors", [])

    print("\nModel Summary:")
    print(f"Number of stocks: {len(stocks)}")
    print("Stocks:", ", ".join([stock["name"] for stock in stocks]))

    print(f"\nNumber of flows: {len(flows)}")
    print("Flows:", ", ".join([flow["name"] for flow in flows]))

    print(f"\nNumber of auxiliary variables: {len(auxiliaries)}")
    print("Auxiliaries:", ", ".join([aux["name"] for aux in auxiliaries]))

    print(f"\nNumber of connectors: {len(connectors)}")

def main():
    # Path to your stock and flow diagram image
    image_path = "image.png"

    # Step 1: Extract the components using the OpenAI API
    model_data = extract_components_from_image(image_path)
```

```
    print("Extracted model data:")
    print(json.dumps(model_data, indent=2))

    # Print summary of the extracted model data
    print_summary(model_data)

    # Step 2: Generate the XMILE file from the extracted model data
    xmile_filename = "SD model.xmile"
    generate_xmile(model_data, xmile_filename)
    print(f"XMILE file '{xmile_filename}' generated successfully!")


if __name__ == "__main__":
    main()
```

How It Works:

1. **Image Extraction:**

- The function *extract_components_from_image* reads your image file, encodes it as a base64 string, and sends it along with a prompt to the OpenAI API.
- The prompt instructs the model to extract stocks, flows, auxiliaries, and connectors from the diagram and suggest equations in a specified JSON format.
- The response is parsed into a Python dictionary (*model_data*).

2. **XMILE Generation:**

- The function *generate_xmile* takes the *model_data* dictionary and creates an XMILE format structure.
- It creates XML elements for stocks, flows, auxiliary variables, and connectors.
- Finally, it writes the XMILE file (*model.xmile*).

3. **Main Function:**

- The *main* function ties the extraction and XMILE generation together.
- It first extracts the model data from the image, prints it for verification, then generates and saves the XMILE file.

This provides a complete end-to-end solution for converting an image of a stock and flow diagram into a structured simulation model in XMILE format. The code can be run in Python environment, and users or developers need to apply for their own API key for use from the OpenAI website[5].

[5] OpenAI API website: https://platform.openai.com/api-keys

For the use of other chatbots like Geminin, Claude, Grok, and so on, it is recommended to adjust the client and message parts according to the development requirements of corresponding platforms.