December 2024

PSSimPy: A Design Science Approach to Constructing and Implementing a Large-Value Payment System Simulator

Kenneth See
National University of Singapore, see.k@u.nus.edu

Hanzholah Shobri
Bank Indonesia, hanzholahs@gmail.com

Nicholas Garcia
National University of Singapore, 1nicholasgarcia@gmail.com

PSSimPy: A Design Science Approach to Constructing and Implementing a Large-Value Payment System Simulator

Completed Research Paper

Kenneth See
Asian Institute of Digital Finance
NUS FinTech Lab
National University of Singapore
see.k@u.nus.edu

Hanzholah Shobri[1]
Bank Indonesia
hshobri@outlook.com

Nicholas MacGregor Garcia
Department of Information Systems and Analytics
NUS FinTech Lab
National University of Singapore
ngarcia@nus.edu.sg

Abstract

This paper introduces PSSimPy, a Python-based simulator for Large-Value Payment Systems developed through a Design Science approach, addressing the challenges posed by blockchain and distributed ledger technologies. PSSimPy serves as a sophisticated tool for analyzing and modeling the dynamic behaviors within financial systems. Our study establishes a comprehensive set of design principles, with a particular emphasis on modularity, accessibility, and data privacy. These principles, not extensively covered in existing literature, enhance the simulator's adaptability and reliability in rapidly changing financial environments. By incorporating these alongside other critical principles, our work provides a robust framework for advancing the design of simulation tools and contributes to the complex adaptive systems literature, offering innovative solutions for managing the intricacies of modern financial systems.

Keywords: CBDC, Simulation, Payment Systems, Design Science Research

Introduction

As the financial sector increasingly adopts blockchain technology, firms must consider the potential risks of such decentralized systems. These risks may include negative externalities within the network, misaligned goals, and competitive pressures among counterparties (Seebacher & Schüritz, 2019). To address these risks, firms have historically used complex behavioral modeling to simulate the interactions between agents and their motivations in real-world settings (e.g., Bech & Garratt, 2003).

Within IS, literature on complex adaptive systems (CAS) offers guidelines for simulation models that capture dynamic interactions between agents and their emergent outcomes (Nan, 2011). Simulations allow researchers to conduct what-if analyses, model agent behaviors, and explore potential scenarios in a controlled environment (Haki et al., 2020). However, current simulation tools may not fully capture the

[1] The views stated herein are those of the author and do not necessarily reflect the views of Bank Indonesia.

complexities or needs of real-world financial settings, limiting their effectiveness in studying and analyzing these systems.

To effectively model and simulate the complexities of novel IS systems, simulation tools must adhere to certain principles. Research has shown that effective simulation models for CAS must incorporate heterogeneous agents, an environment, and a mechanism for agents to interact with each other and the environment (Raczynski, 2020). They should also support a sufficient runtime length and repetitions to achieve stable and accurate outputs in the face of uncertainty and stochastic factors (Long & Li, 2014). Additionally, simulation technology that allows for counterfactual analysis can provide valuable insights for managerial decisions (Eabrasu, 2021). For simulation models to be effective, their outputs must also be validated through the recreation of real-world results or by demonstrating key characteristics (known as "stylized facts") of the system being studied (Rand & Rust, 2011). This ensures that the outputs can be trusted and used for decision-making purposes.

While the current literature provides a foundation in how to model complex behavior, less attention is given to usability considerations for end users, thus constraining their use in real-world decision-making. Other assumptions, such as the operational environment remaining static, may rarely hold true in man-made environments where change is constant and inevitable.

In this paper, we contribute to the literature on CAS simulations for blockchain-based fintech systems by providing a set of design principles and features for simulation system design, with a focus on addressing the identified gaps. Adopting a Design Science approach (Hevner et al., 2004), we use the simulation of Large-Value Payment Systems (LVPSs) as a case study to demonstrate the inadequacy of existing simulation software in addressing contemporary challenges.

LVPSs are specialized financial systems designed to handle high-value and time-critical transactions between financial institutions, corporations, or government entities. These systems are crucial for the stability and efficiency of the financial markets, particularly in managing the transfer of large sums of money, usually in the millions or billions of dollars, between banks and other financial institutions. In recent years, central banks have been increasingly considering the adoption of blockchain technology within LVPS, with some of the leading central banks in the late stages of the pilot phase. Such non-trivial technological change demands advanced simulation tools capable of assessing the implications of such significant system modifications. Among the critical areas of study is the management of liquidity risk within the system (Angelini et al., 1996). Current simulation tools, however, fall short of fully capturing the complexities introduced by blockchain features. These features include the capability to remove information asymmetry through consensus mechanisms (Cong & He, 2019), dynamic transaction and lending costs via protocols (Auer et al., 2023), and more efficient oversight of Delivery versus Payment transactions (Chiu & Koeppl, 2019).

We use a Design Science approach to constructing an artifact directed toward addressing this gap, aiming to provide a comprehensive understanding of how an LVPS simulator can be designed and implemented. Beyond extending our understanding of CAS models in the blockchain domain, this artifact is intended to support both current and future research on LVPSs, thereby enhancing scholars and industry's ability to explore and innovate with novel technologies. To address these challenges, we pose the **research question**: How can an LVPS simulator be designed to effectively support the study of liquidity-related risks in system design?

Guided by Goal-Oriented Requirements Engineering (GORE) (van Lamsweerde, 2001) as the kernel theory and a comprehensive review of academic and central bank literature, we identify the design requirements, design principles, and design features to construct an LVPS simulator artifact. We then implement our artifact, *PSSimPy*, as a Python library that is open to the public and fully addresses the identified requirements of a comprehensive LVPS simulator. Our research process will seek to reveal key necessary features that are not previously highlighted in the existing IS literature and important considerations when implementing simulator artifacts for blockchain-based fintech systems.

Background and Motivation

Large-Value Payment Systems

LVPSs are systems designated to execute the transfer of funds for large-value and high-priority payments. All LVPSs share a common purpose: to facilitate the settlement of urgent transactions with finality (CPMI, 2005). Estimates place LVPS transaction volumes in excess of US$1,000 trillion annually (Bech et al., 2008), underpinning vital economic activities such as liquidity management and interbank transactions, which are essential for the stability of financial markets (Corbae & Gofman, 2020).

Historically, large-value transactions were settled using deferred net settlement systems, which aggregate and offset payments at a specified future time, thereby reducing liquidity demands. Nonetheless, the failure of one participant to meet payment obligations could trigger a cascade of defaults among others. To address these risks, the 1990s saw the adoption of real-time gross settlement (RTGS) systems in many countries (CPMI, 1997), which guarantee immediate settlement of payments individually. This transition to RTGS introduced concerns, and consequently a wave of seminar papers in the field, about liquidity risks due to the unpredictability of payment flows. Central banks, as the operator or overseer of LVPS, are constantly improving LVPS design based on the guidance of ongoing research.

Role of Simulation

The utilization of simulation tools in the study of LVPSs plays a key role in understanding payment system design and dynamics. In a seminar paper by Bech & Garratt (2003), participants within LVPS are identified as strategic agents, where both endogenous and exogenous factors determine the strategy adopted by a participant, and thus affect the overall dynamics of the system. Subsequent work has considered heterogeneity among participants (Martin & McAndrews, 2008) and exogenous shocks (Nellen, 2019) as closer depictions of actual LVPSs. This revelation shows that LVPSs are CAS, where simulations could offer insights into their operation and resilience. The practical application of simulations in this domain is well-documented, with Hong Kong's and Finland's use of simulations prior to their transitions from net to gross settlements serving as prime examples (HKMA, 1997; Koponen & Soramäki, 1998). Simulations have also been used in academic research to identify potential fragility in LVPSs, depending on the characteristics of the participants (Galbiati & Soramäki, 2011).

BoF-PSS (Leinonen & Soramäki, 2003), developed by the Bank of Finland, stands out as a prominent tool in this field for its comprehensiveness, accommodating a wide array of simulation setups. The application has been utilized by several central banks, including those of Canada (Arjani, 2006), Lithuania (Bakšys & Sakalauskas, 2006), and Norway (Enge & Øverli, 2006), to analyze and refine their payment systems. Despite its advantages, BoF-PSS has several notable shortcomings. Its closed-source nature and reliance on Java limit its accessibility, especially for non-technical users who may find customization challenging. Furthermore, the simulator's limited integration capabilities with other technologies, reliance on static input data (e.g., transactions), and the potential difficulties in reproducing and sharing results due to its graphical user interface nature, pose significant constraints. These limitations highlight the need and present an opportunity for more open, flexible, and user-friendly simulation tools in the field of LVPS research. Furthermore, BoF-PSS seems to be unable to accommodate potential blockchain scenarios due to its rigidity. This is evidenced by the lack of research that has used BoF-PSS in the rapidly growing field of blockchain applications to LVPSs.

Blockchain in LVPS

In recent years, central banks have shown increased interest in wholesale central bank digital currencies (wCBDCs), involving the integration of blockchain and distributed ledger technology (DLT) into LVPSs. Over 80% of surveyed central banks are actively investigating wCBDCs (Di Lorio et al., 2024).

The motivations for adopting blockchain and DLT include enhanced security, efficiency, scalability, resiliency, and transparency (Opare & Kim, 2020). Traditionally, LVPSs have settled only the cash leg of transactions, with underlying transactions conducted on separate systems. Blockchain allows for the unification of these settlement layers, enabling both cash delivery and underlying transactions to be processed within a single system via smart contracts (Bech et al., 2020). Depending on the implementation, a wCBDC system could alter the information available to participants and system operators, potentially altering interactions among participants as well as between participants and their operational environment.

Research Methodology

Hevner et al. (2004) state that artifacts should solve an unaddressed and important problem, be rigorously evaluated, and effectively communicated to the intended users. We use the process model proposed by Peffers et al. (2007) as the framework for conducting this research.

Our research entry point is problem identification and motivation. We identified that there is a critical need for an accessible and comprehensive LVPS simulator, especially with the upcoming significant changes that wCBDCs would bring, and there is no existing solution that adequately addresses this issue. Based on the established problem and overall research question, we identified three research objectives:

1. Identify essential requirements for an LVPS to facilitate effective simulations.
2. Specify the architecture needed for a robust LVPS simulator capable of accurately replicating both current and future technologies employed in payment systems.
3. Implement an LVPS simulator based on the key functionalities and architecture identified.

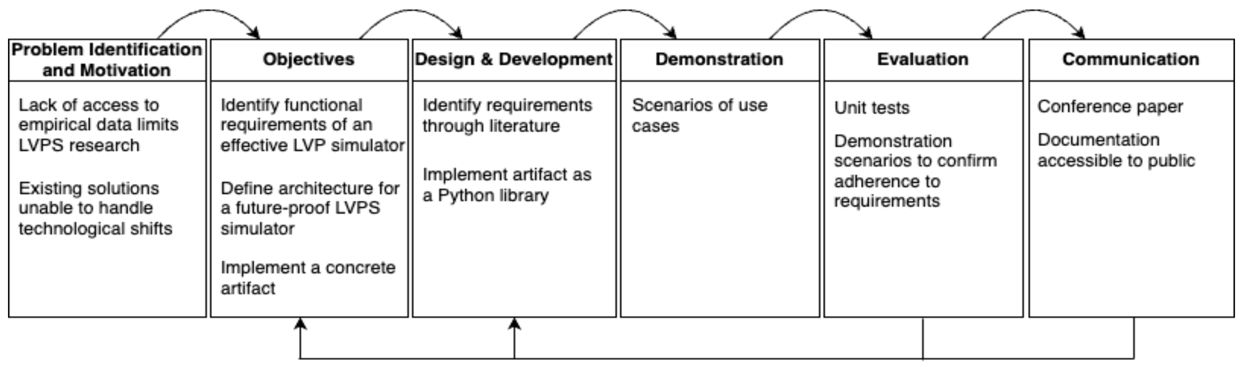

Problem Identification and Motivation
Lack of access to empirical data limits LVPS research
Existing solutions unable to handle technological shifts
Objectives
Identify functional requirements of an effective LVP simulator
Define architecture for a future-proof LVPS simulator
Implement a concrete artifact
Design & Development
Identify requirements through literature
Implement artifact as a Python library
Demonstration
Scenarios of use cases
Evaluation
Unit tests
Demonstration scenarios to confirm adherence to requirements
Communication
Conference paper
Documentation accessible to public

Figure 1. Design Science Research Methodology Process Model Adapted from Peffers et al. (2007)

Our artifact's design closely follows the guidelines proposed by Gregor & Jones (2007), focusing on a methodical approach to creating information systems. To construct an artifact that addresses our established problem within LVPS simulation, we first gather the Design Requirements through a comprehensive review of relevant academic and central banking literature. We then synthesize the requirements into Design Principles that provide high-level guidance on how an artifact should be constructed to address the defined problem. Subsequently, these principles guided the derivation of specific Design Features during the Artifact Implementation phase. This phase involved detailed iterative prototyping, where each feature was developed, tested, and refined based on continuous feedback from initial testing rounds, ensuring alignment with the overarching Design Principles.

Evaluation and demonstration were systematically conducted through a blend of unit testing and scenario analysis. Unit testing allowed us to discover discrepancies between the anticipated and actual performance of our artifact, facilitating iterative refinement to align closely with the defined design specifications. The scenarios we constructed fulfilled a twofold objective: firstly, to illustrate practical applications of our artifact for prospective users, and secondly, to act as a comprehensive test bed for integration assessment.

We disseminate the outcomes of our research through several avenues. This paper thoroughly outlines our contributions, offering a valuable reference for future investigations in this domain. Additionally, we have made the documentation for our Python library publicly available in the README file. This documentation provides comprehensive guidance on utilizing the library for simulations, alongside detailed technical descriptions of essential classes and functions. The source code, unit test cases, and documentation can all be found in the *PSSimPy* GitHub repository[2].

Artifact Design

Purpose and Scope

Our design artifact aims to simplify the complex process of simulating payment and settlement systems for two main groups: academic researchers and central bankers. These individuals often share their insights through academic journal articles, conference proceedings, and central bank working papers and reports. To ensure our artifact adheres to their needs, we adopt a method known as GORE (van Lamsweerde, 2001) as the kernel theory, guided by the way simulations are typically conducted in Information Systems Research (Beese et al., 2019).

We conduct a detailed review of academic and central bank literature, searching keywords like “Interbank Settlement”, “Wholesale Payments”, “LVPS”, and “RTGS” within the Scopus database and working paper series published by central banks, to identify the specific needs users have from a simulation system. By applying GORE, we ensure that each feature of our artifact serves a distinct purpose, addressing the real-world challenges that researchers and central bankers encounter with LVPSs. This approach guarantees that our design is not only practical but directly relevant to the needs of our target users.

Design Requirements

In line with the framework of GORE, we derive our requirements from both the functional and non-functional goals of the user. Functional requirements refer to what the user can expect to do in the system while non-functional requirements refer to the more general attributes and behaviors of the system.

Functional Requirements

Understanding the liquidity needs and risk profiles of participants within an LVPS is essential, where the size of participants in the system could impact the system liquidity requirements (Koponen & Soramäki, 1998). Additionally, balance sheet characteristics may provide useful information for risk assessment (Anderson et al., 2019). These insights inform our first design requirement (**DR1**): to enable users to define a network of participants, each with customizable accounts and starting balances.

At its core, an LVPS facilitates payments between financial institutions (CPSS, 2005). Intuitively, this would mean that at a minimum, participants and transactions need to be able to be modeled. Growing interest in blockchain and DLT for LVPSs requires us to consider that transactions could contain different degrees of information depending on the system design and technology used (Cong & He, 2019). For example, using blockchain for Delivery versus Payment would entail that transactions could capture information verifiable by the system when an obligation arises from a securities settlement (Auer, 2022). This background leads to (**DR2**): allowing users to specify transactions between participants with configurable information content.

LVPSs can charge fees for the provision of settlement services, where the fees in many real-world LVPSs vary depending on the transaction size or time of settlement. For example, Singapore’s wholesale payment system, MEPS+, increases fees towards the end of the day to encourage earlier settlements (MAS, 2020). Holthausen & Rochet (2006) suggest that the way fees are priced can influence banks' transaction routing decisions, which could be an important subject of study when assessing the adoption of a new system. Thus, our third design requirement (**DR3**) is to enable users to define and adjust transaction costs within the simulation, reflecting fixed cost structures or variable cost structures based on time or predefined conditions.

The role of intraday credit in managing liquidity risk and the potential for moral hazard (Zhou, 2000) necessitates a nuanced approach to simulating credit mechanisms. Bech & Garratt (2003) illustrate how different credit regimes—priced versus collateral—can impact system dynamics. Accordingly, our fourth

[2] Accessible at: https://bit.ly/pssimpy

design requirement (**DR4**) is to allow users to simulate various intraday credit mechanisms, enabling an analysis of their effects on system risk and behavior.

The design of the settlement mechanism is a critical component within the broader architecture of an LVPS. This choice significantly influences the system's risk profile; opting for DNS systems subjects participants to the risk of contagious defaults (Angelini et al., 1996), whereas RTGS systems are associated with heightened liquidity risks (Nakajima, 2011). Other components of settlement mechanics are queueing (Guo et al., 2015), where payment requests to be settled could have varying priority (De Caux et al., 2016), and liquidity-saving mechanisms (Martin & McAndrews, 2008). We hence derive our fifth design requirement (**DR5**): enable users to specify the settlement mechanism that governs how transactions are settled within a simulation scenario.

Stress tests are used by regulators to evaluate the resilience of an LVPS. One such study is the assessment of LVPS liquidity following the September 11 attacks (McAndrews & Potter, 2002). Another source of systemic risk is settlement risk, which could cause bank contagion and consequently lead to systemic failure (Furfine, 2003). Such risk can be stochastic in nature, in the sense that it can impact different participants with varying probability (Mills & Nesmith, 2008). This background allows us to formulate our sixth design requirement (**DR6**), which is to allow the simulation of deterministic or stochastic exogenous shocks, facilitating an examination of LVPSs' robustness under adverse conditions.

In line with the operational norms and regulatory requirements outlined by the Committee on Payment and Settlement Systems (CPSS), our design requirements aim to accurately replicate the key aspects of LVPSs. Specifically, the CPSS (2005) report highlights the importance of enforcing minimum account balance requirements and establishing transaction size limits to ensure LVPS operations are liquid. As indicated in the name, LVPSs should also be focused on its primary objective of facilitating large-value transactions. Thus, there needs to be a minimum transaction size to control the volume of transactions being processed. Consequently, our seventh and eighth design requirements (**DR7** & **DR8**) enable users to define minimum account balances and specify minimum and maximum transaction amounts within the simulation.

Some LVPSs adopt throughput rules, which are requirements that stipulate that the system participants have to settle a certain percentage of their average daily transaction volume by some prescribed time in the day (CPSS, 2005). The way throughput rules are implemented within an LVPS could have potential effects on the amount of liquidity recycling in the system (Buckle & Campbell, 2003). We hence determine our ninth design requirement (**DR9**): allow users to determine the mechanisms of any throughput rules.

While we have allowed users to define several constraints in DR7, DR8, and DR9, there is bound to be a scenario where a transaction would violate a constraint. How a transaction would be handled upon a constraint violation would be up to the nature of the violation and the design of the system. For example, when a transaction exceeds the size limit constraint, some LVPSs split the transaction into multiple transactions while other systems may not let the transaction be processed (CPSS, 2005). Another example is when a payment request is placed by a bank with insufficient balance to have the transaction settled. One approach could be to have the transaction fail due to the minimum balance constraint, but if the system has a liquidity-saving mechanism, such a transaction could instead be first placed in such a mechanism to potentially net against incoming payments (Martin & McAndrews, 2008). We arrive at our tenth design requirement (**DR10**): allow users to determine the handling of a transaction when the transaction meets a constraint violation.

LVPS participants can be thought of as strategic agents, whose behaviors, such as the timing of their payments, significantly influence liquidity and risk within the system, as evidenced by empirical studies (McAndrews & Rajan, 2000; Perlin & Schanz, 2011). The strategic nature of participants have been recognized and incorporated into theoretical works (Bech & Garratt, 2003; Mills & Nesmith, 2008; Nellen, 2019). Recognizing these strategic behaviors, Agent-Based Modeling (ABM) has emerged as a popular method for studying LVPSs, offering insights into how participants interact with their environment and each other. A distinct advantage of using ABM is its ability to model heterogenous agents, adding complexity and depth to facilitate more rigorous analysis. Existing uses of ABM in the context of LVPS studies include crisis modeling (Arciero et al., 2008), payment system selection (Mayo et al., 2021), and liquidity management strategies (Galbiati & Soramäki, 2011). Therefore, our eleventh design requirement (**DR11**) is to allow users to model participants as homogenous or heterogenous agents, where their settlement behavior is influenced by the information available at a given moment within a simulation scenario.

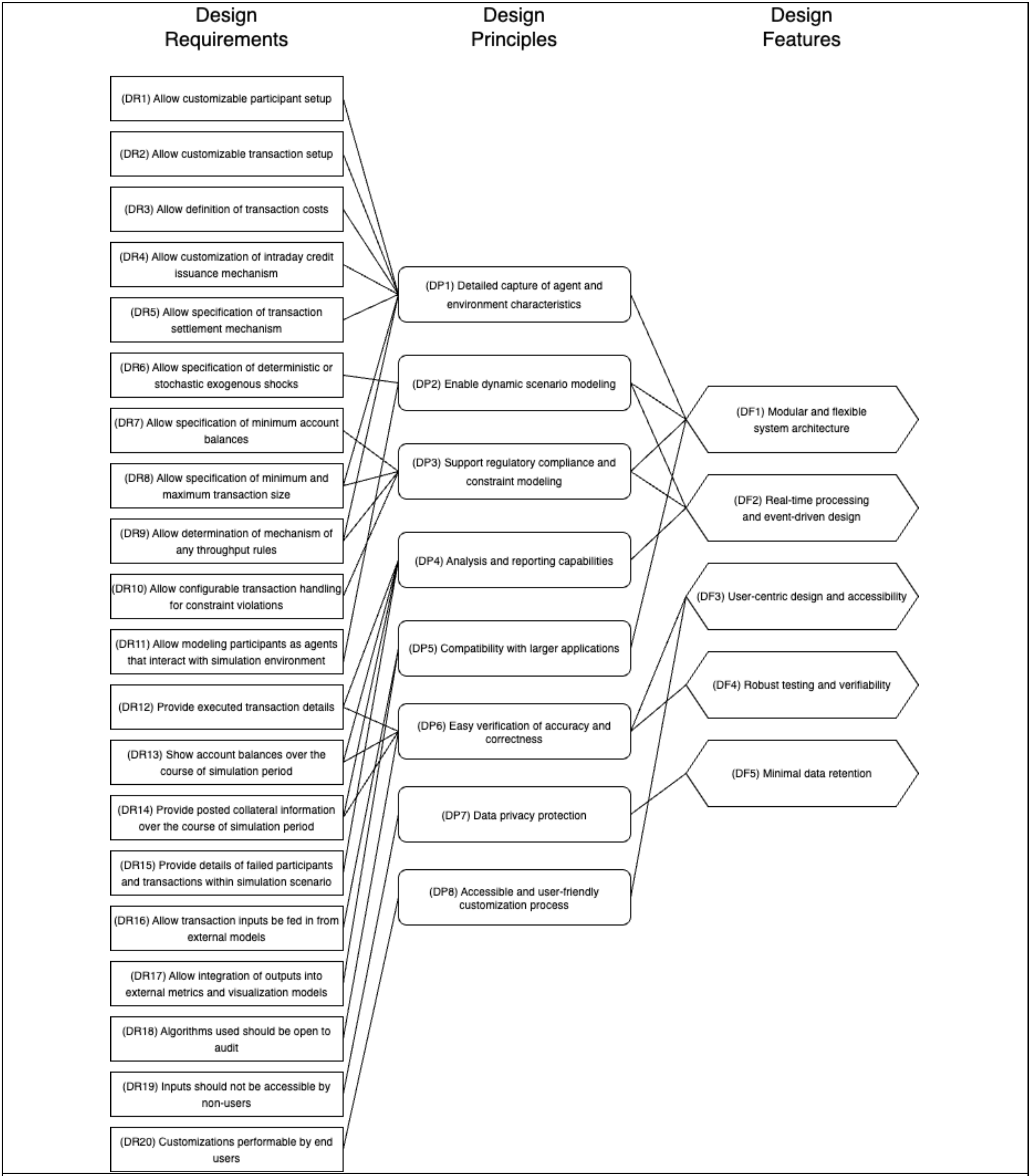

Design Requirements
Design Principles
Design Features
(DR1) Allow customizable participant setup
(DR2) Allow customizable transaction setup
(DR3) Allow definition of transaction costs
(DR4) Allow customization of intraday credit issuance mechanism
(DR5) Allow specification of transaction settlement mechanism
(DR6) Allow specification of deterministic or stochastic exogenous shocks
(DR7) Allow specification of minimum account balances
(DR8) Allow specification of minimum and maximum transaction size
(DR9) Allow determination of mechanism of any throughput rules
(DR10) Allow configurable transaction handling for constraint violations
(DR11) Allow modeling participants as agents that interact with simulation environment
(DR12) Provide executed transaction details
(DR13) Show account balances over the course of simulation period
(DR14) Provide posted collateral information over the course of simulation period
(DR15) Provide details of failed participants and transactions within simulation scenario
(DR16) Allow transaction inputs be fed in from external models
(DR17) Allow integration of outputs into external metrics and visualization models
(DR18) Algorithms used should be open to audit
(DR19) Inputs should not be accessible by non-users
(DR20) Customizations performable by end users
(DP1) Detailed capture of agent and environment characteristics
(DP2) Enable dynamic scenario modeling
(DP3) Support regulatory compliance and constraint modeling
(DP4) Analysis and reporting capabilities
(DP5) Compatibility with larger applications
(DP6) Easy verification of accuracy and correctness
(DP7) Data privacy protection
(DP8) Accessible and user-friendly customization process
(DF1) Modular and flexible system architecture
(DF2) Real-time processing and event-driven design
(DF3) User-centric design and accessibility
(DF4) Robust testing and verifiability
(DF5) Minimal data retention

Figure 2. Mapping of Design Requirements to Design Principles to Design Features

The ability to meticulously analyze and monitor risk over time when performing LVPS simulations is of large importance. This can be done through the use of indicators (Heijmans & Heuver, 2014). Bank Indonesia adopts metrics such as Turnover Ratio, Throughput Zone, and Interconnectedness for liquidity monitoring (Peranginangin, 2020). Based on the formulas provided in the paper, deriving these indicators would require knowing the transaction details and participant account balances. Additionally, monitoring collateral levels can also be important for assessing participants' resilience and identifying signs of potential failure (Heijmans & Heuver, 2014). The management of collateral involves applying haircuts to the securities used, which are adjusted to account for liquidity risk and to reflect uncertainties related to the security or broader macroeconomic conditions (Jasova et al., 2024). Given that intraday credit facilities may allow for a variety of securities to be pledged as collateral, having the capability to specify and display the types of securities used is essential for a comprehensive assessment of participants' resilience to financial shocks. Against this backdrop, we identify the next three design requirements in succession. The twelfth requirement (**DR12**): be able to provide users with the executed transaction details of a simulation scenario. The thirteenth requirement (**DR13**): be able to display account balances throughout a simulation period. The fourteenth requirement (**DR14**): be able to display collateral, along with the types of securities used as collateral, posted to the intraday credit facility.

Being able to build indicators concerning failing participants is integral to financial stability monitoring (Heijmans & Wendt, 2023). Net multilateral flow and net bilateral flow are identified as important indicators to measure for this purpose. To measure these indicators, failed participants must be able to be identified and the transactions canceled as a result of the failure must also be known to exclude them from the indicator value calculations. Hence, our fifteenth design requirement (**DR15**) is to be able to provide the details of failed participants and transactions within a simulation scenario.

Non-functional Requirements

There are certain situations where it would be insufficient to simply expect a static set of transactions as inputs to the LVPS simulator. As exogenous factors may be important in the study of liquidity risks in LVPSs (CPSS, 1997), it is of interest to researchers to explore the spillover effects of these exogenous factors on risk in LVPSs. Therefore, we define our sixteenth design requirement (**DR16**) as the simulator having to be able to have transaction inputs fed in from external models.

On a similar line of reasoning, the outputs of the LVPS simulator could be used in external models as well, such as correlating the value and volume of transactions a wholesale payment system processes with the country's economic growth (Rooj & Sengupta, 2020). Furthermore, depending on the use case and individual or organization needs, the outputs of an LVPS simulator need to be visualized in different ways. With these needs in consideration, we specify our seventeenth design requirement (**DR17**) as the simulator having to be able to have its output seamlessly integrated into external metrics and models.

When data and code are not shared, it could lead to substandard quality of shared resources (Colliard et al., 2023). The sharing of code allows for reproducibility and streamlining of research methodologies (Harvey, 2019). Additionally, in the finance industry, the adoption of new technologies, particularly in risk management, fundamentally requires that these technologies be explainable and transparent (Chen et al., 2019). Recognizing these challenges, we propose our eighteenth design requirement (**DR18**): algorithms used in the simulator should be open to audit.

Central banks use simulators for stress testing (MIPC, 2017) or empirical research (Triepels et al., 2018; Sabetti & Heijmans, 2021), which would involve feeding in real participant and transaction data as inputs. Such data is typically commercially sensitive and restricted to individuals with affiliations to the central bank (Docherty & Wang, 2010). With this privacy concern in mind, our nineteenth design requirement (**DR19**) is that inputs to the simulator should not be accessible to non-users.

Most customizations are done by end-users and thus customizable software should be designed to allow end-users to safely and effectively extend the software to meet their needs (Ko et al., 2011). Additionally, the ease of customization of artifacts leads to increased adoption (Venkatesh et al., 2003). This is especially true in the context of central banks which typically do not maintain large technological capabilities in-house (Fung & Halaburda, 2016). Hence, our final design requirement (**DR20**) is that permitted customization should be performable by the same users who use the simulator to conduct analyses.

Design Principles

Detailed capture of agent and environment characteristics (DP1). DR1 calls for a customizable participant setup to enable users to capture participant-level information that they may believe is relevant to their modeling efforts. DR2, DR3, and DR8 all contribute to how transactions should be modeled in the simulator. From DR4, DR5, and DR9, we acknowledge the many variations in possible intraday credit issuance and transaction settlement mechanisms. We thus deduce DP1 as: "*Provide the ability for users to* *capture the details of the simulation model's agent and environment characteristics, based on their intended use case.*" The principle emphasizes the importance of letting the user determine the granularity and complexity of the simulation model. This should be achieved with configurable entity representations within the simulation system.

Enable dynamic scenario modeling (DP2). From DR6 and DR11, we arrive at DP2: "*Let users model scenarios dynamically so that they can run experiments with varying conditions*." Both practitioners and academics benefit from using simulation systems for what-if analyses. Thus, it is important that a simulation system can support a wide range of hypothetical scenarios, which should be achieved by allowing users to easily toggle a model's inputs and parameters.

Support regulatory compliance and constraint modeling (DP3). DR7, DR8, DR9, and DR10 are requirements that address constraints imposed by system design or regulations. From these, we derive DP3: "*Ensure that constraints can be modeled in the system so that simulation models can accurately account for relevant regulation or external requirements*." This principle acknowledges that exogenous factors could impose constraints on the simulated environment. Simulation systems should hence account for this by incorporating mechanisms for users to define logic that can throttle certain actions from being taken.

Analysis and reporting capabilities (DP4). From DR12, DR13, DR14, and DR15 we specify DP4: "*Facilitate robust analysis and reporting by ensuring data from simulation scenarios is easily accessible and usable for users*." The utility of a simulation system relies on the accessibility and usability of its data outputs, which enable users to conduct thorough analyses and extract actionable insights. The system should therefore ensure that relevant simulation data is readily retrievable. Additionally, designers should actively determine which data are most relevant, guided by a deep understanding of the users' goals."

Compatibility with larger applications (DP5). Based on DR16 and DR17, we derive DP5: "*Offer seamless compatibility with other applications to enhance the simulation system's utility*." Beyond individual simulation tasks, users often need to incorporate outputs into broader systems. To facilitate this, the simulation system should provide well-defined interfaces for inputs and outputs, ensuring they can be easily connected to external models or systems.

Easy verification of accuracy and correctness (DP6). Based on critical outputs as highlighted in DR12, DR13, and DR14, and the need for verifiable algorithms specified in DR18, we define DP6 as: "*Ensure the accuracy and correctness of simulation outputs are verifiable*." Users must trust that the simulation system accurately executes its mechanisms. To support this, the system should enable users to audit both the simulation logic and the quality assurance processes in place."

Data privacy protection (DP7). From DR19, we formulate DP7: "*Provide strong safeguards over the privacy of input data*". Trust in data security is essential for software adoption (Brunotte, 2023), and regulatory compliance may restrict software use if not met. Simulation software must therefore limit its data use to what is necessary and enforce strong security measures for stored data.

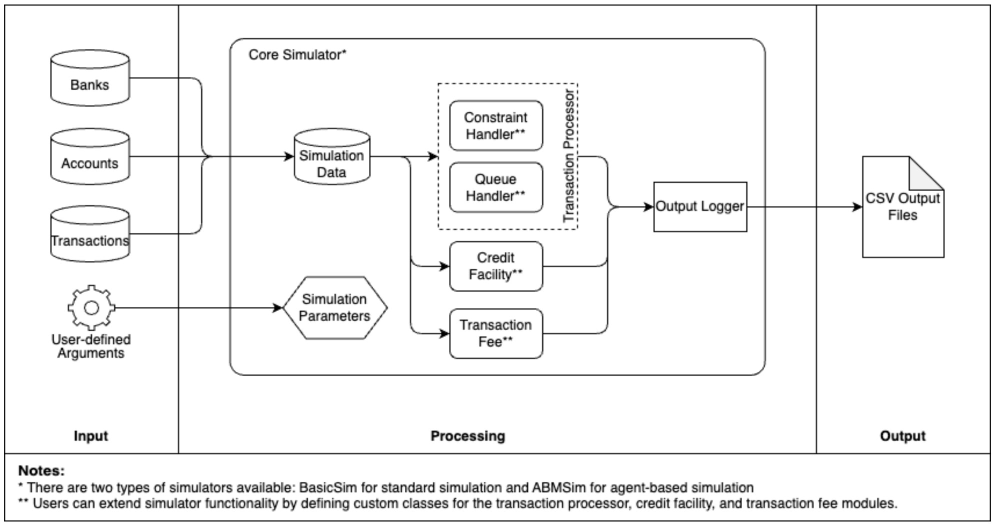
Accessible and user-friendly customization process (DP8). From DR20, we establish DP8 as: "*Enable accessible and user-friendly customization processes*." This principle ensures that end users can effortlessly configure the simulator, irrespective of their technical expertise. The system should support intuitive adjustments and modifications, utilizing technology that is familiar to a broad user base to enhance ease of use and accessibility.

Artifact Implementation

We implement our artifact, *PSSimPy*, as a Python library that users can import to incorporate into their scripts or Jupyter notebooks. The artifact is implemented based on a set of design features derived from the design principles outlined in the previous section. At the time of writing this paper, our implemented artifact is at version 0.1.1 and the overall architecture is presented in Figure 3. We made an intentional decision to build our artifact using Python as it is the most popular programming language (TIOBE, 2024), where the simple syntax and gentle learning curve make it accessible to non-technical users (Wyner & Lubin, 2011).

Modular and flexible system architecture (DF1). We derive this principle from DP1, DP2, and DP3, which are concerned with the ability of users to customize system objects and scenario parameters, and

DP5, which is concerned with the inputs to and outputs from the system. To comply with this principle, the entire library is built using Object-Oriented Programming (OOP) principles, where all objects are represented as Python classes. Abstract classes are provided for objects that are intended to be customizable, where users can inherit and implement the abstract methods for these classes to create their own customizable logic. Additionally, we allow the Transaction and Account classes to be seeded with keyword arguments (*kwargs*) that users can utilize to include additional data points they may deem necessary to capture in their simulation models. This design feature was further reinforced as we crafted the demonstration examples. The initial implementation of *PSSimPy* failed to fully comply with this design feature, and we quickly realized that it would not be able to demonstrate the blockchain use case that we identified. The current version of the artifact is after we refactored the code and design to ensure proper modular and flexible architecture, such that it can support the simulation of current and future states of LVPSs.



Notes:
* There are two types of simulators available: BasicSim for standard simulation and ABMSim for agent-based simulation
** Users can extend simulator functionality by defining custom classes for the transaction processor, credit facility, and transaction fee modules.

Figure 3. *PSSimPy* Architecture

Real-time processing and event-driven design (DF2). This is derived from DP2, DP3, and DP4, which dictate the flow and reporting requirements of the simulation system. We make use of the *SimPy* library to support discrete-event simulation. Transactions are grouped into batches based on time windows defined by the user. Each batch is added to the pool of outstanding transactions awaiting processing. Ordinarily, outstanding transactions would be processed in the same period they are added to the pool. However, in compliance with DR11, our artifact also supports ABM which may result in outstanding transactions being processed in a different order. When the ABM functionality is used, the outstanding transactions that get processed within a time window are determined by the defined bank strategies. Outstanding transactions that are identified for processing are sent to the Transaction Processor module to complete the settlement process, as shown in Figure 3. Figure 4 provides a more detailed depiction of the settlement process for each batch of transactions. At the end of each period, relevant logging data is printed to comma-separated value files.

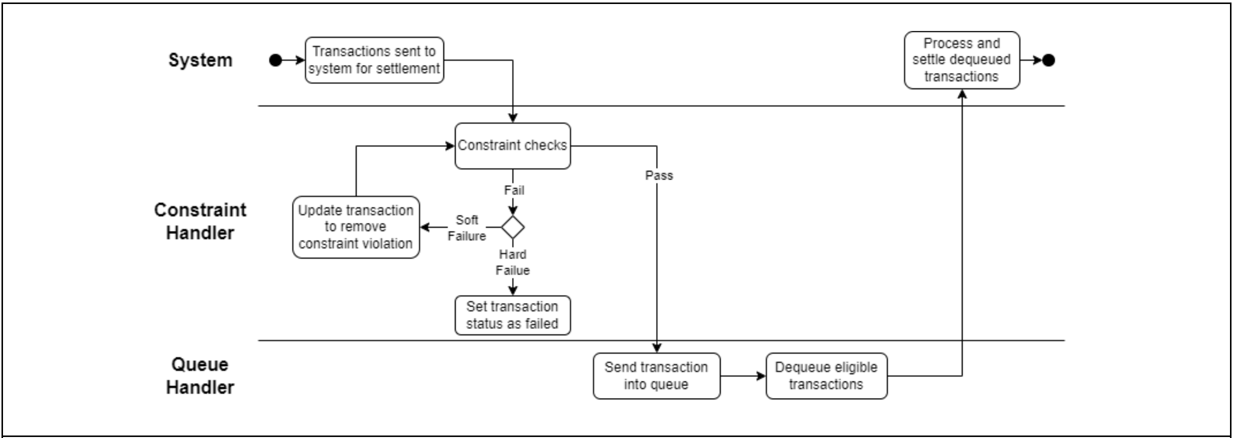

System
Transactions sent to system for settlement
Process and settle dequeued transactions
Constraint Handler
Constraint checks
Pass
Fail
Soft Failure
Update transaction to remove constraint violation
Hard Failue
Set transaction status as failed
Queue Handler
Send transaction into queue
Dequeue eligible transactions

Figure 4. Transaction Settlement Process in a Single Time Period

User-centric design and accessibility (DF3). This feature is formulated from DP6 and DP8, which covers the ease of customization and validation by end users. We satisfy this principle through the choice and openness of our artifact. As mentioned previously, the choice of Python makes customization readily accessible to even non-technical users. However, we acknowledge that some users may not require the full extent of customizations available and thus, we keep the actual simulation interface simple. By having default settings for the simulator, basic users can painlessly utilize our artifact by simply providing the details of the simulation participants and transactions, as well as basic parameters such as the simulation period. Additionally, we provide users with the option to not provide transaction inputs and instead, have the system generate random transactions between banks. This is useful in certain ABM use cases where the goal is to identify emergent patterns and trends and thus it is not as important for there to be real transaction input. This functionality makes the artifact more accessible to such users as this eliminates the need to manually generate a sufficiently rich set of transaction input data, which can be a rather time-consuming task.

Robust testing and verifiability (DF4). This is linked to DP6, which stipulates that users should be able to verify the accuracy of the simulation. The open-source nature of our artifact allows anybody to view our source code. This not only allows users to view our abstract class templates for customization work but also provides auditors the ability to validate our algorithms. Furthermore, we write extensive unit tests to demonstrate that the functionalities of the artifact are in line with the expected outcomes. The unit tests are also publicly accessible for auditors to assess the robustness of the test cases.

Minimal data retention (DF5). This is formulated from the privacy concerns highlighted in DP7. Our artifact is compliant with this principle by ensuring that all data is processed and stored locally on the device the client runs the simulation on. No data is collected, sent to a database, or sent to external parties, as can be validated through our publicly available source code, without the user explicitly writing code to do so.

Demonstration

No.	Scenario	DP1	DP2	DP3	DP4	DP5	DP8
1	Liquidity effects of a blockchain-based LVPS implementation	✓		✓	✓	✓	✓
2	Effect of a bank failure		✓		✓	✓	
3	Agent-based modeling	✓	✓		✓	✓	✓

Table 1. Alignment of Demonstration Scenarios with DPs (Excluding DP6 and DP7)

To illustrate how our artifact has covered the DPs, we constructed three scenarios demonstrating the use of the artifact. We devised the scenarios based on issues commonly faced by central bankers and researchers when assessing and designing an LVPS. Table 1 summarizes the alignment of each scenario with the DPs, excluding DP6 and DP7. We addressed DP6 in the unit testing by ensuring correct output of the artifact, while DP7 is exemplified with the input-output process using flat CSV files, both of which can be accessed in the documentation and verified through the source code.

Scenario 1: Liquidity effects of a blockchain-based LVPS implementation

In this scenario, we capitalize on the tokenization feature of blockchain. Specifically, we explore the possibility of a collateral-based credit facility, in addition to accepting standard securities as collateral, also accepting tokenized incoming transactions as collateral.

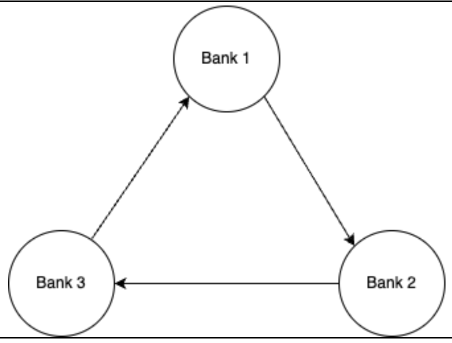


Figure 5. Illustration of Gridlock Scenario Between Three Banks

We use the abstract credit facility class that is shipped with our artifact and implement a new credit facility class that accepts incoming transactions as collateral for intraday credit issuance if an account does not have sufficient traditional securities to post as collateral. We then set up a classic triangle gridlock scenario between three banks, as shown in Figure 5. Each bank has an account balance of 0 and no traditional collateral securities. Thus, under the traditional collateral credit facility mechanism, all the transactions would have failed. However, when we use the new credit facility class that we implemented, we observe that the transactions are now successful. The simulation output of this scenario is documented in Table 2.

Normal Collateralized Credit Facility				**Collateralized Transactions Credit Facility**			
Sender	**Receiver**	**Amount**	**Status**	**Sender**	**Receiver**	**Amount**	**Status**
Bank 1	Bank 2	1	Failed	Bank 1	Bank 2	1	Success
Bank 2	Bank 3	1	Failed	Bank 2	Bank 3	1	Success
Bank 3	Bank 1	1	Failed	Bank 3	Bank 1	1	Success

Table 2. Transaction Processing Results of Gridlock Scenario with Normal Collateralized and Collateralized Transactions Credit Facilities

Therefore, our artifact is flexible enough to accommodate hypothetical features from novel technologies such as blockchain that have never been implemented in any actual LVPS. Users are able to test out new technology features and assess their impact on important metrics such as transaction success and system liquidity.

Scenario 2: Effect of a bank failure

We illustrate how *PSSimPy* can be used to perform stress testing. In this case, we observe the credit facility usage when a bank suddenly fails. We set up a simple scenario with three banks, starting with an account balance of 225, 25, and 25 respectively, with the transaction arrivals detailed in Table 3.

We then have each bank fail at 08:15 and observe how the failure affects the total system's credit facility usage over the course of the simulation. The results are plotted in Figure 6. From the results, an analyst might be able to identify that Bank 1's failure significantly affects the system's requirement for additional liquidity and deem it systemically important.

Arrival Time	Sender Bank	Recipient Bank	Amount
08:00	Bank 1	Bank 2	30
08:00	Bank 1	Bank 3	30
08:15	Bank 1	Bank 2	30
08:15	Bank 1	Bank 3	30
08:15	Bank 3	Bank 2	50
08:30	Bank 1	Bank 2	30
08:30	Bank 1	Bank 3	30
08:30	Bank 3	Bank 2	30
08:45	Bank 1	Bank 2	30
08:45	Bank 1	Bank 3	30
08:45	Bank 2	Bank 1	120

Table 3. Arriving Transactions in Bank Failure Scenario

This illustration demonstrates our artifact's capability to evaluate the impacts of bank failures, serving as a valuable resource for regulators monitoring systemically important financial institutions. Although we focus on credit facility usage as a primary metric to assess system illiquidity, the artifact's design accommodates deeper analysis through the provision of a comprehensive set of data points, enabling the creation of customized metrics. This flexibility highlights its potential as a tool for enhancing regulatory oversight and financial stability assessments.

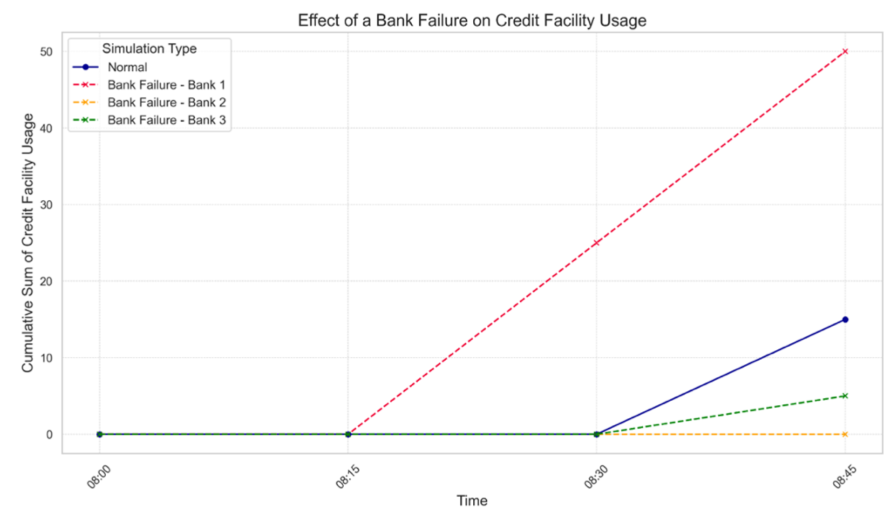


Figure 6. Effect of a Bank Failure on the Cumulative Credit Facility Usage

Scenario 3: Agent-based modeling

ABM is a powerful tool that can uncover dynamics in complex systems where analytical solutions may not be feasible. We demonstrate our simulator's ABM capability through a simple scenario involving three banks with a modified strategy. For simplicity of demonstration, we model homogenous agents, but we would like to highlight that our simulator also supports heterogenous agents. The detailed code implementation for this ABM example can be found in our artifact's technical documentation. To start, we create a new kind of bank agent, called *PettyBank*. *PettyBank*'s strategy is that it will not settle any outgoing transactions to a counterparty bank if there are incoming outstanding transactions from the counterparty bank.

We set up our scenario to involve three banks, all *PettyBank* agents, each with one account. We define a set of transactions and arrival times, where the transaction arrival times dictate when the obligations arise, while the actual settlement time of the transactions depends on the strategies of the respective banks. We then run the ABM simulator module and observe the results in terms of cumulative transaction amounts of arrived and processed transactions, as depicted in Figure 7.

As we can observe, we see that there is a divergence at 08:15 between the arrived and processed transaction amounts. This is because, at 08:30 of our scenario, all the banks have outstanding transactions with each other. This results in a gridlock and thus, based on the defined *PettyBank* strategy, banks stop paying each other. This scenario shows that our simulator can be used to model banks as strategic agents to conduct ABM. Users can customize their own agents for their own modeling purposes.

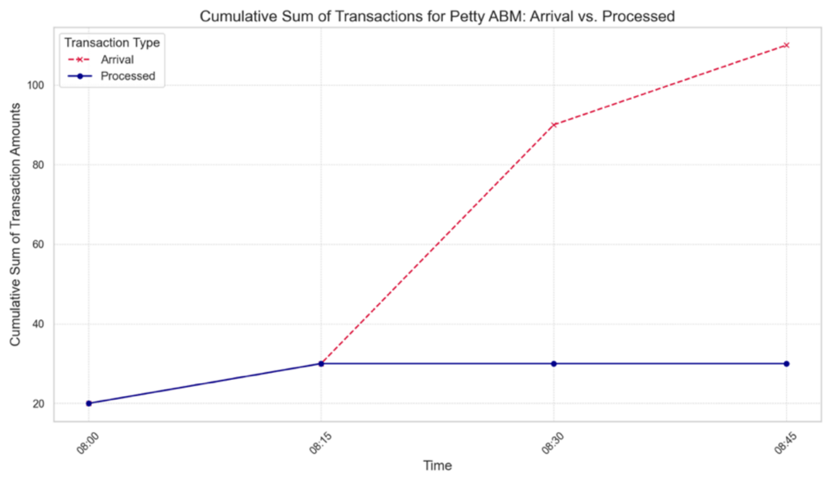


Figure 7. Cumulative Transaction Amounts in ABM Scenario for *PettyBank*

Discussion and Limitations

Overall, this paper provides both practical and theoretical contributions. We introduce *PSSimPy*, a comprehensive LVPS simulator developed through a Design Science approach. *PSSimPy* addresses the need for advanced simulation tools in the study of LVPSs, particularly with the emergence of blockchain and DLT. As a publicly available Python library, it equips researchers and practitioners with the means to explore LVPS designs and behaviors in depth. The design principles and features formulated may also be generalized to the wider simulation design literature and provide a base for future implementations of simulators. When comparing the design requirements identified by our process with those previously identified by CAS literature, three major features emerge as likely to be generally important for any CAS simulation that is intended to be used by real-world participants in the finance or blockchain setting: modularity, accessibility, and data privacy. Modularity is vital because CAS environments constantly change, and modularity allows simulations to adapt over time. Accessibility is important because simulations confined to their creators have limited impact; making them accessible enables reproduction and innovation by other researchers. Data privacy is essential as many CAS simulations require sensitive data for realism, and ensuring privacy leads to more accurate and credible output.

The case of the LVPS model illustrates how useful these simulations are when modeling adversarial or seemingly irrational behavior in financial systems which is a core concern for proper mechanism design within the blockchain space. Our findings regarding the design principles and features may also be extended to the design of simulations more broadly. However, the extent of this generalization depends on the specific nature of the problem being simulated, as the applicability of these principles and features may vary based on their similarities with the LVPS context. Furthermore, in the current design cycle, design requirements could only be formulated based off publicly available literature. It may be possible that there could be some requirements that may not have been discovered as issues or desired use cases have been kept to internal communications among practitioners.

Conclusion and Future Work

In this study, we employed a Design Science approach to develop a simulator that addresses the complex challenges of LVPS research, particularly relevant with the emerging changes in wholesale payments prompted by the introduction of wCBDCs. Our work enriches CAS literature by establishing design guidelines for constructing effective simulators tailored for CAS studies.

Design Science is inherently iterative, and we intend to continually refine our simulator through further evaluations and design revisions. A subsequent design cycle could include testing the prototype with researchers and central bankers, leveraging their structured feedback to enhance its functionality. Moreover, our initial case study utilized a simplified and abstract definition of blockchain. Future research should explore specific blockchain technologies pertinent to wCBDC initiatives more closely, assessing whether *PSSimPy* is sufficiently robust to support simulations of such complex environments.

Acknowledgements

We would like to thank the NUS FinTech Lab for their support and guidance for this research.

References

Anderson, H., Paddrik, M., & Wang, J. J. (2019). Bank Networks and Systemic Risk: Evidence from the National Banking Acts. *American Economic Review*, *109*(9), 3125–3161.

Angelini, P., Maresca, G., & Russo, D. (1996). Systemic Risk in the Netting System. *Journal of Banking & Finance, 20*(5), 853–868.

Arciero, L., Biancotti, C., D'Aurizio, L., & Impenna, C. (2008). Exploring Agent-Based Methods for the Analysis of Payment Systems: A Crisis Model for StarLogo TNG. *Journal of Artificial Societies and Social Simulation*, *12*(1).

Arjani, N. (2006). Examining the Trade-Off between Settlement Delay and Intraday Liquidity in Canada's LVTS: A Simulation Approach. *Staff Working Papers*, Bank of Canada.

Auer, R., Frost, J., Gambacorta, L., Monnet, C., Rice, T., & Shin, H. S. (2022). Central Bank Digital Currencies: Motives, Economic Implications, and the Research Frontier. *Annual Review of Economics*, *14*(1), 697–721.

Auer, R., Haslhofer, B., Kitzler, S., Saggese, P., & Victor, F. (2023). The Technology of Decentralized Finance (DeFi). *Digital Finance*.

Bakšys, D., & Sakalauskas, L. (2006). Modelling of Interbank Payments. *Technological and Economic Development of Economy*, *12*(4), 269–275.

Bech, M. L., & Garratt, R. (2003). The Intraday Liquidity Management Game. *Journal of Economic Theory, 109*(2), 198–219.

Bech, M., Hancock, J., Rice, T., & Wadsworth, A. (2020). On the Future of Securities Settlement. *BIS Quarterly Review*.

Bech, M. L., Preisig, C., & Soramäki, K. (2008). Global Trends in Large-Value Payments. *Economic Policy Review*, *14*(2), 59–81.

Brunotte, W., Specht, A., Chazette, L., & Schneider, K. (2023). Privacy Explanations – a Means to End-User Trust. *Journal of Systems and Software*, *195*.

Buckle, S., & Campbell, E. (2003). Settlement Bank Behavior and Throughput Rules in an RTGS Payment System with Collateralized Intraday Credit. *Bank of England Working Paper Series*, Bank of England.

Chen, M. A., Wu, Q., & Yang, B. (2019). How Valuable Is FinTech Innovation?. *The Review of Financial Studies*, *32*(5), 2062–2106.

Chiu, J., & Koeppl, T. V. (2019). Blockchain-Based Settlement for Asset Trading. *The Review of Financial Studies*, *32*(5), 1716–1753.

Colliard, J. E., Hurlin, C., & Pérignon, C. (2023). Reproducibility Certification in Economics Research. *HEC Paris Research Paper*, HEC Paris.

Committee on Payment and Settlement Systems (CPSS). (1997). Real-Time Gross Settlement Systems. Bank for International Settlements.

Committee on Payment and Settlement Systems (CPSS). (2005). New Developments in Large-Value Payment Systems. Bank for International Settlements.

Cong, L. W., & He, Z. (2019). Blockchain Disruption and Smart Contracts. *The Review of Financial Studies*, *32*(5), 1754–1797.

Corbae, D., & Gofman, M. (2020). Interbank Trading, Collusion, and Financial Regulation. *Proceedings of the Meetings of the American Economic Association.*

De Caux, R., Brede, M., & McGroarty, F. (2016). Payment Prioritisation and Liquidity Risk in Collateralised Interbank Payment Systems. *Journal of International Financial Markets, Institutions and Money*, *41*, 139–150.

Di Lorio, A., Kosse, A., & Mattei, I. (2024). Embracing Diversity, Advancing Together – Results of the 2023 BIS Survery on Central Bank Digital Currencies and Crypto. *BIS Papers*, Bank for International Settlements.

Docherty, P., & Wang, G. (2010). Using Synthetic Data to Evaluate the Impact of RTGS on Systemic Risk in the Australian Payments System. *Journal of Financial Stability*, *6*(2), 103–117.

Eabrasu, M. (2021). What If? Fine-Tuning the Expectations of Business Simulation Technology Through the Lens of Philosophical Counterfactual Analysis. *Organization, 30*(4), 694-711.

Enge, A., & Øverli, F. (2006). Intraday Liquidity and the Settlement of Large-Value Payments: A Simulation-Based Analysis. *Economic Bulletin*, *77*(1), 41–47.

Fung, B. S. C., & Halaburda, H. (2016). Central Bank Digital Currencies: A Framework for Assessing Why and How. *Bank of Canada Staff Discussion Paper 2016-22*, Bank of Canada.

Furfine, C. (2003). Interbank Exposures: Quantifying the Risk of Contagion. *Journal of Money, Credit, and Banking*, *35*(1), 111–128.

Galbiati, M., & Soramäki, K. (2011). An Agent-Based Model of Payment Systems. *Journal of Economic Dynamics and Control*, *35*(6), 859–875.

Gregor, S., & Jones, D. (2007). The Anatomy of a Design Theory. *Journal of the Association for Information Systems*, *8*(5), 312–335.

Guo, Z., Kauffman, R. J., Lin, M., & Ma, D. (2015). Mechanism Design for Near Real-Time Retail Payment and Settlement Systems. *Proceedings of the 48th Hawaii International Conference on System Sciences*.

Haki, K., Beese, J., Aier, S., & Winter, R. (2020). The Evolution of Information Systems Architecture: An Agent-Based Simulation Model. *MIS Quarterly*, *44*(1), 155-184.

Harvey, C. R. (2019). Editorial: Replication in Financial Economics. *Critical Finance Review*, *8*(1–2), 1–9.

Heijmans, R., & Heuver, R. (2014). Is This Bank Ill? The Diagnosis of Doctor TARGET2. *The Journal of Financial Market Infrastructures*, *2*(3), 3–36.

Heijmans, R., & Wendt, F. (2023). Measuring the Impact of a Failing Participant in Payment Systems. *Latin American Journal of Central Banking*, *4*(4).

Holthausen, C., & Rochet, J.-C. (2006). Efficient Pricing of Large Value Interbank Payment Systems. *Journal of Money, Credit, and Banking*, *38*(7), 1797–1818.

Hong Kong Monetary Authority (HKMA). (1997). Hong Kong's Real Time Gross Settlement System. *Quarterly Bulletin*, Hong Kong Monetary Authority.

Jasova, M., Laeven, L., Mendicino, C., Peydró, J.-L., & Supera, D. (2024). Systemic Risk and Monetary Policy: The Haircut Gap Channel of the Lender of Last Resort. *The Review of Financial Studies, 37*(7), 2191-2243.

Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., Rosson, M. B., Rothermel, G., Shaw, M., & Wiedenbeck, S. (2011). The State of the Art in End-User Software Engineering. *ACM Computing Surveys*, *43*(3), 1–44.

Koponen, R., & Soramäki, K. (1998). Intraday Liquidity Needs in a Modern Interbank Payment System: A Simulation Approach. *Bank of Finland Scientific Monographs*, Bank of Finland.

van Lamsweerde, A. (2001). Goal-Oriented Requirements Engineering: A Guided Tour. *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, 249–262.

Leinonen, H., & Soramäki, K. (2003). Simulating Interbank Payment and Securities Settlement Mechanisms with the BoF-PSS2 Simulator. *Bank of Finland Discussion Papers*, Bank of Finland.

Long, Q., & Li, S. (2014). The Innovation Network as a Complex Adaptive System: Flexible Multi-agent Based Modeling, Simulation, and Evolutionary Decision Making. *Proceedings of the Fifth International Conference on Intelligent Systems Design and Engineering Applications*, 1060–1064.

Market Infrastructure and Payments Committee (MIPC). (2017). Stress-Testing of Liquidity Risk in TARGET2. *ECB Occasional Paper Series*, European Central Bank.

Martin, A., & McAndrews, J. (2008). Liquidity-Saving Mechanisms. *Journal of Monetary Economics*, *55*(3), 554–567.

Mayo, K., Fozdar, S., & Wellman, M. P. (2021). An Agent-Based Model of Strategic Adoption of Real-Time Payments. *Proceedings of the Second ACM International Conference on AI in Finance*, Association for Computer Machinery.

McAndrews, J., & Potter, S. (2002). Liquidity Effects of the Events of September 11, 2001. *Economic Policy Review*, *8*(2), 59–79.

McAndrews, J., & Rajan, S. (2000). The Timing and Funding of Fedwire Funds Transfers. *Economic Policy Review*, *6*(2), 17–32.

Mills, D. C., Jr., & Nesmith, T. D. (2008). Risk and Concentration in Payment and Securities Settlement Systems. *Journal of Monetary Economics*, *55*(3), 542–553.

Monetary Authority of Singapore (MAS). (2020). MEPS+: Principles for Financial Market Infrastructures Disclosure. Monetary Authority of Singapore.

Nakajima, M. (2011). DTNS System and RTGS System. In *Advances in Finance, Accounting, and Economics* (pp. 30–41), IGI Global.

Nan, N. (2011). Capturing Bottom-Up Information Technology Use Processes: A Complex Adaptive Systems Model. *MIS Quarterly, 35*(2), 505-532.

Nellen, T. (2019). Intraday Liquidity Facilities, Late Settlement Fee and Coordination. *Journal of Banking & Finance*, *106*, 124–131.

Opare, E., & Kim, K. (2020). Design Practices for Wholesale Central Bank Digital Currencies from the World. *Proceedings of the 2020 Symposium on Cryptography and Information Security*.

Peranginangin, F. (2017). Payment System Statistics to Support Policy Formulation in Indonesia. *Proceedings of the International Statistical Institute Regional Statistics Conference*.

Perlin, M., & Schanz, J. F. (2011). System-Wide Liquidity Risk in the United Kingdom's Large-Value Payment System: An Empirical Analysis. *Bank of England Working Paper Series*, Bank of England.

Raczynski, S. (2020). Agent-Based Models: Tools. In *Interacting Complexities of Herds and Social Organizations: Agent Based Modeling* (pp. 1–18). Springer Singapore.

Rand, W., & Rust, R. (2011). Agent-Based Modeling in Marketing: Guidelines for Rigor. *International Journal of Research in Marketing, 28*(3), 181-193.

Rooj, D., & Sengupta, R. (2020). The Real-Time Impact on Real Economy— a Multivariate Bvar Analysis of Digital Payment Systems and Economic Growth in India. *ADBI Working Paper Series*, Asian Development Bank Institute.

Sabetti, L., & Heijmans, R. (2021). Shallow or Deep? Training an Autoencoder to Detect Anomalous Flows in a Retail Payment System. *Latin American Journal of Central Banking*, *2*(2).

Seebacher, S., & Schüritz, R. (2019). Blockchain–Just Another IT Implementation? A Comparison of Blockchain and Interorganizational Information Systems. *Proceedings of the Twenty-Seventh European Conference on Information Systems*.

Spinellis, D. (2006). *Code Quality: The Open Source Perspective*. Adobe Press.

TIOBE. (2024). TIOBE Index. *TIOBE*. https://www.tiobe.com/tiobe-index/, accessed February 27, 2024

Triepels, R., Daniels, H., & Heijmans, R. (2018). Detection and Explanation of Anomalous Payment Behavior in Real-Time Gross Settlement Systems. *Proceedings of the 19th International Conference on Enterprise Information Systems*, 145–161.

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User Acceptance of Information Technology: Toward a Unified View. *MIS Quarterly, 27*(3), 425–478.

Wyner, G. M., & Lubin, B. (2011). From Hello World to Interface Design in Three Days: Teaching Non-Technical Students to Use an API. *Proceedings of the Seventeenth Americas Conference on Information Systems*.

Zhou, R. (2000). Understanding Intraday Credit in Large-Value Payment Systems. *Economic Perspectives*, *24*(3), 29–44.