Urbanite: A Dataflow-Based Framework for Human-AI Interactive Alignment in Urban Visual Analytics

Gustavo Moreira, Leonardo Ferreira, Carolina Veiga, Maryam Hosseini, and Fabio Miranda

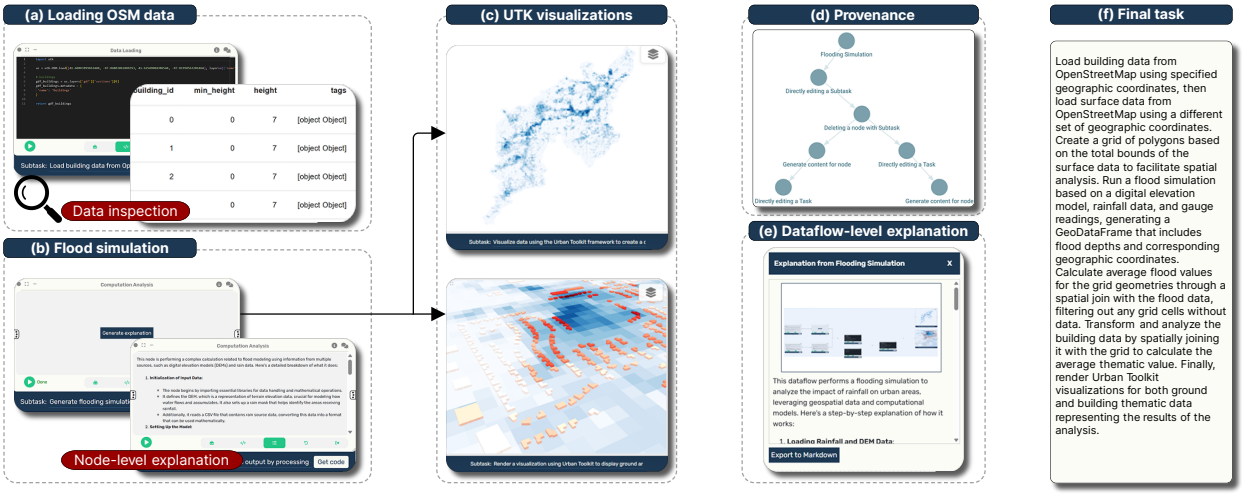

Fig. 1: Using Urbanite to analyze flood simulations. (a) The user loads a building layer and examines outputs through **provenance and data inspection**. (b) They create flood simulation nodes for a Chicago-area town and document them using **dataflow- and node-level explanations**. (c) Results and impacted buildings are visualized using UTK nodes. (d) The user modifies simulation parameters, branching the dataflow with **provenance and data inspection**. (e) To enhance documentation, they describe the dataflow using **dataflow- and node-level explanations**. (f) The final task is generated via **task & subtask definition**, summarizing the dataflow.

Abstract—With the growing availability of urban data and the increasing complexity of societal challenges, visual analytics has become essential for deriving insights into pressing real-world problems. However, analyzing such data is inherently complex and iterative, requiring expertise across multiple domains. The need to manage diverse datasets, distill intricate workflows, and integrate various analytical methods presents a high barrier to entry, especially for researchers and urban experts who lack proficiency in data management, machine learning, and visualization. Advancements in large language models offer a promising solution to lower the barriers to the construction of analytics systems by enabling users to specify intent rather than define precise computational operations. However, this shift from explicit operations to intent-based interaction introduces challenges in ensuring alignment throughout the design and development process. Without proper mechanisms, gaps can emerge between user intent, system behavior, and analytical outcomes. To address these challenges, we propose Urbanite, a framework for human-AI collaboration in urban visual analytics. Urbanite leverages a dataflow-based model that allows users to specify intent at multiple scopes, enabling interactive alignment across the specification, process, and evaluation stages of urban analytics. Based on findings from a survey to uncover challenges, Urbanite incorporates features to facilitate explainability, multi-resolution definition of tasks across dataflows, nodes, and parameters, while supporting the provenance of interactions. We demonstrate Urbanite's effectiveness through usage scenarios created in collaboration with urban experts. Urbanite is available at urbantk.org/urbanite.

Index Terms—Urban analytics, urban data, dataflow, large language models, visualization framework, visualization system.

1 Introduction

The growing availability of urban data, coupled with increasing societal challenges, has driven the need for systems to support data-driven tasks across a range of domains, including urban planning, architecture, environmental and climate sciences, and public health. Urban visual analytics (VA) systems have been repeatedly shown to be a key component in supporting these tasks [12,15,16,34,68], distilling complex data analytics workflows into accessible visual interfaces. These systems enable urban experts to gain insights [27], detect patterns [19], or evaluate potential urban interventions [31]. Despite their growing importance, urban VA systems are still complex and resource-intensive, requiring expertise in urban science and multiple areas of computer science. Beyond technical challenges, their design is an iterative, collaborative process that must align expectations across diverse stakeholders [1].

Recently, the visualization community has been highlighting tensions and technical hurdles in the process of authoring these systems [1,21,58,59]. These can be broadly categorized into three key interconnected areas. First, urban data is inherently complex, encompassing heterogeneous, multi-scale, spatiotemporal datasets from diverse sources such as sensors, satellite and street-level imagery, and authoritative records. The domain tasks are equally intricate, involving

- *Gustavo Moreira, Leonardo Ferreira, and Fabio Miranda are with the University of Illinois Chicago. E-mail: {gmorei3, lferr10, fabiom}@uic.edu.*
- *Carolina Veiga is with the University of Illinois Urbana-Champaign. E-mail: carolvfs@illinois.edu.*
- *Maryam Hosseini is with the University of California, Berkeley. E-mail: maryamh@berkeley.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

multiple coordinated steps [16, 34]. These complexities can stiffen the iterative and ideation cycles essential to visualization design [37], constraining flexibility and slowing down exploration of the visual analytics design space. Second, developing VA systems demands diverse technical skills, often creating barriers for interdisciplinary teams. Domain experts may lack visualization expertise, while visualization experts may not fully understand domain-specific workflows, resulting in misaligned expectations and slower iteration. These knowledge gaps limit domain experts' contributions, diminishing the richness of collaboration by overlooking the valuable insights they can offer during the creative and conceptual phases of visualization design. Third, developing such systems poses a high barrier to entry, and although urban experts are increasingly adopting data-driven approaches, they still face challenges in contributing to their design [1, 58]. Consequently, they are often treated as end-users rather than active participants in system development. To address these challenges, many approaches now use low- or no-code paradigms to support VA [25, 26, 29, 35, 44]. However, these frameworks often lack the flexibility needed for complex urban data analytics workflows and the usability required for meaningful expert involvement.

Recent advancements in large language models (LLMs) provide an opportunity to further tackle these challenges by allowing users to focus on specifying intent instead of computational operations, helping non-technical users articulate goals without programming expertise. However, this shift from explicit operations to intent-based interaction introduces challenges in ensuring alignment throughout the design and development process. Without proper mechanisms, gaps can emerge between user intent, system behavior, and analytical outcomes. For example, LLMs can generate plausible but incorrect outputs [49], and they lack transparency and structured mechanisms for iterative refinement of results. Addressing these challenges requires new approaches that combine the strengths of LLMs with more structured design mechanisms, giving users the control to iteratively refine processes while benefiting from AI-assisted intent specification. Additionally, the complex and iterative nature of these processes calls for tracking mechanisms to help users explore alternative design scenarios and recall their own process. Previous frameworks leveraged provenance [36, 46, 48, 55] as an alternative to provide a structured and easy-to-access approach to explore user actions and, in this case, interactions with the LLM.

To address these challenges, we propose Urbanite, a framework for human-AI collaboration in urban VA. Conceptually, we ground the design of Urbanite in the need for user-centered human-AI alignment across three objectives, as recently proposed by Terry et al. [52]: specification (what the AI should do), process (how the AI should do it), and evaluation (verifying and understanding what was done). We also incorporate insights from a survey of experienced system builders, identifying key pain points in designing and developing urban VA systems. Building on the theoretical framework and empirical findings, Urbanite integrates novel features designed to enhance human-AI collaboration in urban VA. At its core, Urbanite follows a dataflow model, structuring systems as a series of nodes and edges that represent modular and interoperable VA components. In other words, rather than treating VA systems as static software artifacts, we treat them as dataflows that can be iteratively constructed, adapted, and refined. Such an approach enables the creation of lightweight, task-specific "VA-lite" systems. Urbanite's dataflows are organized around a specification language, which ensures consistency in defining and refining analytical workflows. Additionally, this language serves as a layer between the LLM and the user, constraining the model's output to ensure alignment with analytical goals and system constraints. To support alignment between user intent and system behavior, Urbanite introduces design features that operationalize the alignment principles. First, it enables users to specify their analytical goal in natural language through a series of guided prompts that result in a task, subtasks, and a dataflow sketch. Second, it generates contextualized code to fulfill the predefined task and implement the sketch, as well as continuous node and edge suggestions to guide the process. Third, explainability and provenance mechanisms support the evaluation of the final result and the exploration of alternative scenarios. Our contribution also lies in demonstrating how LLMs, embedded within a dataflow system, can enhance urban VA by bridging the semantic gap between the complex (and often ill-defined) intents of experts and the formal, structured requirements of VA systems in this domain. Specifically, we show how such integration lowers the barrier for domain experts to translate high-level goals into executable workflows, facilitates iterative exploration while maintaining transparency, and enables human-AI alignment through the dataflow acting as an interpretable intermediary between LLM outputs and expert understanding. We evaluate Urbanite through usage scenarios, experts' feedback, and a quantitative evaluation. Urbanite is available at urbantk.org/urbanite.

2 Related Work

2.1 Urban visual analytics systems

Urban VA combines computational methods and interactive visualization to help experts analyze complex urban data [67]. Recent surveys highlight the field's growth and complexity [3, 12, 15, 16, 34, 68]. Recently, Deng et al. [12] surveyed over 200 visualization papers, focusing on how different models and analysis approaches are integrated in the visualization workflow, while our recent reviews surfaced over 450 papers on 3D urban data [34] and over 130 papers with requirements for urban-specific frameworks [16]. These works have demonstrated the importance of supporting diverse stakeholders, including urban planners, architects, social scientists, climate researchers, public health professionals, and community advocates, each with distinct analytical needs and priorities. Despite the proliferation of urban VA tools, they still face fundamental challenges, particularly their difficulty in construction and their siloed nature, as detailed in Section 3. As a result, urban VA remains fragmented, with many systems operating in isolation instead of being part of unified ecosystems.

Given these challenges, experts increasingly turn to computational notebooks, which provide an accessible, flexible, and reproducible way to analyze urban data without the burden of developing full-fledged VA systems. However, while notebooks offer ease of construction and experimentation [23], they lack the domain-specific capabilities, scalability, and interactivity of bespoke systems. Addressing these challenges requires a shift towards more modular, extensible, and adaptable frameworks that can bridge the gap between bespoke applications and broadly applicable urban analytics solutions. In our previous work, Curio [36], we focused on enabling human-human collaboration by supporting teams of experts in jointly exploring and analyzing data. With Urbanite, we extend this vision to human-AI collaboration, leveraging a natural language interface that allows users to engage with the system more intuitively. Urbanite combines the flexibility of custom VA systems with accessibility, modularity, and reproducibility, letting experts build and share analyses without traditional development overhead. By integrating an AI assistant through LLMs, Urbanite further lowers barriers by assisting users in navigating complex workflows, automating data steps, and generating visualizations.

2.2 LLM- and NLI-enhanced visual analytics

LLMs have recently proven effective for tasks like text summarization, report generation [30], and programming support [9]. Their integration with natural language interfaces in VA is emerging as a way to reduce barriers to complex data analysis. Shen et al. [43] provided a comprehensive survey of NLI applications in visualization, outlining emerging trends and challenges, including provenance tracking of prompts for interpretability. Basole and Major [4] further explored the role of generative AI and natural language prompts across the visualization workflow, identifying key opportunities and obstacles in integrating AI-driven assistance into analytical processes. While prior studies highlight broad challenges in integrating NLIs and LLMs into visualization workflows, Terry et al. [52] examine the alignment of human and AI capabilities within interactive systems. Their work proposes a conceptual framework for effective human–AI integration, complementing more focused efforts that embed LLMs directly into the visualization process. Zhao et al. [66] proposed using LLMs to support VA across multiple stages of user workflows, including onboarding, exploration, and summarization. They also introduced a specification framework for VA systems that enables LLMs to interpret visual views and their interrelationships. Tian et al. [53] explored the use of LLMs

for generating charts, proposing a step-by-step reasoning pipeline that decomposes visualization tasks into more manageable sub-tasks. Tang et al. [51] investigated the use of LLMs for summarization using visual workspaces as a steering mechanism. Sah et al. [40] used LLMs to generate visualization specifications by detecting data attributes, inferring tasks, and recommending visualizations. L'Yi et al. [25] built a visualization system that combines multimodal interfaces, including natural language, to improve usability.

We draw upon research on alignment in interactive systems [2, 52, 62], particularly the shift towards the specification of outcomes rather than operations. Urbanite builds on these principles by introducing new mechanisms for aligning AI assistance with user intent in urban analytics. Influenced by Zhao et al. [66], we also introduce an urban VA-specific specification for the creation of systems, which acts as a mediator between the AI and the user prompt. Urbanite embeds LLM guidance into structured dataflows, enabling transition between natural language interaction and interactive visual exploration.

2.3 Dataflow-based approaches in visual analytics

The use of dataflows as a structured paradigm for data analysis has long been explored across various topics, including database [47, 50], information visualization [14, 20, 64], scientific visualization [7, 57], and visualization education [45]. Systems employing this paradigm use diagrams with nodes representing data transformations and visualizations, and edges defining the dataflow. Users then interactively compose data pipelines to specify their intent, often through drag-and-drop interfaces that allow visual construction and modification without requiring extensive programming expertise. Importantly, dataflow systems promote transparency, modularity, and reusability, allowing analytical components to be adapted across different usage scenarios [36]. Recent works have extended the applicability of dataflows within the context of VA. Ulbrich et al. introduced sMolBoxes [54], a dataflow model designed for molecular dynamics exploration, demonstrating how structured dataflows can support specialized scientific tasks. Yu and Silva presented VisFlow [64], a framework that supports the creation of interactive charts within a dataflow. The framework was later extended into FlowSense [65] by incorporating a natural language interface featuring a semantic parser that recognizes specific utterances within the dataflow environment. Curio [36] builds upon these principles by introducing a collaborative dataflow environment. Unlike traditional dataflow systems that primarily focus on individual data pipeline creation, Curio facilitates collaboration through discussion spaces, annotation mechanisms, and provenance tracking.

Building on Curio, our system enhances dataflow-based VA by integrating AI-assisted guidance for urban analytics. While Curio focused on human-human collaboration, our system introduces human-AI collaboration, allowing users to describe analytical intent in natural language, which is then translated into structured dataflows.

3 Background: Challenges in Urban Visual Analytics

Urban analytics has emerged as a valuable approach to support decision-making in urban contexts, enabling domain experts to explore and analyze diverse urban data sources [11, 13, 19, 24, 32, 33]. These experts, such as urban planners, engineers, climate scientists, and public health professionals, primarily rely on three environments to support these analyses: off-the-shelf tools, computational notebooks, and urban VA systems. Off-the-shelf tools, such as ArcGIS, offer standardized functionalities but often come with steep learning curves [69]. Computational notebooks require programming skills [56, 63]. In contrast, urban VA systems simplify workflows into interactive tools to analyze data. However, development remains challenging due to the complexity of heterogeneous, multimodal data, and the need to support collaboration among stakeholders with diverse expertise and priorities.

These challenges complicate the creation of broadly applicable VA systems suited for real-world urban settings. For example, large-scale streaming and image data often require intensive wrangling or costly feature extraction, slowing iteration and locking in early design choices. Without standardized, reusable pipelines, efforts are often duplicated, leaving most urban VA systems bespoke and narrowly tailored. The visualization community has recognized this foundational challenge [58], but it is especially pronounced in urban VA due to unique contextual constraints. Unlike other domains with standardized data formats, urban data varies by region due to local practices and policies. Combined with the black-box nature of bespoke systems, this restricts their adaptability and keeps the broader applicability of urban VA systems limited. As a result, urban experts struggle to build on prior work, and insights from one context rarely transfer, reinforcing fragmentation.

Recognizing these limitations, recent years have seen a push towards more modular and adaptable urban VA frameworks [8, 35, 36, 42]. Inspired by toolkits and frameworks in other domains [6, 39, 61], researchers have begun decomposing complex systems into modular building blocks, offering greater reusability and extensibility. This modular decomposition works well because VA systems can be modeled and interpreted, fundamentally, as complex dataflows. With Curio [36], we took a first step towards this vision with a dataflow-based framework grounded in a knowledge base of reusable VA components, primarily supporting human-human collaboration between visualization researchers and urban experts. Curio bridges monolithic VA systems and computational notebooks by making operations transparent to both urban experts and visualization researchers. However, its dataflow and drag-and-drop interface still demand that users understand data operations and manually build workflows. LLMs present a promising avenue to overcome these barriers. By enabling natural language interactions for the authoring of VA components, LLMs provide domain experts with a more intuitive way to specify their intent, facilitating expert-AI collaboration. Unlike traditional interfaces that bridge the gaps of execution and evaluation, however, LLM-based systems introduce a qualitatively different interaction paradigm: *intent-based outcome specification* [38]. Instead of specifying operations, users describe desired outcomes, requiring well-designed human-AI interfaces to bridge additional alignment gaps in specification, process, and evaluation [52]. Our work designs mechanisms to bridge human-AI collaboration gaps, leveraging LLMs for multi-level intent specification. This enables urban experts to define goals, implement dataflows to fulfill them, and understand the process via automated documentation and provenance.

4 Understanding Urban Visual Analytics Challenges

With Urbanite, we aim to simplify dataflow creation for urban VA by bridging technical complexity and domain expertise. As a step towards this vision, we were particularly interested in exploring what it would mean to have an AI-enhanced system that assists urban experts in authoring their own VA interfaces. While our objective centers on supporting urban experts directly, our goal with this survey was to understand, from the perspective of visualization experts, the key gaps and collaborative dynamics that shape the development of urban VA systems. We explored how teams bridge expertise, define goals, document work, and coordinate across design phases. By gathering insights from recent projects, we aimed to surface challenges, best practices, and opportunities to guide Urbanite's design.

Survey design and participants. The survey had two parts. First, participants reflected on working with urban experts, covering project duration, code availability, collaboration barriers, iteration frequency, expectations, and documentation practices. Then, participants estimated time spent across four design stages (Understand, Ideate, Make, Deploy), following McKenna et al. [28], describing key activities, tools, and decisions. We recruited 10 participants, targeting first or second authors of urban VA system papers, based on our prior surveys [16, 34]. The group included 2 faculty members, 4 PhD students, 3 research staff, and 1 postdoc. We focused on visualization experts, as they are part of the VA system-building processes we want to support.

Next, we synthesize the key themes identified through our survey.

Time spent in each design stage. On average, participants spent 24% on Understand, 23% on Ideate, 33% on Make, and 20% on Deploy. Notably, two spent 40% on Understand (deep exploration), while two others spent 40% on Deploy (system refinement).

Bridging the expertise gap between domains. Five participants reported misunderstandings or communication breakdowns between urban experts (e.g., architects, urban planners) and visualization experts. The most cited barrier (40%) was unclear or incompatible terminology

between visualization and domain experts. Another 30% reported multiple barriers, including technical limitations, expertise gaps, differing methods, and communication issues.

Narrowing down analytical goals. Some participants (40%) experienced evolving or confusing goals, highlighting variability in how analytical objectives were established. Reflecting on their iterative approach, one participant noted: "I began by implementing simple, easy-to-understand visualizations based on my understanding of the data. I then consulted domain experts to gather insights and feedback. Using their feedback, I experimented with combining multiple visualizations into single views." While most participants felt the iterative process of refining their systems was as expected (60%), 40% experienced either more or fewer iterations than anticipated, pointing to inconsistent expectations about the collaborative design process.

Tool availability and dissemination. Survey responses revealed that open-source dissemination of urban VA systems remains limited. Only one participant explicitly indicated that both the system and its code were made publicly available in a way that qualifies as open source. Two others shared only the system interface. Four participants reported that neither the system nor the code was made publicly accessible, typically due to institutional, privacy, or operational constraints. Fully open dissemination of tools is still relatively uncommon, limiting opportunities for reuse, validation, and community uptake.

Provenance of analyses & artifacts. Most participants (60%) had clearly defined project goals, while 40% faced evolving or unclear ones, highlighting the difficulty of setting consistent objectives in interdisciplinary work. Documentation practices varied considerably, with manual documentation most commonly used. One respondent noted that much of the work involved "unglamorous software engineering." Dedicated provenance tools, such as version control or specialized tracking systems, were utilized infrequently. One participant described a more structured approach that "at various stages (...) stakeholders were shown the current version of the system."

Key takeaways. The survey highlights that the most significant challenge across projects was bridging domain expertise, primarily due to persistent communication and terminology issues between urban and visualization experts. Participants mentioned the key role of defining analytical goals, noting that projects with ambiguous or evolving objectives often resulted in inefficiencies and misaligned expectations. Furthermore, the results indicate gaps in documentation practices, with many teams relying on informal or manual methods rather than structured provenance tracking. Overall, addressing communication barriers, refining goal-setting practices, and improving documentation emerged as central themes for enhancing outcomes in urban VA.

5 The Urbanite Framework

Building on the challenges and design opportunities identified in previous work and our survey, we present Urbanite, a framework designed to support the authoring of modular, transparent, and adaptable VA workflows for urban data. Urbanite addresses the limitations of current approaches, including the opacity of monolithic systems and the technical barriers of computational notebooks, by enabling users to co-construct dataflows using reusable components and intuitive interfaces. A central aim of Urbanite is to lower the barriers that often prevent urban experts from directly engaging in the creation and adaptation of VA systems. To this end, Urbanite integrates an LLM to enable intent-based interactions throughout the authoring process, which is modeled as a dataflow. This dataflow serves as a transparent intermediary, allowing users to understand and validate the LLM's interpretation. It explicitly supports a human-in-the-loop paradigm, supporting users to iteratively refine LLM suggestions and data transformations. Furthermore, dataflow components are inherently modular and reusable, enhancing efficiency and consistency across projects. To facilitate meaningful human-AI alignment, we introduce a framework for multi-level intent specification, operationalizing this alignment across scopes and modes of expression. By allowing users to specify intent and interpret system responses at varying levels of specificity, Urbanite supports AI-driven assistance that is both traceable and adaptable to the user's analytical context. Urbanite leverages conceptual and technical foundations laid by UTK [35] and Curio [36] while incorporating an LLM-based intent translation pipeline. See supplementary material for key differences between Urbanite and these frameworks. We first introduce the framework's theoretical foundations (Section 5.1) and design goals (Section 5.2). We then present the interaction space for building dataflows (Section 5.3), detail the multi-level authoring mechanisms (Section 5.4), followed by an overview of the system (Section 5.5).

5.1 Conceptual foundations

Urbanite is built on the premise that human-AI alignment can be achieved through thoughtful interaction design. Specifically, it draws from the conceptual framework of *interactive alignment* recently introduced by Terry et al. [52], which reconceptualizes the classic human-computer interaction cycle in light of modern, intent-based AI systems. Traditional interfaces rely on direct manipulation, requiring users to choose and sequence actions to reach their goals. In contrast, human-AI interaction uses a declarative approach: users state desired outcomes, and the system interprets and executes them. This lowers entry barriers but introduces ambiguity around system capabilities, behavior, and intent alignment. Terry et al. [52] propose three distinct stages to support more transparent and effective interaction in a declarative paradigm. We adopt their conceptual framework to operationalize the human-AI interaction cycle in the context of dataflow authoring through three key alignment objectives:

(1) Specification alignment. *What the dataflow should accomplish*, that is, users' analytical intent and goals (e.g., "identify areas with high heat vulnerability"), and how these translate into system behavior;

(2) Process alignment. *How the dataflow should be constructed or operations performed*, the data transformations, analytical operations, and visualizations that the AI proposes to fulfill users' intent;

(3) Evaluation alignment. *Verifying & refining the dataflow and outputs*, ensuring that the responses are correct and meet users' needs.

In Urbanite, this alignment framework directly informs our system architecture and design. A user's analytical goals are translated into data pipelines composed of data transformations, analytical operations, and visualizations. Interaction mechanisms at multiple levels of granularity and abstraction then guide users through the authoring, refining, and evaluation of dataflows, ensuring that alignment is always met.

5.2 Design goals

Building on the previously identified challenges in urban VA and the conceptual foundations of human-AI alignment, we articulate a set of design goals that guide the development of Urbanite. These design goals (DGs) are directly informed by the real-world challenges described in Section 3 and the empirical findings from our survey in Section 4. In particular, the high technical barriers to authoring systems (DG1, DG2, DG6), the need for transparency (DG3, DG4), and the evolving nature of urban VA workflows (DG5, DG6). Each design goal reflects a specific response to misalignment risks in **specification** (S), **process** (P), **evaluation** (E), or all of them (cross cuts, CC).

DG1-S Enable natural language specifications of modular dataflows. Dataflows are inherently modular, composed of discrete components for data processing, analytical operations, and visualization. However, translating analytical goals into these components is a key barrier for domain experts. Urbanite must allow users to specify intent in natural language. Through LLM integration, Urbanite must interpret these goals and map them to partial or complete dataflows, reducing reliance on technical expertise.

DG2-P Support user control during dataflow authoring. Users must be able to control *how* the framework translates goals into operations. Rather than treating the dataflow generation as a black box, Urbanite must provide access to intermediate decisions made by the LLM, and users should accept, reject, or revise these decisions. Control must be supported through natural language (e.g., "filter the data and perform a spatial join") and manipulation (e.g., dragging a module into place).

DG3-E Enable inspection of AI-generated outputs. Outputs generated by LLMs and other AI components must be made transparent and inspectable to support evaluation alignment. Urbanite must allow users to verify whether the system's responses align with their original intent, not only in terms of final results but also in how those results were

derived. For example, if a user asks "visualize the most heat-vulnerable neighborhoods using census and weather data", Urbanite must be able to not only inspect a heatmap, but also which datasets were used and how the vulnerability index was computed.

DG4-E Trace and visualize the provenance of data transformations and human-AI interactions. Urbanite must expose the provenance of data transformations and LLM outputs throughout the lifecycle of the dataflow. This must support the evaluation and comparison of different states of the dataflow, as well as the user prompts used to generate them. For example, if an expert is exploring accessibility datasets, they may ask the system "highlight neighborhoods that are outliers in terms of accessibility problems." Over time, the expert might modify the definition of outlier or adjust the spatial resolution of the analysis. Provenance tracking must allow the expert to see which revision introduced changes, reuse, or revert earlier steps without reconstructing the dataflow from scratch.

DG5-CC Support iterative refinement across alignment stages. Authoring is inherently iterative: goals evolve, understanding deepens, and constraints shift over time. To maintain alignment between user intent and AI behavior, Urbanite must support refinement across all stages: specification, process, and evaluation. This includes the ability to revisit and revise prior steps: rephrasing analytical goals, modifying AI-generated dataflow components, or adjusting outputs based on observed results. These iterations should be fluid, allowing users to go back and forth between stages as needed. Urbanite must support these refinements through both natural language and direct manipulation.

DG6-CC Support multi-level interaction within alignment stages. While DG5 focuses on iteration across stages, DG6 focuses on depth within each stage. Given the range of user expertise and task complexity, Urbanite must support interaction at multiple levels of abstraction within specification, process, and evaluation. For example, during specification, users may articulate a high-level objective ("analyze heat vulnerability") or fine-tune specific prompts; during process, they might inspect the dataflow or dive into the inner logic of individual modules; during evaluation, they may explore summaries or individual outputs.

5.3 A flexible interaction space for building dataflows

In Urbanite, the user's primary goal is to construct a dataflow: a data pipeline of analytical operations that serves as a model of an urban VA system. The authoring revolves around two core responsibilities: (1) defining the structure of the dataflow (i.e., breadth), and (2) specifying or modifying the inner workings of each module within the structure (i.e., depth). Users may begin by sketching out a pipeline, either through natural language or by manually composing dataflow nodes. This defines the high-level flow of data and operations.

However, authoring is rarely linear. As users dive into the details of a specific module, they may realize that the structure itself needs to change. Likewise, while building out the structure, users may iteratively define or revise the behavior of individual nodes to test their ideas or explore alternative paths. To support this interaction space in Urbanite, we define a two-dimensional conceptual framework with the *scopes of specification* and *modes of expression*. The scopes of specification refer to the granularity of *what* is being authored, ranging from the entire dataflow to individual parameters within a module. The modes of expression capture *how* that specification is articulated, whether through natural language, visual manipulation, declarative grammar, or executable code. Changes made at one scope with one mode are reflected across others. For instance, updating a parameter in a node's code will be reflected in the natural language explanation. Urbanite continuously synchronizes these representations, giving users the flexibility to shift between modes and scopes while preserving consistency and intent. Table 1 and Figure 2 summarize the supported interactions and how they contribute to Urbanite's integrated, multi-modal authoring experience. The specific design features that support these dimensions in practice are presented in Section 5.4.

5.3.1 Dataflow preliminaries

Urbanite's dataflow model follows from the one first introduced in our Curio framework [36]. A dataflow is defined as a composition of modular computing nodes, connected through data and interaction dependencies, and designed to express VA workflows. Each data node takes as input and produces zero or more data layers. Although the formalism treats all nodes uniformly, each node can semantically represent a distinct operation type within the workflow. Urbanite adopts the types defined in Curio, including data wrangling and transformation, analysis and modeling, and visualization nodes. To reduce ambiguity in how goals are interpreted and operationalized, Urbanite introduces a structured *specification layer* that builds directly on this formal model. This specification serves as a machine-readable representation of the current dataflow and provides a bridge between high-level user intent and low-level implementation. It enables the system to capture both the evolving scope of specification, from entire workflows to individual parameters, and the mode of expression, whether defined through natural language, UI, declarative grammar, or executable code.

Table 1: Combinations of specification scopes and expression modes.

	Modes of expression (*how*)			
Scopes of specification (*what*)	**NL**	**UI**	**Grammar**	**Code**
Dataflow	●	●	●	
Module	●	●	●	●
Parameter		●	●	●

While Curio used this dataflow model for collaborative execution, Urbanite repurposes it as the backbone for LLM-assisted, multimodal authoring. It serves both as an execution plan and as the foundation for prompt-based refinement, traceable suggestions, and semantic alignment. The dataflow also acts as a transparent intermediary between user intent and LLM output, supporting in-the-loop refinement and reusable components. See supplementary material for full specification details.

5.3.2 Scopes of specification

We define the *scopes of specification* as the granularity in which the user interacts with Urbanite to create a dataflow. This ranges from dataflow definitions that capture the overall structure of the data pipeline, to mid-level specifications of individual modules, down to fine-grained parameter settings within those modules.

● **Dataflow-level specification** . At the dataflow level, users specify the overall structure of the workflow. This includes defining the key **nodes** that will be part of the dataflow, the data **inputs** and **outputs**, and how these nodes are connected through data and interaction dependencies. The result is a blueprint that outlines the logical flow of data and operations. Operations at this level restrict interactions to coarse-grained structural elements: users can create, modify, or remove **nodes** and **edges**, as well as define or update the **task** that describes the dataflow's high-level task. At this level, the dataflow structure provides context for downstream refinement and serves as a scaffold for LLM-driven suggestions and user guidance.

● **Module-level specification**. At the module level, users focus on specifying the behavior and configuration of a single node within the dataflow. Each node is a discrete data, analytical, or visual operation. At this level, users are restricted to modifying the internal contents of a node. This includes the **subtask** that describes the node's purpose, the **content** field that encodes the operation logic (e.g., grammar or code). The structure of the dataflow remains unchanged at this level: users are not adding new modules or rerouting connections, but refining how a specific module contributes to the overall dataflow.

● **Parameter-level specification**. At the parameter level, users engage with fine-grained aspects of the dataflow behavior: individual settings and thresholds that define how components operate. The parameters are made available to the user through UI elements using code annotations [36]. When operating at this level, user modifications are restricted to the code annotations within a single node. The structure of the dataflow and the purpose of the node remain fixed.

Provenance and interoperability across scopes. All interactions, regardless of the scope, are captured and stored in a hierarchical data structure. This interoperability is made possible by considering the dataflow specification as a central artifact.

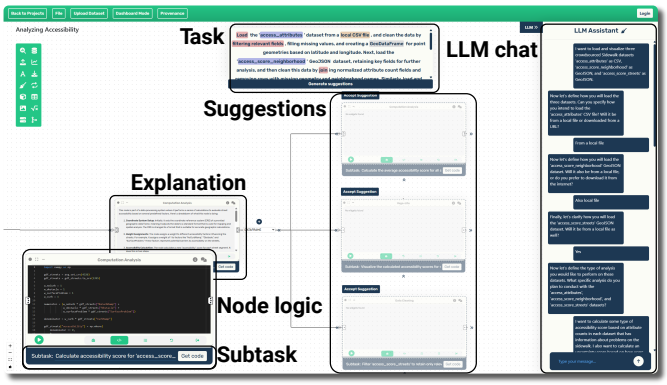

Fig. 2: Urbanite's interface, with design features for human-AI alignment.

5.3.3 Modes of expression

Scopes define *what* part of the dataflow the user edits; *modes of expression* define *how* they convey intent. In Urbanite, users can use natural language, UI, code, or grammar, with changes reflected across all modes through the shared specification.

Natural language. This is the most accessible entry point for authoring in Urbanite. Users can describe high-level tasks (e.g., "map accessibility gaps across neighborhoods") or specific prompts (e.g., "map median income to color"). These are interpreted by the LLM and translated into modifications to the dataflow specification. Natural language can be used across scopes, from defining dataflows to adjusting parameters.

User interface. The interface lets users build and edit dataflows by directly dragging nodes and adjusting parameters. Based on Curio's components, it supports extensibility: users can add elements via code annotations as reusable widgets. The UI and diagram remain the specification's anchor, whether authored visually or in code.

Code & grammar. Urbanite supports authoring via Python and declarative grammars (Vega-Lite, UTK [35]) to define node logic. Each node executes its logic using imported libraries, meaning Urbanite's performance is tied to the efficiency of the specific libraries. This enables flexible and transparent node authoring, but can constrain performance for large datasets or complex visual encodings.

5.4 Design features for human-AI alignment

Building on the earlier foundation, Urbanite implements features that align user intent with AI behavior during dataflow authoring. These features support alignment across specification, process, and evaluation, contributing to key design goals and tied to the scopes of specification.

5.4.1 Specification features

🔴 **Dataflow generation and scaffolding**. The process of dataflow creation in Urbanite begins with a conversational interaction between the user and the LLM. Through this dialogue, the user articulates their analytical goals in natural language, DG1-S. The system incrementally builds a structured task description from this input. This occurs through guided prompts, with the LLM reformulating user input into a machine-readable schema and requesting clarification as needed. This exchange supports DG5-CC. To ensure the task is well-defined, Urbanite performs schema checks that verify key elements (e.g., presence of a dataset, whether the task can be mapped to one or more supported nodes). Once requirements are met, the system transitions from intent capture to dataflow construction. At this point, Urbanite uses the finalized specification to scaffold an initial dataflow. This involves creating nodes for key analytical operations and edges for data or interaction flow. Each node is assigned a subtask (based on the main task) describing its role in natural language. Stored in metadata, it guides LLM assistance for code, grammar, and UI generation.

🔴🔵 **Task & subtask definition**. After the initial dataflow is scaffolded through the conversational interaction with the LLM, Urbanite assigns a natural language task description to the entire dataflow (supporting DG1-S), along with subtask descriptions to each node. The task captures the user's overarching analytical goal (e.g., "analyze accessibility issues across neighborhoods"), while each subtask expresses the local intent of a node within that broader workflow (e.g., "join accessibility data with demographic layers"). Such a multi-level interaction within specification supports DG6-CC. These descriptions are editable: users can revise the task to reflect updated goals or change the subtask to clarify, refine its behavior. To ensure coherence across the dataflow, Urbanite uses a two-way synchronization mechanism between task and subtasks. When a user edits the task, Urbanite triggers a decomposition routine that prompts the LLM to suggest revised subtasks for each node in the dataflow, further supporting DG6-CC. Conversely, when a user modifies a subtask, Urbanite re-evaluates the full set of subtasks to determine whether the overall task description is still aligned. This uses an LLM summarization pass to generate a natural language summary of current subtasks and check for coherence, implementation issues, or opportunities to refine or split subtasks. If found, a warning is added. All synchronization steps are tracked via the provenance model.

5.4.2 Process features

🔵 **Code generation**. For each node, Urbanite uses the natural language subtask as the primary prompt for generating the node's logic. Depending on the node type, this logic may be rendered as Python code or declarative grammar (Vega-Lite or UTK). The prompt is specified with both subtask text and contextual metadata (expected inputs and outputs, node type) to generate a candidate implementation. Users remain in control throughout the process DG2-P. They may inspect, modify, or fully rewrite the generated logic. Alternatively, the user can request Urbanite to revise the implementation by updating the subtask. To maintain consistency across abstraction levels, Urbanite synchronizes edits made with different modes of expression, supporting DG5-CC. When a user makes changes to the code or grammar, Urbanite prompts the LLM to infer and update the corresponding subtask, avoiding drifting between the descriptive and functional representations of each node.

🔴 **Connection suggestions**. As the user iteratively builds out their dataflows, they may reach decision points where the next analytical step is unclear or open-ended. To support exploration in these moments, Urbanite allows users to request connection suggestions from the LLM. Urbanite generates these suggestions by prompting the LLM with the current state of the dataflow specification, including existing nodes, their subtasks, and their interconnections. Urbanite then returns candidate nodes (e.g., new analysis, transformation, or visualization steps), each annotated with a subtask that explains its intended logic. These proposals are presented as options rather than decisions, supporting DG2-P. Users maintain full control over which nodes to integrate.

5.4.3 Evaluation features

🔴🔵 **Dataflow- or node-level explanations**. To support transparency, Urbanite enables users to request natural language explanations for individual nodes or entire sections of the dataflow. These are generated by prompting the LLM with the relevant parts of the specification and asking it to summarize the logic, purpose, and expected behavior of that component. Users can also prompt for debugging ideas when facing actual or potential errors. These requests trigger focused prompts to the LLM, which in turn generates context-aware responses. This aligns with DG3-E, ensuring that users can inspect and verify the AI-generated behavior of the system. This is particularly useful when users are unfamiliar with certain operations or wish to validate nodes before proceeding, supporting DG5-CC.

🔴🔵🟢 **Provenance and data inspection**. Urbanite extends Curio's provenance model to capture versioned snapshots of the dataflow specification during key interactions. Every time a user accepts an LLM suggestion, modifies a task or subtask, or changes a node's content, Urbanite automatically creates a snapshot of the full specification, including nodes, edges, subtasks, and metadata. These snapshots are then organized into a history tree that allows users to trace how their dataflow has evolved over time, supporting DG4-E. Users can browse earlier versions, inspect changes, and revert to previous versions if needed. Each snapshot is timestamped and labeled with the triggering event or user prompt that led to the change. Urbanite also supports data-level inspection for each node. Based on the specification schema, it offers standard visualizations for possible outputs. Users can click a magnifying glass icon on a node to view its data, supporting DG3-E.

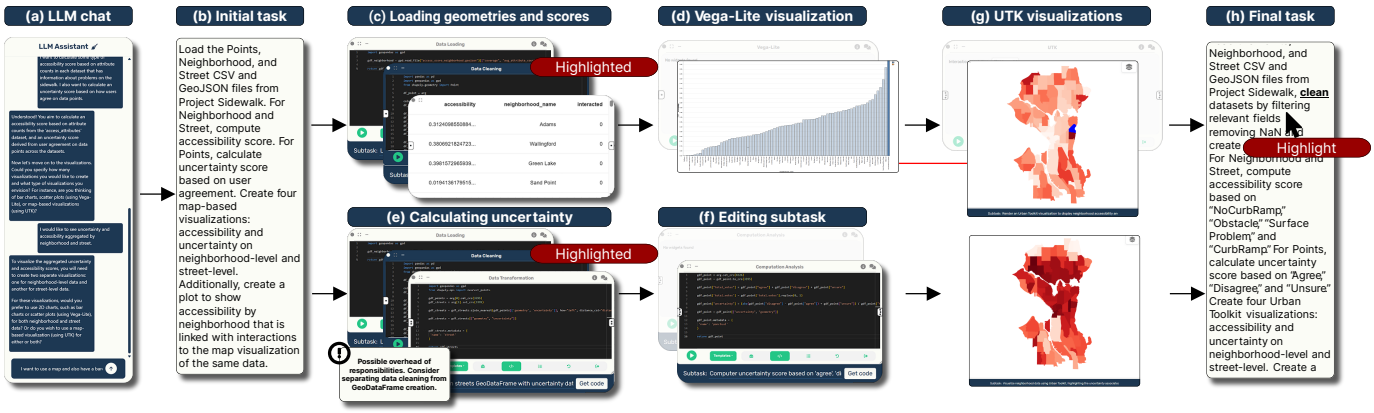

Fig. 3: Using Urbanite to analyze Project Sidewalk accessibility data. (a) The user starts with **dataflow generation** to define a task. (b) Based on the task, the LLM proposes a first sketch. (c) The user accepts nodes for geometry and score calculation, using **code generation** to fill them in. (d) The same is done for a neighborhood-level bar chart. (e) In a parallel flow, uncertainty is then calculated, and (f) the user edits the subtask to aggregate by neighborhood using **task & subtask definition**. (g) UTK suggestions are accepted to visualize accessibility (top) and uncertainty (bottom). (h) As nodes and subtasks were edited, the task was automatically refined. Hovering task keywords highlights related dataflow parts.

5.5 The Urbanite system

Urbanite builds on Curio's dataflow framework [36], adding alignment-driven features and a language-based interaction model.

5.5.1 System architecture

Urbanite comprises a Python-based backend server and sandboxed execution server, with a React frontend. Most nodes use Python, while visualization nodes rely on Vega-Lite or UTK. The backend manages dataflow logic, LLM interactions, data, provenance, and specification. Data remains in the backend in binary format, transferred internally between nodes, and moves to the frontend when a visualization node requests it. The sandbox server executes user or LLM-generated Python code in isolation. The frontend provides an interactive interface for dataflow authoring, inspection, and LLM communication.

5.5.2 LLM integration and prompting strategies

Urbanite relies on the LLM as a semantic engine for generating, modifying, and explaining parts of the dataflow. It supports multiple actions: task definition, subtask refinement, dataflow generation, debugging, and suggestions. To make these actions reliable, the backend constructs each LLM request from three components: (1) A preamble, describing Urbanite's internal logic, specification schema, UTK and Vega-Lite grammar structure, and system constraints; (2) A feature-specific prompt, tailored to the user's current action; (3) The input context, including the current dataflow specification, task and subtasks, expected input / output types, and node content. The dataflow specification will be used to constrain the LLM responses. To improve response quality, we use: (1) few-shot prompting with examples, (2) contextual priming with specs and metadata, (3) subprompt decomposition for multi-turn tasks, (4) role prompting to guide the LLM as an assistant, and (5) negative prompting to discourage vague or irrelevant outputs. The LLM can access dataset names and metadata, while persistent interaction is limited to referencing previous messages within the assistant chat. Currently, Urbanite uses OpenAI's gpt-4o-mini.

5.5.3 Frontend and interaction features

The frontend interface presents a canvas-based authoring environment, integrated with LLM-powered and manual editing controls (Figure 2).

Canvas area. The canvas lets users build dataflows by dragging and connecting nodes. LLM-suggested nodes appear semi-transparent with an "Accept Suggestion" button. The node area provides node types and file controls for creating, uploading, and exporting dataflows. A "Dashboard Mode" hides edges for presentation. Users define or edit tasks in natural language in the task editor, which highlights key terms and can be hovered to reveal their inferred connections. A "Generate Suggestions" prompts the LLM to scaffold the dataflow.

Node area. A node exposes interaction panels for specifying its purpose, input, and output types. Users can describe a subtask in natural language and prompt the LLM to generate node logic. A "Suggest Connection" button allows the LLM to propose next steps. An "Explanation Tab" provides an LLM-based summarization of the node. A magnifying glass lets users preview the output with a standard visualization.

LLM assistant chat. The right side of the interface hosts the LLM assistant chat, where the user can interact with the framework by revising tasks, clarifying suggestions, or exploring alternatives. This is the only LLM interaction that preserves chat history across turns.

Provenance controls. A version viewer lets users track dataflow provenance. LLM-triggered edits create snapshots of the specification, allowing users to inspect, compare, or revert changes (Figure 1(d)).

6 Evaluation

6.1 Usage scenarios

We present three real-world scenarios co-developed with urban experts who co-authored this paper. We use inline indicators for interactions at different scopes: 🔴 dataflow, 🔵 node, and 🟢 parameter. Each scenario uses a different entry point with LLM augmentation: Scenario 1 starts with a chat task and dataflow request; Scenario 2 with sketching a dataflow from a paper diagram; and Scenario 3 with creating nodes to load data.

6.1.1 Scenario 1: Analyzing access with Project Sidewalk

Crowdsourced platforms like Project Sidewalk [41] help fill data gaps by having volunteers annotate sidewalk features. However, such data introduces uncertainty, especially from contributor disagreements. Using Urbanite, we build a dataflow that combines multiple analytical and visualization layers, such as comparing scores across neighborhoods and inspecting confidence levels. In this scenario, the user begins by engaging with the LLM assistant in the chat-based interaction (Figure 3(a)). The user starts with a vague task: "I want to load and visualize three datasets from Project Sidewalk with data observations as CSV, neighborhoods as GeoJSON, and streets as GeoJSON." The LLM uses metadata from uploaded files to ground the conversation and follows up with clarification questions to help the user structure a task 🔴. Once satisfied (Figure 3(b)), the user applies it to the dataflow 🔴. The system proposes nodes for data loading, spatial joins, filtering, aggregation, and multi-view visualization (Figure 3(c,d,e)). The user reviews and accepts most suggestions, editing subtasks to better reflect evolving goals (Figure 3(f), 🔵). These edits automatically update the high-level task (Figure 3(h), 🔴), keeping the specification aligned. For each node, the user prompts the LLM to generate code 🔵, specifies expected input/output types 🟢, and refines as needed. During this process, the user writes a custom subtask for a new module 🔵 and receives a warning indicating that the logic could be modularized into separate operations, splitting a transformation that filters and normalizes data into two nodes (Figure 3(e), 🔵). By the end, the dataflow supports dynamic comparison of accessibility and uncertainty. Tight task-subtask-dataflow alignment led to a clearer final task description, showing how iterative refinement improved human-AI alignment (Figure 3(b,e)).

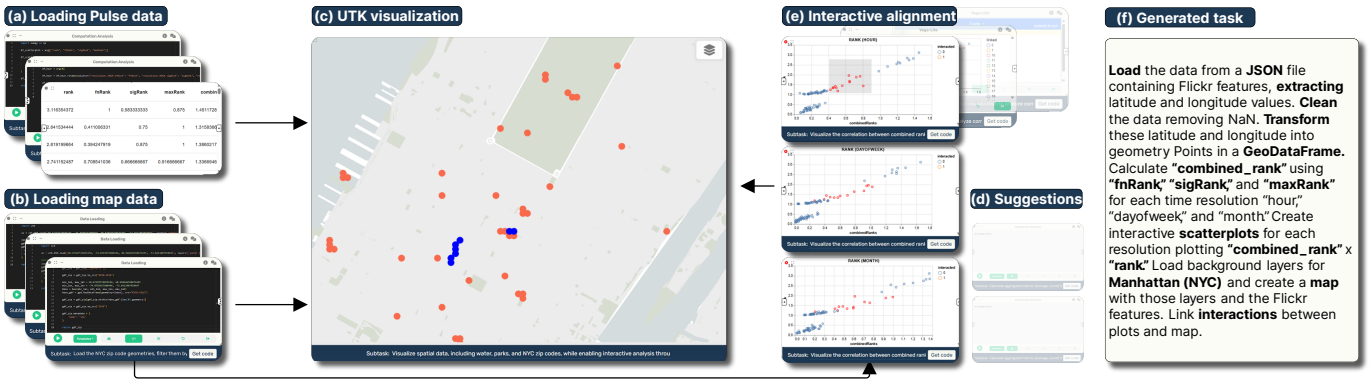

Fig. 4: Using Urbanite to reproduce Urban Pulse. (a) The user writes code to load and parse pulse data. (b) Background layers are added, and subtasks are generated using **task & subtask definition**. (c) A UTK node is added to visualize the data. (d) The user requests **connection suggestions** from the node and accepts a Vega-Lite recommendation. (e) The Vega-Lite subtask is refined to include interactive selection, and the user uses **code generation** to implement it. (f) The task is continuously updated via **task & subtask definition**, resulting in a final task.

6.1.2 Scenario 2: Topology exploration of social media data

Urban Pulse [32] is a VA framework using topological techniques to identify and compare urban activity "pulses." It models activity as a spatiotemporal scalar field, extracting peaks across time scales. The system includes coordinated views: a map of pulses, a scatter plot for comparison, and a line plot for trends. Urbanite enables users to recreate and extend such workflows by combining visual scaffolding, AI-assisted code generation, and synchronization between process-level actions and high-level task definitions. In this scenario, we illustrate how a user reproduces key elements of Urban Pulse's workflow while leveraging Urbanite's process alignment features to ensure that the resulting system remains coherent, modular, and explainable (Figure 4). The user begins by uploading the Flickr dataset used in the original study. Unlike the previous scenario, they choose not to define a task up front via the LLM assistant. Instead, they take a bottom-up approach, sketching the initial dataflow structure (Figure 4(a,b), 🔴) based on the original paper's diagrams. For simple preprocessing steps, such as loading and formatting the data, the user writes Python code into node editors and executes them (Figure 4(a,b), 🔵). As each node is run, the LLM analyzes the node content and automatically generates a subtask (Figure 4(a,b), 🔵), incrementally constructing a task description in the background (Figure 4(f), 🔴). Throughout, the user frequently uses the connection suggestion to explore next steps (Figure 4(e), 🔴). These LLM-generated prompts help expand the dataflow based on the task and dataflow (Figure 4(d), 🔴). The user accepts or edits suggestions and updates subtasks 🔵 as needed, resulting in three dataflow branches for Urban Pulse's time resolutions: hour, day, and month (Figure 4(d), 🔴). The LLM recommends visualizing each branch with a scatter plot (Figure 4(d), 🔵). After accepting this suggestion, the user modifies the subtask to include interactive selections. The final output mirrors the Urban Pulse system, supporting multi-scale activity exploration.

To build the map-based view, the user adds nodes for loading background layers (Figure 4(b), 🔵), merges them with pulse point data 🔵, and visualizes the result using a UTK node (Figure 4(c), 🔵), with parameters exposed to the user 🟢. They complete the workflow by creating interaction edges between the scatter plots and the map view 🔴, allowing selections in one view to update the others, turning what would normally be a technically demanding coordination task into a visual connection. Even though the user never explicitly defined a task in advance, the task and subtasks are continually synchronized (Figure 4(f), 🔴🔵), preserving intent and clarity across modules. This scenario demonstrates Urbanite's support for process-first workflows, where a user incrementally constructs a system and achieves strong alignment through bottom-up iteration and AI collaboration (Figure 4).

6.1.3 Scenario 3: Impacted houses in flooding simulation

Urban VA is crucial for environmental risk assessment [5, 10]. Climate experts use models to simulate real-world phenomena, and libraries like SynxFlow [60] simplify flooding simulations. However, non-experts still struggle with terminology and data, while domain experts need better ways to explore and visualize results. We show how a user can use Urbanite's dataflows, UTK visualizations, and LLM-powered features to interpret simulations and build complex visualizations.

The user starts by creating nodes to load data for a small village in the Chicago area 🔴. The data is used in a computation node (Figure 1(b), 🔵) to run a SynxFlow flooding simulation for an extreme storm event. As nodes execute, the LLM generates subtask descriptions 🔵 which are automatically incorporated into the task, maintaining the broader context (Figure 1(f), 🔴). An additional node (Figure 1(a), 🔵) fetches 3D building data from OpenStreetMap using the UTK API. To check the result, the user uses the data inspection tool (Figure 1(a), 🟢). Finally, two UTK visualizations (Figure 1(c), 🔵) are created: one providing an overview of the simulation and another highlighting flood risk for individual houses. Wanting to change key simulation parameters, the user goes back to a previous dataflow version using provenance (Figure 1(d), 🔴), updating the task description (Figure 1(f), 🔴) and consequently the subtasks of respective nodes 🔵. Based on the new subtasks, the user requests updated code for the simulation nodes 🔵 from the LLM. The provenance tree now has two branches (Figure 1(d), 🔴), allowing easy comparison of simulation configurations. To share this workflow with collaborators, the user generates node-level explanations (Figure 1(b), 🔵) for simulation nodes and a dataflow-level explanation (Figure 1(e), 🔴). Finally, the dataflow is exported, and markdown files with explanations are generated. In the end, the user creates a dataflow to simulate floods in the Chicago area, with an overview visualization and a 3D view of affected buildings. This scenario demonstrates Urbanite's ability to enhance evaluation alignment and collaboration through provenance tracking and automatic documentation (Figure 1).

6.2 Experts' feedback

To evaluate Urbanite, we conducted semi-structured interviews with six urban experts using one usage scenario, gathering feedback on each feature. None were paper authors; all had programming experience and had used LLMs. Participants included three civil engineering PhD students E1-3, an urban planning faculty member E4, a water resource engineering researcher E5, and a computer scientist specializing in urban accessibility E6. Overall, they expressed positive impressions towards Urbanite, appreciating its ease of use and accessibility. Regarding the **dataflow generation**, E1 positively remarked, "It's really amazing because usually we have to spend a long time defining these things by ourselves," highlighting that the framework "gives us an initial version we can work around." Similarly, E2 noted the novelty of the **dataflow generation** feature, emphasizing, "this doesn't exist in tools that we have." E4 appreciated the conversational nature of the dataflow generation, mentioning that "being prompted on figuring out more detailed tasks is helpful," since in her experience, "LLMs sometimes do something that has nothing to do with what you're asking."

Concerning the **task and subtask definition**, E6 found the suggestion of subtasks useful, observing, "The idea of suggesting subtasks is interesting, and the user will even learn through them by visualizing

the isolated subtasks." E4 also praised the concept, emphasizing the benefit of syncing code and task, "that's where information gets lost in a lot of projects (...) you update the code but you don't update the notes, so keeping them synced is very helpful." E2 aligned positively with the subtasks feature, as it closely matches his workflow of "breaking down a problem into distinct parts, visually plotting it, seeing what needs to be done next." E5 mentioned that this feature "simplifies the task" of creating dataflows, stressing that "from a project perspective, this immediately cleans up how we understand our dataflow."

On **code generation**, E4 underscored its practical value, appreciating "the simplicity of having LLM helping me with the task." Similarly, E3 found significant value in context integration, stating, "Usually you have your VS Code, and then you go to ChatGPT, but I appreciate that the code generation happens inside the tool and in the context of a node." E4 particularly appreciated the structured coding environment provided by the tool: "Usually I use individual scripts for coding, Urbanite structures them clearly within the same canvas." With respect to the **connection suggestions**, E1 acknowledged how this feature enables incremental refinement of workflows. E3 further highlighted the creative potential of this capability, calling it "a very interesting feature for brainstorming and exploring analytical possibilities." E4 especially valued the **explanation** and **provenance** features for supporting reproducibility and transparency: "My mind is going to things like reproducibility and open science, linking code with clear descriptions of what it does." She also highlighted the clarity: "It contextualizes the subtasks within the big dataflow," the ease of sharing: "You can easily share with others," and support for iteration: "I can go back and iterate on specific aspects easily." The urban experts also expressed certain reservations. For example, E5 mentioned that technical terms differ across countries, so that terminology might not be factored in by LLMs. Both E3 and E5 mentioned concerns regarding the short-term caching of LLM context, with E5 mentioning that "when the short-term cache ends, if the user returns later, the context might already be lost."

6.3 Quantitative evaluation

To assess Urbanite's effectiveness in translating high-level user intent into executable visual analytics workflows, we conducted an evaluation focusing on the core stages of LLM-powered generation: task semantic alignment, subtask coverage, and dataflow quality. In order to achieve this, we first selected ten representative urban VA papers. From each of these papers, we manually extracted the core user intent that the corresponding VA system aimed to support. Using Urbanite, we then: (1) generated a task description based on this user intent, (2) decomposed it into subtasks, and (3) matched these subtasks to a dataflow graph.

Methodology. We recruited five experienced Computer Science researchers as evaluators; none are authors of this paper: 3 faculty members and 2 PhD students. They were instructed to assess the generated outputs and how they translated intent into a dataflow. Each paper was assigned to two evaluators, who scored three aspects on a 3-point ordinal scale, where higher scores indicate better quality: **Semantic alignment** measured how well the generated task description reflected the original user intent, ranging from "Misaligned" (0) to "Fully aligned" (2). **Coverage of subtasks** assessed whether the subtask decomposition included all essential steps for task completion, from "Incomplete" (0) to "Complete" (2). **Flow quality** evaluated whether the dataflow's structure supported task execution, ranging from "Invalid flow" (0) to "Functional and coherent" (2). Additionally, evaluators could flag outputs for hallucination if fabricated elements were present.

Results & takeaways. Urbanite achieved strong alignment with user intent across the evaluated cases, with an average **semantic alignment score of 1.65 (SD=0.32)** on the 0–2 scale. In most cases, the generated task descriptions captured the essence of the original intent effectively. For **subtask coverage**, Urbanite achieved an average score of **1.6 (SD=0.37)**, suggesting that the system was generally able to decompose user intent into comprehensive subtasks, although a few cases exhibited partial coverage where additional subtasks would enhance completeness. The **flow quality** evaluation had an average of **1.5 (SD=0.45)**, indicating that the generated dataflows were typically functional and coherent, but with some instances requiring refinement. Across all evaluations, we identified three noteworthy mismatches between the extracted tasks and the original user intent: For Ferreira et al. [18], the extracted task incorrectly inferred that the analysis was conducted using hourly or daily patterns, which was not specified in the original intent. For Ferreira et al. [17], the extracted task incorrectly assumed that sky exposure data was available through an open data portal and omitted a necessary computation step. Finally, for Konev et al. [22], the extracted task failed to capture a critical aspect of the user intent: the need to run and control simulations. Additionally, the generated dataflow was overly vague and lacked sufficient detail to support execution. Full results are in the supplementary material.

7 Conclusions

Reflection on design goals. The design goals of Urbanite were defined to bridge the gap between human intent and AI behavior across all stages of an urban VA dataflow. Regarding DG1-S, by allowing users to articulate analytical intent in natural language, Urbanite lowers the entry barrier to the creation of these dataflows. Urbanite does not require users to translate ideas into technical components from the outset. Instead, the framework shifts the burden of interpretation to the AI. Despite this automation, the framework still gives users control over the authoring process, as highlighted by DG2-P; Urbanite offers suggestions rather than decisions, so all AI-generated content is inspectable and editable. As highlighted in our experts' feedback, transparency is key in fostering trust in AI-assisted workflows. Urbanite directly addresses DG3-E; whether through explanations or data inspections, users are given visibility into what the dataflow is doing. To support trust, transparency, and reproducibility, Urbanite incorporates provenance of dataflow specifications, which aligns with DG4-E. By tracing both data transformations and human-AI interactions over time, the framework allows users to compare past states, revert changes, and understand the evolution of their dataflows. As noted in DG5-CC, Urbanite supports rephrasing tasks and revising AI-generated dataflows, enabling alignment as an iterative dialogue where human and AI gradually converge. For alignment stages in DG6-CC, Urbanite supports multi-level interaction by keeping scopes consistent. Users can start with a broad task, refine subtasks, switch between dataflow and node logic, and explore results at different levels. Changes in one scope automatically update others to ensure consistency. While our evaluation revealed that Urbanite occasionally made mistakes (e.g., inferring unavailable datasets, omitting specific steps, overly broad flows), these instances were limited. These results suggest that Urbanite can effectively translate high-level user intent into executable workflows, providing a strong starting point for urban VA. Importantly, such imperfections highlight the value of *Urbanite's in-the-loop, iterative refinement process*, where users can easily identify and adjust AI-generated outputs. Rather than seeking a perfect one-shot generation, Urbanite is designed to facilitate progressive alignment between human intent and system behavior.

Limitations. While Urbanite presents a novel approach to urban VA, we acknowledge a few limitations. The visualizations generated are primarily constrained by the capabilities of Vega-Lite and UTK. This can make it challenging to create more complex, custom-layered visualizations that require very specific visual encodings. Additionally, Urbanite's performance is primarily constrained by the underlying libraries utilized within each dataflow node. We also note that the framework does not yet support the automatic positioning or reorganization of dataflow nodes through LLM interaction, which could pose a visual burden for very comprehensive dataflows.

Future work. Building upon Urbanite's foundational capabilities, we envision several avenues for future work. We are particularly interested in exploring intelligent dataflow layout, alongside a detailed investigation into the cognitive burden of dataflow interaction, especially for complex analytical workflows. Another compelling research avenue lies in a comparative study of different interaction modalities, including dataflow-centric approaches, notebook-based systems, grammar-based interfaces, and our LLM-driven natural language interaction. In addition, investigating these avenues with a larger cohort of users would be valuable for expanding upon our findings and ensuring broader applicability across different expertise levels.

Acknowledgments

We thank the reviewers for their constructive feedback. This work was supported by the U.S. National Science Foundation (Awards #2320261, #2330565, #2411223) and conducted as part of the Open-Source Cyberinfrastructure for Urban Computing (OSCUR) project.

References

[1] D. Akbaba, D. Lange, M. Correll, A. Lex, and M. Meyer. Troubling collaboration: Matters of care for visualization design study. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, article no. 812, pp. 1–15, 2023. 1, 2

[2] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, et al. Guidelines for human-AI interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2019. 3

[3] G. Andrienko, N. Andrienko, W. Chen, R. Maciejewski, and Y. Zhao. Visual analytics of mobility and transportation: State of the art and further research directions. *IEEE Transactions on Intelligent Transportation Systems*, 18(8):2232–2249, 2017. 2

[4] R. C. Basole and T. Major. Generative AI for visualization: Opportunities and challenges. *IEEE Computer Graphics and Applications*, 44(2):55–64, 2024. 2

[5] S. Boorboor, Y. Kim, P. Hu, J. M. Moses, B. A. Colle, and A. E. Kaufman. Submerse: Visualizing storm surge flooding simulations in immersive display ecologies. *IEEE Transactions on Visualization and Computer Graphics*, 30(9):6365–6377, 2024. 8

[6] W. Büschel, A. Lehmann, and R. Dachselt. MIRIA: A mixed reality toolkit for the in-situ visualization and analysis of spatio-temporal interaction data. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, article no. 470, pp. 1–15, 2021. 3

[7] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. VisTrails: Visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pp. 745–747, 2006. 3

[8] J. Chen, Q. Huang, C. Wang, and C. Li. SenseMap: Urban performance visualization and analytics via semantic textual similarity. *IEEE Transactions on Visualization and Computer Graphics*, 30(9):6275–6290, 2024. 3

[9] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 2

[10] C. V. F. de Souza, S. M. Bonnet, D. de Oliveira, M. Cataldi, F. Miranda, and M. Lage. ProWis: A visual approach for building, managing, and analyzing weather simulation ensembles at runtime. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):738–747, 2024. 8

[11] Z. Deng, D. Weng, J. Chen, R. Liu, Z. Wang, J. Bao, Y. Zheng, and Y. Wu. AirVis: Visual analytics of air pollution propagation. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):800–810, 2019. 3

[12] Z. Deng, D. Weng, S. Liu, Y. Tian, M. Xu, and Y. Wu. A survey of urban visual analytics: Advances and future directions. *Computational Visual Media*, 9(1):3–39, 2023. 1, 2

[13] J. Dong, H. Zhang, M. Cui, Y. Lin, H.-Y. Wu, and C. Bi. TCEVis: Visual analytics of traffic congestion influencing factors based on explainable machine learning. *Visual Informatics*, 8(1):56–66, 2024. 3

[14] N. Elmqvist, J. Stasko, and P. Tsigas. DataMeadow: A visual canvas for analysis of large-scale multivariate data. In *2007 IEEE Symposium on Visual Analytics Science and Technology*, pp. 187–194. IEEE, 2007. 3

[15] Z. Feng, H. Qu, S.-H. Yang, Y. Ding, and J. Song. A survey of visual analytics in urban area. *Expert Systems*, 39(9):e13065, 2022. 1, 2

[16] L. Ferreira, G. Moreira, M. Hosseini, M. Lage, N. Ferreira, and F. Miranda. Assessing the landscape of toolkits, frameworks, and authoring tools for urban visual analytics systems. *Computers & Graphics*, 123:104013, 2024. 1, 2, 3

[17] N. Ferreira, M. Lage, H. Doraiswamy, H. Vo, L. Wilson, H. Werner, M. Park, and C. Silva. Urbane: A 3D framework to support data driven decision making in urban development. In *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 97–104. IEEE, 2015. 9

[18] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva. Visual exploration of big spatio-temporal urban data: A study of New York City taxi trips. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2149–2158, 2013. 9

[19] G. Garcia, J. Silveira, J. Poco, A. Paiva, M. B. Nery, C. T. Silva, S. Adorno, and L. G. Nonato. CrimAnalyzer: Understanding crime patterns in São Paulo. *IEEE Transactions on Visualization and Computer Graphics*, 27(4):2313–2328, 2019. 1, 3

[20] W. Javed and N. Elmqvist. ExPlates: Spatializing interactive analysis to scaffold visual exploration. *Computer Graphics Forum*, 32(3pt4):441–450, 2013. 3

[21] S. Jänicke, P. Kaur, P. Kuzmicki, and J. Schmidt. Participatory visualization design as an approach to minimize the gap between research and application. In C. Gillmann, M. Krone, G. Reina, and T. Wischgoll, eds., *VisGap - The Gap between Visualization Research and Visualization Software*. The Eurographics Association, 2020. 1

[22] A. Konev, J. Waser, B. Sadransky, D. Cornel, R. A. Perdigão, Z. Horváth, and M. E. Gröller. Run Watchers: Automatic simulation-based decision support in flood management. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1873–1882, 2014. 9

[23] S. Lau, I. Drosos, J. M. Markel, and P. J. Guo. The design space of computational notebooks: An analysis of 60 systems in academia and industry. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 1–11. IEEE, 2020. 2

[24] J. Li, S. Chen, K. Zhang, G. Andrienko, and N. Andrienko. COPE: Interactive exploration of co-occurrence patterns in spatial time series. *IEEE Transactions on Visualization and Computer Graphics*, 25(8):2554–2567, 2019. 3

[25] S. L'Yi, A. van den Brandt, E. Adams, H. N. Nguyen, and N. Gehlenborg. Learnable and expressive visualization authoring through blended interfaces. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):459–469, 2025. 2, 3

[26] S. L'Yi, Q. Wang, F. Lekschas, and N. Gehlenborg. Gosling: A grammar-based toolkit for scalable and interactive genomics data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):140–150, 2022. 2

[27] Y. Lyu, H. Lu, M. K. Lee, G. Schmitt, and B. Y. Lim. IF-City: Intelligible fair city planning to measure, explain and mitigate inequality. *IEEE Transactions on Visualization and Computer Graphics*, 30(7):3749–3766, 2024. 1

[28] S. McKenna, D. Mazur, J. Agutter, and M. Meyer. Design activity framework for visualization design. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2191–2200, 2014. 3

[29] A. M. McNutt. No grammar to rule them all: A survey of JSON-style DSLs for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):160–170, 2023. 2

[30] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2025. 2

[31] F. Miranda, H. Doraiswamy, M. Lage, L. Wilson, M. Hsieh, and C. T. Silva. Shadow Accrual Maps: Efficient accumulation of city-scale shadows over time. *IEEE Transactions on Visualization and Computer Graphics*, 25(3):1559–1574, 2018. 1

[32] F. Miranda, H. Doraiswamy, M. Lage, K. Zhao, B. Gonçalves, L. Wilson, M. Hsieh, and C. T. Silva. Urban Pulse: Capturing the rhythm of cities. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):791–800, 2016. 3, 8

[33] F. Miranda, M. Hosseini, M. Lage, H. Doraiswamy, G. Dove, and C. T. Silva. Urban Mosaic: Visual exploration of streetscapes using large-scale image data. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2020. 3

[34] F. Miranda, T. Ortner, G. Moreira, M. Hosseini, M. Vuckovic, F. Biljecki, C. T. Silva, M. Lage, and N. Ferreira. The state of the art in visual analytics for 3D urban data. *Computer Graphics Forum*, 43(3):e15112, 2024. 1, 2, 3

[35] G. Moreira, M. Hosseini, M. N. A. Nipu, M. Lage, N. Ferreira, and F. Miranda. The Urban Toolkit: A grammar-based framework for urban visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):1402–1412, 2024. 2, 3, 4, 6

[36] G. Moreira, M. Hosseini, C. Veiga, L. Alexandre, N. Colaninno, D. de Oliveira, N. Ferreira, M. Lage, and F. Miranda. Curio: A dataflow-based framework for collaborative urban visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1224–1234, 2025. 2, 3, 4, 5, 7

[37] T. Munzner. *Visualization analysis and design*. CRC press, 2014. 2

[38] J. Nielsen. AI: First new UI paradigm in 60 years. https://www.nngroup.com/articles/ai-paradigm/, 2023. Accessed: 2025-03-

31. 3

[39] R. Qiu, Y. Tu, Y.-S. Wang, P.-Y. Yen, and H.-W. Shen. DocFlow: A visual analytics system for question-based document retrieval and categorization. *IEEE Transactions on Visualization and Computer Graphics*, 30(2):1533–1548, 2024. 3

[40] S. Sah, R. Mitra, A. Narechania, A. Endert, J. Stasko, and W. Dou. Generating analytic specifications for data visualization from natural language queries using large language models. *arXiv preprint arXiv:2408.13391*, 2024. 3

[41] M. Saha, M. Saugstad, H. T. Maddali, A. Zeng, R. Holland, S. Bower, A. Dash, S. Chen, A. Li, K. Hara, et al. Project Sidewalk: A web-based crowdsourcing tool for collecting sidewalk accessibility data at scale. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2019. 7

[42] K. Salinas, T. Gonçalves, V. Barella, T. Vieira, and L. G. Nonato. CityHub: A library for urban data integration. In *2022 35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, vol. 1, pp. 43–48, 2022. 3

[43] L. Shen, E. Shen, Y. Luo, X. Yang, X. Hu, X. Zhang, Z. Tai, and J. Wang. Towards natural language interfaces for data visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 29(6):3121–3144, 2023. 2

[44] R. Sicat, J. Li, J. Choi, M. Cordeil, W.-K. Jeong, B. Bach, and H. Pfister. DXR: A toolkit for building immersive data visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):715–725, 2019. 2

[45] C. T. Silva, E. Anderson, E. Santos, and J. Freire. Using VisTrails and provenance for teaching scientific visualization. *Computer Graphics Forum*, 30(1):75–84, 2011. 3

[46] C. T. Silva, J. Freire, and S. P. Callahan. Provenance for visualizations: Reproducibility and beyond. *Computing in Science & Engineering*, 9(5):82–89, 2007. 2

[47] V. Silva, D. de Oliveira, P. Valduriez, and M. Mattoso. Dfanalyzer: Runtime dataflow analysis of scientific applications using provenance. *Proceedings of the VLDB Endowment*, 11(12):2082–2085, 2018. 3

[48] T. Spinner, U. Schlegel, H. Schäfer, and M. El-Assady. explAIner: A visual analytics framework for interactive and explainable machine learning. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1064–1074, 2020. 2

[49] M. Steyvers, H. Tejeda, A. Kumar, C. Belem, S. Karny, X. Hu, L. W. Mayer, and P. Smyth. What large language models know and what people think they know. *Nature Machine Intelligence*, pp. 1–11, 2025. 2

[50] Z. Sun, D. Xu, Y. Zhang, Y. Qi, Y. Wang, Z. Zuo, Z. Wang, Y. Li, X. Li, Q. Lu, W. Peng, and S. Guo. BigDataflow: A distributed interprocedural dataflow analysis framework. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, p. 1431–1443, 2023. 3

[51] X. Tang, E. Krokos, C. Liu, K. Davidson, K. Whitley, N. Ramakrishnan, and C. North. Steering LLM summarization with visual workspaces for sensemaking. *arXiv preprint arXiv:2409.17289*, 2024. 3

[52] M. Terry, C. Kulkarni, M. Wattenberg, L. Dixon, and M. R. Morris. Interactive AI alignment: Specification, process, and evaluation alignment. *arXiv preprint arXiv:2311.00710*, 2023. 2, 3, 4

[53] Y. Tian, W. Cui, D. Deng, X. Yi, Y. Yang, H. Zhang, and Y. Wu. ChartGPT: Leveraging LLMs to generate charts from abstract natural language. *IEEE Transactions on Visualization and Computer Graphics*, 31(3):1731–1745, 2025. 2

[54] P. Ulbrich, M. Waldner, K. Furmanová, S. M. Marques, D. Bednář, B. Kozlíková, and J. Byška. sMolBoxes: Dataflow model for molecular dynamics exploration. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):581–590, 2022. 3

[55] R. Walker, A. Slingsby, J. Dykes, K. Xu, J. Wood, P. H. Nguyen, D. Stephens, B. W. Wong, and Y. Zheng. An extensible framework for provenance in human terrain visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2139–2148, 2013. 2

[56] S. Wang, F. Lyu, S. Wang, C. E. Catlett, A. Padmanabhan, and K. Soltani. Integrating CyberGIS and urban sensing for reproducible streaming analytics. *Urban Informatics*, pp. 663–681, 2021. 3

[57] J. Waser, H. Ribicic, R. Fuchs, C. Hirsch, B. Schindler, G. Bloschl, and E. Groller. Nodes on Ropes: A comprehensive data and control flow for steering ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1872–1881, 2011. 3

[58] A. Wu, D. Deng, M. Chen, S. Liu, D. Keim, R. Maciejewski, S. Miksch, H. Strobelt, F. Viégas, and M. Wattenberg. Grand challenges in visual analytics applications. *IEEE Computer Graphics and Applications*, 43(5):83–90, 2023. 1, 2, 3

[59] A. Wu, D. Deng, F. Cheng, Y. Wu, S. Liu, and H. Qu. In defence of visual analytics systems: Replies to critics. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1026–1036, 2023. 1

[60] X. Xia and X. Ming. SynxFlow: Synergising high-performance hazard simulation with data flow, 2025. 8

[61] T. Xie, Y. Ma, J. Kang, H. Tong, and R. Maciejewski. FairRankVis: A visual analytics framework for exploring algorithmic fairness in graph mining models. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):368–377, 2022. 3

[62] Q. Yang, A. Steinfeld, C. Rosé, and J. Zimmerman. Re-examining whether, why, and how human-AI interaction is uniquely difficult to design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2020. 3

[63] W. Yap, P. Janssen, and F. Biljecki. Free and open source urbanism: Software for urban planning practice. *Computers, Environment and Urban Systems*, 96:101825, 2022. 3

[64] B. Yu and C. T. Silva. VisFlow-web-based visualization framework for tabular data with a subset flow model. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):251–260, 2016. 3

[65] B. Yu and C. T. Silva. FlowSense: A natural language interface for visual data exploration within a dataflow system. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1–11, 2020. 3

[66] Y. Zhao, Y. Zhang, Y. Zhang, X. Zhao, J. Wang, Z. Shao, C. Turkay, and S. Chen. LEVA: Using large language models to enhance visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 2, 3

[67] Y. Zheng, L. Capra, O. Wolfson, and H. Yang. Urban computing: Concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology*, 5(3):1–55, 2014. 2

[68] Y. Zheng, W. Wu, Y. Chen, H. Qu, and L. M. Ni. Visual analytics in urban computing: An overview. *IEEE Transactions on Big Data*, 2(3):276–296, 2016. 1, 2

[69] P. Ziegler and S. E. Chasins. A need-finding study with users of geospatial data. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, article no. 862, pp. 1–16, 2023. 3