10.48047/jocaaa.2026.35.05.17

Unified Interaction Models for Complex Analytical Workflows

Sonali Priya
LB Networks, LLC, USA

Abstract

Enterprise analytics platforms serve many functions, from operational monitoring and compliance reporting to investigative analysis and high-stakes decision-making. As these platforms expand through successive development cycles and distributed teams, the interaction layer that connects users to data becomes increasingly fragmented. Filtering behaviors, drilldown semantics, state management rules, and workflow sequencing vary across modules, imposing cognitive friction that degrades analytical accuracy, erodes user trust, and lengthens task completion times. This paper proposes unified interaction models as a system-level architectural discipline for addressing interaction fragmentation in large-scale enterprise analytics ecosystems. The proposed discipline comprises four interlocking components: an interaction grammar that defines a vocabulary of analytical primitives and their behavioral rules; a workflow state architecture that governs context persistence and traceability across navigational boundaries; reusable workflow patterns that encode the sequential logic of common analytical tasks; and a governance framework that ensures behavioral consistency across feature builds and product cycles. Drawing on empirical research in usability engineering, cognitive load theory, spatial cognition, and structured interaction pattern analysis, this paper argues that treating interaction as platform infrastructure, rather than a screen-level design concern, yields a principled and scalable architecture for elevating analytical fidelity, reducing operational friction, and enabling trustworthy, reproducible decision-making across the enterprise.

Keywords: *Interaction Grammar, Workflow State Architecture, Enterprise Analytics Design, Reusable Workflow Patterns, Interaction Governance*

1. Introduction

Enterprise analytics software touches nearly every part of how organizations operate. Teams use it to track live processes, measure performance against targets, produce regulatory reports, and investigate patterns that shape consequential decisions. The people who rely on these platforms are just as varied as the tasks themselves: analysts building and interpreting models, administrators managing data pipelines, operations staff watching real-time dashboards, and senior leaders drawing on summarized outputs to guide planning. Despite their different needs, all of these users depend on the same thing -- an interaction layer that connects them to data reliably, responds promptly, and behaves consistently across the platform [1].

As enterprise analytics platforms grow in scope, the reliability of that interaction layer becomes a determining factor in the quality of analytical outcomes. Platforms rarely emerge as monolithic designs. Instead, they evolve through successive product cycles in which new teams build new modules, each addressing a specific analytical domain or user need. This distributed development model, while organizationally efficient, produces a cumulative liability: without a shared specification for how interactions should behave, each team implements filtering, drilldown navigation, and state management according to its own local conventions. Over time, the result is a platform whose visual surface may appear cohesive, thanks to shared component libraries and design tokens, but whose behavioral layer is fragmented in ways that are invisible at the component level yet deeply disruptive at the workflow level [2].

10.48047/jocaaa.2026.35.05.17

The consequences of this behavioral fragmentation are well documented in usability research. Jakob Nielsen's heuristic of consistency and standards holds that users should not have to wonder whether different words, situations, or actions mean the same thing across different parts of a system [1]. When that heuristic is violated, the effects cascade through several of the core usability objectives identified in the literature: learnability suffers because users cannot transfer expectations from one module to the next; memorability degrades because the behavioral rules users must internalize multiply with each inconsistently designed module; and error rates climb because users apply learned behaviors in contexts where those behaviors produce different, unpredictable results [2]. In complex analytics environments, where people are working through multi-step investigations under real-time pressure, these kinds of usability breakdowns do not just frustrate users—they erode the accuracy of the analysis itself and slow down the decisions that depend on it.

This paper's central argument is that visual standardization alone is not enough to achieve genuine interaction consistency across an analytics platform. Interaction behavior needs to be treated as a system-level architectural concern -- specified, versioned, and governed with the same discipline that organizations already apply to their data pipelines and API contracts. To this end, this paper proposes a unified interaction model comprising four interlocking components: an interaction grammar that defines analytical primitives and their behavioral rules; a workflow state architecture that governs context persistence across navigational boundaries; reusable workflow patterns that encode the sequential logic of common analytical tasks; and a governance framework that ensures behavioral consistency across distributed development teams and product cycles.

The paper proceeds as follows. Section 2 examines the problem of interaction fragmentation and its consequences for analytical workflows. Section 3 introduces the unified interaction model and its core concept, the interaction grammar. Sections 4 and 5 develop the remaining components: workflow state architecture and reusable workflow patterns with their governance mechanisms. Section 6 discusses practical implications and limitations, and Section 7 concludes.

2. Interaction Fragmentation: The Problem of Decoupling

Design systems have proven remarkably effective at standardizing the visual language of enterprise platforms. Shared component libraries, typographic scales, spacing systems, and color tokens ensure that buttons look like buttons, cards look like cards, and data tables maintain a consistent surface appearance regardless of which team built them. However, visual consistency and behavioral consistency are distinct properties, and the gap between them is particularly consequential in complex analytical contexts. Two modules may share identical component implementations yet diverge entirely in how they handle filter scoping, drilldown propagation, time range inheritance, and state resetting. These behavioral differences are invisible at the component level but become acutely disruptive at the workflow level, where users must carry forward analytical context across multiple screens and modules [3].

The ramifications of behavioral divergence are amplified in time-critical, multi-step analytical workflows. Consider an analyst investigating an anomaly in operational data who must move between a monitoring dashboard, a detailed drill-down view, and a comparative analysis panel. If each of these modules implements different rules for how filters propagate, how drilldown paths are preserved, and how state resets on navigation, the analyst must dedicate cognitive resources not to interpreting data but to reorienting to each module's interaction logic. Research in spatial cognition has established that users develop implicit conventions based on mappings between position and meaning, direction and sequence, and layout and

10.48047/jocaaa.2026.35.05.17

value. Pitt and Casasanto demonstrated that even the lateral position of interface elements produces systematic, measurable bias in user behavior, showing that incidental spatial associations influence selection patterns by as much as 15 percentage points [4]. When interaction architectures violate these implicit expectations inconsistently across modules, the cognitive cost of reorientation compounds with each transition.

Empirical research on interface consistency across devices corroborates this concern. Li et al. studied 40 responsive websites using eye tracking and questionnaire surveys to understand which design factors matter most for interaction consistency. They found that layout consistency, stable navigation patterns, and predictable interactive behavior were the strongest drivers of user confidence and task efficiency -- and that inconsistency in any one of these areas led to measurable drops in both satisfaction and task completion [5]. Their study looked at cross-device consistency, but the principle applies just as directly to cross-module consistency within enterprise analytics platforms: users build expectations about how interactions will work, and when those expectations are violated, the cognitive costs add up across every step of an analytical workflow.

A further dimension of the fragmentation problem concerns how users develop analytical competence over time. The intuition that repeated exposure to a platform will naturally produce proficiency assumes that the platform's interaction behaviors are consistent enough to permit generalization. When they are not, users develop localized proficiency, learning how a specific module behaves without acquiring a transferable understanding of how the platform as a whole operates. Gabry et al., writing about visualization in iterative Bayesian workflows, observed that vague specifications produce outputs that are, in their words, entirely implausible for the application domain [6]. The analogy to interaction architecture is direct: vagueness in behavioral specification produces variable platform behavior that users can neither predict nor control, which in turn prevents the formation of reliable analytical intuition at scale.

Table 1. Dimensions of Interaction, Fragmentation and Their Analytical Consequences

Fragmentation Dimension	**Behavioral Symptom**	**Usability Heuristic Violated**	**Analytical Consequence**
Filter Scoping	Filters apply globally in one module, locally in another	Consistency and Standards [1]	Misattribution of filtered results; silent data exclusion
Drilldown Semantics	Drilldown resets context in some views but preserves it in others	User Control and Freedom [1]	Broken investigative continuity; repeated context reconstruction
Time Range Inheritance	Selected time range does not carry across module transitions	Visibility of System Status [1]	Cross-temporal comparison errors; invalid trend analysis
State Resetting	Navigation events trigger unpredictable default-state resets	Recognition Rather Than Recall [1]	Cognitive burden of state reconstruction; workflow abandonment
Comparison Logic	Baseline selection and comparison panel layout differ by module	Consistency and Standards [1]	Incorrect baseline assumptions;

10.48047/jocaaa.2026.35.05.17

			unreliable comparative analysis

Table 1 summarizes the principal dimensions of interaction fragmentation, their behavioral symptoms, the usability heuristics they violate, and their downstream consequences for analytical work. What emerges from this mapping is that fragmentation is not a single defect but a syndrome: a pattern of distributed behavioral inconsistencies that, taken together, undermine the reliability of the analytical process across the platform.

3. Defining the Unified Interaction Model

A unified interaction model is a system-level specification that governs how users carry out analytical actions across an entire enterprise analytics platform. The distinction from a design system matters here: a design system defines how interface components look and fit together, but a unified interaction model defines how those components behave in practice. It spells out what happens when someone applies a filter, drills into a data hierarchy, selects a comparison baseline, or moves between modules -- and it ensures those behaviors are consistent, predictable, and documented across the platform.

At the foundation of this model is what we call an interaction grammar: a structured vocabulary of analytical primitives, paired with rules that govern how they behave across different platform contexts. The concept of an interaction grammar draws on a methodological parallel from the work of Diehl et al., who developed a hierarchical framework of user interface design recommendations through an iterative, multi-step expert validation process. Their framework distilled an initial set of 244 design recommendations into a validated set of 174, organized across two hierarchical levels: 69 generic recommendations applicable across all contexts and 105 domain-specific recommendations tailored to particular interaction scenarios [7]. The interaction grammar proposed here adopts an analogous hierarchical structure: a generic layer of platform-wide behavioral rules and a specific layer of module-scoped behavioral refinements.

The primitives governed by this grammar encompass the core interaction types found in analytical workflows: filtering and scoping, which control which data subsets are visible and active; drilldown and hierarchical navigation, which enable movement between levels of aggregation; comparison and baseline selection, which support side-by-side evaluation of data segments; sorting and grouping, which organize data along chosen dimensions; dimensional pivoting, which restructures analytical views around alternative axes; annotation, which allows users to attach interpretive context to data points; result export, which enables extraction of analytical outputs; and state preservation, which maintains the configuration of an analytical session across navigational events. For each primitive, the grammar specifies its triggering condition, its scope of effect (distinguishing between global scope, which applies across the platform, and local scope, which applies only within the current view), its interaction rules with other concurrently active primitives, and its conflict resolution protocol when multiple primitives compete for the same analytical context.

This level of behavioral specification transforms interaction from an implicit social contract between designers and users into an explicit, verifiable system contract. The practical significance of such formalization is supported by recent research on AI-assisted design workflows. Zhu et al. demonstrated that structured AI-assisted workflows reduced total operation time by 48.6% when designers used AI tools to complete design briefs, while simultaneously improving average task scores across multiple evaluation dimensions, including market analysis, economic feasibility, and budgeting [8]. While their study addressed design brief creation rather than interaction specification, the underlying mechanism is analogous:

10.48047/jocaaa.2026.35.05.17

structured, well-defined interaction models, whether governed by human architectural discipline or augmented by computational assistance, reduce the cognitive demands of complex tasks and improve the quality and consistency of outcomes.

The interaction grammar also supports role-based differentiation without sacrificing consistency. A novice user executing a constrained workflow template is governed by the same underlying grammar as an expert analyst conducting a multi-step investigative workflow; the difference lies only in the subset of primitives accessible to each role. This design principle was corroborated by Diehl et al.'s finding that 91 of 105 domain-specific recommendations in their framework were judged by experts to be mandatory, suggesting that behavioral specifications at all levels of complexity can and should be formalized and that such formalization is feasible across a wide range of user profiles [7].

Table 2. Interaction Grammar Layer Structure and Recommendation Distribution

Grammar Layer	Scope	Recommendation Count (Diehl et al.)	Mandatory Rate	Interaction Grammar Application
Generic	Platform-wide	69 [7]	High	Universal primitives: filter scope, drilldown behavior, state persistence
Domain-Specific	Module-scoped	105 [7]	86.7% (91/105) [7]	Module-specific: comparison baselines, alert thresholds, export formats
Role-Based Overlay	User-scoped	N/A (derived)	Inherits from above	Constrained primitive access per role: novice templates, expert full access

Table 2 maps the interaction grammar's hierarchical structure onto the empirical recommendation distribution from Diehl et al. [7], illustrating how the generic and domain-specific layers correspond to validated categories of design specification. The role-based overlay extends this hierarchy to support differentiated access without introducing behavioral inconsistency.

4. Workflow State Architecture

Among the many dimensions of enterprise analytics interaction, state management is perhaps the most consequential yet least documented. Workflow state encompasses all aspects of analytical context that a user configures during a session: active filters, selected time frames, entity scopes, view modes, drilldown path histories, and in-progress comparative configurations. In the absence of a platform-level state model, this context is inherently ephemeral. It can be lost through module transitions, overwritten by default behaviors, or silently invalidated by navigation events that the platform does not document or the user does not anticipate.

The cognitive effects of state loss are structural rather than incidental. Research on interface navigation for users with limited literacy and cognitive processing capacity has demonstrated that poor mental spatial organization and low processing speed significantly increase trial-and-error navigation in hierarchical

10.48047/jocaaa.2026.35.05.17

interfaces. Guimaraes et al. found that as menu depth increased, users became progressively more disoriented, and navigation errors compounded rather than diminished with experience [9]. The parallel to enterprise analytics is direct: when an investigative workflow breaks state context at a module boundary, the analyst is forced into trial-and-error reconstruction of their previous analytical configuration, which raises error rates and degrades the timeliness and accuracy of the analytical output in operational settings.

The workflow state architecture proposed here addresses this problem by defining explicit rules for state management at the platform level. Its first structural distinction is between global state and local state. Global state is cross-module and persistent: it survives navigation events and is accessible from any analytical context within the platform. Examples include the active entity scope (which organization, account, or case is under investigation), the selected time range, and any active compliance or audit filters. Local state is view-scoped and ephemeral: it applies only to the current analytical context and is expected to reset when the user navigates away. Examples include sort order within a data table, the expansion state of a hierarchical tree, and in-progress annotation drafts.

This distinction, while conceptually straightforward, resolves a significant class of usability failures. When a user applies a time range filter in a monitoring dashboard and then navigates to a detailed investigation module, the state architecture determines whether that time range persists (global) or resets to a default (local). Without an explicit architectural decision, this behavior is left to the implementing team's discretion, which is precisely how fragmentation originates. The state architecture removes that discretion by making the persistence rules explicit, documented, and testable.

Beyond persistence rules, the architecture specifies three additional capabilities. First, drill path history: the platform maintains a visible record of the navigational path the user has followed through hierarchical data, enabling both backtracking and contextual awareness of how the current view relates to higher-level aggregations. Second, state repository: analytical configurations can be saved, named, and retrieved, enabling users to resume interrupted investigations without reconstructing their context from scratch. Third, state conflict resolution: when someone moves between modules that have different active filter contexts, the architecture provides clear rules for reconciling those conflicts instead of quietly falling back to one module's defaults.

These capabilities are directly linked to the principle of visibility of system status, identified by Guimaraes et al. as one of 12 core interaction principles organized across four categories: Design, Language and Information, Navigation, and Error Prevention [9]. Users must be able to see their current analytical context, including active filters, drill path history, and state provenance, at all times. Without this visibility, users have no way to tell whether what they are seeing reflects real patterns in the data or is simply an artifact of a state conflict they do not know about.

The necessity of formal derivation and empirical validation for state management rules is reinforced by research on universal design principles. Gao et al. conducted a quantitative three-phase study on universal design principles for architecturally innovative products, demonstrating that their systematically derived measurement scale achieved a Cronbach's alpha reliability coefficient of 0.960, substantially exceeding the 0.7 threshold for measurement excellence [10]. Their conclusion that systematically derived and validated frameworks produce superior results to design-by-convention approaches applies directly to the state architecture of analytics platforms. Formal state management rules, validated against actual user workflows, will outperform ad hoc conventions precisely because they are explicit, testable, and revisable in response to empirical evidence.

The state architecture also supports compliance and auditability requirements that are increasingly important in regulated industries. In financial services, healthcare, and law enforcement analytics, the

10.48047/jocaaa.2026.35.05.17

ability to reproduce the exact analytical context that produced a given result is not merely a usability convenience but a regulatory necessity. The state model embeds traceability features into the platform's interaction infrastructure, ensuring that analytical reproducibility is a system property rather than a function of user discipline.

Table 3. State Architecture Validation Metrics and Framework Reliability

Framework / Study	Validation Method	Reliability Metric	Implication for State Architecture
Universal Design Scale (Gao et al.) [10]	Three-phase quantitative study	Cronbach's alpha = 0.960	Systematically derived state rules outperform ad hoc conventions
Low-Literacy Navigation (Guimaraes et al.) [9]	User study with hierarchical menus	Error rate increased with depth	State loss forces trial-and-error navigation, compounding errors
UI Design Recommendations (Diehl et al.) [7]	Multi-step expert validation	86.7% mandatory rate at specific level	State specifications can and should be formalized across complexity levels
AI-Assisted Design Workflows (Zhu et al.) [8]	Controlled experimental study	48.6% reduction in operation time	Structured interaction models reduce cognitive demand on complex tasks

Table 3 aggregates key validation metrics from the empirical literature, illustrating the convergent evidence that formally specified, systematically validated interaction frameworks produce superior outcomes to convention-based approaches across multiple dimensions of design quality and user performance.

5. Reusable Workflow Patterns and Governance

In addition to analytical primitives and state management rules, the unified interaction model is instantiated through a set of reusable workflow patterns that encode the sequential logic commonly found in enterprise analytical tasks. These patterns are not prescriptive scripts that constrain user behavior; rather, they are structural templates that provide an expected interaction path for common analytical objectives, reducing the navigational burden on users and allowing them to concentrate cognitive resources on data interpretation rather than interface navigation.

Two canonical patterns illustrate this concept. The investigative pattern follows the sequence: monitor, isolate, compare, validate, act. It supports anomaly detection and root cause analysis workflows in which an analyst identifies an anomalous signal in a monitoring view, isolates the relevant data subset through filtering and drilldown, compares the isolated data against a baseline or historical reference, validates the finding through additional analytical perspectives, and initiates an appropriate response. The operational monitoring pattern follows a related but distinct sequence: observe, alert, drill down, resolve. It supports time-critical operational workflows in which rapid context switching and efficient state preservation are essential to maintaining situational awareness while investigating alert conditions.

The predictive power of structured interaction sequences has been empirically demonstrated in adjacent domains. Yu et al. applied micro-interaction pattern analysis to an online learning platform and identified six types of interactive behavioral patterns, including structured exploration sequences and forum engagement sequences, that explained 82% of the variance in learner achievement across a cohort of 193

10.48047/jocaaa.2026.35.05.17

participants [11]. Their finding that structured, recurring interaction patterns are strongly predictive of performance outcomes has direct implications for enterprise analytics. When users are guided toward consistent interaction patterns through platform-level workflow templates, the resulting analytical performance becomes more predictable, more measurable, and more amenable to systematic improvement. Conversely, when each user must discover their own interaction path for each analytical task, the behavioral variance increases to a point where neither the user's performance nor the platform's effectiveness can be meaningfully assessed.

The predictive accuracy of pattern-based analysis also has implications for governance. Yu et al. demonstrated that micro-interaction data could predict with 91% to 95% accuracy which users were low-performing, a capability that is possible only when the interaction primitives being measured are clean, well defined, and capable of generating interpretable behavioral signals [11]. In enterprise analytics, governance mechanisms that promote the adoption and reuse of standardized workflow patterns create precisely this precondition: consistent interaction behaviors that can be studied, evaluated, and optimized at the platform level. Without governance, interaction patterns degenerate locally, deviating from platform standards in ways that generate behavioral noise and obscure the performance signals that would otherwise enable data-driven platform improvement.

The architectural risks of ungoverned, modular interaction design are further illuminated by research on pipeline architectures in human-computer dialogue systems. Wang and Yuan demonstrated that dialogue systems built on pipeline architectures with multiple independent sub-modules suffer from compounding error propagation, where errors introduced in early processing stages amplify through subsequent stages in ways that cannot be controlled after the fact. Their study showed that supervised classification methods trained jointly across a unified model achieved greater than 90% accuracy on intent detection tasks, compared to substantially lower accuracy when the same classification tasks were distributed across independently trained pipeline components [12]. The enterprise analytics governance lesson is clear: behavioral rules applied jointly across a cross-functional set of modules, through a versioned and shared specification discipline embedded in each feature development cycle, produce measurably superior platform behavior compared to rules applied independently to individual modules with minimal cross-functional coordination.

Table 4. Structured Interaction Patterns and Governance Implications

Study / Framework	Domain	Key Metric	Result	Governance Implication
Yu et al. [11]	Online Learning	Variance in achievement explained	82%	Structured patterns yield predictable, measurable outcomes
Yu et al. [11]	Online Learning	Prediction accuracy for low performers	91-95%	Clean interaction signals enable performance diagnostics
Wang and Yuan [12]	Dialogue Systems	Intent detection accuracy (joint model)	>90%	Joint governance outperforms independent module rules

10.48047/jocaaa.2026.35.05.17

Zhu et al. [8]	Enterprise UX	Reduction in operation time with structured workflows	48.6%	Formalized workflows reduce cognitive overhead

Table 4 synthesizes the empirical evidence for the relationship between structured interaction patterns and measurable performance outcomes, drawing parallels between the governance requirements of enterprise analytics and the pattern-based optimization demonstrated in adjacent domains. Good governance also encompasses role-based access to workflow complexity, ensuring that the flexibility afforded by the unified model does not produce uncontrolled variation in interaction behavior that undermines the consistency benefits the model is designed to deliver.

6. Discussion

The unified interaction model presented in this paper should be understood not as four independent components but as an interlocking system in which each component depends on and reinforces the others. The interaction grammar defines the vocabulary and behavioral rules; the state architecture ensures that those rules operate over a consistent and persistent analytical context; the workflow patterns encode how primitives are sequenced to accomplish analytical objectives; and the governance framework ensures that all three remain coherent as the platform evolves. Removing any one component weakens the others: a grammar without state architecture produces well-defined primitives that lose context at module boundaries; workflow patterns without governance degenerate into local variations; and governance without an interaction grammar has no behavioral specification to enforce.

From a practical standpoint, the most important implication for platform teams concerns timing. Usability engineering research has long established that design problems with interaction consistency are easier and less costly to address early in the development process [2]. Introducing a unified interaction model after a platform has accumulated years of fragmented interaction behaviors requires not only the engineering cost of retrofitting existing modules but also the organizational cost of retraining users who have adapted their workflows to the current inconsistencies. Teams building new analytics platforms have an opportunity to embed interaction architecture from the outset; teams managing existing platforms must weigh the costs of incremental harmonization against the ongoing costs of fragmentation.

The relationship between unified interaction models and emerging interaction paradigms also warrants consideration. As enterprise analytics platforms increasingly incorporate conversational interfaces, AI-assisted query builders, and agentic workflows that autonomously execute analytical tasks on behalf of users, the need for a consistent interaction grammar becomes more rather than less pressing. Hamdani and Chihi demonstrated that adaptive human-computer interaction models for Industry 5.0 contexts, which dynamically adjust to user behavior, cognitive load, and environmental conditions, produced improved efficiency, reduced cognitive load, and better fault detection in empirical validation [13]. These adaptive systems depend on consistent behavioral signals from the interaction layer; without a unified interaction model providing those signals, adaptation becomes unreliable because the system cannot distinguish between intentional user behavior and artifacts of interaction inconsistency.

Several limitations of this work should be acknowledged. The framework proposed here is conceptual and architectural; it does not include a controlled experimental validation of the unified interaction model's effects on specific analytical performance metrics. Additionally, organizational adoption barriers, including

10.48047/jocaaa.2026.35.05.17

the political and resource challenges of coordinating interaction specifications across distributed development teams, are substantial and are not fully addressed by the architectural framework alone. The generalizability of the model beyond enterprise analytics to other complex software domains remains an open question that would benefit from cross-domain empirical investigation.

7. Conclusion

Interaction consistency is a non-functional architectural requirement of enterprise analytics platforms. It is the condition under which such platforms can scale in scope and user diversity without incurring operational friction that progressively degrades analytical performance and decision quality. The interaction burden of an enterprise analytics platform is proportional to its number of modules, user roles, and distinct analytical intents; and broken interaction consistency creates relearning costs, promotes workflow abandonment, and produces inaccurate interpretations of analytical results due to unpredictable behavioral variation across the platform.

The unified interaction model addresses these liabilities through four interlocking components. An interaction grammar defines the analytical primitives and their behavioral rules, transforming interaction from an implicit social contract into an explicit, verifiable system specification. A workflow state architecture controls the preservation, propagation, and resolution of analytical context across navigational boundaries. Reusable workflow patterns encode the sequential logic of common investigative and operational tasks, providing structural interaction paths that reduce cognitive overhead. And a governance framework ensures that behavioral consistency is maintained across distributed development teams and product cycles, preventing the local deviations that compound into platform-wide fragmentation.

By integrating these four components, the unified interaction model provides a well-defined, testable, platform-level interaction infrastructure that creates consistent and learnable analytical experiences, enables reliable decision-making, and supports the long-term scalability and extensibility of enterprise analytics platforms across the organization.

References

[1] Nielsen, J. (1994). *10 usability heuristics for user interface design.* Nielsen Norman Group (updated Jan. 2024). https://www.nngroup.com/articles/ten-usability-heuristics/

[2] Holzinger, A. (2005). *Usability engineering methods for software developers.* Communications of the ACM, 48(1), 71-74. https://dl.acm.org/doi/10.1145/1039539.1039541

[3] Budiu, R. (2024). *10 usability heuristics applied to complex applications.* Nielsen Norman Group. https://www.nngroup.com/articles/usability-heuristics-complex-applications/

[4] Pitt, B. and Casasanto, D. (2022). *Spatial metaphors and the design of everyday things.* Frontiers in Psychology, 13, 1019957. https://doi.org/10.3389/fpsyg.2022.1019957

[5] Li, W., Zhou, Y., Luo, S., and Dong, Y. (2022). *Design factors to improve the consistency and sustainable user experience of responsive interface design.* Sustainability, 14(15), 9131. https://doi.org/10.3390/su14159131

[6] Gabry, J., Simpson, D., Vehtari, A., Betancourt, M., and Gelman, A. (2019). *Visualization in Bayesian workflow.* Journal of the Royal Statistical Society: Series A, 182(2), 389-402. https://doi.org/10.1111/rssa.12378

10.48047/jocaaa.2026.35.05.17

[7] Diehl, C., Martins, A., Almeida, A., Silva, T., Ribeiro, O., Santinha, G., Rocha, N., and Silva, A. G. (2022). *Defining recommendations to guide user interface design: Multimethod approach.* JMIR Human Factors, 9(3), e37894. https://doi.org/10.2196/37894

[8] Zhu, Z., Lee, H., Pan, Y., and Cai, P. (2024). *AI assistance in enterprise UX design workflows: Enhancing design brief creation for designers.* Frontiers in Artificial Intelligence, 7, 1404647. https://doi.org/10.3389/frai.2024.1404647

[9] Guimaraes, L., Martins, N., Pereira, L., Penedos-Santiago, E., and Brandao, D. (2022). *Interface design guidelines for low literacy users: A literature review.* Proceedings of the 2022 6th International Conference on Education and E-Learning, ACM, 29-35. https://doi.org/10.1145/3578837.3578842

[10] Gao, C., Lin, K., and Wu, Z. (2021). *The strategy of universal-design thinking in architecturally innovative product development.* Processes, 9(12), 2254. https://doi.org/10.3390/pr9122254

[11] Yu, H., Harper, S., and Vigo, M. (2021). *Modeling micro-interactions in self-regulated learning: A data-driven methodology.* International Journal of Human-Computer Studies, 151, 102625. https://doi.org/10.1016/j.ijhcs.2021.102625

[12] Wang, X. and Yuan, C. (2016). *Recent advances on human-computer dialogue.* CAAI Transactions on Intelligence Technology, 1(4), 303-312. https://doi.org/10.1016/j.trit.2016.12.004

[13] Hamdani, R. and Chihi, I. (2025). *Adaptive human-computer interaction for Industry 5.0: A novel concept, with comprehensive review and empirical validation.* Computers in Industry, 168, 104268. https://doi.org/10.1016/j.compind.2025.104268