Universal Library of Innovative Research and Studies Research Article

ISSN: 3065-0003 | Volume 3, Issue 3

Open Access | PP: 37-43

DOI: https://doi.org/10.70315/uloap.ulirs.2026.0303007

Evolution of Design Systems as a Tool for Ensuring Scalability of Digital Products: Analysis of the Impact of Architectural Principles, Tokenization, and Component Modeling on Interface Consistency

Volkodav Vladyslav

Senior Product Designer, Betterme, Lithuania, Kaunas.

Abstract

The article is dedicated to examining how contemporary design systems support the scalability of digital products through architectural principles, tokenization practices, and component modeling. The relevance of the study stems from the rapid expansion of multi-platform ecosystems, where fragmented interfaces undermine product coherence and increase development overhead. The novelty lies in treating the design system not as a static guideline set but as an evolving architectural structure that shapes consistency through modular organization, semantic token frameworks, and stable component hierarchies. The work describes the interplay between these elements and studies how they collectively sustain visual and behavioral alignment across distributed teams. Special attention is paid to the role of system governance, bottom-up evolution, and design-to-code convergence. The work sets itself a goal of identifying mechanisms through which design systems maintain cohesion as products and teams scale. To achieve this, analytical and comparative methods are applied. The conclusion describes the stabilizing effect of unified design architectures and their value for long-term product development. The article will be useful for researchers, UX architects, product designers, and engineering teams involved in large-scale digital systems.

Keywords: *Component Modeling, Design Systems, Design Tokens, Interface Consistency, Scalability.*

INTRODUCTION

Digital products now stretch across a widening array of platforms, interaction environments, and feature sets, and this expansion places unusual pressure on interface coherence. Fragmentation tends to surface as teams scale and parallelize their work, especially when design decisions remain isolated within individual product streams. The turn toward design systems reflects an industry-level response to this tension. These systems consolidate visual rules, component structures, and implementation guidelines into a shared design language that supports uniformity across increasingly complex product ecosystems. Their relevance stems from the rising demand for scalable UI solutions that remain maintainable over time, even as organizations accelerate release cycles and diversify their digital portfolios.

The study pursues a clear aim: to investigate how design systems function as architectural mechanisms for sustaining interface consistency in conditions of growth. To move toward this aim, three tasks are set.

1) To examine how architectural principles—particularly modular organization and separation of concerns—shape the scalability of design systems.
2) To analyze the contribution of design tokens to cross-platform coherence and the management of visual variation.
3) To explore how component modeling practices govern the stability of user experience when new features accumulate at speed.

Novelty arises from approaching design systems not as static reference documents but as evolving architectural assemblages whose internal structures—tokens, components, and rules—operate collectively to stabilize UX under expansion. The analysis brings these elements into a single conceptual frame, treating them as interdependent mechanisms that reinforce consistency across distributed design and development teams.

Citation: Volkodav Vladyslav, "Evolution of Design Systems as a Tool for Ensuring Scalability of Digital Products: Analysis of the Impact of Architectural Principles, Tokenization, and Component Modeling on Interface Consistency", Universal Library of Innovative Research and Studies, 2026; 3(3): 37-43. DOI: https://doi.org/10.70315/uloap.ulirs.2026.0303007.

METHODS AND MATERIALS

The section draws on a set of published studies that collectively illuminate the architectural, organizational, and methodological foundations of design systems. The research of Edelberg and Kilrain examines how design systems improve efficiency, collaboration, and interface uniformity in digital product development (Edelberg & Kilrain, 2020). The work of Churchill discusses the role of design systems in scaling user experience across large organizations (Churchill, 2019). The study of Lamine and Cheng provides an empirical account of design system practices, emphasizing architectural structure and operational challenges (Lamine & Cheng, 2022). The study of Yew et al. explores community practices, governance processes, and collaborative models shaping system evolution (Yew et al., 2020). The work of Patel analyzes style tokens as instruments for structuring visual logic and ensuring controlled variability (Patel, 2025). The study of Rose et al. addresses the pedagogical and methodological framing of component-based design (Rose et al., 2023). The study of Moore et al. investigates the extension of design systems to conversational user interfaces (Moore et al., 2020). The work of Kosuru presents a conceptual framework for scalable design in complex digital applications (Kosuru, 2022).

To address the article's objectives, a combination of analytical synthesis, comparative analysis, structural interpretation, and source examination was used. These methods made it possible to generalize architectural principles, align them with documented practices, and formulate a coherent model of how design systems sustain consistency under conditions of rapid growth.

RESULTS AND DISCUSSION

At its core, a design system is more than a style guide – it is an evolving product in its own right, meant to scale alongside the applications it serves. Modern design systems encapsulate a comprehensive "design language," including a repository of UI components, visual styles, and usage guidelines functioning as a single source of truth. Such systems are explicitly intended to tackle the inconsistency and inefficiency that plague large digital portfolios. In industry surveys, ensuring UI consistency ranks among the top motivations for adopting a design system, alongside efficiency gains in development (Edelberg & Kilrain, 2020). By codifying design decisions into reusable assets, teams avoid the drift that occurs when each product or feature team invents UI solutions in isolation (Churchill, 2019). Indeed, prominent examples like Google's Material Design demonstrate this effect at scale, with over 100 million web pages reportedly incorporating Material components to maintain a coherent look and feel across disparate services (Lamine & Cheng, 2022). The benefit extends beyond visuals: a shared system reduces redundant work and miscommunication between designers and engineers, enabling faster product iterations without sacrificing uniformity. In essence, the design system approach addresses a fundamental scalability challenge – it elevates interface design from a project-by-project effort to a systematic, architecture-driven process. Below is a conceptual structuring of system functions (Table 1).

Table 1. Functional Dimensions of a Modern Design System (compiled by the author based on Edelberg & Kilrain, 2020; Churchill, 2019; Lamine & Cheng, 2022)

Dimension	Description
Design Language Consolidation	Establishes a shared vocabulary of components, patterns, and rules for coherent multi-product ecosystems.
Cross-Team Alignment	Provides a centralized framework that synchronizes design and engineering workflows, reducing drift and ambiguity.
Scalable Governance Mechanisms	Supports controlled evolution of the system through contribution models, reviews, and standard-setting.
Implementation Convergence	Ensures that distributed codebases interpret and apply design guidance uniformly through reusable assets.
Experience Stability	Maintains consistent interaction and visual behavior across expanding feature suites.

A technical mindset underpins successful design systems. Rather than treating design as ad-hoc styling, teams apply architectural principles akin to software engineering. One principle is modularity: interfaces are decomposed into a hierarchy of components, from basic atoms (buttons, inputs) to complex composites (headers, forms). This echoes classic software modularity, promoting separation of concerns and interchangeability. A structured component taxonomy provides stability as systems grow – new interface elements can be constructed from existing modular pieces, preserving consistency by default. The conceptual relationships underlying this component organization are outlined below (Table 2).

Table 2. Structural Model of Component Hierarchy in Scalable Design Systems (compiled by the author based on Lamine & Cheng, 2022; Yew et al., 2020; Rose et al., 2023)

Component Level	Role in the System	Contribution to Scalability
Atomic Elements	Fundamental UI primitives (buttons, inputs, labels).	Provide the smallest stable units for composition and reuse.
Molecules	Small clusters of atoms forming functional UI fragments.	Support consistent assembly of repeatable patterns.
Organisms	Larger, self-contained interface sections.	Enable predictable expansion as features grow in complexity.
Templates / Layout Structures	Content-agnostic page or view structures.	Maintain spatial coherence across products.
Page-Level Compositions	Fully composed screens integrating multiple layers.	Stabilize behavior and interaction flow across teams and platforms.

For example, a design system might define a standard "card" component. Every new feature that needs a content container can use this card, perhaps with slight variants. The underlying structure (padding, corner radius, shadow style) remains uniform because it is architecturally baked into the shared component. This modular reuse is precisely how design systems scale: rather than reinventing patterns, teams instantiate pre-built components, much as developers import tested modules from a library. Studies of design system practice indicate that teams explicitly elevate components from concrete products into the system, merging them into a central library (Yew et al., 2020). This bottom-up cultivation of a component repository reflects an architectural mindset (Lamine & Cheng, 2022). It acknowledges that stable design "building blocks" must be identified and maintained with the same rigor as a codebase. In effect, the design system becomes a structural framework – an interface architecture – that product teams build upon. As new use cases arise, the architecture can flex by adding new components or extending guidelines, but always in a controlled, unified way. This helps avoid the fragmentation often seen when each project hacks together its own UI solutions. Architectural consistency, enforced by the system's rules and component API, ensures scalability does not degrade the user experience. Remarkably, this approach parallels known benefits of software architecture: constraints and patterns at the system level yield more predictable, maintainable outcomes in each implementation.

Alongside component libraries, design tokens form another pillar of scalable design systems. These tokens are essentially named variables for basic design attributes – colors, font sizes, spacings, etc. Instead of hardcoding values (e.g., a specific hex color in a button), designers and developers reference tokens (e.g., color. primary or spacing. small). This abstraction might seem trivial, but it greatly enhances scalability and consistency. By centralizing core values, a design system allows global changes and multi-brand theming with minimal effort. If a company rebrands with a new primary color, updating the token in one place propagates the change everywhere. The tokenization approach has been shown to improve consistency across platforms and products by providing a single source of truth for visual decisions (Patel, 2025). Teams at scale often support light and dark modes, or multiple product brands; tokens enable these variations without altering the underlying components, simply by swapping the token values per theme. In effect, tokens decouple design definition from implementation. This decoupling is crucial when dozens of development squads are consuming the design system. It prevents the divergence that occurs when each team picks slightly different shades of gray or padding values – a subtle entropy that can undermine a unified UX. Research underscores the importance of precise naming and hierarchy for tokens to avoid "sprawl" or ambiguity (Patel, 2025). A well-structured token taxonomy (e.g., category. subcategory.property, as in color.background.primary or spacing.card.margin) scales gracefully, as new tokens can be added in a logical framework without disrupting existing ones. Moreover, tokens serve as a lingua franca between design and code, bridging designers' intent and developers' implementation. This alignment is vital when products scale – a subtle mismatch in interpretation can lead to dozens of minor inconsistencies across an interface. A comparative overview of token categories is presented below (Table 3).

Table 3. Classification of Design Tokens in Scalable Multi-Platform Systems (compiled by the author based on Churchill, 2019; Patel, 2025; Moore et al., 2020)

Token Category	Purpose	Impact on System Consistency
Color Tokens	Encode semantic color roles across brands and modes.	Sustain coherent palettes and reduce subjective variation.
Typography Tokens	Define scalable text hierarchies independent of the platform.	Reinforce readable, predictable typographic rhythm.

Spacing & Layout Tokens	Regulate margins, paddings, and grid rules.	Prevent spacing drift and support harmonious composition.
Interaction Tokens	Capture states such as hover, focus, and press.	Normalize user feedback patterns across interfaces.
Elevation & Shadow Tokens	Describe depth and layering logic.	Ensure visual hierarchy remains stable across components.

By enforcing token usage, design systems ensure that, for example, the notion of "primary background color" or "small spacing" is identical in every context, across web and mobile, across different teams. Token-driven design thus provides both flexibility (easy theming and updates) and control (strict consistency in values), a combination that is otherwise hard to achieve at scale.

A scalable design system must not only house components and tokens but also define how they relate and compose. This is where component modeling practices like Atomic Design come into play (Rose et al., 2023). Influential design methodology has advocated designing from atomic units upward – define atoms (e.g., a button), then molecules (button + label), organisms (complete UI sections), and so on (Rose et al., 2023). The benefit is an inherently consistent system: larger UI assemblies are built from the same small parts, arranged by clear rules. Empirical work on style tokens and component libraries confirms that basing a system on atomic principles yields maintainable and scalable solutions (Lamine & Cheng, 2022). Each component is conceptually isolated, yet they interlock seamlessly because they share fundamental design tokens and adhere to common standards. For instance, if every component in a system uses the same spacing token for internal padding, then when those components are combined on a page, the rhythm of whitespace is harmonious by construction. There is no accidental doubling or misalignment of margins because the components were modeled under a unified spacing scheme. This consistency-by-construction is exactly what organizations seek from component-based design systems. Notably, scaling a product often means different teams working on different features that eventually live side by side. A robust component model curbs the risk of each team's work looking and behaving differently. It provides predefined interaction models and states for components (hover effects, error states, etc.), so teams don't invent their own. A study of design system practitioners found that one challenge was balancing customizability with standardization (Yew et al., 2020). The component modeling approach addresses this: components are made flexible through properties and variants, but not forked into wholly separate designs. Teams can extend a base component for a special use, but within the bounds of the system's style. In practice, this might mean offering a limited set of approved variants (e.g., a "primary" and "secondary" button style, no more) to cover most needs. Such guardrails ensure that, as new UI pieces are introduced for new features, they still feel part of the same family. The evidence of improved consistency is often anecdotal but compelling – for example, Airbnb noted that using their design system halved the time to build new pages and drastically reduced UI bugs, precisely because components and tokens handled the heavy lifting of consistency (Edelberg & Kilrain, 2020). In sum, a carefully modeled component library acts as a scalability multiplier: it allows dozens of contributors to design and code in parallel while the system quietly enforces cohesion and best practices in the background.

As digital products grow, design systems themselves must evolve. Scalability is not a one-time achievement but a continuous process of refinement. One key insight from recent research is the value of a bottom-up evolution strategy. Instead of a top-down decree of what the design system should contain, many organizations let their system grow organically by abstracting real product UI solutions that have proven useful. In interviews, design system leads describe elevating common components from different apps into the global system, merging duplicates and reconciling inconsistencies along the way. This process ensures the system stays relevant and practical, as it is built on actual needs and tested patterns rather than theoretical ones. However, it requires governance to maintain order. Typically, a centralized design system team curates contributions, reviews proposals for new components or tokens, and arbitrates conflicts between teams. Irregular though it may seem, tension often arises between preserving consistency and allowing innovation. The system must be flexible enough to accommodate novel UI ideas when warranted (for instance, introducing a new component type for a groundbreaking feature), yet disciplined enough to reject one-off styles that don't fit the overall architecture. Case studies show that design system governance forums – regular meetings or review sessions – help negotiate these decisions in a collaborative manner (Yew et al., 2020). Interestingly, the culture around a design system can influence its scalability as much as the technical architecture. Organizations with successful systems often cultivate a sense of community ownership, encouraging teams to suggest improvements or flag gaps. This distributed input, when funneled through a clear maintenance process, leads to a living system that scales in step with the product roadmap. Conversely, a design system that remains static or is imposed rigidly can become a bottleneck – teams may begin to bypass it to meet pressing needs, causing fragmentation to creep back in. Thus, ensuring the design system's own

evolution is crucial. In modern practice, some enterprises version their design system like software, releasing updates and changelogs, and even running analytics on system usage to identify unused or problematic components. By treating the system as a product, they apply scalability practices internally: refactoring components, deprecating redundant tokens, and optimizing documentation for onboarding new teams.

The impact of these efforts is evident in the consistency of user interfaces across large platforms. Users benefit from familiar interaction patterns and visual continuity as they navigate an ecosystem of products, even if those products are built by separate teams in parallel. One banking company's experience is illustrative: by uniting their various app teams under a single design system, they discovered and resolved an obscure usability issue (inconsistent balance-check behaviors across interfaces) that was previously masked by siloed development. The system's centralized perspective made this discrepancy visible and fixable. In another case, a bottom-up contribution surfaced when multiple products independently needed a more flexible data table component; rather than three divergent solutions, the design system team worked with product teams to develop one robust, scalable table component and rolled it out to all apps, improving consistency and reducing maintenance overhead. Such anecdotes, though informal, underscore the tangible advantages of a well-architected design system in scaling a digital product suite. The architecture, tokenization, and component modeling principles act like a scaffolding that holds the user experience together even as the edifice grows taller and more complex.

The evolution of design systems reveals a dynamic tension between creative diversity and enforced uniformity. On one hand, these systems thrive on a principled rigidity: a button is a button, adhering to the same rules everywhere. On the other hand, as organizations innovate, design systems are pressured to expand and flex without cracking. This tension surfaces most clearly in the push-and-pull over customization. Teams sometimes chafe against the constraints – a product might demand a unique interaction pattern that the existing system doesn't support. Should the system absorb this novel pattern (thus growing in scope), or should the product conform to the system's way (possibly at the expense of an optimal UX for that feature)? The answer often lies in a nuanced middle ground. Leaders of mature design systems emphasize conversation and iteration: if multiple teams independently voice the need for a new component or variation, it signals a genuine gap in the system's coverage, warranting an addition. But isolated one-off requests might be gently declined or handled with a temporary workaround. One observes an almost organic quality to the design system's evolution. It must be neither ossified nor capriciously malleable. As a living artifact, it develops through cycles of feedback and refactoring much like a codebase. In fact, some organizations explicitly version their design system and deprecate elements over time, introducing breaking changes in a controlled manner to realign the system's foundations when needed – an abrupt but sometimes necessary step to sustain long-term coherence.

Another notable tension is between consistency and creativity. By enforcing a common language of design, do design systems risk making products feel homogenous or templated? This concern is not merely theoretical; designers occasionally express frustration that overly strict systems leave little room for exploratory or brand-specific flourishes. The irony is that the consistency which users appreciate can feel creatively stifling to those building the product. Some companies mitigate this by establishing tiers of the system: a core that is sacrosanct and universal, and a flexible layer where teams can extend or customize within guidelines. For example, a core might dictate grid spacing and typography scales (never to be violated), while a flexible layer might allow custom illustrations or unique micro-interactions that give a product its character without breaking overall alignment. The presence of tokens and theming capabilities in systems directly addresses this – tokens allow a degree of brand or context customization (colors, fonts) without altering structural patterns. Thus, creativity finds channels within the system's structure rather than outside it. Nonetheless, maintaining this balance requires continual stewardship. Too lax, and the design system fragments into "design system-inspired" variants in each team; too strict, and teams implement dark patterns or workarounds outside the system to achieve their goals, also undermining unity. The human element, therefore, is critical: advocacy, education, and even negotiation form the backdrop of design system governance. It's not purely a technical challenge of architecture; it's also a social challenge of aligning diverse stakeholders around a shared vision of the product's UX.

One observes that interpretive shifts in what consistency means in an evolving context. Early in a design system's life, consistency might be a straightforward visual uniformity: ensuring the colors, icons, and typography are the same everywhere. But as systems mature, consistency extends to interaction and behavior. For instance, consistency means that similar user actions yield similar responses across the app suite – error messages appear in a standard way, loading states follow a common pattern, swiping gestures behave predictably across screens. Achieving this behavioral consistency is a subtler challenge than styling. It delves into the realm of experiential architecture. Architectural principles again help: defining interaction models at a system level (e.g., navigation principles, modal dialogs usage) creates a cognitive rhythm for users. However, as new technologies and paradigms enter (say, voice interfaces or augmented reality elements), the design system must reconcile them

with its established patterns (Moore et al., 2020). There is often a lag between technological innovation and design system integration. This lag can introduce temporary inconsistency – for example, a team might implement a voice assistant in one product that doesn't align with the rest of the UX, simply because the design system had no provisions for voice interactions. The tension here is between innovation and uniformity. Ideally, the design system eventually catches up, formalizing patterns for the new paradigm (perhaps guidelines for conversational UI that harmonize with the visual design). But in the interim, products can feel uneven. Some forward-looking design systems are now trying to be more proactive, monitoring emerging design trends and prototyping how they'd incorporate them so that when product teams experiment, they do so in a way that could scale. This adaptive foresight is itself a marker of scalability in the design system practice – recognizing that consistency is not a fixed target but a moving one as the definition of a "digital product" evolves (Kosuru, 2022).

There is also the interesting counterpoint that too much consistency can lead to user interface fatigue or inflexibility. Users may struggle to distinguish services if everything is too uniform, especially in a large ecosystem. Subtle differentiation can be important to signal context (think of how Google's apps share Material Design but each has its own icon color and nuances to fit its purpose). Design systems allow leeway for this via theming or secondary style choices. The discussion in practice revolves around where to draw the line. What constitutes a permissible variation versus a harmful divergence? This can be tension-filled, with debates often cropping up in design critiques: one team's "necessary tweak" is another's "breaking the system." Resolving these disagreements often relies on evidence – user research or data indicating whether a variation improves the experience or simply reflects subjective preference. Interestingly, the design system can be a tool for experimentation here: by rolling out a variation in a controlled way across multiple products, teams can gather data on its impact on usability or engagement. If positive, the variation might be upstreamed into the system; if negative or negligible, it is dropped. Thu, the system acts not only as a source of consistency but also as a framework for testing innovations at scale. This dynamic, experimental aspect of modern design systems underscores that they are not static rulebooks but evolving design infrastructures.

Under the hood, a tension also exists in the technical implementation of design systems – between centralization and performance/independence. A single design system repository that feeds all products is great for consistency, but what if a change breaks something? Coordinating updates across dozens of codebases can be daunting. Some organizations version-lock their products to particular design system releases and upgrade gradually, akin to dependency management in software. This strategy highlights a conceptual shift: the design system is treated as a platform or service. In doing so, it introduces certain inefficiencies (version mismatches mean not all products are perfectly consistent at a given time) in exchange for greater stability. The discussion here is essentially about change management at scale – how to propagate improvements without causing chaos. It's a reminder that scalability is not purely a design concern but also a process one. The most elegant design system won't scale if teams cannot absorb changes and contribute feedback in an orderly way. Many practitioners identify community and documentation as linchpins: thorough documentation and open channels for support encourage teams to actually use the system as intended, rather than bypass it out of confusion or urgency. Inconsistent usage of a consistent system can paradoxically result in inconsistency once again; hence, education and ease of adoption become critical factors. This sociotechnical blend – part architecture, part advocacy – is where a design system either flourishes across an organization or remains underutilized.

Ultimately, the evolution of design systems reflects a maturation in how organizations approach product design. One observes a shift from treating design as an ephemeral project output to treating it as a structured ecosystem asset. This shift is not linear or uniform; it's marked by irregular spurts of growth, occasional realignments, and continuous negotiation of principles. The discourse around design tokens, for instance, was virtually non-existent a decade ago, but has now become central to scaling discussions. Tokenization emerged as a response to multi-platform needs and has since influenced how consistency is conceptualized (not as exact sameness, but as systematic variance – e.g., dark mode vs light mode using the same token set). Likewise, the embrace of bottom-up contributions signals a more democratic evolution model for design systems compared to the earlier top-down style guides. This injects a healthy tension: the system must listen to the products even as the products obey the system. That interplay ensures the system stays alive. And it is in that somewhat messy vitality – with discussions, exceptions, and adjustments – that a design system truly ensures scalability of user experience, not by freezing the interface in time, but by giving it a resilient, flexible skeleton that can grow in any direction without losing shape.

CONCLUSION

A design system at scale resembles an evolving architecture of experience rather than a static rulebook. It is dense with interconnected parts – tokens, components, guidelines – that together impose an abrupt clarity on otherwise chaotic growth. There is no neat finality to its influence; consistency emerges as an ongoing practice, a residue of continuous alignment between teams and principles. In the end, the

scalability of digital product design is achieved not by endless addition but by controlled convergence. The design system becomes the abrupt scaffold that holds the interface together as it expands, allowing the product to grow without losing its identity. The result is an interface consistency that feels almost inevitable to users – a coherence born not of uniformity alone, but of an architecture that absorbs change and enforces order in the same breath. Each new feature or platform integrates without fanfare, guided quietly by the system's architecture and standards. No formal epilogue announces this consistency; it simply persists, woven into the fabric of every interaction as the product line scales ever further.

REFERENCES

1. Churchill, E. F. (2019). Scaling UX with design systems. *Interactions, 26*(5), 22–23. https://doi.org/10.1145/3352681
2. Edelberg, J., & Kilrain, J. (2020). Design systems: Consistency, efficiency & collaboration in creating digital products. In *Proceedings of the 38th ACM International Conference on Design of Communication (SIGDOC '20)*, Article 8, 1–3. https://doi.org/10.1145/3380851.3416743
3. Kosuru, P. (2022). Design systems for scalable digital applications: A framework for consistency and efficiency. *Journal of Mathematical & Computer Applications*, 1–4. https://doi.org/10.47363/JMCA/2024(1)E131
4. Lamine, Y., & Cheng, J. (2022). Understanding and supporting the design systems practice. *Empirical Software Engineering, 27*, 146. https://doi.org/10.1007/s10664-022-10181-y
5. Moore, R. J., Liu, E. Y., Mishra, S., & Ren, G. (2020). Design systems for conversational UX. In *Proceedings of the 2nd Conference on Conversational User Interfaces (CUI '20)*, Article 45, 1–4. https://doi.org/10.1145/3405755.3406150
6. Patel, R. (2025). The role of style tokens in modern design systems: Ensuring consistency and flexibility. *Journal of Information Systems Engineering & Management, 10*(30s), 34–44. https://doi.org/10.52783/jisem.v10i30s.4770
7. Rose, E. J., MacDonald, C. M., & Putnam, C. (2023). Design systems: A scalable model for teaching design systems for UX. In *Proceedings of the 5th Symposium on HCI Education (EduCHI '23)*, 5–7. https://doi.org/10.1145/3587399.3587403
8. Yew, J., Convertino, G., Hamilton, A., & Churchill, E. F. (2020). Design systems: A community case study. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems (CHI EA '20)*, 1–8. https://doi.org/10.1145/3334480.3375204

Copyright: © 2026 The Author(s). This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.