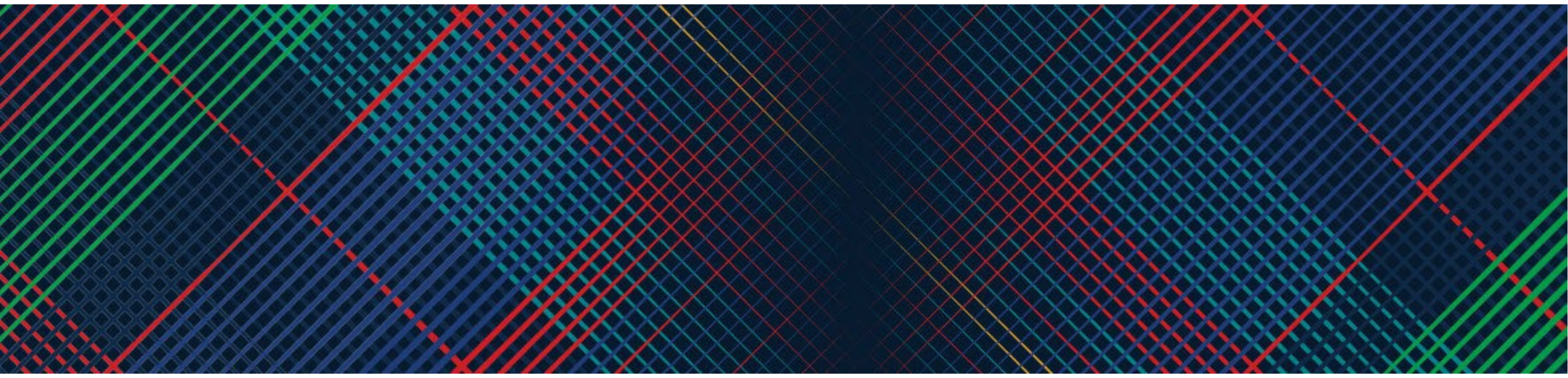
Carnegie Mellon University
Software Engineering Institute



SEI Report

How Design Systems Lead to Accessible and Secure Applications

June 30, 2024

Barbora Batokova

Jason Shimkoski

Daniel Tompkins

Yuliia Sergeeva

Marlon Mejia

Brandon Jabout

Christopher Baum

DOI: 10.1184/R1/XXXXXXX

Table of Contents

Abstract

Abstract— With the increasing complexity of digital products, it has become more challenging to create secure and accessible applications while also delivering a consistent user experience across platforms and brand touchpoints, all at scale and acceptable speed. Design systems provide a solution to manage this complexity by defining a set of reusable components and practices that serve as the single source of truth for design and development. By design and definition, design systems also play a crucial role in ensuring the accessibility and security of applications. They achieve this by (1) minimizing the number of dependencies and thus providing maximum control over the application; (2) offering standardized components optimized for accessibility; (3) implementing secure coding practices to create robust components and reduce vulnerabilities; (4) incorporating a collaborative feedback loop, and (5) leveraging human-centered design (HCD) to enhance system trustworthiness. Using our proprietary SEI Design System (SDS) as an example, we discuss these aspects.

Keywords— accessibility, components, dependency management, design patterns, design systems, feedback loop, human-centered design, secure coding practices, user interfaces

1 Introduction

With the increasing complexity of digital products, it has become more challenging to create secure and accessible applications while also delivering a consistent user experience across platforms and brand touchpoints, all at scale and acceptable speed. Bespoke design patterns and development methods have led to lack of consistency, variable quality, increased technical debt, and security risks like vulnerabilities and compliance issues. Design systems provide a solution to manage this complexity by defining a set of reusable components and practices that serve as the single source of truth for design and development. By design and definition, design systems also play a crucial role in ensuring the accessibility and security of applications through multiple aspects.

2 Managing dependencies

Dependency management involves handling external libraries, frameworks, and modules that software projects rely on, ensuring they are controlled, secure, and integrated. More external dependencies increase the risk of version conflicts, security vulnerabilities, build complexities, and performance issues. By proactively managing dependencies, teams can maintain a stable, secure, and maintainable project. However, reducing the number of external dependencies or using dependencies you fully manage is even more advantageous.

When building custom applications for the SEI, a proprietary design system like the SDS acts as a dependency that we have full control over and deep knowledge of, because we build it ourselves. It undergoes regular scans to detect vulnerabilities and malicious packages, and all dependencies are fetched from a locally-hosted mirror, where each component is scanned and cached before being used, thus enhancing security. By using only essential dependencies for each application, the potential attack surface is reduced. Regular checks with tools like *npm outdated* ensure dependencies are kept up to date, minimizing security risks from outdated versions.

Centralizing dependencies through a proprietary design system simplifies downstream maintenance, as updates are managed more efficiently across the entire project. This approach not only secures software integrity, but also optimizes maintenance efforts, ensuring reliability over time.

3 Implementing WCAG Compliance

Design systems that are compliant with Web Content Accessibility Guidelines (WCAG) provide dependent projects with optimized accessibility, which also contributes to system security. [1] One of the most established WCAG guidelines centers around ensuring an interface is perceivable, or presented, in a way that users can recognize and understand. As an example, colors, which are crucial for creating accessible experiences, are core elements of any design system. For the SDS, we developed tonal ranges that meet various WCAG contrast ratios, and thus established a robust color palette that can be used to create multiple combinations that meet either Level AA or AAA accessibility, depending on the project need.

Design systems can also set the foundation for users to successfully operate and understand digital experiences. Consider a simple form consisting of multiple inputs. Our SDS Form Group component provides structure and labeling for form fields and also defines navigational patterns for keyboard accessibility, so a user can quickly move through the form with their keyboard. It also implements helper text and validation messages, both of which help the user to prevent or recover from mistakes and successfully fill out the form.

WCAG-compliant design systems not only form the foundation for designing accessible and distinguishable interfaces in dependent projects, they also present significant savings in both development and design time.

4 Implementing Secure Code Practices

One of the biggest advantages of design systems is that they provide a common API, which defines a standardized set of interfaces, methods, and conventions that determine how components within the design system interact and can be used by developers. For example, with our SDS 3.0 release, we standardized shared props (and their attributes) such as size, kind, type, and variant across 15 components, significantly increasing consistency, efficiency, and maintainability across all projects that use the SDS.

Design systems also incorporate HTML and JavaScript best practices to prevent vulnerabilities. For example, SDS components can prevent prototype pollution, a JavaScript vulnerability where attackers can inject properties into existing JavaScript prototypes, which can be inherited into other components in the software, by utilizing the Vue framework to handle the security for us. SDS components also protect against Cross Site Scripting (XXS) attacks by utilizing "textContent" over "innerHTML" for input, resulting in the text being read as plain text and not HTML. As another example, the SDS Link component prevents the target blank vulnerability where a malicious website can change the window.opener.location to a phishing page by automatically adding on "noopener noreferrer' tags for an external link, which stops this attack.

Design systems are also built and deployed using static code analysis to detect potential regressions, deprecations, syntax errors, and security issues in the code. Static code analysis, which includes writing unit tests and using linting software, ensures that the components maintain high quality through rigorous testing.

Finally, many components in mature design systems like the SDS are written in Typescript, a framework designed to make explicit object types in code. This improves the developer experience by helping to prevent type errors when writing code and thus preventing buggy and unreliable code.

5 Incorporating Collaborative Feedback Loop

A design system thrives on collaborative feedback between end users, product teams, and maintainers. Tightening this feedback loop results in tailored design solutions that directly support the needs of product teams, maximizing readability and usability, minimizing implementation errors and bugs, and building with best practices by default. Driven by the needs of end users and product teams, as well as the needs of their products, new features and components become inherently more useful to the critical audience.[2]

Creating a dedicated platform for visual and technical documentation is paramount to provide a centralized source of truth and encourage a common language and goal.[3] Detailed, transparent standards for usability and best practices help the design system meet the expectations of their users, especially when that process allows opportunities for feedback.

Setting up design+build roadmaps with expected checkpoints is one example of an intentional practice which can measurably improve usability. For example, on the SDS, we employ a progressive release for new components: starting in Alpha, iterating before moving to Beta, then releasing as Ready. This gives priority to testing and documentation, making it an integral part of building the product and uncovering issues early through a collaborative process.

6 Leveraging HCD to Enhance Trustworthiness

Applying human-centered design (HCD) to create design systems ensures they embody the best user experience practices and are optimized for versatile use cases. Thoughtful intersection of needs from business, technology, and people combined with iterative approach to create the components not only makes design systems intuitive and user-friendly, but also ensures that the interfaces are perceived as safe and trustworthy. This reduces the chances of human errors by minimizing cognitive load, making it easier for users to understand and interact with the system correctly. [4]

Design systems, including the SDS, promote secure behaviors by informing users about the necessity of certain security measures and offering timely feedback, alerts, and visual cues (such as icons, badges, and colors) when potentially risky actions are taken. Thy also help achieve a higher level of error prevention and recovery, which are basic usability heuristics [5] that contribute to system trustworthiness, through features like auto-completion, validation checks, confirmation dialogues, and undo functions. All these attributes enable users to make more informed decisions and helps prevent mistakes that compromise security such as misconfiguring settings or inadvertently disclosing sensitive information.

References/Bibliography

[1] World Wide Web Consortium (W3C), "Web Content Accessibility Guidelines (WCAG) 2.2," W3C Recommendation, 05 Oct. 2023. [Online]. Available: https://www.w3.org/TR/WCAG22/. [Accessed: 23-Jul-2024].

[2] “Closing the Feedback Loop Between UX Design, Software Development, Security Engineering, and Operations | Proceedings of the 20th Annual SIG Conference on Information Technology Education,” *ACM Conferences*, 2019. https://dl.acm.org/doi/10.1145/3349266.3351420 [Accessed: 24-Jul-2024].

[3] “Guiding to Safety: How Technical Documentation Writers Can Encourage Software Security,” *Federal Trade Commission*, May 22, 2019. https://www.ftc.gov/about-ftc/bureaus-offices/bureau-consumer-protection/office-technology-research-investigation/guiding-to-safety-how-technical-documentation-writers-can-encourage-software-security [Accessed: 24-Jul-2024].

[4] N. Sevcenko, T. Appel, M. Ninaus, et al., "Theory-based approach for assessing cognitive load during time-critical resource-managing human–computer interactions: an eye-tracking study," *Journal of Multimodal User Interfaces*, vol. 17, pp. 1-19, 2023. [Online]. Available: https://doi.org/10.1007/s12193-022-00398-y. [Accessed: 23-Jul-2024].

[5] D. A. Norman, *The Design of Everyday Things*. Cambridge, MA: MIT Press, 2013.