Viel System, aber wenig Design?

Wie Design Systeme adaptiver werden können, um gute UX zu fördern

Thomas Immich
Centigrade GmbH
66123 Saarbrücken, Germany
thomas.immich@centigrade.de

ABSTRACT

Da digitale Produkte immer komplexer werden, wächst auch der Bedarf, diese systematisch zu skalieren. Design Systeme unterstützen dabei, wiederverwendbare UI oder Gestaltungs-Elemente synergetisch auch über Abteilungsgrenzen hinweg einzusetzen. Design Systeme entstehen in der Praxis allerdings oft „schubweise“ statt kontinuierlich, weshalb Designer und Engineers oftmals unterschiedliche Verständnisse von Inhalt und Sinnhaftigkeit eines Design Systems haben. Gemeinsames Verständnis besteht oft nur in dem Punkt, dass ein Design System in erster Linie das Entwicklungsteam unterstützen soll, während das zu entwickelnde System in erster Linie den Endnutzer unterstützen soll.

Ein zunächst „adaptierbares“ (nicht „adaptives“) Design System ermöglicht es Designern und Engineers, während der Laufzeit einer Anwendung mit der Darstellung und dem Interaktionsangebot einzelner UI Elemente zu experimentieren. Mittels Meta-Daten-Konfiguration können einzelne technisch umgesetzte UI Komponenten auch von Nicht-Engineers in verschiedene Ansichtsvarianten gebracht werden. Diese Konfigurations-Sets können mit einzelnen Nutzergruppen assoziiert werden, so dass verschiedene Nutzergruppen auch verschiedene UI Darstellungsformen erhalten. Um eine Zuordnung zwischen UI Darstellung und Nutzergruppe herzustellen, enthält ein adaptierbares Design System nicht nur UI- und Gestaltungs-Elemente (aus dem „Lösungsraum“) sondern auch Kontextelemente wie Personas oder User Stories (aus dem „Problemraum“).

Um den Schritt von einem adaptierbaren zu einem adaptiven Design System zu gehen, bringt jedes darin enthaltene UI Element sein eigenes Nutzungsdaten-Logging mit. Dieses generische Logging, welches einfache Interaktionsereignisse aufzeichnet, wird ergänzt durch einen zweiten spezifischen Logging-Mechanismus auf Anwendungsebene. Der zweite Logging-Mechanismus zeichnet auf User Story Ebene auf, in welchem Nutzungskontext sich der Nutzer gerade befindet. Kombiniert man die Ergebnisse der beiden Logging-Mechanismen, lässt dies Rückschlüsse darauf zu, welche UI-Element-Konfiguration für welche Nutzergruppen sinnhaft scheint oder nicht. Ist die Sinnhaftigkeit fraglich, so können Designer oder Engineers die Zuordnung zwischen UI-Element-Konfiguration und Nutzergruppe manuell feinjustieren. Auf Basis dieser kontinuierlichen Schärfungen „lernt“ das adaptive Design System über die Zeit auch selbst Entscheidungen bezüglich dieser Zuordnungen zu treffen. In diesem letzten Schritt wird dann nicht nur das Design System selbst adaptiv, sondern letztlich auch die daraus resultierenden User Interfaces.

KEYWORDS

Lean UX, Continuous UX, Design Systems, System Engineering, Adaptive UIs, Atomic Design, Design Patterns

1 Motivation

Digitale Produkte werden mit jedem Jahr komplexer und stehen zudem selten für sich allein, sondern interagieren miteinander in einem größeren Zusammenhang. Unternehmen denken daher immer häufiger in Plattformen und Service-Ökosystemen statt in einzelnen Apps oder monolithischen Großprodukten. Das ist eine gute und sinnvolle Entwicklung in einer Zeit, die immer mehr Flexibilität und Adaptionsfähigkeit fordert – sie setzt aber auch ein Umdenken voraus hinsichtlich der Vorgehensweise und Systematik, mit der man Plattformen und Ökosysteme aus UX Perspektive aufbauen muss.

Inzwischen sind Design Systeme als strategischer Faktor zur systematischen Entwicklung von UX und UI Landschaften kein unbekanntes Thema in der Branche [1]. Standards zur Etablierung effektiver Design Systeme gibt es allerdings wenige, da diese sehr stark von den Prozessen und Organisationsstrukturen des umsetzenden Unternehmens abhängen. Viel zu häufig werden daher herkömmliche UI Styleguides, die nicht über die Definition von visuellen Details und Gestaltungselementen hinausgehen, einfach in "Design System" umbenannt, ohne den wirklichen Kern und Nutzen der Thematik zu treffen. Auch Strömungen wie „Atomic Design“ [2] sind nur vordergründig eine Erleichterung bei der Entwicklung eines nachhaltigen, unternehmensweiten Design Systems. In der Praxis sind Design Systeme daher oft statische, hierarchische Bausätze, aus denen nicht minder statische und hierarchische

Veröffentlicht durch die Gesellschaft für Informatik e.V. und die German UPA e.V. 2019 in S. Hess & H. Fischer (Hrsg.):
Mensch und Computer 2019 – Usability Professionals, 08.-11. September 2019, Hamburg

https://doi.org/10.18420/muc2019-up-0296

Screens abgeleitet werden. Für die umsetzenden Teams fühlt es sich dann oft an wie „viel System, aber wenig Design“.

Im Folgenden wird daher ein Ansatz zur schrittweisen Etablierung eines adaptiven Design Systems vorgestellt, der in Teilen als Bestandteil des agilen UX Management Baukastens „Continuous UX“ [3] bereits in der Praxis eingesetzt wurde.

2 Design Systeme

Um das Themengebiet der Design Systeme gut ausleuchten zu können, lohnt es sich, zunächst den Begriff des „Systems“ in den Fokus zu nehmen.

In der Domäne des System Engineering wird ein System als eine Menge von Elementen in Interaktion definiert [4] sowie im System- und Software Engineering als eine Kombination von interagierenden Elementen, die einen oder mehrere angegebene Zwecke erfüllen [5]. Unter professionellen System Engineers gibt es daher die vorherrschende Meinung, dass beispielsweise ein Lego-Bausatz kein System darstellt.

> „Was ist denn nun definitiv kein System? Meine Antwort: ein Lego-Bausatz! Das ist eine Menge von Elementen, die aber nicht interagieren. Diese Menge hat zwar auch Eigenschaften, sie ergeben sich aber durch einfache Aufsummierung (etwa das Gesamtgewicht).“ [6]

Die Tatsache, dass die Elemente eines Lego-Bausatzes in beliebiger Art und Weise miteinander kombinierbar sind, scheint in dieser Betrachtung somit kein system-stiftendes Kriterium zu sein.

Diese Darstellung mag für Außenstehende zunächst nicht nachvollziehbar sein, zumal in der UX Online Community ein Design System vordergründig sehr häufig als eine Sammlung verschiedener UI Elemente und Patterns unterschiedlicher Komplexitätsstufen beschrieben wird und dem Grundprinzip eines Lego-Bausatzes somit sehr nah kommt. Tatsächlich beschränken sich UI Design Systeme jedoch sehr oft auf Elementbeschreibungen und deren Kombinationsmöglichkeiten und zielen weniger auf deren dynamische Wechselspiele untereinander ab.

Um eine solide Basis für die Etablierung eines Design Systems in interdisziplinären, agilen Teams zu schaffen, definiert Continuous UX ein Design System daher wie folgt:

> „Ein Design System ist die Gesamtheit von miteinander in Beziehung stehenden Nutzeranforderungen, Interaktionskonzepten, Gestaltungselementen und Code Bausteinen, die multi-disziplinäre Teams dazu befähigen soll, kohärente User Interfaces über Abteilungs- und Produktgrenzen hinweg effizient und effektiv zu realisieren.“

Diese Definition orientiert sich stark an der originären Definition aus dem System Engineering und fokussiert daher nicht vordergründig auf UI Elemente, sondern vor Allem auf das Wechselspiel zwischen Elementen, Artefakten und Benutzern, sowie auf den Zweck, der mit Hilfe des Design Systems erfüllt werden soll.

Um beispielsweise als Product Owner oder gar Chief Product Owner Sinn und Zweck eines Design Systems besser einschätzen zu können, gibt es eine Reihe von Indikatoren, die signalisieren, ob Kosten und Nutzen eines Design Systems in einem guten Verhältnis zu einander stehen. An dieser Stelle seien daher die wichtigsten Indikatoren aufgelistet, die in der Regel für die Etablierung eines Design Systems sprechen, sofern eine gute User Experience eines der strategischen Ziele des Unternehmens ist:

- **Teamkonstellation** (Teams größer als 16 Mitglieder, abteilungsübergreifende Teams, interdisziplinäre Teams)
- **Plattform-Diversifizierung** (z.B. Mobile, Desktop, Industrieanlage)
- **Produkt-Linien** (ähnlich Word, Excel, PowerPoint)
- **Ineffiziente oder ineffektive Übergaben** zwischen Designern und Engineers (häufig kopierter Code oder schlechte Qualität im Endergebnis)

Typische Elemente in Design Systemen

Unabhängig von der Definition sowie dem Sinn und Zweck eines Design Systems, stehen UX Designer regelmäßig vor der Aufgabe, Design System Elemente auf den einzelnen Komplexitätsstufen mit eindeutigen Begriffen zu belegen. Eine Betrachtungsweise, die sich in der Online Community überraschenderweise recht gut durchgesetzt hat, entleiht dabei Begriffe aus der Chemie und teilt Elemente daher unter dem Oberbegriff „Atomic Design“ in folgende Kategorien ein:

- Atome
- Moleküle
- Organismen
- Templates
- Seiten
- Patterns

Diese Einteilung greift an vielen Stellen zu kurz. Zum einen fallen „Templates“ und „Seiten“ aus der Begriffshemisphäre der Chemie ohne nachvollziehbare Begründung heraus, was es Themen-Neulingen und gerade Software Engineers erschwert, einem „roten Faden“ zu folgen. Zum anderen beziehen sich diese Kategorisierungen ausschließlich auf UI- und Gestaltungs-Elemente. Auch der „Pattern“ Begriff avanciert beim Atomic Design in der Praxis oftmals ungewollt vom „Lösungsmuster“ zur „konkreten Lösung“ – der Pattern Begriff entwickelt sich somit anders als ursprünglich von Christopher Alexander initiiert und beschrieben [7].

Continuous UX sieht es als essenziell an, dass ein Design System nicht nur UI- und Gestaltungs-Elemente enthält (also Elemente aus dem „Lösungsraum“), sondern auch Kontext-Elemente (also Elemente aus dem „Problemraum“). Es handelt sich um jene Elemente, die eher der Anforderungsentwicklung und -Analyse entspringen. Diese Elemente unterstreichen die Natur eines „echten“ Systems, da erst mit ihrer Hilfe Sinn und Zweck anderer Elemente definiert werden können. In diesem Sinne empfiehlt Continuous UX, folgende Artefakte aus dem

Problemraum ebenfalls zum Inhalt eines Design Systems zu machen:

- Personas
- Nutzungsszenarien
- User Needs
- User Requirements
- User Stories

Doch egal, ob wir uns im Problem- oder Lösungsraum befinden: wie die einzelnen Elemente im Design System benannt werden, muss letztlich dem eigentlichen Zweck des Design Systems dienen. Wenn dieser Zweck in erster Linie darin besteht, die Zusammenarbeit in interdisziplinären Teams effizienter und effektiver zu gestalten, so muss sich die Begriffsbelegung auch daran ausrichten. Somit gibt es an dieser Stelle keine „guten“ oder „schlechten“ Element-Bezeichnungen, sondern lediglich solche, die für das gesamte Team, also Product Owner, genauso wie UX Designer, User Researcher oder Software Engineers, gleichermaßen verständlich sind.

Adaptierbarkeit vs. Adaptivität

Adaptive Systeme sind bereits seit einiger Zeit Gegenstand der Forschung. Im UX Kontext geht es dabei in erster Linie um die Adaptivität eines User Interface gegenüber dem Endnutzer [8]: Ändern sich die Eigenschaften des Benutzers – beispielsweise dessen kultureller Hintergrund – so ändern sich auch die Eigenschaften der Benutzeroberfläche – beispielsweise Sprache und Farbgestaltung [9]. In diesem Zusammenhang ist vor allem die Unterscheidung zwischen „Adaptierbarkeit“ und „Adaptivität“ wichtig. Ein „adaptierbares“ User Interface ermöglicht es dem Benutzer, aktiv Teile des User Interface an die eigenen Bedürfnisse anzupassen. Ein „adaptives“ User Interface, im Gegensatz dazu, macht diese Anpassung automatisch auf Grundlage von Informationen zu diesem Benutzer.

Nun ist Fokus unserer Betrachtung aber nicht das User Interface selbst, sondern eben das Design System – also dasjenige System, aus dem User Interfaces nach entsprechenden Umsetzungsaktivitäten entspringen. Da die direkte Zielgruppe eines Design Systems nicht die Endnutzer, sondern die Mitglieder des Entwicklungsteam sind [10], passt sich ein adaptives Design System somit auch zunächst an die Bedürfnisse und Eigenschaften dieser Teammitglieder an und erst in der Folge an die Eigenschaften der Endnutzer. Diese wichtige Unterscheidung soll im Weiteren tiefer betrachtet werden.

3 Adaptivität von Design Systemen

Kern-Philosophie des adaptiven Design System Ansatzes ist die Auflösung harter Übergänge zwischen Analyse, Design, Spezifikation und Implementierung. Somit gibt es generell keine „Developer Handoffs“, wie sie viele Design-zentrische Tools unterstützen, so z.B. Figma [11] oder Zeplin [12]. Das Rollen-Verständnis innerhalb dieser Philosophie setzt bereits anders an und orientiert sich eher an der Computerspiele-Industrie. Diese Industrie muss seit Jahr und Tag Design und Engineering aufs Äußerste vereinen, um erfolgreich zu sein [13]. Daher bezeichnet die Rolle „Game Developer“ hier eine übergeordnete Verantwortlichkeit, unterhalb derer sich alle anderen spezifischeren Rollen wie Level Designer, Animator, Storywriter, Engine Programmierer etc. einfinden. Diese zwar spezialisierten Rollen verstehen sich im übertragenen Sinne dennoch als kontinuierlich im Wechsel agierende Finger eines einzelnen Schöpfers und nicht als verschiedene Hände unterschiedlicher Köpfe.

Abbildung 1: Traditioneller Ansatz mit „Developer Handoff“ (links) „Designer sind Developer, nur eben keine Engineers“ (rechts)

Aus diesem Grund hat ein adaptives Design System auch keine explizite Geburtsstunde oder Übergabestation, sondern wächst mit Beginn der Entwicklung eines oder mehrerer Produkte als eigenständiges Teilprodukt organisch mit.

Bevor ein Design System aber „adaptiv“ sein kann, muss es zwangsläufig zunächst einmal „adaptierbar“ werden. Man könnte ein adaptierbares Design System mit einem Level-Editor für ein Videospiel (oder eine Serie von Videospielen) vergleichen: während das Spiel eine Reihe von Elementen, wie Charaktere, Objekte, Dialoge und Regeln im Rahmen konkreter Szenen, zum Leben erweckt, stellt der Level-Editor eine kostensparende, synergetische Unterstützung dar, um dieselben Elemente auch in jegliche anderen Kontexte zu setzen und den Designer darin experimentieren zu lassen. Der experimentelle Faktor ist dort entscheidend für den Erfolg, da Design und Engineering jederzeit in Wechselwirkung stehen und deren Einfluss auf die Player Experience daher auch für erfahrene Designer oft nicht vorhersagbar ist.

Bei der Entwicklung von Videospielen entstehen auch eher selten seitenbasierte Klick-Dummys sondern hochdynamische Prototypen, da erstere eben keinen Erkenntnisgewinn hinsichtlich der beschriebenen Wechselwirkungen liefern. Zwar nutzen UX Designer bei der Entwicklung mensch-zentrierter Produkte oft andere Elemente als Videospiele, die es gilt in verschiedenste Kontexte zu setzen (also z.B. Patterns, Controls, Charts oder Icons), das grundsätzliche Vorgehen bei der Entwicklung von UIs kann aber zum großen Teil übernommen werden.

In einem adaptierbaren Design System entsteht bereits mit der Identifikation eines UI Elementes, also beim frühen Interaktionsdesign, eine „technische Komponente“ – und dies, obwohl zu diesem Zeitpunkt weder Visual Design noch Detailspezifikationen für das Element vorliegen oder die eigentliche Implementierung noch gar nicht geplant sein mag. Sobald also beispielsweise ein Interaktionsdesigner feststellt,

dass ein UI Konzept in jedem Fall eine „Auftragsliste“ („Build Job List“) zur Verwaltung von Produktionsaufträgen erfordert, wird von den Engineers bereits eine technische Komponente angelegt, die dieses UI Element repräsentiert, ohne dass eine solche „Build Job List“ zu diesem Zeitpunkt durch den Endnutzer verwendbar wäre.

Die technische Komponente „Build Job List“ kann dank der zwar unfertigen, aber technisch bereits integrierfähigen Hülle direkt im Anschluss als eine Art „Keimzelle“ in lebendige Kontexte eingespielt und dort bewertet, optimiert oder „zur Reife gebracht“ werden. In unserem Beispiel wäre die „Build Job List“ also vielleicht nichts weiter als ein graues Rechteck, das seinen eigenen Titel zeigt, aber noch keinerlei Effekt verursacht, wenn der Benutzer es klickt. Dennoch kann es bereits Teil des Layouts eines sich gerade in der Entstehung befindlichen Screens werden (siehe Abbildung 2). Man kann dieses Vorgehen vergleichen mit einer „Interface Definition“ oder „abstrakten Klasse“ aus dem Software Engineering, also der Beschreibung von technischen Schnittstellen und der teilweisen Nutzung dieser Schnittstellen, ohne bereits eine vollständige Implementierung zu besitzen [14].

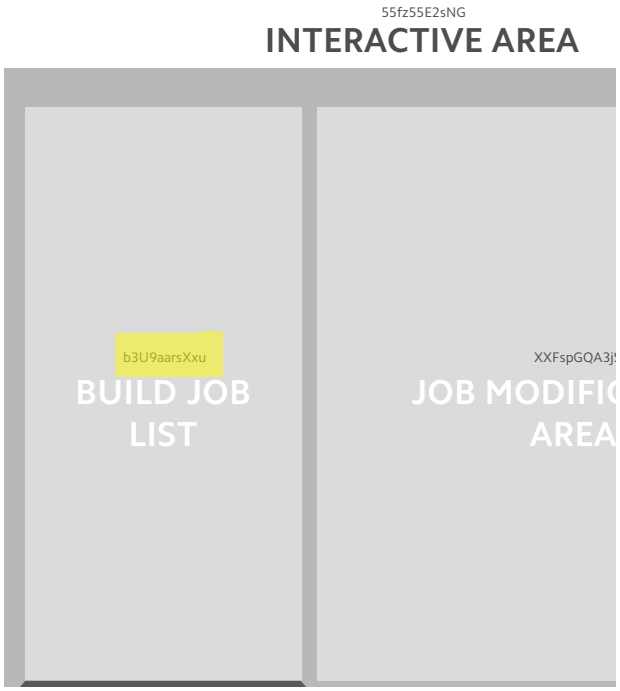


Abbildung 2: Die unfertige UI Komponente „Build Job List“, ist technisch bereits in einen Gesamtkontext integriert. Über dem Titel der UI Komponente steht die eindeutige Identifikationsnummer („b3U9aarsXxu“, hier gelb markiert), aber die Komponente ist noch nicht für den eigentlichen Benutzer verwendbar.

Da jedes neue Element automatisch eine eindeutige Identifikations-Nummer zugewiesen bekommt (z.B. „b3U9aarsXxu“), sind auch wiederholte Namens- oder Detailänderungen des Elementes im Laufe der späteren Entwicklung jederzeit möglich – auch unabhängig von der Hierarchie-Ebene des Elementes und ohne vorhandene Beziehungen zwischen Elementen zu brechen.

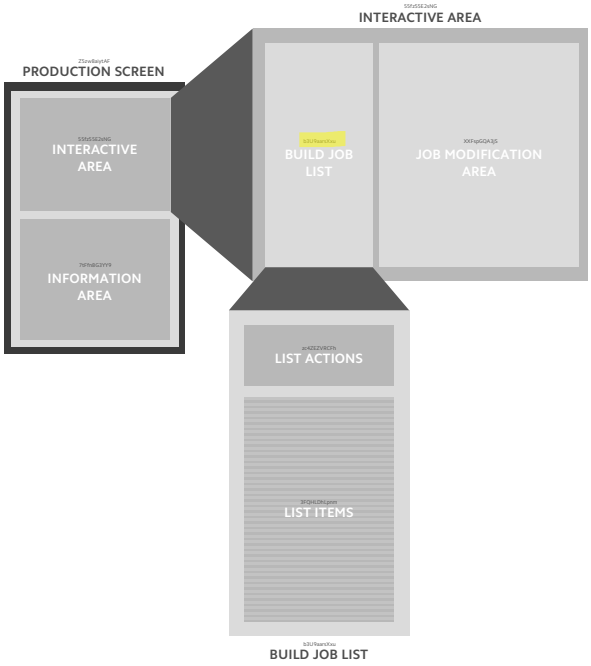


Abbildung 3: Entfaltete Darstellung hierarchischer UI Elemente, die zum frühen Zeitpunkt der Entwicklung als „Keimzelle“ lediglich über eine eindeutige Identifikation und eine temporäre Bezeichnung verfügen

Im Beispiel in Abbildung 3 wird nun auch jede weitere unfertige UI Komponente durch ein graues Rechteck mit Titel und Identifikationsnummer repräsentiert. Sollte sich im Verlauf der Entwicklung beispielsweise die Bezeichnung „Interactive Area“ in „Interaction Pane“ ändern, so stellt dies kein Problem dar, da jegliche hierarchische Zuordnung lediglich über die eindeutige Identifikations-Nummer geschieht und somit von der eigentlichen Element-Bezeichnung unabhängig ist.

4 Konfiguration von Adaptionsmöglichkeiten

Die zweite wichtige Komponente eines zunächst nur adaptierbaren Design Systems besteht in der integrierten Meta-Beschreibung und Konfiguration von UI Elementen. In herkömmlichen Design Systemen existiert das realisierte und noch zu konfigurierende UI Element an einem Ort (z.B. als Teil eines UI Toolkits wie „Material Design“ [15]) und die Beschreibung und Konfigurationsmöglichkeiten des Elementes an einem anderen Ort (z.B. als Teil eines web-basierten Design System Guide Manuals).

Im adaptierbaren Design System hingegen, ist sowohl die Beschreibung als auch Konfiguration Teil des Elementes selbst. Dies eröffnet dem UX Designer die Möglichkeit, zur Laufzeit

einer Anwendung – also genau in dem Kontext, in dem das Element auch tatsächlich dem Benutzer begegnen wird – das Element in verschiedenen Varianten durchzuspielen, um die treffendste Designauswahl für verschiedene Anwendungskontexte zu tätigen.

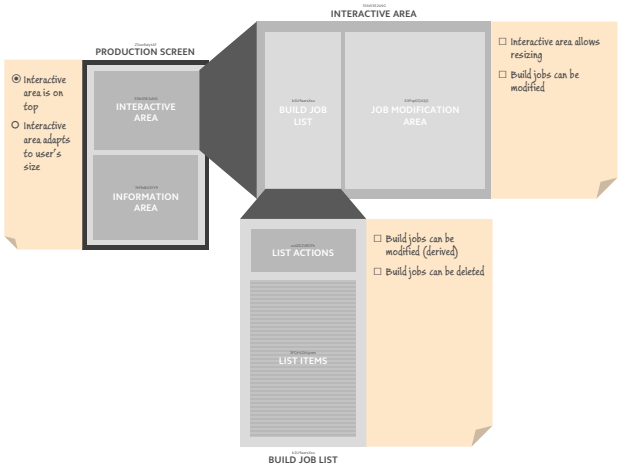


Abbildung 4: Meta-Konfigurationen in adaptierbaren Design Systemen als integraler Bestandteil des UI Elementes selbst

Abbildung 4 zeigt die einzelnen Konfigurationsmöglichkeiten pro UI Element jeweils als Notizzettel angehängt. Die Konfigurationsmöglichkeiten sind in der Regel durch „One of Many" oder „Many of Many" Checklisten repräsentiert, aber auch numerische Konfigurationen, wie etwa Grenzwerte für Eingabeparameter, sind denkbar.

Rollenspezifische Adaption

Natürlich ist es nicht immer zielführend, jede mögliche UI-Element-Konfiguration auch jederzeit für jedes Teammitglied frei wählbar zu lassen. Ein adaptierbares Design System sieht daher vor, dass der UX Designer eine Beziehung zwischen einzelnen Konfigurations-Sets und möglichen Nutzerrollen etablieren kann.

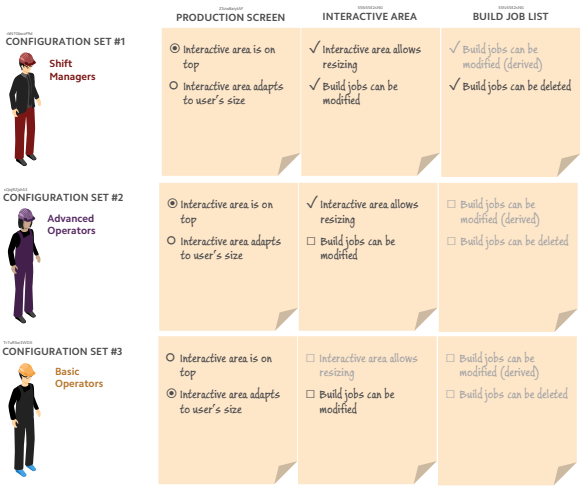


Abbildung 5: Zuordnung zwischen Benutzerrollen und Konfigurations-Sets

Während in dem Beispiel aus Abbildung 5 der Schichtführer („Shift Manager") beliebigen Einfluss auf den Produktionsplan sowie das Produktionsergebnis nehmen darf, da die UI-Element-Konfiguration ihm ermöglicht, Aufträge sowohl zu löschen als auch zu modifizieren, können fortgeschrittene Benutzer („Advanced Operators") nur die Reihenfolge von Aufträgen anpassen. Alle anderen Benutzer („Basic Operators") können hingegen keinerlei Änderungen am Produktionsplan vornehmen.

Es sei an dieser Stelle darauf hingewiesen, dass es sich bei der Konfiguration von UI-Elementen keinesfalls um eine klassische Rechteverwaltung handelt, auch wenn im Beispiel eines industriellen Produktivsystems natürlich Gemeinsamkeiten vorhanden sind. Jedoch ändern sich durch eine UI-Elementkonfiguration lediglich Darstellung und Interaktionsangebot eines UI Elementes – eine solide Rechteverwaltung auf der Applikations- und Datentransfer-Schicht ersetzt dies nicht [16].

Manuelle Adaption

Neben den klassischen Bedienerrollen können auch andere Stakeholder potenzielle Nutznießer eines adaptierbaren Design Systems sein. Beispielsweise könnte es für ein eher technisch orientiertes Mitglied des Entwicklungsteams in der Rolle des Service- oder Frontend Engineers sinnvoll sein, einzelne UI-Elemente direkt im Feld an die individuellen Bedürfnisse eines bestimmten Kunden anzupassen. Hierfür sollte ein „wirklich" adaptierbares Design System die Möglichkeit bereitstellen, auch zur Laufzeit auf die einzelnen Konfigurationsmöglichkeiten zugreifen zu können, um Darstellungsoptionen und Interaktionsangebote im realen Kontext durchzutesten und die optimale Konfiguration schließlich zum Einsatz zu bringen.

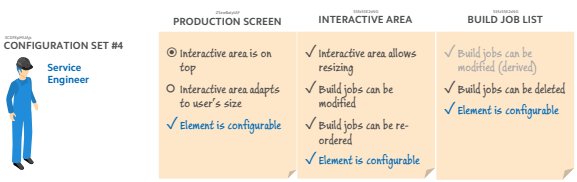
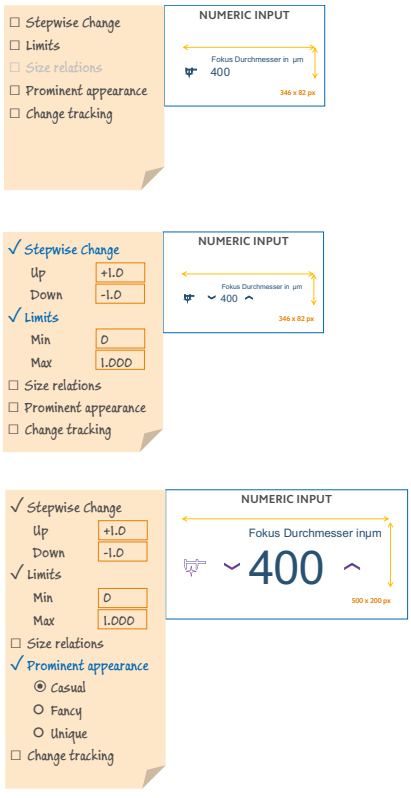


Abbildung 6: Konfigurierbarkeit von Elementen zur Laufzeit

Ein typisches Einsatzgebiet für diese Art von manueller Konfiguration zur Laufzeit ist beispielsweise die produktionsspezifische Anpassung von Eingabeformularen. Eingabeformulare können auf diese Art und Weise aus dem gleichen, immer wiederkehrenden Standard-Element (z.B. ein Element für die Nummerneingabe) vorbereitend angelegt und erst später, innerhalb der vom UX-Designer vorgegebenen Möglichkeiten, feinjustiert werden, wie in Abbildung 7 gezeigt. Hier beinhaltet das „adaptierbare Design System" also „adaptive UI Elemente".



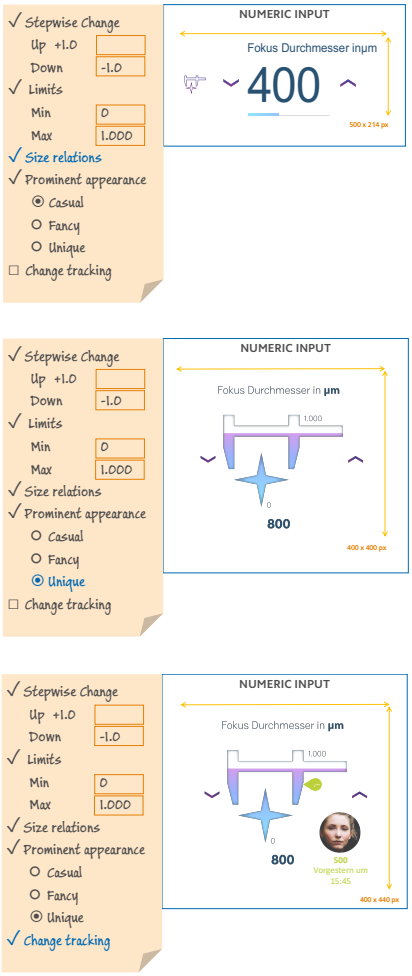


Abbildung 7: Adaptionsfähigkeit eines einzelnen Standard-UI-Elementes auf Basis verschiedener Konfigurationsmöglichkeiten

Benutzerspezifische Adaption

Wie bereits an früherer Stelle angedeutet, liegt ein adaptives UI System (im Gegensatz zum adaptiven Design System) dann vor, wenn Darstellung und Interaktionsangebote der UI-Elemente nicht nur durch manuelle Konfiguration definiert werden, sondern dies automatisch geschieht: sobald sich der Benutzer des UIs ändert, werden die passenden Konfigurationen automatisch angewendet – und alle UI-Elemente ändern dementsprechend Darstellung und Interaktionsangebot.

In unserem Beispiel bedeutet dies, dass jedes Mal, wenn ein Benutzer von kleiner Körpergröße mit dem UI interagiert, der interaktive Bereich automatisch in der unteren Bildschirmhälfte angezeigt wird, während der gleiche Bereich bei großen Bedienern in der oberen Bildschirmhälfte erscheint (Abbildung 8).

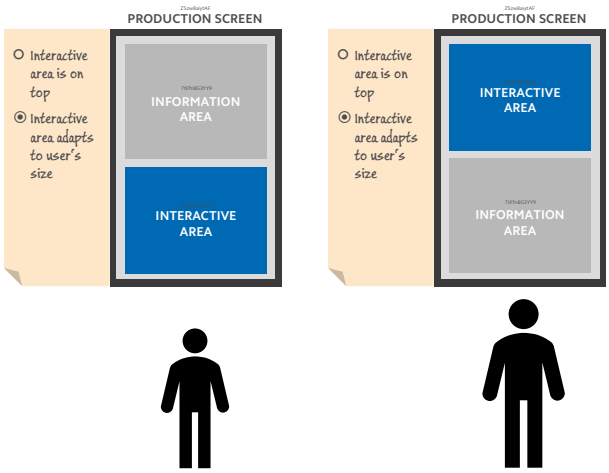


Abbildung 8: Gleiche Konfiguration, unterschiedliche Darstellungsweisen für Bediener unterschiedlicher Körpergröße

Adaptive Design Systeme

Erfolgreiche adaptive User Interfaces sind in der Praxis kaum zu finden. Das liegt aus Sicht des Autors unter anderem daran, dass die zur automatischen Adaption benötigten Kontext-Informationen oft nicht in ausreichendem Maße zur Verfügung stehen und zu sehr auf eine „Vollautomation" gesetzt wird. An dieser Stelle kann das „adaptive" (im Gegensatz zum „adaptierbaren") Design System eine Brücke zwischen Entwicklungsteam und Endnutzer bilden.

In einem adaptiven Design System bringt jedes UI-Element zunächst seinen eigenen Logging-Service für die Aufzeichnung von Nutzungsdaten mit. Beispielsweise zeichnen Buttons auf, dass sie geklickt, oder Listen, dass sie gescrollt wurden. Da diese Logging-Daten auf generischer UI Element Ebene noch wenig Rückschlüsse auf die anwendungs-spezifische Nutzung zulassen, gibt es einen weiteren Logging-Service auf Applikationsebene. Dieser zweite Logging-Service markiert Einstiegs- und Ausstiegspunkte für bestimmte Nutzungskontexte, so dass die Nutzungsdaten der generischen UI Elemente entsprechend in diese spezifischen Nutzungskontexte eingeordnet werden können [17]. In agilen Umfeldern werden diese Nutzungskontexte idealerweise durch Anfang und Ende einer einzelnen „User Story" repräsentiert [18].

Eine beispielhafte (und an dieser Stelle stark vereinfachte) User Story könnte lauten, dass Schichtführer Aufträge aus der Auftragsliste löschen möchten. Diese Story beginnt just mit der Verwendung des generischen „Löschen"-Buttons in der Toolbar der Auftragsliste und endet, sobald die Bestätigung des Löschen-Dialogs durch den Nutzer erfolgt ist. Der Löschen-Button wurde im Rahmen der genannten User Story verwendet und die User Story wiederum lässt Rückschlüsse auf die Nutzerrolle zu (Abbildung 9).

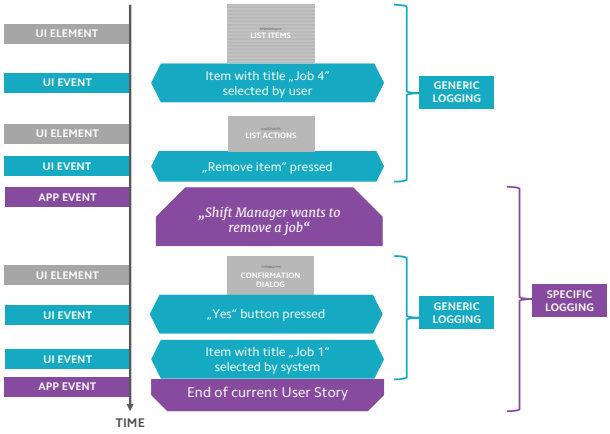


Abbildung 9: Das Logging generischer Ereignisse auf UI Element Ebene sowie spezifischer Ereignisse auf User Story Ebene liefert Teammitgliedern Rückschlüsse bzgl. der Sinnhaftigkeit von UI Element Konfigurationen

Auf Grundlage der so aufgezeichneten Nutzungsdaten kann dem Entwicklungsteam schließlich transparent gemacht werden, ob die Konfiguration eines UI Elementes auch im Sinne des Benutzers durchgeführt worden ist. Wenn Schichtleiter das Löschen in fast 100% aller Fälle mit „Ja" bestätigen, kann eine sinnvollere UI Lösung sein, generell auf die Sicherheitsabfrage zu verzichten und dieser Nutzergruppe stattdessen eher eine „Undo"-Funktion zur Wiederherstellung eines versehentlich gelöschten Items anzubieten.

Ein weiteres Beispiel: Wurde ein Nummerneingabefeld von einem Frontend Engineer so konfiguriert, dass es allen Nutzergruppen die Möglichkeit bietet, historische Werte nachzuverfolgen, von dieser Möglichkeit wird aber in der Praxis von einer bestimmten Nutzergruppe nie Gebrauch gemacht, so kann das Design System eben diesem Frontend Engineer vorschlagen, dieses UI-Element in Zukunft etwas kompakter anzuzeigen. Jeder Vorschlag muss von dem Frontend-Engineer mit „Ja" bestätigt oder mit „Nein" abgelehnt werden, wie in Abbildung 10 dargestellt. Natürlich ist diese Verantwortung auch auf andere Teammitglieder in anderen Rollen wie z.B. UX Designer übertragbar.

Abbildung 10: Selbstlernendes UI Element eines adaptiven Design Systems, welches dem Teammitglied vorschlägt, ungenutzte Features für den Endnutzer auszublenden

Durch den kontinuierlichen Frage-Antwort-Austausch mit den Teammitgliedern wird das Design System sukzessive „klüger" und kann schließlich eigene Konfigurations-Entscheidungen treffen. In dieser finalen Ausbaustufe des selbstlernenden Agenten [19] ist das Design System am adaptivsten und begünstigt in dieser Eigenschaft natürlich auch die Adaptivität der daraus entstehenden User Interfaces selbst.

5 Zusammenfassung und Ausblick

Um ein klassisches Design System zu einem adaptiven Design System auszubauen, reicht es nicht aus, mit einer Sammlung von UI Elementen oder Patterns ins Rennen zu gehen. Nur wenn alle möglichen Nutzungskontexte in Form von Szenarien, Personas und User Stories scharf umrissen sind, können Sinn und Zweck einzelner UI-Elemente überhaupt erst bewertet werden. In einem adaptiven Design System müssen UI Elemente zudem nicht nur auf Instanz-Ebene, sondern auch auf Meta-Ebene beschreibbar sein, denn zunächst muss das Design System „adaptierbar" werden, bevor es „adaptiv" werden kann. Die Meta-Daten ermöglichen Teammitgliedern, UI Elemente während der Laufzeit der Anwendung in verschiedene Darstellungs- und Interaktionsmodi zu versetzen. Da UI Elemente in adaptiven Design Systemen bereits sehr früh zu technisch integrierbaren Komponenten werden, können UX Designer ähnlich wie Level Designer mit verschiedenen Varianten und Konfigurationen experimentieren.

Schließlich wird es durch das spezifische Logging von User Story Anfangs- und Endpunkten auf Applikationsebene in Kombination mit dem generischen Logging von Interaktionsereignissen auf UI Element Ebene möglich, eine Zuordnung zwischen UI Elementen und deren Nutzung in verschiedenen Kontexten zu erhalten. Hieraus lassen sich sinnvolle UI-Element-Konfigurationen von unsinnigen unterscheiden und durch das Entwicklungsteam bewerten. Mit der kontinuierlichen Bewertung dieser Konfigurationen wird das Design System sukzessive adaptiver und kann in der finalen Ausbaustufe eigene Entscheidungen bezüglich der Konfigurationsauswahl treffen.

Gerade diese finale Ausbaustufe kann in Zeiten von Künstlicher Intelligenz und selbstlernenden Assistenzsystemen eine große Perspektive für die Etablierung „wirklich" adaptiver und mensch-zentrierter UIs darstellen, da sich UI-Elemente dann tatsächlich auf individuelle Benutzer einstellen könnten. Letztlich kommt bereits ein „nur" adaptierbares Design System mit rein manuellen Konfigurationsansätzen dem Bestreben ein gutes Stück näher, dass ein Design System mehr wird, als die Summe seiner Einzelteile.

ACKNOWLEDGMENTS

Ich danke meinen Kollegen bei Centigrade, die aufgrund ihrer professionellen Projektarbeit die Erkenntnisse in diesem Beitrag möglich gemacht haben.

REFERENCES

[1] Kholmatova, A. (2017). Design Systems. A practical guide to creating design languages for digital products. Smashing eBooks: https://www.smashingmagazine.com/design-systems-book/
[2] Frost, B. (2016). Atomic Design Methodology. Blog-Post: http://atomicdesign.bradfrost.com/chapter-2/
[3] Immich, T. (2018). Continuous UX - "Lean" und "Large" unter einem Dach. In: Hess, S. & Fischer, H. (Hrsg.), Mensch und Computer 2018 - Usability Professionals. Bonn: Gesellschaft für Informatik e.V. Und German UPA e.V. (S. 115-126).
[4] Guide to the Systems Engineering Body of Knowledge (SEBoK). (2017, March 27). in BKCASE Editorial Board, Guide to the Systems Engineering Body of Knowledge (SEBoK), version 1.8, R.D. Adcock (EIC), Hoboken, NJ: The Trustees of the Stevens Institute of Technology ©2017.
[5] Walden, D., Roedler, G., Forsberg, K., Hamelin, R. & Shortell, T. (2017). INCOSE Systems Engineering Handbuch: ein Leitfaden für Systemlebenszyklus-Prozesse und -Aktivitäten. München: GfSE e.V.
[6] Scheithauer, A. (2017). Was ist kein System? Blog-Post: https://www.oose.de/blogpost/was-ist-kein-system/
[7] Alexander, C. (1979). The Timeless Way of Building. New York: Oxford University Press. ISBN: 0195024028
[8] Morzy T., Wojciechowski M., Zakrzewicz M., Dachtera P., Jurga P. (2003) Implementing Adaptive User Interface for Web Applications. In: Kłopotek M.A., Wierzchoń S.T., Trojanowski K. (eds) Intelligent Information Processing and Web Mining. Advances in Soft Computing, vol 22. Springer, Berlin, Heidelberg
[9] Heimgärtner, R. (2012). Cultural Differences in Human-Computer Interaction – Towards Culturally Adaptive Human-Machine Interaction (Paperback B: Einband - flex. (Paperback) ed. Vol. 1): Oldenbourg Verlag.
[10] Curtis N. (2017). Defining Design Systems. Blog-Post: https://medium.com/eightshapes-llc/defining-design-systems-6dd4b03e0ff6
[11] Figma: https://www.figma.com/
[12] Zeplin: https://zeplin.io/
[13] Bethke, E. (2003). Game development and production. Texas: Wordware Publishing, Inc. p. 4. ISBN 1-55622-951-8.
[14] Riel, A. (1996). Object-Oriented Design Heuristics. Addison-Wesley Professional. p. 89. ISBN 0-201-63385-X
[15] Google Material Design: https://material.io/design/
[16] Richardson, L, Amundsen, M. (2013). RESTful Web APIs, O'Reilly Media, ISBN 978-1-449-35806-8
[17] Elberzhager F., Holl K., Karn B., Immich T. (2017). Rapid Lean UX Development Through User Feedback Revelation. In: Felderer M., Méndez Fernández D., Turhan B., Kalinowski M., Sarro F., Winkler D. (eds) Product-Focused Software Process Improvement. PROFES 2017. Lecture Notes in Computer Science, vol 10611. Springer, Cham
[18] Cohn, M. (2004) User Stories Applied. Addison Wesley, ISBN 0-321-20568-5.
[19] Russell, S., Norvig, P. (2003). Artificial Intelligence: A Modern Approach (2nd ed.). Prentice Hall. ISBN 978-0137903955.