# Basic UI/UX Guide

## Grundlagen für UI-Entwickler

Jürgen Messerer und Patrick Labud, bbv Software Services AG

**Im Zeitalter der Smartphones und Tablets wird auch im industriellen Umfeld zunehmend mehr Wert auf eine optisch schöne sowie leicht und geschmeidig zu bedienende Oberfläche gelegt. Damit das User Interface, kurz UI, auch von den Kunden akzeptiert wird, müssen einige Punkte in Betracht gezogen werden. Aber wie erreicht man das Ziel, wenn ein UI/UX-Experte fehlt?**

**In diesem Artikel wird gezeigt, wie Stolpersteine für die Nutzer verhindert werden. Zusätzlich wird gezeigt, wie ein effizientes und für die Nutzer nachvollziehbares User Interface konzipiert wird.**

### Einführung

Themen wie Benutzerfreundlichkeit respektive Benutzererfahrung (User Experience) sind keine Erfindung der Neuzeit. Schön früh erkannten Menschen, dass durch die Erfahrung der Nutzer ein System in seiner Funktion optimiert werden konnte. Ein gutes Beispiel dafür ist die Zeitung. Seit ihrer Erfindung im 16. Jahrhundert durchlief die Zeitung einige Veränderungen und Anpassungen, bis sie so wurde, wie wir sie heute kennen. Besonders im Format, aber auch im Design wurde und wird die Zeitung stetig weiterentwickelt.

Die User-Interface-Entwicklung verläuft sehr ähnlich. Zwei der Personen, die den Entwicklungsprozess massgeblich geprägt haben, sind Ben Shneiderman, der 1986 die *«8 goldenen Regeln des Interface-Designs»* niederschrieb. Und Jakob Nielsen, der 1994 eine Heuristik zur Überprüfung der Benutzerfreundlichkeit entwickelte.

**Die 8 goldenen Regeln des Interface-Designs**
Zwar sind die Prinzipien nicht vollständig und eingeschränkt, doch durch Anpassung und Erweiterung verhelfen sie einem Entwickler zu einem vereinfachten Start im UI-Design.

| |
|---|
| **1. Strebe nach Konsistenz:** In ähnlichen Situationen sollten die Aktionssequenzen immer konsistent sein. Z.B die Position eines Cancel Buttons, Layout, Farbe, Schriftart etc. |
| **2. Sorge für universelle Bedienbarkeit:** Erkennen der Anforderungen der verschiedenen Benutzer. Die verschiedenen Benutzer können sich in Alter, Erfahrung und allfälligen Behinderungen unterscheiden. Das System respektive Programm soll durch jeden Benutzer schnell und einfach bedient werden können. Zum Beispiel können für Experten Abkürzungen und für ungeübte Benutzer Hilfestellung angeboten werden. |
| **3. Biete informative Rückmeldungen:** Genügend Feedback über laufende Funktionen oder den Systemstatus muss stets gewährleistet sein. |
| **4. Entwerfe abgeschlossene Dialoge:** Hiermit soll erreicht werden, dass dem Benutzer bewusst wird, wann eine Aktion, Funktion oder Befehlskette gestartet wird und wann sie abgeschlossen ist. Zum Beispiel führt eine Shop-Website den Benutzer vom Selektieren des Produkts bis zum Zahlen der Ware. Beendet wird der Vorgang mit einer Bestätigungsseite, wonach die Transaktion vollständig und abgeschlossen ist. |
| **5. Biete einfache Fehlerbehandlung:** Der Benutzer darf nicht verzweifeln, wenn ein Fehler auftaucht. Nach Möglichkeit sollten verständliche Informationen über potentielle Ursachen abgegeben werden. Sehr wichtig ist es, einen Ausweg anzubieten, um in den normalen System- respektive Programmbetrieb zurückzugelangen. |
| **6. Lass die einfache Umkehrung von Aktionen zu:** Dem Benutzer soll die Möglichkeit gegeben werden, getätigte Aktionen wieder rückgängig zu machen. |
| **7. Vermittle ein Gefühl der Kontrolle:** Der Nutzer sollte immer das Gefühl haben, Kontrolle über das System respektive Programm zu haben. Kurz gesagt, der Benutzer agiert und das System reagiert und nicht umgekehrt. |
| **8. Entlaste das Kurzzeitgedächtnis:** Es sollte vermieden werden, dass der Benutzer sich zu viele Informationen merken muss, damit er zum Beispiel nachfolgende Aktionen auf einer nächste Seite ausführen kann. Als Richtlinie gilt, dass der Mensch sich zwischen fünf bis sieben verschiedene Informationen sicher merken kann. |

Tabelle 1: Die 8 goldenen Regeln des Interface-Designs

**Heuristik zur Überprüfung der Benutzerfreundlichkeit**
Auch sehr nützlich für die UI-Entwicklung ist die Heuristik, die Jakob Nielsen 1994 niederschrieb.

**Sichtbarkeit des Systemstatus** Genügend Feedback über laufende Funktionen oder den Systemstatus muss stets gegeben sein.

**Verknüpfung zwischen dem System und der realen Welt**
Es ist hilfreich, entsprechende Konzeptmodelle zu erarbeiten, die auf einer Analogie aus der realen Welt basieren. Dadurch ist die Funktionsweise für den Benutzer viel schneller nachvollziehbar oder sogar vorhersehbar.

**Kontrolle und Freiheit des Benutzers**
Der Benutzer sollte immer das Gefühl haben, die Kontrolle auszuüben. Dennoch sollte man dem Benutzer nicht grenzenlose Freiheit einräumen, da er ansonsten überfordert sein und Fehler machen könnte.

**Konsistenz und Standards**
Man sollte immer auf entsprechende interne und externe Konsistenz achten. Darüber hinaus sind, ausser in speziellen Ausnahmefällen, immer bestehende Standards zu berücksichtigen.

**Fehler-Vorbeugung**
Potentielle Fehlerquellen sollten frühzeitig eliminiert werden. Der Benutzer muss zudem ausreichend Anleitung erhalten, um keine Fehler zu verursachen.

**Wiedererkennen vor Überlegen**
Bevor der Benutzer nachdenken muss, wie eine Funktion zu bewerkstelligen oder wo sie zu finden ist, sollte er sie direkt anhand des Interfaces wiedererkennen können.

**Flexibilität und Effizienz der Benutzung**
Dies bezieht sich auf die nötige Balance zwischen Abkürzungen (engl. shortcuts) für Experten und ausführlicher Hilfestellung für Anfänger.

**Ästhetisches und minimalistisches Design**
Ein Design sollte stets ästhetisch ansprechend, aber dennoch minimalistisch sein, um unnötige Verwirrung und Übersichtsverlust zu vermeiden.

**Benutzerhilfe und Support bei Fehlern**
Fehlermeldungen müssen klar und verständlich sein. Sie müssen das Problem und eine mögliche Lösung aufzeigen.

**Hilfe und Dokumentation**
Auch wenn es besser ist, ein System ohne Dokumentation betreiben zu können, so ist es manchmal doch nötig, eine Hilfe und eine Dokumentation anzubieten. Diese sollten einfach zu durchsuchen sein und sollten Schritte hin zur Lösung eines Problems aufzeigen.

Tabelle 2: Die 10 Usability-Heuristiken

**Layout-Design**
Mit den 8 goldenen Regel und der Heuristik von Jakob Nielsen hat ein Entwickler gute Voraussetzungen, um ein User Interface so zu designen, dass es vom Benutzer verstanden wird und gut bedient werden kann.

Wichtig ist auch die Aufteilung des Bildschirms in Spalten und Reihen. Fast alle UI-Frameworks bieten, unabhängig von der Sprache, ein sogenanntes Gridlayout an. In diesem können dann Eigenschaften, beispielsweise Margin und Spacing, eingestellt werden. Bei der Layout-Einteilung haben sich zwei Grundpositionen etabliert: Man nimmt alle Höhen und Breiten als ein Vielfaches von entweder fünf Einheiten oder vier Einheiten an, wobei als Einheiten Pixel, Points, em oder Sonstiges dienen kann.

Wird die Einteilung in 5er Einheiten gewählt, dann kann nur eine geradzahlige Spalteneinteilung verwendet werden. Ungerade Einteilungen wie drei Spalten sind so nicht möglich. Aus diesem Grund empfiehlt es sich, alles auf einer Basis von vier Einheiten zu wählen. Dadurch können gerade wie auch ungerade Spalteneinteilungen gewählt werden.

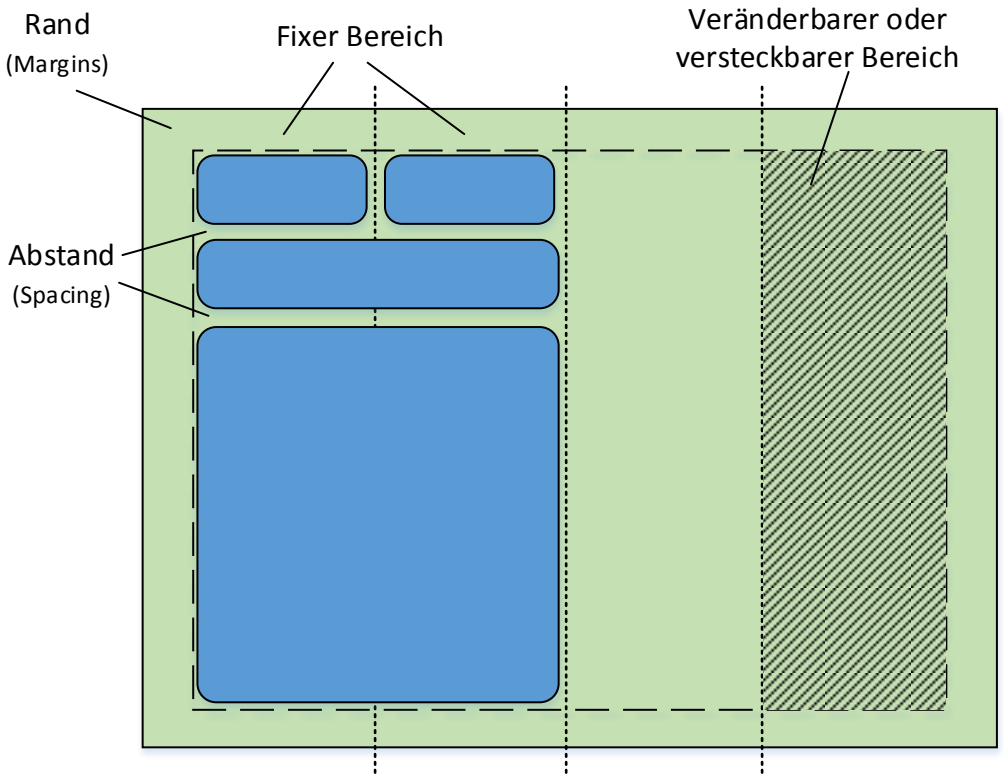

Abbildung 1: Gridlayout mit variablem Bereich

**Layouts**
Früher wurden die Applikationen zumeist mit einem starren Layout fix für eine bestimmte Bildschirmgrösse entwickelt. Durch die zunehmende Vielfalt an verschiedenen Displaygrössen und -auflösungen entstand nach und nach das Problem, dass die Applikation für jeden neu gewählten Bildschirm auch neu übersetzt werden muss.

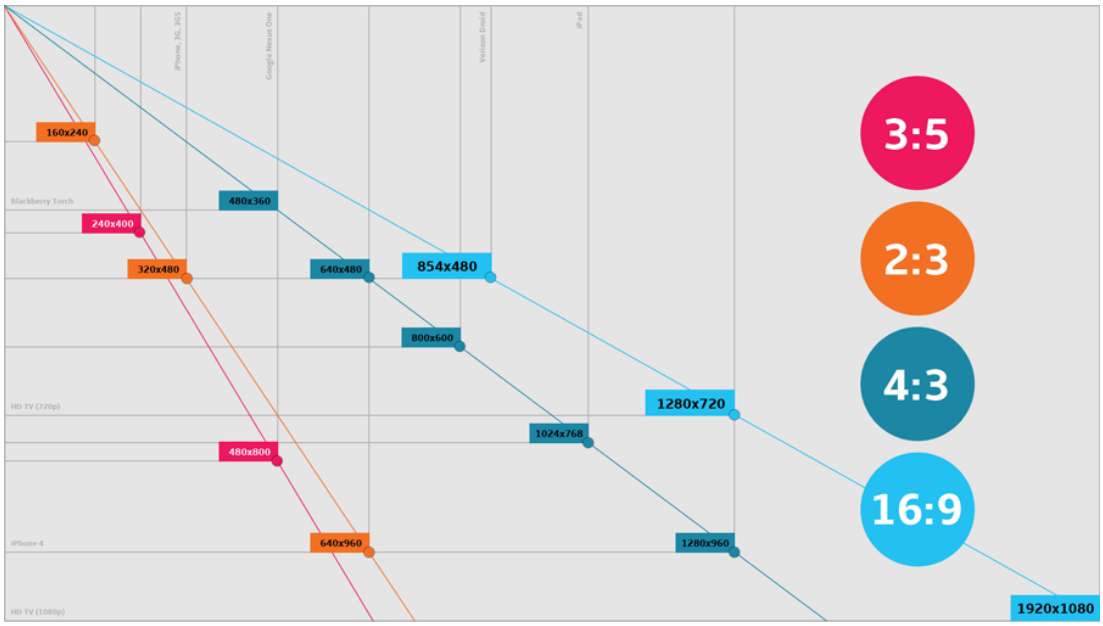

Abbildung 2: Bildschirmgrößen

Um dieses Problem zu beheben, sind neben dem fixen Layout noch zwei Mechanismen zur Gestaltung variabler Layouts gängig.

Folgende Layouts respektive Designs werden unterschieden:

- Static/Fixed: Das Layout ist fix auf eine bestimmte Displaygröße eingestellt. Es ist sehr einfach zu implementieren, benötigt aber auf der anderen Seite eine manuelle Anpassung für jede weitere Displaygröße.

- Fluid/Liquid: Das Layout passt sich der Bildschirmgröße in der Breite und Höhe an. Die Elemente werden entweder vergrößert oder verkleinert. Für Bildschirme, die sich in der Grösse nur geringfügig unterscheiden, kann diese Art von Design gut eingesetzt werden. Bei deutlichem Bildschirmunterschied werden die Elemente zu sehr gestreckt. Dann muss die Applikation manuell im Code angepasst werden.

- Responsive: Das Layout passt sich auch hier der Bildschirmgröße an. Es ändert sich aber nicht nur die Grösse der Elemente, sondern auch die Anordnung der Inhalte in den Elementen. Zusätzlich werden Elemente ein- oder ausgeblendet bzw. in ein Menü verschoben. Das Implementieren und das Testen eines solchen Layouts sind deutlich aufwändiger als bei den anderen Arten. Dafür passt sich die Applikation dynamisch laufend an jede gewünschte Bildschirmgröße an.

**Zusammenfassung**
Immer häufiger wünschen sich Kunden aus dem industriellen Umfeld User Interfaces, wie sie sie von Smartphones her kennen. Diese Interfaces müssen effizient und effektiv zu bedienen sein, damit die Kunden zufrieden sind.

Mit Hilfe der «8 goldenen Regeln des Interface Design» und unter Berücksichtigung der 10 Heuristiken von Jakob Nielsen kann ein Softwareentwickler sich vor einigen Stolperfallen der UI-Entwicklung schützen.

**Literatur- und Quellenverzeichnis**
https://www.cs.umd.edu/users/ben/goldenrules.html
http://www.nngroup.com/articles/ten-usability-heuristics/
Responsive Design Example: http://www.liquidapsive.com/

**Autoren**
Jürgen Messerer arbeitet bei der bbv Software Services AG als Fachleiter C++ und Qt sowie als Embedded Senior Software Engineer. Seine Schwerpunkte liegen in Embedded-Linux-Systemen sowie in der Applikationsentwicklung mit C++ und Qt.

**Kontakt**
Internet: www.bbv.ch
Email: juergen.messerer@bbv.ch

Patrick Labud arbeitet bei der bbv Software Services AG als Fachleiter UX/Usability. Er unterstützt Kunden dabei, die Disziplin in der Firma aufzubauen und in IT-Projekten zu etablieren.

**Kontakt**
Email: patrick.labud@bbv.ch