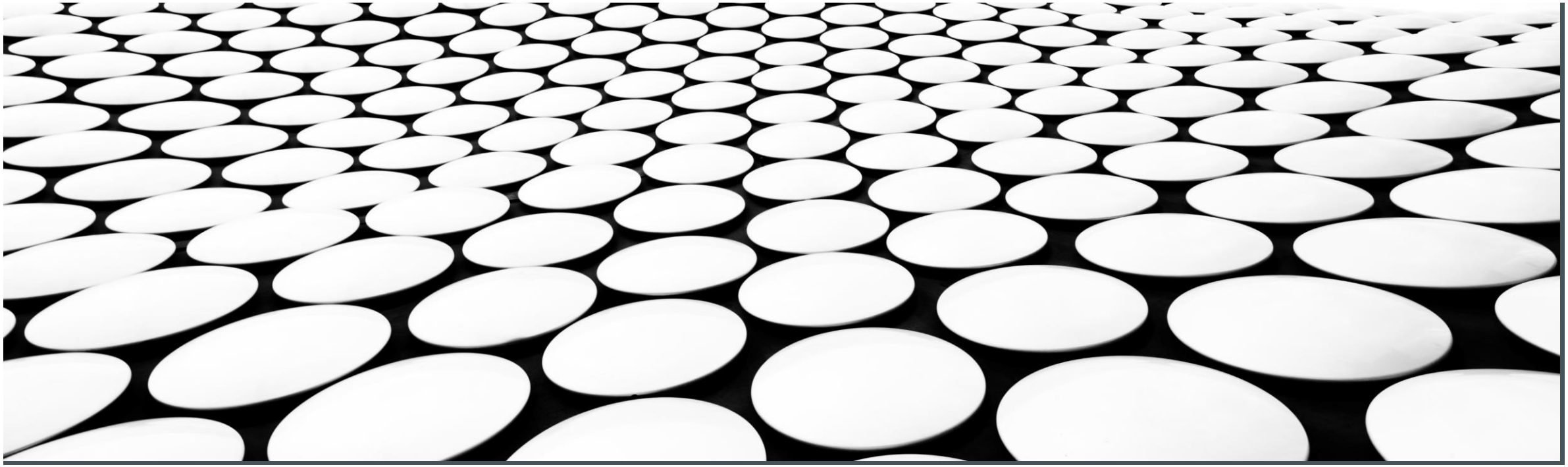
SOFTWARE ENGINEERING

ENTWURFSPHASE



ZIELSETZUNG DES SOFTWAREENTWURFS

Zielsetzung des Software-Entwurfs ist eine Software-Architektur.

Dabei geht es um die Design-Entscheidung, welche Teile der „Strukturierten Analyse“ in Software umgesetzt werden sollen und wie die Struktur des künftigen Software-Systems aussehen soll.

Üblicherweise spricht man dabei von Systemkomponenten und deren Beziehungen untereinander.

Eine Systemkomponente ist ein abgegrenzter Teil eines Software-Systems und dient als Baustein für die physikalische Struktur einer Software-Anwendung.

Beispiele für Systemkomponenten sind:

- Funktionen für prozedurale Entwürfe
- Klassen für objektorientierte Entwürfe

VORGEHEN BEI DER ERSTELLUNG DES SOFTWAREENTWURFS

1. Bestimmung der Zielplattform:

- Abhängigkeit zwischen den Endgeräten und dem Betriebssystem definieren.
  - Desktop-Anwendung, Web-basierte Anwendung, App…

VORGEHEN BEI DER ERSTELLUNG DES SOFTWAREENTWURFS

2. Architekturdesign:

- Schnittstellen
- Frontend
- Backend
- RESTful API für die Kommunikation zwischen dem Frontend und dem Backend
- Die Nutzung von den Technologien (z.B. Programmiersprachen)
- Die Nutzung vom MVC-Muster
- Die Darstellung, wie Systemteile miteinander kommunizieren werden
- Die Darstellung wie das System das MVC-Muster verwenden wird
- die Einrichtung einer Datenbank

VORGEHEN BEI DER ERSTELLUNG DES SOFTWAREENTWURFS

3. Entwurf der Benutzeroberfläche:

- Erstellung der Mockups (Skizze, Prototypen, Layouts!)
- Detailliierte Beschreibung aller Elemente Der Benutzeroberfläche

4. Datenbankdesign:

- Planung der Datenbankschema
- DBMS auswählen
- Erstellung des ERM (Entity-Relationship-Model)
- Erstellung des Tabellenmodells

VORGEHEN BEI DER ERSTELLUNG DES SOFTWAREENTWURFS

5. Geschäftslogik:

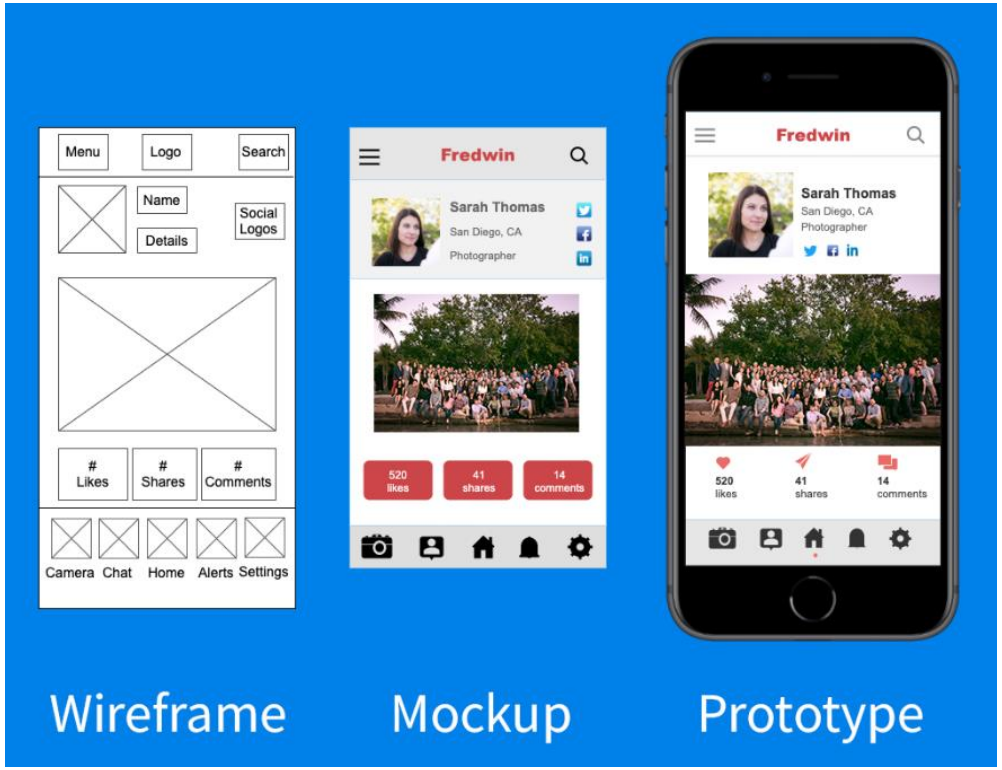
- Beschreibt, wie der Benutzer mit der Anwendung interagiert (UML-Anwendungsfalldiagramm)
- Beschreibt die Objekte und deren Interaktion (Objektdiagramm)

6. Mockup-Daten:

- Erstellung von Mockup-Daten in „JSON“ oder „XML“ Formate als Test-Daten während der Implementierung zum Test, danach wird die Datenbank und deren Tabellen implementiert

SKETCHING, WIREFRAMES, MOCKUPS, PROTOTYPING

SKETCHING, WIREFRAMES, MOCKUPS, PROTOTYPING



SKETCHING

Sketches oder zu deutsch „Skizzen“ sind immens wichtig und sollten im Designprozess nicht fehlen. Im Unterschied zu einem Wireframe sind Skizzen nur grob gehalten und unsauber. Es ist sogar erwünscht, dass die Skizzen keine Perfektion ausstrahlen. Es geht hier ausschließlich um das Verständnis was der Benutzer sieht. Man kann es auch als eine Art grafisches Brainstorming bezeichnen.

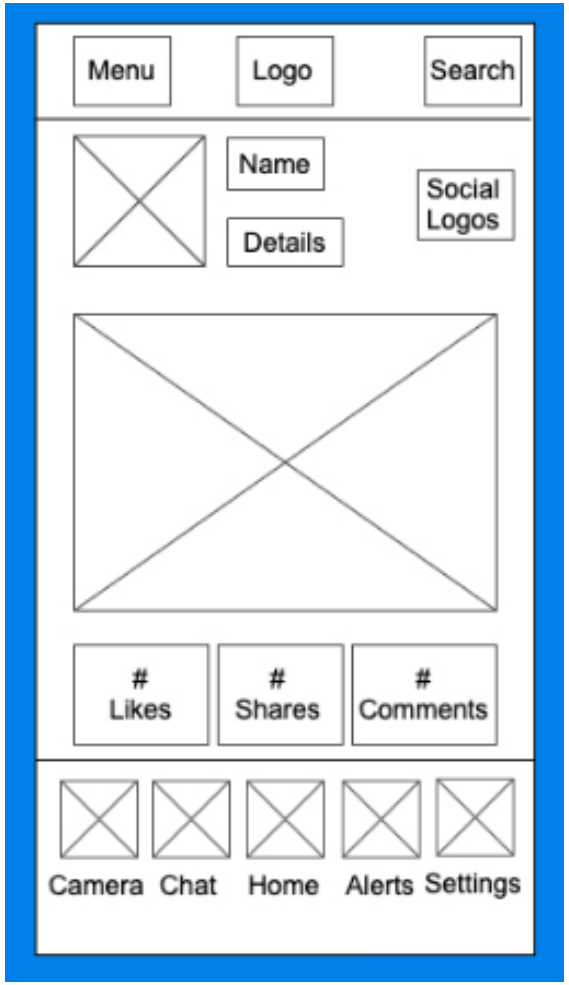
SKETCHING

Vorteile

- Zeitersparnis im Workflow-Prozess.
- Ideal zum Brainstorming von Ideen und zur Zusammenarbeit im Team.
- Verfeinert den späteren Wireframing-Prozess
- Hilft bei der Bewertung der Machbarkeit von Features und beim Ausschluss von Layout- und Funktionalitätsproblemen.

WIREFRAMES

Wireframe bedeutet so viel wie „Drahtgerüst“, welches den Ursprung im Begriff der Schneiderpuppe hat. Als Wireframe wird der konzeptionelle Entwurf einer Applikation oder Internetseite während der Planungsphase bezeichnet. In diesem Entwurf werden nur die nötigsten Elemente der Anwendung dargestellt.



WIREFRAMES

Man kann diese wieder in zwei Gruppen einteilen:

1. Statische Wireframes

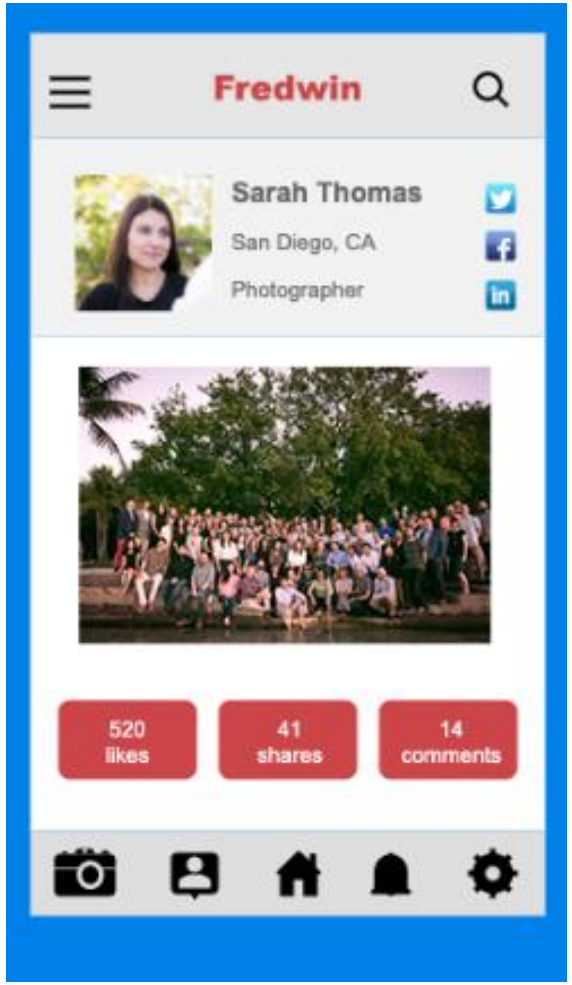
Unter einem statischen Wireframe versteht man eine schematische Darstellung von nur einer einzelnen Oberfläche.
Für diese werden die grundlegenden Elemente definiert.
Das Layout eines statischen Wireframes sollte konzeptionell sein.

2. Dynamische Wireframes

Dynamische Wireframes bestehen aus mehreren Seiten. Die einzelnen Seiten werden dynamisch miteinander verknüft und bilden somit einen grob funktionellen Prototypen. Man kann hier auch von „click-dummies“ sprechen.

MOCKUPS

Mockups sind Attrappen und ähneln dem fertigen Produkt schon etwas. Dies liegt an der Verwendung von richtiger Typographie, ersten Farben und teilweise schon korrekten Bildern, sollte es sich um eine Webseite handeln. Die Texte können Blindtexte in Form von *„Lorem Ipsum“* sein oder schon korrekte Texte.



MOCKUPS

In der Regel spiegeln Mockups die Designauswahl für Layouts, Farbschemas, Typografie, die Visualisierungen der Navigation und die Gesamtatmosphäre des Produkts wider.

Sie sind meist eine Darstellung des Erscheinungsbildes eines Produkts in mittlerer bis hoher Wiedergabequalität und zeigt Grundlagen seiner Funktionalität. Mockups füllen die visuellen Details (Typographie, Farben, Branding) aus und sind in der Regel statisch. Wenn man sich solch ein Modell ansieht, bekommt man eine gute Vorstellung davon, wie das Endprodukt aussehen wird und eine ungefähre Vorstellung davon, wie es funktionieren könnte. Auch wenn die Funktionen noch nicht funktionieren. Ein Mockup kann als ein hochwertiger visueller Entwurf angesehen werden.

PROTOTYP

Der Begriff Prototyping bezeichnet die Entwicklung eines Musters oder Prototyps im Bereich Software-Engineering. Prototyping kann als schrittweise Annäherung an das fertige Endprodukt betrachtet werden: Ein Prototyp wird im Laufe des Projekts zu einem fertigen Produkt – zum Beispiel zu einer Website, einer App oder einer komplexeren Software-Anwendung.

PROTOTYPING

Protoyping ist eine Vorgehensweise, die sehr früh Feedback von beteiligten Entwicklern und vor allem von den Endanwendern ermöglicht, indem ein starkes Augenmerk auf die Kommunikation während des Entwicklungs-Prozesses gelegt wird.

ARTEN DES PROTOTYPING

Allgemein können Methoden im Bereich Prototyping danach unterschieden werden, ob sie vertikal oder horizontal angelegt sind.

- Horizontales Prototyping fokussiert sich auf einen speziellen Bereich einer Anwendungssoftware – zum Beispiel auf die Benutzerschnittstelle. Dabei fehlt der Bezug zu den technischen Funktionalitäten des Gesamtsystems und deren Implementation. Ziel ist es, den Nutzer oder Auftraggeber mit der GUI in Berührung zu bringen und erstes Feedback zu erhalten.
- Vertikales Prototyping greift sich einen speziellen Bereich einer Software heraus und zeigt die Wechselwirkungen mit anderen Komponenten des Systems. Eine Benutzerschnittstelle würde hier bereits mit der Datenverwaltung und anderen Teilen des Systems gemeinsam abgebildet werden. Das Ziel besteht darin, komplexe Funktionalitäten zu erläutern und die Software in Teilen durch den Anwender überprüfen zu lassen.

ARTEN DES PROTOTYPING

Darüber hinaus lassen sich Prototyping-Ansätze mithilfe von Anwendungszwecken voneinander trennen.

Exploratives Protoyping: Das Anforderungsprofil einer Software wird Schritt für Schritt geklärt, indem Prototypen iterativ und schnell in einer Testumgebung erzeugt werden. Anschließend werden die Funktionalitäten verfeinert, um zu beurteilen, ob die Software das angenommene Problem löst und einen Nutzerbedarf abdeckt. In diesem Zusammenhang wird auch von Demonstratoren, Rapid sowie Paper Prototyping gesprochen. Demonstratoren werden in der Akquise und in frühen Phasen eines Projekts verwendet, um abstrakte Anforderungen und Problemstellungen bei der Entwicklung deutlich zu machen und zu kommunizieren.

ARTEN DES PROTOTYPING

Darüber hinaus lassen sich Prototyping-Ansätze mithilfe von Anwendungszwecken voneinander trennen.

Experimentelles Protoyping: Ein Entwurf wird mit den grundlegenden Funktionen erstellt und im Hinblick auf dessen Realisierbarkeit überprüft. Die gewonnenen Erkenntnisse dieses Experiments oder Tests fließen in das tatsächliche Produkt ein. Oft wird hier von Labormustern, Throw-Away-Protoypes gesprochen. Diese Muster sollen dabei helfen, technische Fragestellungen zu beantworten und das Projekt als solches auf dessen Realisierbarkeit hin überprüfen zu können.

ARTEN DES PROTOTYPING

Darüber hinaus lassen sich Prototyping-Ansätze mithilfe von Anwendungszwecken voneinander trennen.

Evolutionäres Protoyping: Die Software wird sukzessive erstellt. Bei jedem Entwicklungsschritt sorgen Feedbackschleifen mit Nutzern, Entwicklern und Auftraggebern dafür, dass das Endprodukt das Anforderungsprofil erfüllt. In der Regel wird eine Version der Software stets lauffähig gehalten. Hier wird auch von Pilotsystemen gesprochen. Sie beinhalten bereits eine Vielzahl der anvisierten Funktionen und können von Anwendern sorgfältig geprüft werden.

ARTEN DES PROTOTYPING

Die Wahl eines geeigneten Modells (vertikal, horizontal) und eines Anwendungszweckes ist im Einzelfall von vielen verschiedenen Faktoren abhängig. Das Budget, das Ziel des Projekts und die beteiligten Akteure (zum Beispiel externe Agenturen) bilden den Rahmen bei der Ausrichtung des Prototypings. In der Praxis können Modell und Anwendungszweck so gewählt werden, dass Mischformen der oben gemachten Unterscheidungen entstehen und einzelne Aspekte wie das Nutzer-Feedback besonders hervorgehoben werden.

ARCHITEKTURMODELL MVC

MODEL VIEW CONTROLLER

MODEL-VIEW-CONTROLLER MUSTER

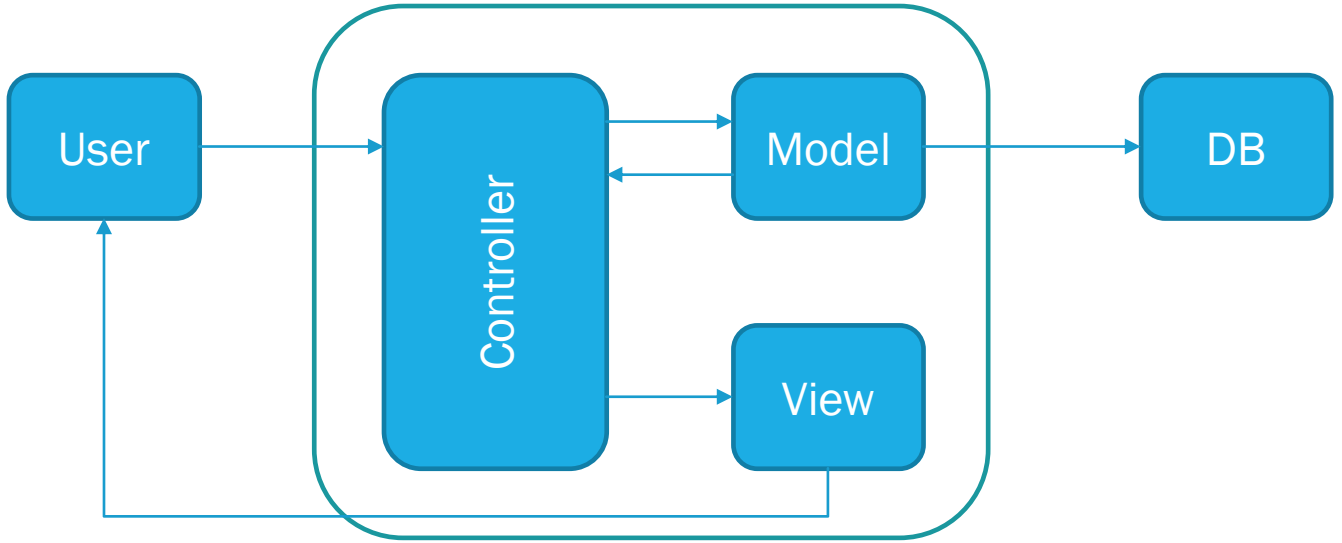
Anwendbar für interaktive Systeme

Aufteilung des Systems in drei Bereiche:

Model: Kernfunktionalität und Daten

View: Anzeige der Informationen

Controller: Verarbeitung der Nutzereingaben



MODEL-VIEW-CONTROLLER MUSTER

Problem:

- Benutzerschnittstellen unterliegen häufigen Änderungsbedarfen. Funktionserweiterungen erfordern Anpassungen der Menüs etc.. Nutzer verlangen nach verschiedenen Darstellungsmöglichkeiten desselben Inhaltes und möchten das "look and feel" auf ihre Bedürfnisse anpassen.

Kräfte/Forces

- Verschiedene Anzeigeformate für dieselben Informationen in verschiedenen Fenstern (z.B. Kreisdiagramm oder Balkendiagramm)
- Umgehende Verarbeitung von Datenänderungen
- Einfaches und Schnelles Ändern der Benutzerschnittstelle zur Laufzeit
- Anbieten verschiedener "look and feel"-Standards

MODEL-VIEW-CONTROLLER MUSTER

Lösung:

- Aufteilung in processing, output und input
- Model kapselt die Kerndaten und -funktionalität, ist unabhängig vom Output, dem Input und der Repräsentation
- View stellt dem Nutzer die Informationen zur Verfügung; holt sich die Daten vom Model; ein Model kann mehrere Views haben
- Controller empfängt Input, i.d.R. eventgetrieben
- Verändert der Benutzer das Model über einen Controller, zeigen alle Views die Änderungen unmittelbar an

MODEL-VIEW-CONTROLLER MUSTER

Vorteile:

- Verschiedene Views auf dasselbe Model
- Synchronisierte Views
- Nachträgliches Hinzufügen von weiteren Views
- Wechselbare Oberflächeneinstellungen

MODEL-VIEW-CONTROLLER MUSTER

Nachteile:

- Komplexität
- Abhängigkeit zwischen Controller und View