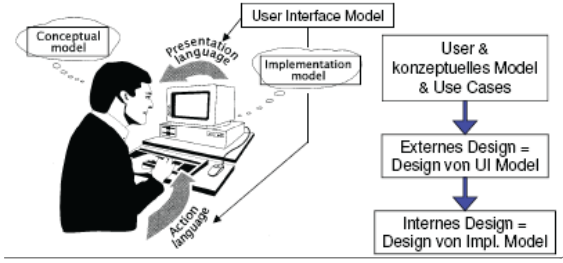
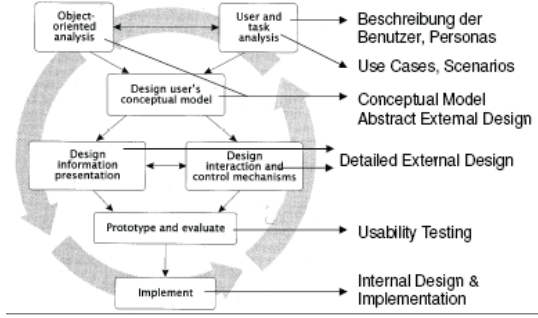
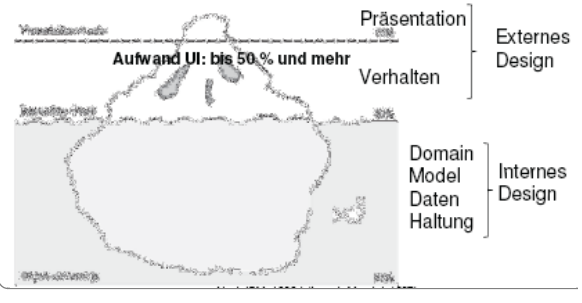
- Modelle
  - Interaktionen
    - Formen
      - Instructing (Tell system what to do)
      - Conversing (Have a dialogue with the system)
      - Manipulation (Interact with physical or virtual objects)
      - Exploration (Move through physical or virtual environment)
    - 7 Schritte der Interaktionen (Norman)
      - Forming the goal 1
      - Forming the intention 2
      - Specifying the action 3
      - Executing the action 4
      - Percieving the state of the world 5
      - Interpreting the state of the world 6
      - Evaluating the outcome 7
    - Direct Manipulation
      - + Novices can lern easily
      - + Immediate feedback
      - + Sense of mastery and control
      - - not always appropriate
    - Interaktions-Modell (Collins)



  - UI Design Prozess (Collins)



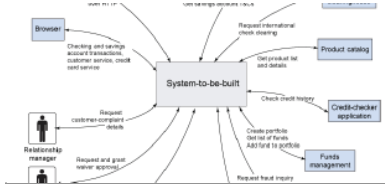
  - Begriffe
    - Model
      Struktur die eine Struktur in anderem Bereich erklärt, simuliert, vorhersagt und kontrolliert.
    - Metapher
      Analogie zwischen Unterschiedlichem, das eine erklärt das andere.
    - Kongruenz
      Modell, Metapher erklärt das Andere.
    - Inkongruenz
      Modell, Metapher erklärt nur einen Teil, stimmt in anderen Teilen nicht überein.
  - Gebrauchstauglichkeit
    - Nach ISO 9241-11
      - Effektivität
      - Effizienz
      - Zufriedenheit
    - Nach Queensbery
      - Effecive
      - Efficient
      - Engaging
      - Easy to Learn
      - Error Tolerant
  - Messbare Usability Kriterien
    - Time to learn
    - Speed of performance
    - Rate of errors by users
    - Retention over time
    - Subjective satisfaction
  - Iceberg Schichten Modell



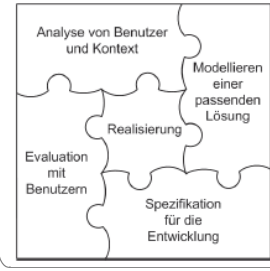
- External Design
  - konkret
    - Persona (Beispielhafter Benutzer) Nicht "durchschnittliche Benutzer"
      - Primary
      - Secondary
      - Pro Persona ein UI
    - Scenario (Fasst beobachtetes Benutzerverhalten zusammen)
  - abstrakt
    - Use Cases
      - Benutzerrollen
      - Ziele formulieren, nicht Handlungen beschreiben
      - Was beschreiben, nicht wie.
  - Kontext Diagramm
    - Ähnlich wie Domain Modell
  - User Environement Model
    - Zeigt Operationen und Objekte + Verbindungen
  - Konzeptionelles Modell
    - Zeigt Objekte + Verbindungen
    - Eigenschaften für ein gutes Konzeptionelles Modell
      - Minimieren Evaluationskluft
      - Minimieren Ausführungskluft
      - Bauen auf Bekanntem auf
      - Unterstützen Zielerreichung
  - Prototype (Paper, Visio Sketch, GUI Builder)
  - Layout Regeln
    - Balance
    - Symmetrie
    - Regularität
    - Vorhersehbarkeit
    - Sequenzialität
      - oben->unten, links->rechts, hell->dunkel, ...
  - Gutes UI Design ist wichtig weil: ⭐
    - Gutes UI kommt nicht von selbst
    - Wahrgenommene Qualität ist stark durch das UI geprägt
    - Usability ist von zentraler Bedeutung für Qualitätskriterien
    - Hersteller können für schlechte Usability haftbar gemacht werden
    - Hersteller können auf einen "UserCentred Design Prozess" verpflichetet werden
  - Use-Case Matrix

Use Case \ Entity	Order	Chemical	Requester	Vendor Catalog
Place Order	C	R	R	R, L
Change Order	U, D		R	R, L
Manage Chemical Inventory		C, U, D		
Report on Orders	R	R, L	R, L	
Edit Requesters			C, U, L	

  - System Kontext Diagram (Use-Case Diagram) Zeigt Rollen & Use-Cases in der Übersicht



  - ⭐ 12 Schritte des GUI Designs (Galitz)
    1. Know your user or client
    2. Understand the business function
    3. Understand the principles of good screen design
    4. Select the proper kinds of windows
    5. Develop system menus
    6. Select the proper device-based controls
    7. Choose the proper screen-based controls
    8. Organize and lay-out windows
    9. Choose the proper colors
    10. Create meaningful icons
    11. Provide effective messages, feedback, ...
    12. Test , test and retest
  - Gestaltungsregeln (Galitz)
    - Each screen element (control, icon, color, animations, messages, ...) must have meaning to users
  - Good Design (Galitz)
    - Reduce visual work
    - Reduce intellectual work
    - Reduce memory work
    - Reduce motor work
    - Reduce "management" work
  - Idiom: Browse Idiom, Add-and-Remove Idiom, Row Sorting
  - Auswahl von Widgets: Radio Button, Check Box, Spin Button, Slider, Drop-down List Box
  - Gadgets
  - 8 golden Rules (Schneiderman)
    - Strive for consistency
    - Enable frequent users to use shortcuts
    - Offer informative feedback
    - Design dialogs to yield closure
    - Offer error prevention and simple error handling
    - Permit easy reversal of actions
    - Support internal locus of control
    - Reduce short-time memory load
  - Vorgehensmodell Usability Kompakt



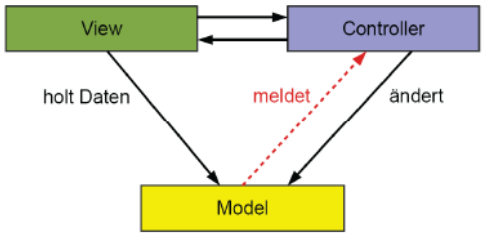
Analyse
- Aktivität: Benutzer Beobachtung
- Aktivität: Workshops

Modellieren
- Ergebnis: Personas
- Ergebnis: Szenarios
- Ergebnis: Low-Fi GUI Design

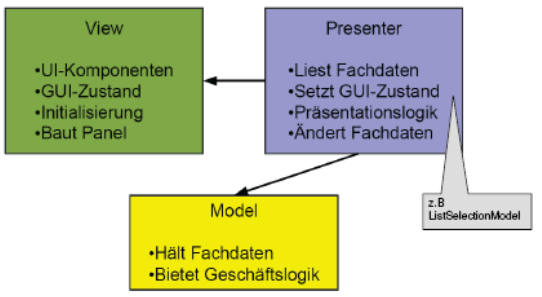
Spezifikation
- Ergebnis: Use Cases, Feature List
- Ergebnis: Requirements Document
- Ergebnis: Detailed GUI Design

Realisierung
- Ergebnis: Internal Design

Evaluation



- Internal Design
  - MVC Model View Controller
    - In Swing ist der Controller häufig in der View
    - Jeder Listener kann als Controller im Kleinen angesehen werden



  - MVP Model View Presenter
  - Distributed Systems
  - 3 Schichten-Architektur
    - Presentation (User Interface)
    - Domain (Problem Domain)
    - Persistance (Datenhaltung)
  - 2 Schichten-Architektur
    - User Interface
    - Problem Domain
  - Event Handling
    - Anonymer listener bearbeitet den Event
    - Callback, anonymer listener ruft Methode der View auf
    - Eigene abgeleitete listener Klasse
  - Observer Pattern

```
Subject = extends Observable
  import
  attach = addObserver
  detach = deleteObserver
  notify = setChanged & notifyObservers
Observer = implements Observer
  import
  anmelden = addObserver(this)
  update(...)
```

  - Code Beispiele

```
semesterComboBox.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        semesterComboBoxActionPerformed(evt);
    }
});
```

```
searchTextField.getDocument().addDocumentListener(new DocumentListener(){
    public void changedUpdate(DocumentEvent e){
        searchTextFieldChanged();
    }
    public void insertUpdate(DocumentEvent e){
        searchTextFieldChanged();
    }
    public void removeUpdate(DocumentEvent e) {
        searchTextFieldChanged();
    }
});
```