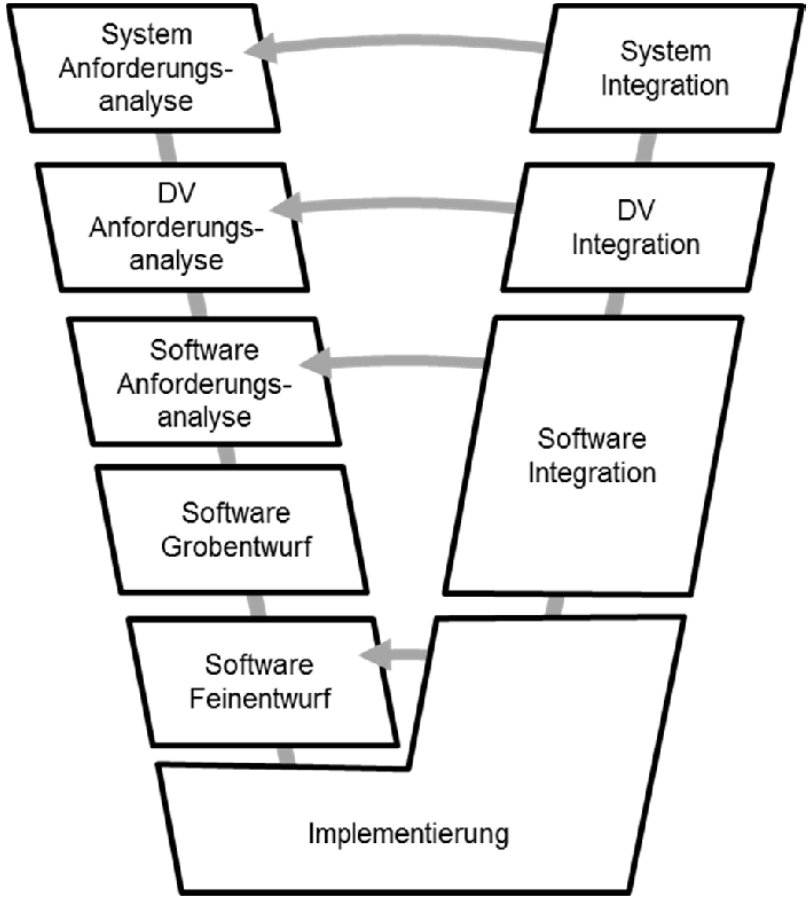
# Kapitel 2

# Einführung in die Programmierung

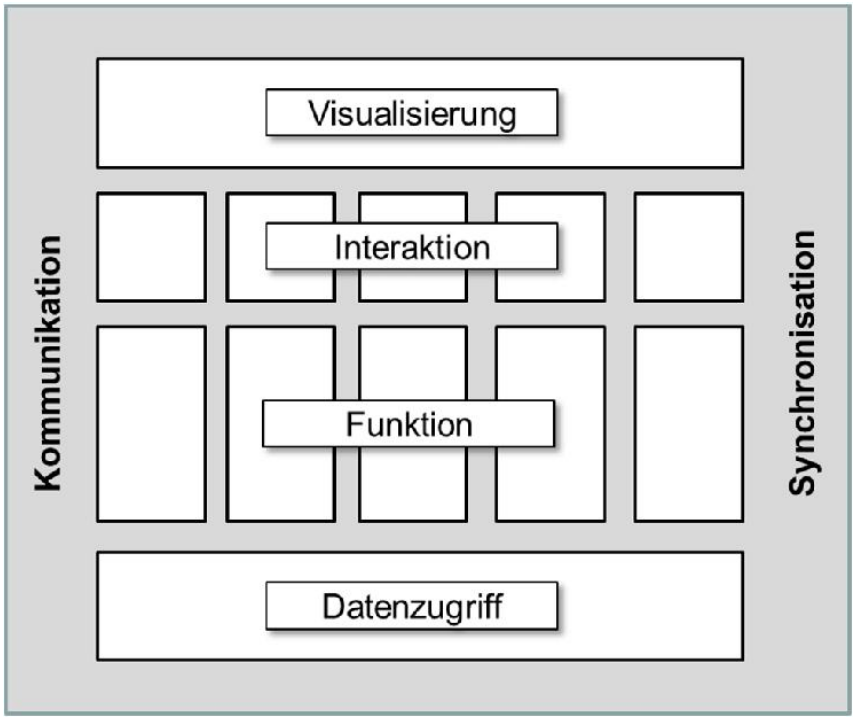
## Softwareentwicklung ist mehr als Programmierung

Es gibt wesentliche, der Programmierung vorgelagerte (z.B. **Anforderungsanalyse**, **Systementwurf**), die Programmierung begleitende (z. B. **Test**, **Qualitätssicherung**) und der Programmierung nachgelagerte (z.B. **Integration**, **Wartung**) Aktivitäten der Softwareentwicklung.

Beispiel: V-Modell



Prinzipieller Aufbau eines Softwaresystems:



Auf der Ebene der **Visualisierung** werden die Elemente der Benutzerschnittstelle (Dialoge, Menüs), aber auch Grafikfunktionen bereitgestellt. Angestrebt wird eine konsequente Trennung von »Form« und »Inhalt«. Das Layout der Benutzerschnittstelle wird getrennt von der eigentlichen Funktionalität des Systems.

Unter **Interaktion** sind Funktionen zusammengefasst, die die Steuerung der Benutzerschnittstelle ausmachen. In der Regel werden die Funktionen zur Interaktion über den Benutzer (Mausklick auf einen Button etc.) angestoßen und vermitteln dann zwischen den Benutzerwünschen und den eigentlichen **Funktionen** des Anwendungssystems. Häufig zerfällt ein System in unabhängige, vielleicht sogar parallel laufende Module, die auf einem gemeinsamen Datenbestand arbeiten.

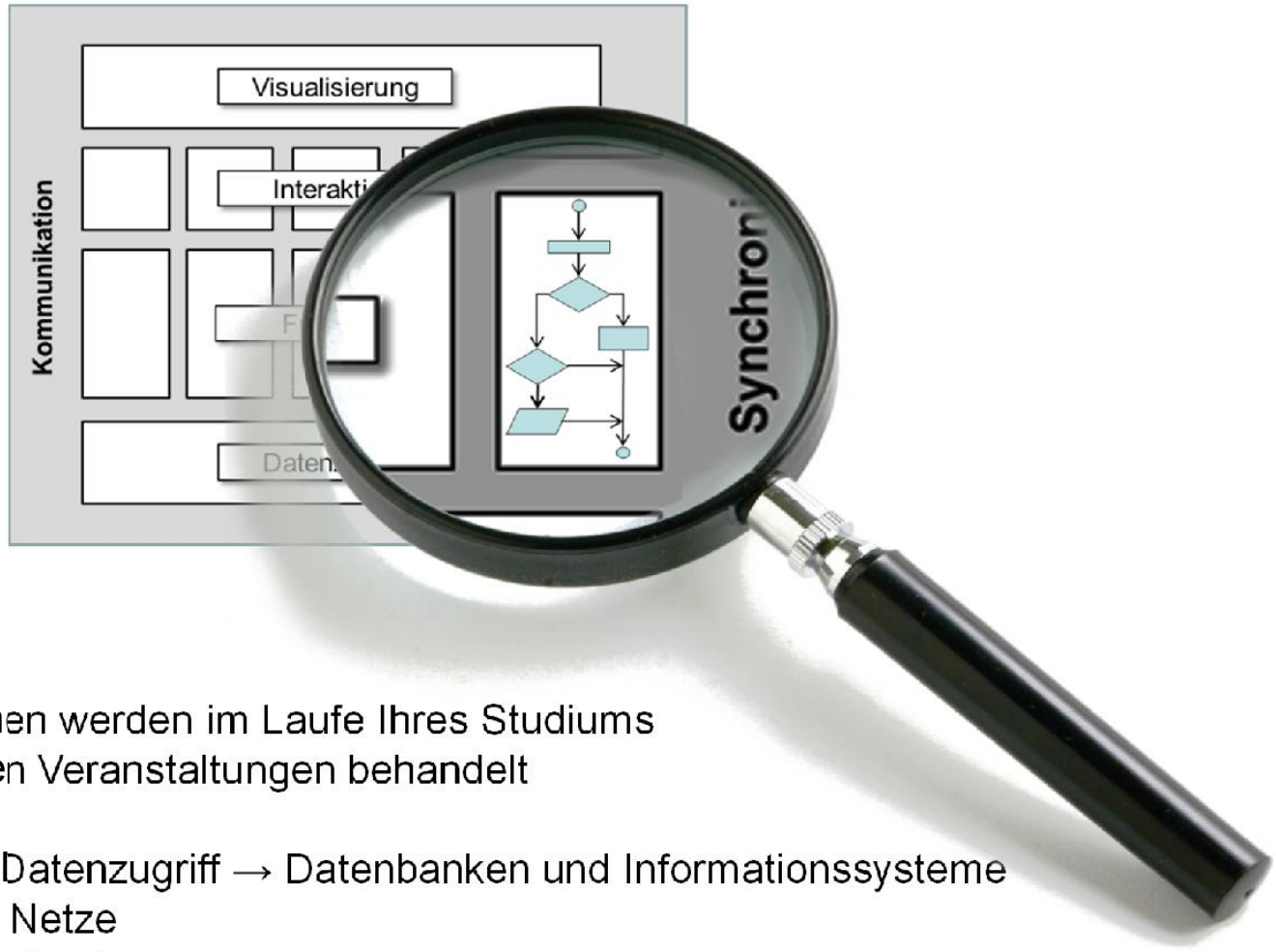
Die **Datenhaltung** und der **Datenzugriff** werden häufig in einer übergreifenden Schicht vorgenommen, denn hier muss sichergestellt werden, dass unterschiedliche Funktionen trotz konkurrierenden Zugriffs einen konsistenten Blick auf die Daten haben. Bei großen Softwaresystemen kommen Datenbanken mit ihren Management-Systemen zum Einsatz. Diese verfügen über spezielle Sprachen zur Definition, Abfrage, Manipulation und Integritätssicherung von Daten.

Unterschiedliche Teile eines Systems können verteilt in einem lokalen oder weltweiten Netz laufen. Wir sprechen dann von einem »verteilten System«.

Unter dem Begriff **Kommunikation** werden Funktionen zum Datenaustausch zwischen verschiedenen Komponenten eines verteilten Systems zusammengefasst.

Mittels Funktionen zur **Synchronisation** werden parallel arbeitende Systemfunktionen, etwa bei konkurrierendem Zugriff auf Betriebsmittel, koordiniert.

Der Fokus dieser Veranstaltung liegt auf dem Bereich **Funktion**



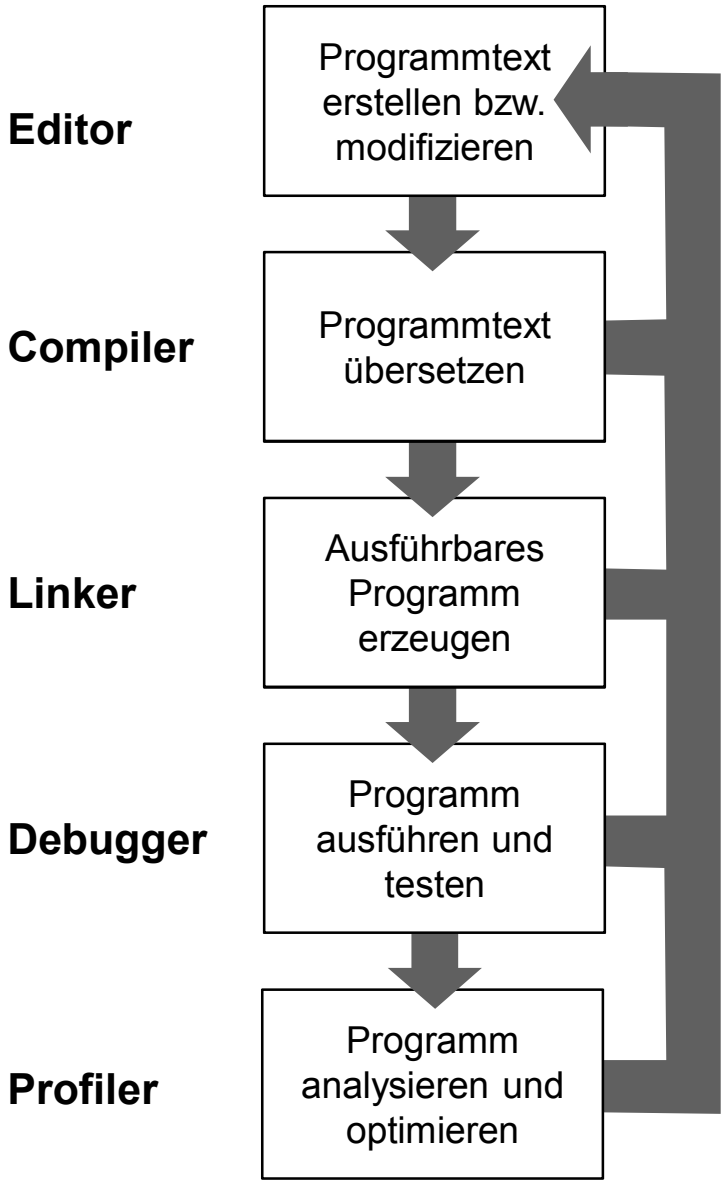

Die anderen Themen werden im Laufe Ihres Studiums
in unterschiedlichen Veranstaltungen behandelt

Datenhaltung und Datenzugriff → Datenbanken und Informationssysteme
Kommunikation → Netze
Synchronisation → Betriebssysteme
Visualisierung und Interaktion → Programmierung grafischer Benutzeroberflächen
Softwareentwicklung allgemein → Softwaretechnik

**Programmierwerkzeuge**:

Die Werkzeuge sind in eine Entwicklungsumgebung (IDE = Integrated Development Environment) integriert.

Die Entwicklungsumgebung und die Arbeit mit den Werkzeugen der Entwicklungsumgebung lernen Sie im begleitenden **Praktikum** kennen.



Mit dem **Editor** als Werkzeug erstellen wir unseren Programmcode, den wir in Dateien ablegen. Im Zusammenhang mit der C-Programmierung sind dies:

- Header-Dateien und
- Quellcode-Dateien

**Header-Dateien** (engl. Headerfiles) sind Dateien, die Informationen zu Datentypen und Datenstrukturen, Schnittstellen von Funktionen etc. enthalten. Es handelt sich dabei um allgemeine Vereinbarungen, die an verschiedenen Stellen (d. h. in verschiedenen Source- und Headerfiles) einheitlich und konsistent benötigt werden. Headerfiles stehen im Moment noch nicht im Mittelpunkt unseres Interesses. Spätestens mit der Einführung von Datenstrukturen werden wir jedoch die große Bedeutung dieser Dateien erkennen.

Die **Quellcode-Dateien** (engl. Sourcefiles) enthalten den eigentlichen Programmtext und stehen für uns zunächst im Vordergrund.

Den Typ (Header oder Source) einer Datei können Sie bereits am Namen der Datei erkennen. Header-Dateien sind an der Dateinamenserweiterung ».h«, Quellcode-Dateien an der Erweiterung ».c« in C beziehungsweise ».cpp« oder ».cc« in C++ zu erkennen.