Design-Erwägungen

- C gehört zur Gruppe **imperativer** Programmiersprachen.

  Diese sind eine Umsetzung der von-Neumann-Architektur: Befehle sind im selben Medium gespeichert wie die Daten, die sie bearbeiten.

  Im Gegensatz dazu beschreiben („deklarieren“) d**eklarative** (funktionale, logikbasierte oder regelbasierte) Sprachen ein gewünschtes Ergebnis, dessen konkrete Verarbeitungsschritte von einem Compiler oder Interpreter abgeleitet und angesteuert werden.
- Die Gruppe imperativer Programmiersprachen umfaßte nacheinander die **maschinenorientierten** Sprachen (z.B. Assembler), die **prozeduralen** Sprachen (FORTRAN, ALGOL, C, Pascal) und die **objektorientierten** Sprachen (z.B. Smalltalk, EIFFEL, C++, Java, C#).

Design-Erwägungen

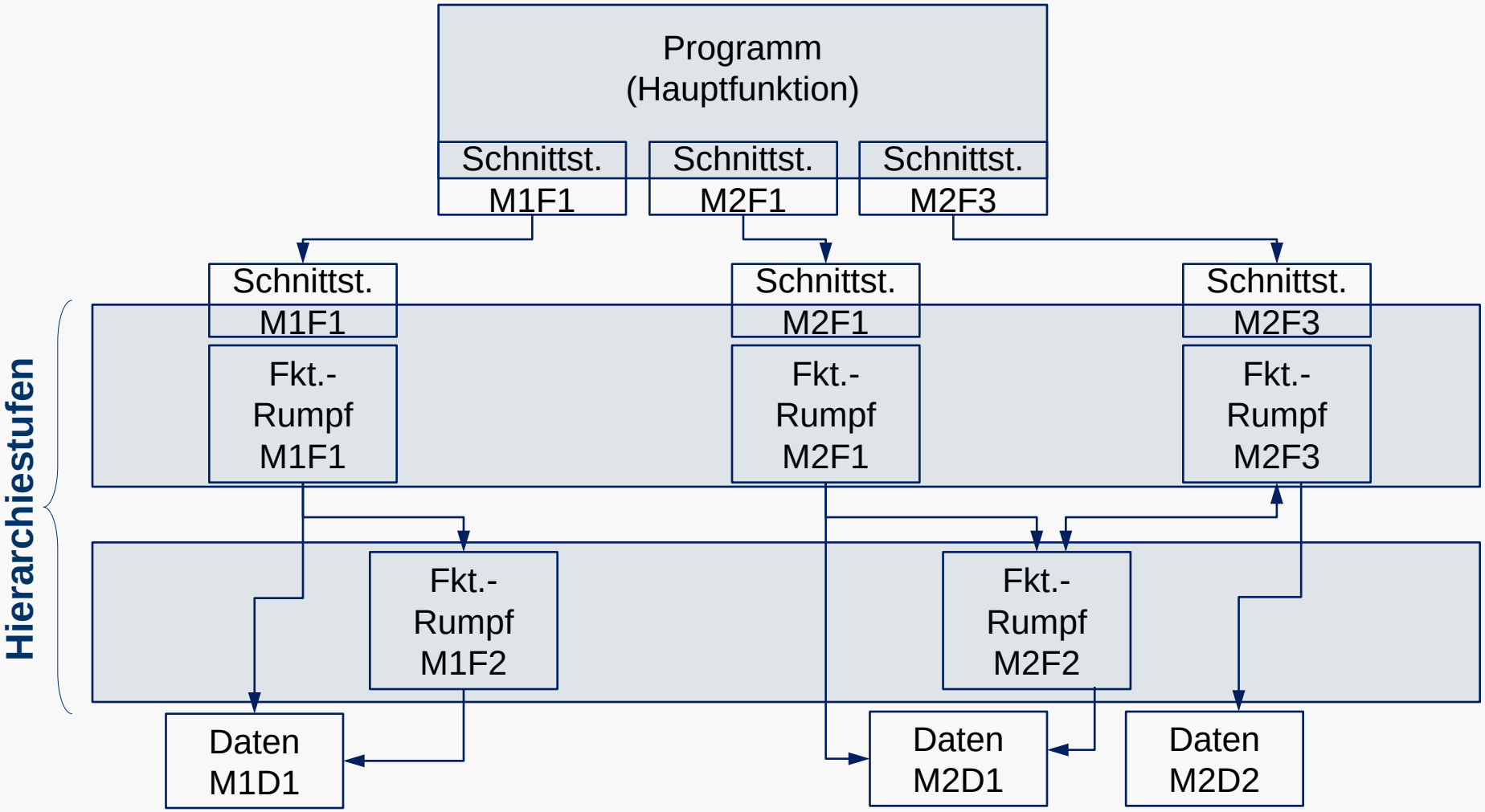
- **Imperativen**, **prozeduralen** Sprachen (wie C) liegt als Konzept die Programmierung von Prozeduren zugrunde –

  von Befehlsfolgen zur Lösung wiederkehrender Aufgaben, die voneinander unabhängig unter einem (Funktions-) Namen eingesetzt (aufgerufen) werden können.
- **Strukturiertes Softwaredesign** (*structured)* liegt vor, wenn ein Programm in **Funktionen** (Unterprogramme) zerlegt wird, die zueinander möglichst wenige Querbeziehungen haben. Dadurch entsteht in der Software eine Aufrufhierarchie der Funktionen.

  Man spricht vom **„Programmieren im Kleinen“**.
- Für größere Softwarepakete („Programmieren im Großen“ – z.B. Plattformen) eignet sich besser ein **modulares Softwaredesign**, bei dem **Funktionen und Daten** zu größeren Einheiten (Modulen) zusammengefaßt werden.

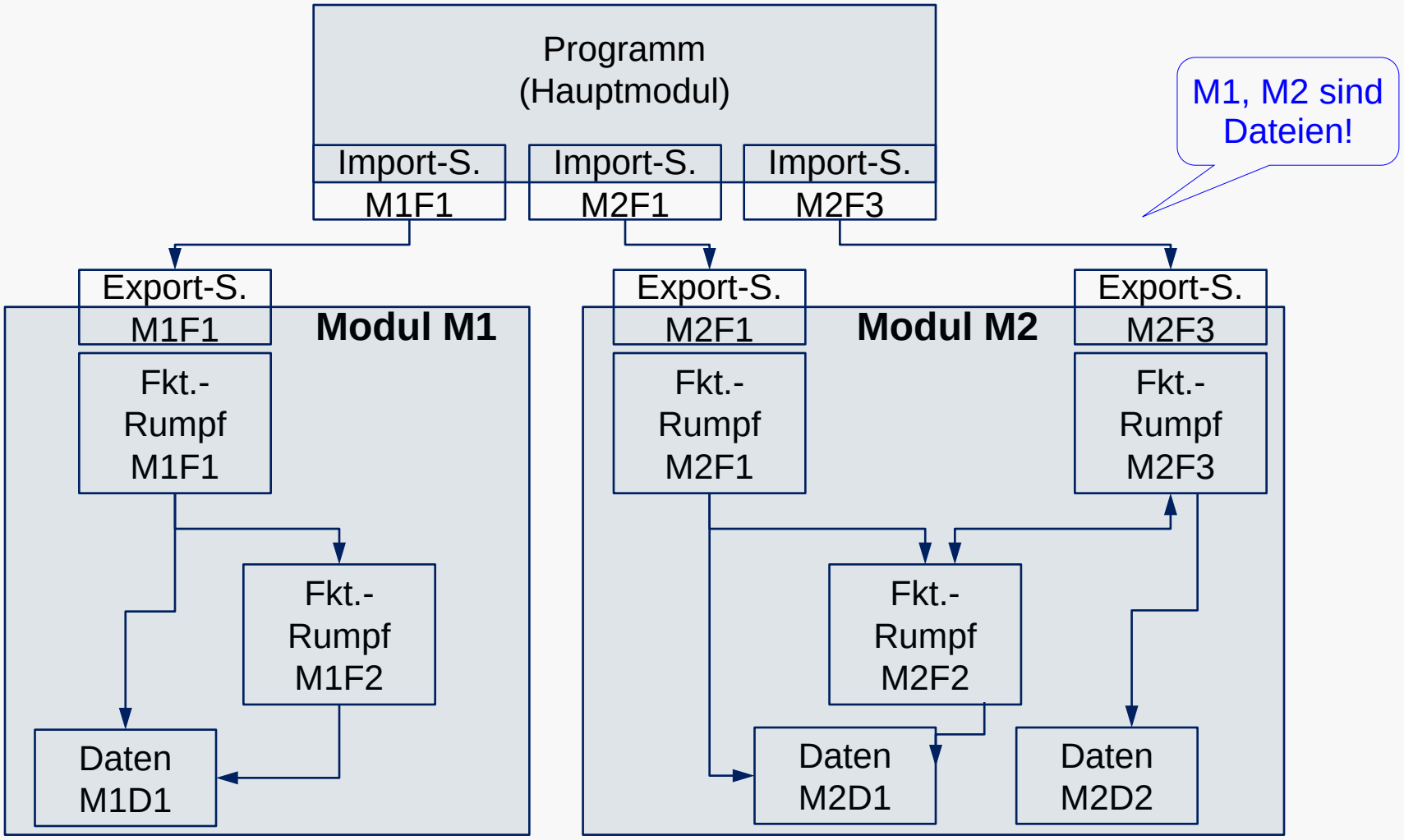
Design-Erwägungen

Schema strukturiertes Softwaredesign:



Design-Erwägungen

Schema modulares Design mit Import-/ Export-Schnittstellen:



Design-Erwägungen

Modulares Softwaredesign wird als Vorstufe der Objekt-orientierung betrachtet: Es faßt Funktionen und Daten zu eigenständigen Einheiten zusammen (Kapselung) und

- erhöht Abstraktionsniveau (Black-Box)
- erleichtert Austauschbarkeit/ Versionsverwaltung, solange die (Export-/Import-) Schnittstellen eingehalten werden
- verbirgt interne Struktur (Information Hiding / Copyright)
- verbessert Prüfbarkeit (ggf. in unterschiedlichem Kontext);

letztere ist bei Implementierung größerer Softwarepakete (z.B. Fenstersysteme) zusätzlich erschwert durch die vielfach benötigte **Wiedereintrittsinvarianz** (engl. *reentrancy*):

Reentrante (z.B.: Fenster-) Funktionen erlauben mehrfache, geschachtelte Aufrufe (die keine Rekursion darstellen).

SPprintf-Aufruf
- aus redisplay
- aus SPprintf