Physikalische Chemie, Universität Rostock

Vorlesung: Wissenschaftliche Softwareentwicklung

Wintersemester 2024/25

Dr. habil. Till Biskup

— Glossar zu Lektion 30: „Datenpräsentation: Darstellungs- und Berichterstellung“ —

Hinweis: Die nachfolgend genannten Begriffe und Definitionen erheben keinen Anspruch auf formale Korrektheit, sondern dienen lediglich dem besseren Verständnis der in der Vorlesung behandelten Themen und sind im jeweiligen Kontext zu sehen. Mehrfache, voneinander abweichende Definitionen in unterschiedlichen Kontexten sind daher möglich. Englische Begriffe werden zwar nach Möglichkeit übersetzt, erscheinen aber ggf. unter ihrem englischen Namen in der Liste. Verweise untereinander sind durch ↑ gekennzeichnet.

Abhängigkeit *dependency*, im Quellcode durch explizite Nennung hervorgerufene ↑Kopplung von Programmteilen (↑Funktionen, ↑Objekte, …), die dazu führt, dass der aufgerufene Programmteil nicht mehr ohne Veränderung des aufrufenden Teils verändert werden kann.

Abstraktion Nach Edsger Dijkstra [1] das einzige mentale Werkzeug, das es erlaubt, eine große Vielzahl von Fällen abzudecken. Zweck der Abstraktion ist es nicht, vage zu sein, sondern im Gegenteil ein neues Bedeutungsniveau zu schaffen, das präzise Beschreibungen erlaubt.

Aggregation *aggregation*, „klassische“ Form der ↑Zusammensetzung in der ↑objektorientierten Programmierung, definiert eine klare „*hat ein*“-Beziehung

ANSI *American National Standards Institute*, private, gemeinnützige, amerikanische Organisation zur Koordinierung der Entwicklung freiwilliger Normen in den U.S.A.

Assoziation *association*, Interaktion von ↑Objekten/↑Klassen auf die Weise, dass ein Objekt/eine Klasse einen Service für ein anderes Objekt/eine andere Klasse bereitstellt.

Attribut im Kontext der ↑objektorientierten Programmierung eine Variable, die innerhalb einer ↑Klasse definiert wird. ↑Methoden operieren auf den Attributen einer ↑Klasse bzw. dem daraus erzeugten ↑Objekt.

Auszeichnungssprache *markup language* (ML) maschinenlesbare Sprache für die Gliederung und Formatierung von Texten und anderen Daten. Der bekannteste Vertreter ist die *Hypertext Markup Language* (↑HTML), die Kernsprache des World Wide Webs.

Clean Code „sauberer Code“, letztlich lesbarer Code, der insbesondere im Kontext der naturwissenschaftlichen Datenauswertung die essentiellen Kriterien von Wiederverwendbarkeit, Zuverlässigkeit und Überprüfbarkeit erfüllt.

CSV *comma-separated values*, ↑Datenformat für zeilenweise Speicherung zusammengehöriger Daten, ähnlich der Zeilen in einer Tabelle. Die einzelnen Felder in einer Zeile werden oft durch Komma (daher der Name), ggf. aber auch durch Semikolon getrennt. Im Gegensatz zu ↑DSV wurde CSV nie standardisiert, weshalb es Tabellenkalkulationen großer Hersteller gibt, die zwar CSV exportieren, aber den eigenen Export nicht mehr importieren können. Insbesondere der Umgang mit Feldtrennern innerhalb eines Feldes ist nicht festgelegt.

Datenformat digitales Speicherformat für Daten jeglicher Form. Grundsätzlich werden bi-

näre und Textformate unterschieden. Während erstere meist mit deutlich geringerem Speicherbedarf auskommen, sind sie im Gegensatz zu letzteren nicht ohne Hilfsmittel lesbar. Textformate hingegen sind, ein beliebiger Texteditor vorausgesetzt, prinzipiell menschenlesbar. Wichtige Vertreter binärer Datenformate in den Naturwissenschaften sind ↑HDF5 und ↑IEEE 754 (eigentlich ein Standard für die Darstellung von Gleitkommazahlen). Wichtige Vertreter von Textformaten sind ↑CSV, ↑DSV, ↑JSON, ↑Windows-INI, ↑XML und ↑YAML.

Dependency Inversion Umkehr der ↑Abhängigkeiten gegenüber der intuitiven Implementierung. Abhängigkeiten sollten häufig entgegen dem ↑Kontrollfluss verlaufen.

Dependency-Inversion-Prinzip (DIP) Anwendung der ↑Dependency Inversion: Abstraktionen sollten nicht von Details abhängen. Umgekehrt sollten Details auf Abstraktionen aufbauen.

DIP ↑Dependency-Inversion-Prinzip, vgl. ↑SOLID

DokuWiki freie, quelloffene ↑Wiki-Software, die in PHP geschrieben ist, eine hierarchische Organisation der Inhalte erlaubt und als Textdateien mit zugehörigen Medien direkt in einer Verzeichnisstruktur abbildet. Der Verzicht auf eine Datenbank und die Verwendung des Dateisystems macht DokuWiki sehr flexibel und robust.

DSV *delimiter-separated values*, ↑Datenformat für zeilenweise Speicherung zusammengehöriger Daten, ähnlich der Zeilen in einer Tabelle. Im Gegensatz zu ↑CSV ist das Format eindeutig definiert. Das Trennzeichen (*delimiter*) ist beliebig wählbar. Sollte es innerhalb eines Feldes auftauchen, wird es durch Backslash („\“) geschützt.

Entwurfsmuster *design patterns*, erprobte und bewährte Lösungen für wiederkehrende Probleme in der Softwareentwicklung. Beschreibungen miteinander kommunizierender ↑Objekte und ↑Klassen, die maßgeschneidert sind, um ein generelles Entwurfsproblem in einem bestimmten Kontext zu lösen. [2, S. 3] Entwurfsmuster liefern eine (↑abstrakte) Beschreibung des dahinterstehenden Konzepts und haben meist einen etablierten Namen, der die Kommunikation erleichtert. Es gibt ganze Kataloge solcher Muster, und viele der ursprünglich beschriebenen Entwurfsmuster sind heute in vielen Programmiersprachen fest etabliert.

Funktion im Kontext der strukturierten Programmierung eine Liste von Anweisungen, die eine bestimmte Aufgabe erfüllt und der Programmiersprache unter einem festen Namen bekannt ist. Vgl. ↑Methode.

GoF *Gang of Four*, „Viererbande“, Bezeichnung für die Autoren des Werkes „*Design Patterns. Elements of Reusable Object-Oriented Software*“ [2], namentlich Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides. Das Erscheinen dieses Buches gilt als eigentliche Geburtsstunde der ↑Entwurfsmuster in der Softwareentwicklung.

größeres Projekt hier: Alles, was mehr als zwei Wochen Arbeit kostet und deutlich mehr als zweihundert Zeilen (reinen) Quellcode bzw. mehr als eine Handvoll Unterfunktionen umfasst. Wichtig ist der Fokus: Sobald ein Programm über längere Zeit und/oder von anderen verwendet werden soll (was eher die Regel statt die Ausnahme ist), ist es ein größeres Projekt.

HDF5 *Hierarchical Data Format*, ↑Datenformat, das insbesondere in wissenschaftlichen Anwendungen für die Speicherung großer Datenmengen verwendet wird. Optimierte Strukturen und Algorithmen erlauben das effiziente Speichern und Auslesen von ein- und mehrdimensionalen Tabellen, ohne dass jeweils der komplette Inhalt der Datei in den Speicher geladen werden muss. Tabellen und andere Daten können in ein und derselben Datei in einer beliebigen Verzeichnisstruktur abgelegt werden. Das ermöglicht u.a. die gleichzeitige Speicherung von Messwerten und zugehörigen ↑Metadaten. Das Format wurde vom *Natio-*

nal Center for Supercomputing Applications (NCSA) entwickelt und wird u.a. von der NASA für Missionen verwendet.

HTML *Hypertext Markup Language*, textbasierte ↑Auszeichnungssprache zur Strukturierung elektronischer Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten. HTML-Dokumente sind die Grundlage des World Wide Web und werden von Webbrowsern dargestellt. Neben den vom Browser angezeigten Inhalten können HTML-Dateien zusätzliche Angaben in Form von Metainformationen enthalten, z. B. über die im Text verwendeten Sprachen, den Autor oder den zusammengefassten Inhalt des Textes.

IEEE *Institute of Electrical and Electronics Engineers*, weltweiter Berufsverband von Ingenieuren hauptsächlich aus den Bereichen Elektrotechnik und Informationstechnik. Der Verband veranstaltet Fachtagungen, gibt diverse Fachzeitschriften heraus und bildet Gremien für die Standardisierung von Techniken, Hardware und Software.

IEEE 754 Norm des ↑IEEE, die die Standarddarstellungen für binäre Gleitkommazahlen in Computern definiert und genaue Verfahren für die Durchführung mathematischer Operationen, insbesondere für Rundungen, festlegt. In der Fassung ↑ANSI/IEEE 754-1985 ist nur der Standard für 64 Bit eindeutig.

Infrastruktur personelle, sachliche und finanzielle Ausstattung, um ein angestrebtes Ziel zu erreichen. Im Kontext der Softwareentwicklung die Gesamtheit der Hilfsmittel, die (manche) Abläufe formalisieren und für Struktur und Überprüfbarkeit sorgen. Erleichtert die Arbeit des Programmierers, indem sie viele Aspekte festlegt, die so zur Routine werden (und keine Denkleistung absorbieren).

Interface Segregation Aufteilung der ↑Schnittstelle einer ↑Klasse oder eines ↑Moduls mit dem Ziel möglichst geringer ↑Kopplung und hoher ↑Kohäsion.

Interface-Segregation-Prinzip (ISP), Anwendung der ↑Interface Segregation: Nutzer sollten nicht dazu gezwungen werden, von Methoden abzuhängen, die sie nicht verwenden.

ISP ↑Interface-Segregation-Prinzip, vgl. ↑SOLID

Iteration eine von zwei Kontrollstrukturen der ↑strukturierten Programmierung, neben der ↑Selektion. Meist als Schleife implementiert, die über alle Elemente einer Liste läuft und für jedes Element bestimmte Anweisungen ausführt.

JSON *JavaScript Object Notation*, hierarchisches ↑Datenformat, das ursprünglich zur einfachen Persistierung (↑Persistenz) von JavaScript-Objekten entwickelt wurde. Heute erfreut es sich als Austauschformat für Objekte und Datenstrukturen großer Beliebtheit, wird aber gerade für die Ablage von Konfigurationen in menschenlesbaren (und schreibbaren) Dateien aufgrund dessen noch einfacherer und übersichtlicherer Syntax zunehmend von ↑YAML abgelöst.

Kapselung *encapsulation*, ein ↑Objekt enthält Daten (↑Attribute) *und* zugehöriges Verhalten (↑Methoden) und kann beides nach Belieben vor anderen Objekten verstecken.

Klasse *class*, im Kontext der ↑objektorientierten Programmierung die Blaupause für die Erzeugung eines ↑Objektes; Definition der Daten (↑Attribute) und des zugehörigen Verhaltens (↑Methoden).

Kohäsion *cohesion*, innerer Zusammenhalt; hier: Zusammenhang einzelner Aspekte einer Softwareeinheit zueinander. Ein Ziel der Softwareentwicklung ist starke Kohäsion (*strong cohesion*). Jede Einheit (z.B. ↑Methode, ↑Funktion, ↑Klasse) hat eine Aufgabe, und alle Teile dieser Einheit dienen dem Zweck, diese eine Aufgabe zu erfüllen.

Kontrollfluss *flow of control*, Reihenfolge des Aufrufs von Programmteilen (↑Funktionen, ↑Objekte, ...), um eine gegebene Aufgabe zu erfüllen.

Kopplung *coupling*, in Software der Grad der Verbindung zweier Komponenten; enge Bindung

mehrerer Einheiten einer Software aneinander, so dass sie nicht unabhängig wiederverwendbar (bzw. ggf. auch nicht testbar) sind. Programmierkonzepte zielen generell auf eine lose Kopplung (*loose coupling*) einzelner Komponenten ab, da so die Wiederverwendbarkeit erleichtert wird.

LaTeX auf dem Textsatzsystem ↑TeX aufbauendes Makropaket, das seit 1980 von Leslie Lamport entwickelt wurde (LaTeX = Lamport TeX) und dem Anwender erlaubt, TeX komfortabel zu verwenden. Folgt dem Konzept der logischen Textauszeichnung: Elemente werden hinsichtlich ihrer Funktion markiert (z.B.: Überschrift, Zitat), die zugehörige Formatierung wird zentral in einer Stildatei (Dokumentklasse, Zusatzpaket) festgelegt. Die Verwendung reiner Textdateien als Eingabe und die Trennung von logischer Textauszeichnung und Formatierung resultiert in einer großen Flexibilität, Plattformunabhängigkeit und Stabilität. Nahezu beliebig große Dokumente lassen sich (ggf. weitgehend automatisiert) erzeugen.

Liskov-Substitution Einsatz von Subtypen anstelle ihrer Basistypen ohne Beeinträchtigung der Funktionalität.

Liskov-Substitutionsprinzip Anwendung der ↑Liskov-Substitution: Subtypen müssen durch ihre Basistypen ersetzbar sein. Grundlegendes Prinzip für die ↑Vererbung in der ↑objektorientierten Programmierung, das auf Barbara Liskov [3] zurückgeht.

LSP ↑Liskov-Substitutionsprinzip, vgl. ↑SOLID

Markdown vereinfachte ↑Auszeichnungssprache, deren Ziel die einfache Lesbarkeit der Ausgangsform ohne weitere Konvertierung ist. Als Auszeichnungselemente wurden daher vor allem Auszeichnungsarten verwendet, die in reinem Text und E-Mails üblich sind.

Markup-Format Format einer ↑Auszeichnungssprache

Mehrfachvererbung *multiple inheritance*, eine ↑Klasse erbt (↑Vererbung) von mehr als einer ↑Superklasse. Wird von den wenigsten Programmiersprachen unterstützt, oftmals behilft man sich hier aber des Konzeptes einer ↑Schnittstelle (*interface*) (3.) und kann dann mehr als eine solche implementieren (bzw. davon erben). Konzeptionell lassen sich diese beiden Ansätze quasi identisch einsetzen.

Metadaten Informationen zu den numerischen Daten, notwendige Voraussetzung für eine sinnvolle Verarbeitung der Daten im Kontext eines ↑Systems zur Datenverarbeitung und für ↑reproduzierbare Wissenschaft.

Methode im Kontext der ↑objektorientierten Programmierung eine ↑Funktion, die innerhalb einer ↑Klasse definiert wird und auf den ↑Attributen einer ↑Klasse bzw. dem daraus erzeugten ↑Objekt operiert.

Modularisierung Aufteilung der Gesamtaufgabe in kleinere Abschnitte. Die Aufteilung wird so lange fortgesetzt, bis die Lösung für den aktuellen Abschnitt unmittelbar in Form von Quellcode offensichtlich ist. Setzt die Definition von ↑Schnittstellen voraus.

monolithisch aus einem Stück bestehend; zusammenhängend und fugenlos

OASIS *Organization for the Advancement of Structured Information Standards*, internationale, nicht-gewinnorientierte Organisation, die sich der Weiterentwicklung von E-Business- und Webservice-Standards widmet. Zu den bekannten Standards der OASIS gehören OpenDocument (↑ODF) und DocBook.

Objekt *object*, im Kontext der ↑objektorientierten Programmierung der grundlegende Baustein eines Programms, bestehend aus den Daten (↑Attribute) und dem zugehörigen Verhalten (↑Methoden). Ein Objekt ist in diesem Kontext immer die Instanz einer ↑Klasse.

objektorientierte Programmierung (OOP) ein ↑Programmierparadigma, bei dem Daten (Variablen zugewiesene Werte, als ↑Attribute bezeichnet) und Funktionen (↑Methoden), die auf diesen Daten (Attributen) operieren, eine Einheit bilden. Die in den ↑Attributen gespeicherten Daten lassen sich i.d.R. nur vermittelt

durch (öffentlich zugängliche) ↑Methoden der ↑Klasse bzw. des daraus erzeugten ↑Objektes ansprechen. Es gibt eine klare Trennung zwischen öffentlicher ↑Schnittstelle und internen Verarbeitungsroutinen. Wichtige Vertreter objektorientierter Programmiersprachen sind Smalltalk, C++ und Java, aber auch Python.

OCP ↑Open-Closed-Prinzip, vgl. ↑SOLID

ODF *OpenDocument Format*, *OASIS Open Document Format for Office Applications*, international genormter quelloffener Standard für Dateiformate von Bürodokumenten wie Texten, Tabellendokumenten, Präsentationen, Zeichnungen, Bildern und Diagrammen. Eine OpenDocument-Datei ist entweder eine einzelne XML-Datei oder eine Sammlung verschiedener XML-Dateien und anderer Objekte (z. B. eingebundener Bilder), die zu einer Datei im ZIP-Format zusammengefasst werden.

Open Closed Offenheit einer Software-Einheit für Erweiterungen bei gleichzeitiger Abgeschlossenheit gegenüber Abänderung

Open-Closed-Prinzip Anwendung von ↑Open Closed: Software-Einheiten (↑Klassen, ↑Module, ↑Funktionen etc.) sollten offen für Erweiterung, aber verschlossen gegenüber Abänderung sein.

Paradigma nach Thomas S. Kuhn [4] ein Satz allgemein anerkannter wissenschaftlicher Leistungen, der für eine gewisse Zeit einer Gemeinschaft von Fachleuten maßgebende Probleme und Lösungen liefert

parsbar von einem ↑Parser verarbeitbar, d.h. insbesondere, dass die zu parsende Eingabe in ein für die elektronische/digitale Weiterverarbeitung geeignetes Format umgewandelt werden kann.

Parser (engl. *to parse*, analysieren) Programm, das für die Zerlegung und Umwandlung einer Eingabe in ein für die Weiterverarbeitung geeigneteres Format zuständig ist.

Persistenz Fähigkeit, Daten (oder ↑Objekte) oder logische Verbindungen über lange Zeit (insbesondere über einen Programmabbruch hinaus) bereitzuhalten; benötigt ein nichtflüchtiges Speichermedium.

Polymorphie *polymorphism*, „Vielgestaltigkeit“, ähnliche ↑Objekte können auf die gleiche Botschaft (den Aufruf einer gleichnamigen ↑Methode) in unterschiedlicher Weise reagieren.

Programmierparadigma ein ↑Paradigma der Art zu programmieren. Wichtige Beispiele sind ↑strukturierte Programmierung, ↑objektorientierte Programmierung und funktionale Programmierung.

Repräsentation Darstellung von Daten bzw. Zusammenhängen in grafischer oder tabellarischer Form. Entscheidender Aspekt der wissenschaftlichen Tätigkeit, auf den entsprechend viel Sorgfalt verwendet werden sollte. Insbesondere sollten (grafische) Darstellungen nicht mehr implizieren, als die Daten hergeben.

reproduzierbare Wissenschaft *reproducible science*, seit der Etablierung rechnergestützter Datenauswertung eigentlich nie mehr erreichter, aber für die Wissenschaft konstituierender Aspekt, dass sich Ergebnisse und Auswertungen unabhängig reproduzieren lassen, weil alle dazu notwendigen Aspekte vollständig und ausreichend beschrieben wurden. Motivation für die Vorlesung, deren Ziel es ist, die Hörer mit Konzepten vertraut zu machen, die letztlich eine ernstzunehmende reproduzierbare Wissenschaft ermöglichen.

Schnittstelle *interface*, Begriff mit mehreren leicht unterschiedlichen Bedeutungen; (1.) ↑Signatur einer ↑Funktion oder ↑Methode. (2.) Im weiteren Sinne die Gesamtheit der öffentlichen ↑Attribute und ↑Methoden einer ↑Klasse bzw. eines ↑Objekts. Der Nutzer kennt nur die Schnittstelle, die Implementierung ist irrelevant und kann sich problemlos jederzeit ändern, solange die Funktionalität erhalten bleibt. Das dient der Trennung von Verantwortlichkeiten und ermöglicht ↑Modularisierung und ist in der Folge ein wesentlicher Aspekt der ↑Softwarearchitektur. (3.) In einer weiteren Bedeutung

wird der Begriff (auch im Deutschen dann häufig mit seinem englischen Pendant) für (abstrakte) Klassen verwendet, die lediglich eine Schnittstelle (im Sinne von 2.) definieren. Das ist hauptsächlich dann von Bedeutung, wenn die Programmiersprache keine ↑Mehrfachvererbung unterstützt, aber das Implementieren von „*Interfaces*“.

Selektion eine von zwei Kontrollstrukturen der ↑strukturierten Programmierung, neben der ↑Iteration. In den meisten Sprachen über Bedingungen (`if...else...end`) realisiert, die sich ggf. beliebig verschachteln lassen.

Serialisierung in der Informatik eine Abbildung von strukturierten Daten auf eine sequenzielle Darstellungsform. Serialisierung wird hauptsächlich für die Persistierung von Objekten in Dateien und für die Übertragung von Objekten über das Netzwerk bei verteilten Softwaresystemen verwendet.

Signatur hier: Name und Parameter einer ↑Funktion bzw. ↑Methode, also alles, was ein Nutzer braucht, um diese Funktion oder Methode verwenden zu können.

Single Responsibility ↑Verantwortlichkeit gegenüber genau einer Sache und damit nur ein Grund für Änderungen

Single-Responsibility-Prinzip Anwendung der ↑Single Responsibility: eine ↑Klasse sollte nur einen Grund haben, sich zu ändern.

Softwarearchitektur Aufteilung eines größeren Projektes in einzelne kleinere Projekte bzw. Aufgaben (↑Modularisierung), Definition klarer ↑Schnittstellen und Anforderungen sowie der Interaktion der einzelnen Teile miteinander. Nach Robert C. Martin die Gestalt eines Systems, die ihm von seinen Entwicklern gegeben wird: Unterteilung des Systems in Komponenten, ihre Anordnung, und die Art ihrer Interaktion miteinander. [5, S. 136]

SOLID von Robert C. Martin eingeführtes Akronym aus den fünf Anfangsbuchstaben wichtiger Prinzipien für ↑Softwarearchitektur: ↑Single-Responsibility-Prinzip, ↑Open-Closed-Prinzip, ↑Liskov-Substitutionsprinzip, ↑Interface-Segregation-Prinzip, ↑Dependency-Inversion-Prinzip. Die von der „Gang of Four“ (↑GoF) vorgestellten ↑Entwurfsmuster beruhen großenteils (implizit) auf einer Umsetzung dieser fünf Prinzipien.

SRP ↑Single-Responsibility-Prinzip, vgl. ↑SOLID

Steuercode hier: in Vorlagen (*templates*) eingesetzte Zeichenfolgen, die eine „Programmierung“ innerhalb der Vorlage erlauben. Meist werden die beiden Kernkonzepte der ↑strukturierten Programmierung, also ↑Iteration (Schleife) und ↑Selektion (bedingte Ausführung) implementiert.

strukturierte Programmierung ein ↑Programmierparadigma, das die Zahl möglicher Kontrollstrukturen auf nur zwei (↑Iteration, ↑Selektion) beschränkt, insbesondere den `goto`-Befehl eliminiert (E. Dijkstra). Idealerweise hat ein Codeblock nur jeweils genau einen Ein- und Ausgang. Nach D. Knuth der systematische Einsatz von ↑Abstraktion, der es ermöglicht, große Programme aus kleine(re)n Komponenten zusammenzusetzen. Wichtige frühe Vertreter strukturierter Programmiersprachen sind C und Pascal. Die meisten heutigen Programmiersprachen (mit Ausnahme der funktionalen Programmiersprachen) unterstützen die strukturierte Programmierung.

Subklasse ↑Klasse, die von einer anderen Klasse (der ↑Superklasse) ↑Attribute und ↑Methoden erbt. Die ↑Vererbung geht dabei i.d.R. über die nach außen hin sichtbare ↑Schnittstelle der Superklasse hinaus. Die Subklasse erbt von der ↑Superklasse häufig nur den „kleinsten gemeinsamen Nenner“ und implementiert die spezifische Funktionalität.

Superklasse ↑Klasse, von der andere Klassen (↑Subklassen) ↑Attribute und ↑Methoden erben. Die ↑Vererbung geht dabei i.d.R. über die nach außen hin sichtbare ↑Schnittstelle der Superklasse hinaus. Superklassen implementieren bzw. definieren normalerweise nur das Notwendigste, sozusagen den „kleinsten gemeinsamen Nenner“. Alle spezifische Funktionalität wird in der ↑Subklasse implementiert.

System zur Datenverarbeitung hier: Gesamtsystem für wissenschaftliche Datenverarbeitung von der Datenaufnahme bis zur fertigen Publikation, das alle Aspekte umfasst und das ↑reproduzierbare Wissenschaft möglich macht und gewährleistet. Definitiv ein ↑größeres Projekt, das nicht nur eine ↑monolithische Anwendung umfasst, sondern viele Aspekte darüber hinaus. Setzt entsprechende ↑Infrastruktur und in der Umsetzung der einzelnen Komponenten sauberen Code (↑Clean Code) und eine solide ↑Softwarearchitektur voraus.

TEX von Donald E. Knuth ab 1977 entwickeltes und 1986 fertiggestelltes Textsatzsystem mit dem Schwerpunkt auf typographischer Qualität des Ergebnisses und korrektem mathematischem Formelsatz. Wird heute meist über das auf ihm basierende Makropaket ↑LATEX verwendet.

Typisierung *typing*, Zuweisung eines Typs zu einem Objekt (im abstrakten Sinne) einer Programmiersprache, z.B. Ganzzahl (*integer*) oder Zeichenkette (*string*) im Fall einer Variable. ↑Abstraktion, die die Ausdrucksstärke von Programmiersprachen und Programmen deutlich erhöht, und die Überprüfung der Korrektheit erleichtert sowie Optimierungen ermöglicht. Typisierung kann explizit und implizit erfolgen. Darüber hinaus wird zwischen starker und schwacher Typisierung sowie zwischen statischer und dynamischer Typisierung unterschieden. Jede Art der Typisierung hat ihre Vor- und Nachteile, und unterschiedliche Programmiersprachen verwenden unterschiedliche Arten der Typisierung.

Verantwortlichkeit *responsibility*, im Kontext des ↑Single-Responsibility-Prinzips ein Grund für Veränderung: Wenn man sich mehr als einen Grund vorstellen kann, dann hat das Modul mehr als eine Verantwortlichkeit (gegenüber unterschiedlichen Akteuren).

Vererbung *inheritance*, Weitergabe aller Eigenschaften (↑Attribute, ↑Methoden) von einer ↑Superklasse an eine ↑Subklasse. Die Subklasse ist vom gleichen Typ (↑Typisierung) wie die Superklasse, was wiederum die Grundlage der ↑Polymorphie ist. Änderungen der Superklasse wirken sich allerdings auf die Subklasse aus, die ↑Kapselung wird entsprechend geschwächt. Vgl. ↑Zusammensetzung. Das ↑Liskov-Substitutionsprinzip liefert wichtige Regeln für die Vererbung.

Wiki Website, deren Inhalte Besucher nicht nur lesen, sondern auch direkt im Webbrowser bearbeiten und ändern können. Das Ziel ist häufig, Erfahrung und Wissen gemeinschaftlich zu sammeln und in für die Zielgruppe verständlicher Form zu dokumentieren. Das prominenteste Beispiel ist die Wikipedia. Eines der ersten Wikis wurde von Ward Cunningham 1995 für die Diskussion von ↑Entwurfsmustern in der ↑objektorientierten Programmierung entwickelt. [6] Vgl. ↑DokuWiki.

Windows-INI blockweise strukturiertes ↑Datenformat, das ursprünglich für die Ablage von Konfigurationsoptionen für das Windows-Betriebssystem und darauf laufende Programme entwickelt wurde. Die blockweise Struktur erlaubt zwei Hierarchieebenen: Blöcke und Schlüssel-Wert-Paare. Für weitere Hierarchieebenen und die Ablage beliebig verschachtelter Datenstrukturen müssen hierarchische Datenformate wie ↑XML, ↑JSON oder ↑YAML herangezogen werden.

XML *eXtensible Markup Language*, „erweiterbare Auszeichnungssprache“, ↑Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten im Format einer Textdatei, die sowohl von Menschen als auch von Maschinen lesbar ist. Der große Vorteil von XML ist seine ↑syntaktische Validierbarkeit, der Nachteil das Verhältnis von beschreibender Syntax zu eigentlichem Inhalt, das sowohl die Dateigröße erheblich erhöhen kann als auch die Lesbarkeit durch Menschen einschränkt. Alternativen, die von Menschen einfacher les- und insbesondere schreibbar sind, sind ↑JSON und ↑YAML.

YAML *YAML Ain't Markup Language* (rekursives Akronym, ursprünglich *Yet Another Markup Language*), vereinfachte

↑Auszeichnungssprache (*markup language*) zur Datenserialisierung (↑Serialisierung). Die grundsätzliche Annahme von YAML ist, dass sich jede beliebige Datenstruktur nur mit assoziativen Listen, Listen (Arrays) und Einzelwerten (Skalaren) darstellen lässt. Durch dieses einfache Konzept ist YAML wesentlich leichter von Menschen zu lesen und zu schreiben als beispielsweise ↑XML, außerdem vereinfacht es die Weiterverarbeitung der Daten, da die meisten Sprachen solche Konstrukte bereits integriert haben. Durch weitgehenden Verzicht auf Klammern ist YAML für Menschen les- und insbesondere schreibbarer als ↑JSON und eignet sich daher gut für die Ablage von Metadaten und Konfigurationen. Seit Version YAML 1.2 ist ↑JSON eine vollständige Untermenge von YAML.

Zusammensetzung *composition*, Wechselwirkung zwischen *unabhängigen* ↑Objekten, die ↑Kapselung bleibt im Gegensatz zur ↑Vererbung voll erhalten. Ein Objekt ist aus anderen Objekten zusammengesetzt. Die Objekte können anderweitig vollkommen unabhängig sein. Vgl. ↑Aggregation und ↑Assoziation.

Literatur

[1] Edsger W. Dijkstra. The humble programmer. *Communications of the ACM* 15 (1972), S. 859–865.

[2] Erich Gamma u. a. *Design Patterns. Elements of Reusable Object-Oriented Software*. Boston: Addison-Wesley, 1995.

[3] Barbara Liskov. Data Abstraction and Hierarchy. *ACM Sigplan Notices* 23.5 (1987), S. 17–34. DOI: 10.1145/62139.62141.

[4] Thomas S. Kuhn. *Die Struktur wissenschaftlicher Revolutionen*. Frankfurt am Main: Suhrkamp, 1976.

[5] Robert C. Martin. *Clean Architecture. A Craftman's Guide to Software Structure and Design*. Boston: Prentice Hall, 2018.

[6] Bo Leuf und Ward Cunningham. *The Wiki Way. Quick Collaboration on the Web*. Upper Saddle River, NJ: Addison-Wesley, 2001.