Physikalische Chemie, Universität Rostock

Vorlesung: Wissenschaftliche Softwareentwicklung

Wintersemester 2023/24

Dr. habil. Till Biskup

— Glossar zu Lektion 02: „Motivation: Softwareentwicklung“ —

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.

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.

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

DRY „*Don't Repeat Yourself*“, zentrales Programmierprinzip und Konzept für ↑Clean Code: Wiederholungen (quasi) identischer Code-Teile sind meist ein guter Hinweis auf eine fehlende ↑Abstraktion.

Entwurfsmuster *design patterns*, erprobte und bewährte Lösungen für wiederkehrende Probleme in der Softwareentwicklung. 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.

Infrastruktur 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).

intellektuelle Beherrschbarkeit *intellectual manageability*, nach Edsger Dijkstra [1] das Hauptziel der Softwaretechnik (↑Software Engineering) – und letztlich des Projektmanagements. Unterschiedliche Lösungsansätze für ein Problem sind unterschiedlich gut intellektuell beherrschbar. Entsprechend ist die intellektuelle Beherrschbarkeit das zentrale Kriterium für die Entscheidung, welche Lösung für ein Problem bevorzugt wird.

KISS „*keep it simple, stupid*“, ursprünglich bei Lockheed Martin (Skunk Works) in den 1940er oder 1950er Jahren von Clarence L. (Kelly) Johnson geprägtes Prinzip. Das Ziel damals war: Ein Ingenieur sollte nur mit einem Schweizer Taschenmesser innerhalb von zwei Stunden ein Flugzeug reparieren können. Sehr nah an „Ockhams Rasiermesser“ (Sparsamkeitsprinzip) in den Wissenschaften: Die einfachste mögliche Erklärung, die mit den wenigsten Hilfsannahmen auskommt, ist die zu bevorzugende.

Konvention innerhalb einer Gruppe oder einem (lokalen) Kontext getroffene (temporäre) Fest-

legung. Ziel von Konventionen ist die Vereinheitlichung und damit einhergehend die Befreiung von der Notwendigkeit, jedesmal aufs Neue nachdenken zu müssen, wie z.B. gewisse Prozesse durchgeführt oder Objekte benannt werden sollen. Konventionen sind im Gegensatz zu ↑Standards weniger verbindlich und deutlich flexibler sowie *ad hoc* innerhalb einer Gruppe einführbar.

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.

Muster siehe ↑Entwurfsmuster

Pair Programming „Paarprogrammierung", enge und aktive Zusammenarbeit zweier Programmierer bei der Programmierung von Software. Eine Person sitzt an der Tastatur und programmiert, während die andere aktiv über den entstehenden Code und das Problem nachdenkt und ggf. korrigierend eingreift. Auffallende Probleme werden sofort angesprochen und diskutiert. Die Rollen sollten häufig wechseln (mindestens innerhalb einer Stunde), idealerweise ebenso die Zusammensetzung der Paare.

Schnittstelle mehrfache, leicht unterschiedliche Bedeutungen im Kontext der Softwareentwicklung; hier: Verbindung zwischen einem Stück Software (Programm, Routine) und seiner Umgebung. Dient der Trennung von Verantwortlichkeiten und ermöglicht ↑Modularisierung. Ist deshalb ein wesentlicher Aspekt der ↑Softwarearchitektur.

Softwarearchitektur Aufteilung eines größeren Projektes in einzelne kleinere Projekte bzw. Aufgaben (↑Modularisierung), Definition klarer ↑Schnittstellen und Anforderungen sowie der Interaktion der einzelnen Teile miteinander.

Software Engineering Softwaretechnik, nach ISO: 1. die systematische Anwendung von Kenntnissen, Methoden und Erfahrungen aus Wissenschaft und Technologie auf den Entwurf, die Umsetzung, Überprüfung und Dokumentation von Software [...] 2. die Anwendung eines systematischen, disziplinierten und quantifizierbaren Ansatzes auf die Entwicklung, den Einsatz und die Wartung von Software, d.h. die Anwendung ingenieurswissenschaftlicher Konzepte auf Software (ISO/IEC/IEEE 24765:2010; eigene Übersetzung)

Softwarekrise Erstmals Mitte der 1960er-Jahre auftretendes Phänomen, dass die Kosten für die Software die Kosten für die Hardware überstiegen, weil die vorhandenen Softwareentwicklungsstrategien der steigenden Komplexität der Programme, die durch die deutlich verbesserte Hardware möglich wurden, nicht gewachsen waren.

Standard von einem oft internationalen und anerkannten Gremium definierte Festlegung. Standards sind im Gegensatz zu ↑Konventionen sehr viel starrer und nicht *ad hoc* von einer Gruppe einführbar.

Trennung der Zuständigkeiten *separation of concerns*, grundlegendes Prinzip für ↑Modularisierung, u.a. durch Kapselung und Aufteilung eines Gesamtsystems in Schichten.

Unix-Philosophie „*Do one thing, and do it well*" [2, S. 53] Jeder Programmabschnitt erfüllt genau eine Aufgabe. Durch die einheitliche ↑Schnittstelle vieler Prozesse können Routinen nahezu beliebig hintereinander geschachtelt und so zu einer komplexen Prozesskette verbunden werden (*piping*).

Literatur

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

[2] Peter H. Salus. *A Quarter Century of UNIX*. Reading, MA: Addison-Wesley, 1994.