1 Modellbildung bei der Softwareerstellung

1.1 Grundlagen der Softwareentwicklung

Programme können ganze Produktionsanlagen, Roboter, Fahrzeuge u.v.m. steuern. Man kann aber auch ganz einfache Dinge wie die Addition zweier Zahlen per Computersoftware auf allen möglichen Enderäten ausführen lassen. Diese beiden Extreme zeigen die unterschiedlichen Stufen des Kompelxitätsgrades von Softwareprodukten. Kennzeichen all dieser zu lösenden Aufgaben ist es, Menschen bei der Lösung beruflicher Problemstellungen zu helfen (oder Aufgaben komplett zu übernehmen), oder in der Freizeit in vielfältiger Weise mit den Anwendern zu kommunizieren.

Damit Softwareprodukte (in Zusammenspiel mit Hardware) solche Aufgaben lösen können, müssen Anleitungen (Arbeitsanweisungen) formuliert werden, die als **Programme** bezeichnet werden.

Der Begriff **Software** (engl. „das weiche Zeug“) wurde zuerst von John Tukey im Jahr 1958 verwendet. Man versteht darunter die Summe aller nicht physisch vorhandenen Komponenten von Systemen und die dazugehörenden Daten, die auf einem Computer ausgeführt werden können.[1]

Salopp könnte man sagen: **Software bestimmt, was Hardware erledigen soll.**

Der Sprachgebrauch aus dem Umfeld von Smartphones hat sich auch bei PC-Anwendungen durchgesetzt. Der Begriff **App** (Kurzform für Application Software) hat die gleiche Bedeutung wie Anwendungssoftware.

Aufgrund des extrem unterschiedlichen Komplexitätsgrades von Softwareprodukten können diese zum einen im Handumdrehen erstellt werden, aber auch einen mehrjährige Entwicklungsprozesse durchlaufen, an denen ganz unteschiedliche Teams beteiligt sein können.

Da nur vermeintliche Genies fertige, d. h. lauffähige Anwendungen aus „dem Ärmel schütteln“, bedienen wir uns in diesem Buch den Methoden der Softwarentwicklung[2] in Form eines mehr oder weniger stark ausgeprägten Modellbildungsprozesses, also eines Entwurfsprozesses, genannt **Softwaredesign.**

1.2 Modelle als Mittel der Komplexitätsbewältigung

Stellen Sie sich vor, Ihr Informatiklehrer beauftragt eine Gruppe von Schülern, in Form einer Projektarbeit, ein lauffähiges Programm zu erstellen, welches die Belegung der

---

[1] Vgl. beispielsweise Software Lexikon und Glossar, Business Software Definitionen und Abkürzungen aus der IT-Branche, SoftSelect, Hamburg, hier: „Definition und Erklärung Software“, unter http://www.softselect.de/business-software-glossar/software [15.12.2017]

[2] Häufig wird auch der englische Begriff „Software Engineering“ verwendet.

Schulsporthalle, abhängig nach Sportarten, der benötigten Zeit und den Stammdaten der Nutzer darstellt und eventuelle überschneidende, neu eingehende Belegungswünsche, mittels Eingabe der gewünschten Daten, am Bildschirm ausgibt.

Wie gehen Sie vor, um Schweissausbrüche und durchwachte Nächte zu vemeiden?

Sicher sikizzieren Sie – zusammen mit dem Hausmeister und Sportlehrern – die jetzige Hallenbelegung auf einem Blatt Papier, bevor sie sich Gedanken über mögliche externe Belegungsanfragen, oder gar über die Umsetzung des Projektauftrages in Form eines Comuterprogramms machen.

Bereits das Projekt „Sporthallenbelegung“ hat gezeigt, dass die reale Welt sind viel zu komplex ist, um sie in einer 1:1-Darstellung bewältigen zu können. Wenn der Hausmeister bei jeder Anfrage zu einem neuen Hallenbelegungswusch am Telefon antwortet: „Geben Sie mir 30 Sekunden Zeit, dann sage ich Ihnen …“, dann mag diese Aussage des Hausmeisters bei den anfragenden Kunden Freude ohne Ende erzeugen, aber sicher nicht die Frage beantworten, was ist, wenn dieses Genie krankt ist, in Rente geht …

Die Zeiten sind vorbei, als der Satz noch Gültigkeit hatte: „Ich bin Hausmeister und weiss alles“.

Zur Lösung der gestellten Problemstellung könnten u. a. folgende Aspekte bedeutsam sein:

- **Abstraktion**, d. h. Ausblenden unwesentlicher Elemente.
  Bei der Hallenbelegung kann wohl – von Ausnahmen abgesehen – auf Alter, Geschlecht, Wohnort und weiterer Parameter von Sportlern getrost verzichtet werden.
- **Strukturiererung** durch Gliederung und Aufteilung in sich abgeschlossene Unterstrukturen (Modularisierung).
  Hier eignet sich zuerst die Erstellung einzelner Tagespläne, die erst in einer Endphase zu Wochen- Monats- oder gar Jahresplänen zusammengefügt werden.

Modellbildung bedeutet nicht nur „verkleinerte Abbildung der Realität“, sondern mittels Abstraktion und Strukturierung reale Systeme besser zu verstehen. Ein wesentliches Prinzip dieser modellhaften Vorgehensweise ist die bewußte Reduktion unbedingt notwendige Eigenschaften (Attribute), die für den Modellbetrachter bedeutsam sind, mit dem Ziel, der Erklärung des Gesamtsystems bausteinmäßig näher zu kommen. [3]

Modellarten[4]

Werden Modelle der Realität nachgebildet, so spricht man von **Nachbild-Modellen.**

Beispiele hierfür sind:

---

[3] Vgl. u. a.: Workshop "Modellentwicklung, Modellgestaltung, Modellnutzung, Theorie der Modelle", Christian-Albrechts-Universität Kiel, Abschitt „Modellentwicklung“, veröffentlicht 2010 unter: https://www.is.informatik.uni-kiel.de/events/modellierung_2010/AllgemeineszuModellen.html [15.12.2017]

[4] Vgl. „Grundlagen der Software-Modellierung“, Folie 101, Universität Marburg, veröffentlicht 04.11.2014, unter: https://www.uni-marburg.de/fb12/arbeitsgruppen/swt/lehre/files/est1415/EST141104.pdf [15.12.2017]

aus Bereichen der Geografie:	Landkarten, GPS-Anwendungen
aus Bereichen der Medizin:	Körpermodelle, DNS-Strukturmodelle
aus Bereichen der Soziologie:	Gruppen-Dynamik-Modelle

Entstehen Modelle vor der Realisierung von Aktionen, so spricht man von **Vorbild-Modellen.**

Beispiele hierfür sind:

aus Bereichen der Architekur:	Baupläne
aus Bereichen der Hardwareerstellung:	Chip-Schaltpläne
aus Bereichen der Verwaltung:	Bebauungspläne
aus Bereichen der Softwareerstellung:	Alle Modelle, die der eigentlichen Programmierung vorausgehen.

1.2.1 Vorgehensmodelle als wichtige Hilfen des Softwaremanagementes

Ein **Vorgehensmodell**[5] definiert einen allgemeinen Rahmen für die Organisation von Ablaufprozessen bei der Softwareerstellung. Hier werden u. a. festgelegt:

- Art, zeitlicher Ablauf und Grobgliederung der durchzuführenden Aktivitäten.
- Definition einzelner Entwicklungsstufen und deren Abgrenzung von anderen Phasen im Entwicklungsprozess.
- Festlegung von Faktoren für die Fertigstellung von Teilprojekten innerhalb des Gesamtprojektes.
- Regelung personeller Verantwortlichkeiten und Kompetenzen der handelnden internen bzw. externen Mitarbeiter und daraus folgenden Konsequenzen für den Personaleinsatz einschließlich Planung und Durchführung von Fortbildungsmaßnahmen.

Bekannte Vorgehensmodelle sind u. a.:

1. Phasenmodelle
2. Wasserfall-Modelle
3. Evolutionäre Modelle
4. Wiederverwendungsorientierte Entwicklungen
5. Inkrementelle Entwicklungen
6. Extreme Programming
7. Iterative Modelle (z.°B. Spiralmodell)
8. Unified Process

---

[5] Weitergehende Informationen finden Sie u. a. in der Enzyklopädie der Wirtschaftsinformatik, Hrsg.: Norbert Gronau, Jörg Becker, Elmar J. Sinz, Leena Suhl, Jan Marco Leimeister, hier u. a. der Artikel „Vorgehensmodell“ von Prof. Dr. Michael H. Breitner, Leibniz Universität Hannover, Institut für Wirtschaftsinformatik, Hannover, veröffentlicht am 30.10.2012 unter http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Vorgehensmodell/index.html [15.12.2017].

9. V-Modelle
10. Agile Methoden
11. „DevOps“ (Development + Operation-Methoden

Argumente für den Einsatz von Vorgehensmodellen

- Kontrolle von Zeit, Kosten und Qualität der Einzelprozesse in Bezug auf das zu erstellende Endergebnis
- exakte Planbarkeit von Softwareprojekten durch definierte, strukturierte und standardisierte Vorgehensweisen
- steigende Transparenz und Kontrolle während der Erstellungsphasen von Softwareprodukten
- Optimierung von Entwicklungsprozessen, z. B. schnelle Fehlerbehebung
- Verbesserung der Kommunikation innerhalb der Projektbeteiligten und Kunden
- Automatisierungsmöglichkeiten durch Einsatz von Tools (Entwicklungswerkzeugen)

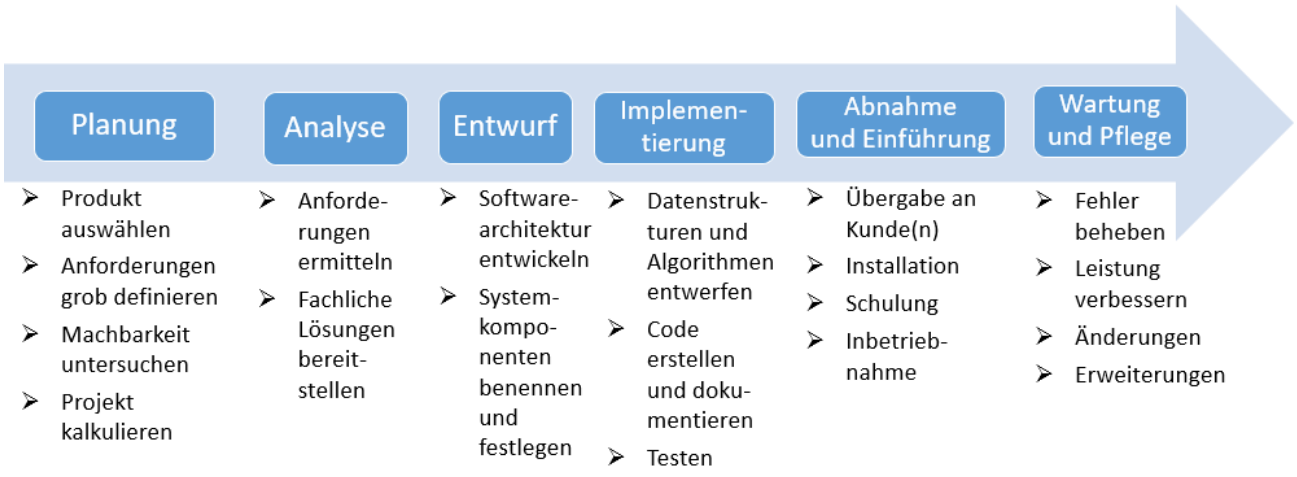
Klassische Fehler in der Software-Entwicklung

- Man entscheidet sich erst verpätet für ein Vorgehensmodell, bzw. befolgt nicht die standardardisierten, vorgegebenen Regeln.
- unrealistische Terminvorgaben
- Die Weiterbildung der Mitarbeiter ist nicht zielgerichtet.
- Es fehlt ein Risikomanagement.
- Es erfolgt keine Abnahme der einzelnen Phasenergebnisse.
- Es wird nicht systematisch bzw. unzureichend getestet.
- Anforderungen und Qualitätsmerkmale werden nicht oder zu spät festgelegt.

Die Vielfalt der in der Praxis eingesetzten Vorgehensmodelle, aber auch die ganz unterschiedlichen Ausgestaltungen dieser erlauben es nicht, dass in diesem Lehrbuch alle Modelle einzeln vorgestellt werden. Lediglich drei – aber typische Vorgehensweisen – werden mit ihren jeweiligen Zielen, Betrachtungsweisen und Vor- und Nachteilen exemplarisch beschrieben.

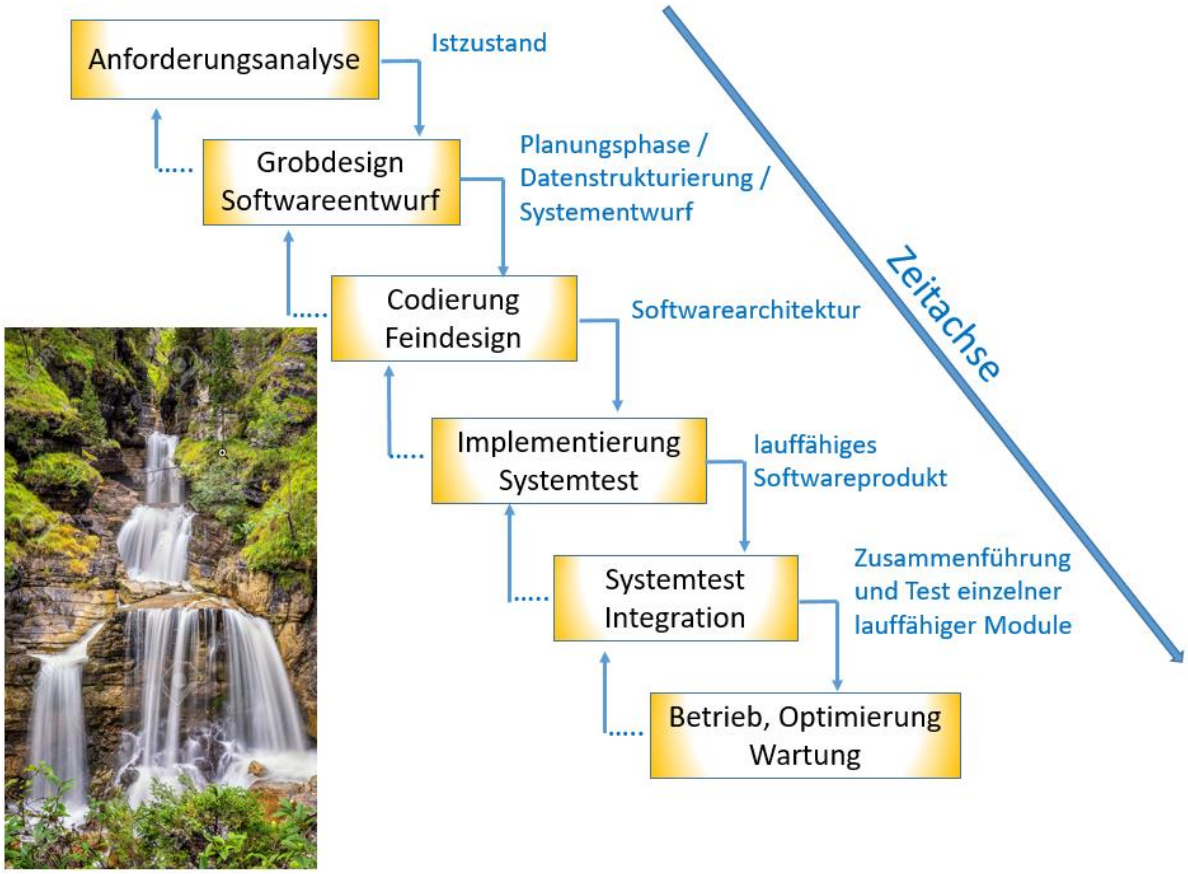
Phasen bei der Erstellung von Softwareprojekten

Folgende Darstellung beschreibt den organisatorischen Ablauf der Softwareerstellung in allgemeiner Form.



1.2.2 Softwareerstellung nach dem Wasserfallmodell

[[Information]]
Das Wasserfallsmodell arbeitet nach dem top-down-Ablaufprinzip, indem Zwischenschritte einzelner Entwicklungsphasen an nachfolgende Stufen weitergereicht werden.
[[Ende Information]]



Da es kein einheitliches Wasserfallmodell gibt, sondern mehrere Ausgestaltungsmöglichkeiten, gilt folgende Grundstruktur:

In der **ersten Phase**, der **Anforderungsanalyse** werden die Aufgabenstellungen und Problemdefinitionen des Auftraggebers ermittelt, und auf ihre Machbarkeit hin geprüft.

Anschließend werden die vorliegenden Daten strukturiert. Das Ergebnis einer Anforderungsanalyse wird meistens in einem **Lastenheft**[6] (Anforderungskatalog) beschrieben.

In der **zweiten Phase** des **Grobdesigns** werden **Algorithmen** formuliert.

Um diesen Begriff mit „Leben zu erfüllen“ lohnt es sich, einen kurzen Blick in die Geschichte zu werfen.

Wussten Sie schon, dass das Wort Algorithmus auf den persisch-arabischen Mathematiker und Astronom *Muhammad Ibn Musa Al-Hwârizmî* nach seinem Werk

„Kitab al jabr w.almuqabala“ („Regeln der Wiedereinsetzung und Reduktion“) um 825 n. Chr. zurückzuführen ist? Das Werk – es ist im Original leider nicht mehr vorhanden – behandelt algebraische Gleichungen, geschrieben mit dem indischen Zahlensystem und das Rechnen in diesem. Eine lateinische Übersetzung aus dem 12. Jahrhundert „Algorithmi de numero Indorum“ existiert noch. [7]

((ibn_Musa_13073113.jpg))
Vgl. 4688, S.74.

Seit Jahrhunderten versteht man unter einem **Algorithmus** eine präzise, programmunabhängige Vorschrift, die aus mehreren elementaren Schritten besteht, die in einer bestimmten Reihenfolge ausgeführt werden müssen.

Algorithmen lassen sich mit unterschiedlichen Mitteln erstellen:

- umgangssprachlich
- mit grafischen Darstellungsmitteln
- mit mathematischen Ausdrücken

In einer **dritten Phase** erfolgt die entscheidende Vorstufe zur Arbeit am Computer. Alle Daten und Arbeitsanweisungen sind in diesem Stadiun so aufzubereiten, dass in der **vierten Phase** diese in einen Programmcode (Quellcode) unter Zuhilfenahme einer Programmiersprache umgewandelt werden können. Diesen Umwandlungsprozess nennt man **Codierung**. Es erfolgt eine **Implementierung** (lat. „implere“ steht für ergänzen, ausführen, realisieren), d. h. das Erstellen eines lauffähigen Softwareproduktes oder das Einbinden in bestehende Programmstrukturen. Die **fünfte Phase** ist die Phase des Testens, des Vergleichens von Soll und Ist-Eigenschaften und (was oft schlaflose Nächte bereiten kann), die Fehlersuche und Fehlerfindung.

Vorteile

---

[6] Näheres erfahren Sie im Abschnitt „Das Wasserfallmodell im Praxiseinsatz“, S. xx

[7] Wenn Sie mehr daüber erfahren wollen, hilft folgende Adresse:
www.kk.s.bw.schule.de/mathge/alhwariz.htm

- Aufgrund seiner einfachen Anwendbarkeit ist das Wasserfallmodell sehr verbreitet und existiert in vielen Abwandlungen.
- Es bietet gut strukturierbare und kontrollierbare Prozessabläufe.

Nachteile

- Es ist ein starres Modell, da ein Austausch nur über und entlang der einzelnen Stufen des Verfahrens erfolgen kann. Das minimiert zwar die Fehleranfälligkeit, schränkt aber die Flexibilität in der Entwicklung ein.
- Sprünge zwischen den einzelnen Phasen (Iterationen) sind nur zwischen aufeinanderfolgenden Bereichen erlaubt.
- Jede Phase muss vollständig durchlaufen werden.

Das Wasserfallmodell im Praxiseinsatz

Das **Wasserfallmodell** ist ein lineares **Vorgehensmodell**, das auf den Vorstellungen eines geradlinigen Entwicklungsprozesses beruht. Dabei gehen die Phasenergebnisse wie bei einem Wasserfall immer als bindende Vorgaben in die nächst tiefere Phase über. In jeder Phase hat dieses Modell vordefinierte Start- und Endpunkte mit eindeutig definierten Ergebnissen. In **Meilensteinsitzungen** am jeweiligen Phasenende werden die Ergebnisse bewertet. Zu den wichtigsten Dokumenten zählen dabei das **Lasten-** und **Pflichtenheft**.

In einem Lastenheft beschreibt der Auftraggeber alle Anforderungen, welche in einem Anforderungskatalog dem Softwareersteller zugehen. Somit erleichtert es dem Auftraggeber, vergleichbare Angebote verschiedener Anbieter einzuholen, da jeder potentielle Auftragnehmer dieselbe Grundlage für ein Angebot vorliegen hat. Dem Auftragnehmer ist es nun dank Lastenheft möglich, ein Pflichtenheft zu erstellen, welches beschreibt, wie und womit der Auftragnehmer das Gesamtvorhaben umsetzen wird.

Oben beschriebene Vorgehensweisen werden an dem Beispiel der Bestellung eines neuen PKW verdeutlicht.

Der Kunde beschreibt ein Fahrzeug mit allen gewünschten Extras. Der Verkäufer hinterfragt im ersten Gespräch die geäußerten Wünsche. In seiner Vorstellung entsteht das Bild eines Wunschfahrzeugs. Der Verkäufer „baut“ die Systemelemente zusammen und liefert es dem Auftraggeber. Was der Händler im Verhältnis zu seinem Großhändler oder gar Produzenten vereinbart, ist Gegenstand eines Pflichtenheftes. Bitte stören Sie sich nicht an der Bezeichnung „Heft“. Diese „Hefte“ sind elementare (meist digitale) Bausteine des Projektmanagements.

Fazit

Die Tatsache, dass das Wasserfallmodell vor allem durch zyklische Entwicklungen ergänzt bzw. abgelöst wurde zeigt, dass sich eine streng sequenzielle Abfolge von

Prozesschritten nur für ein überschaubares Projekt von Kunden mit kleinen Mitarbeiterzahlen anbietet.

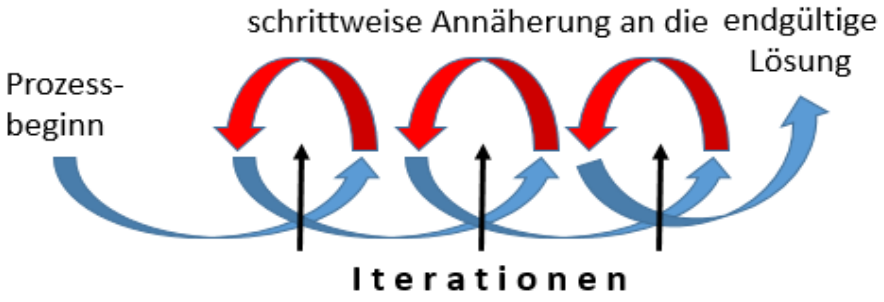
1.2.3 Iterative Verfahren: Vom Vorläufigen zum Perfekten

Haben Sie schon einmal einen **Erlkönig** gesehen?“

… nein, nicht jenigen, der seit 1782, geschrieben von Johann Wolfgang v. Goethe, so spät durch Nacht und Wind reitet, sondern eine Bezeichnung für einen noch unbekannt bleiben sollenden Prototyp eines neu entworfenen Fahrzeuges, der von *Erlkönig-Jägern* gejagt wird.

Ob Auto, Smartphone, Computer, Fotoapparat oder Software: Sehr viele Produkte werden iterativ entwickelt, d. h. es wird eine vorläufige Version erstellt, anschließend sofort hinterfragt und auf Basis des Feedbacks weiter verbessert, bis das gewünschte Endergebnis erreicht ist.

Das Iterationsmodell (lat. iterare = wiederholen) geht davon aus, dass es unwahrscheinlich ist, dass ein Produkt gleich beim ersten Versuch perfekt dem Kunden übergeben werden kann. Stattdessen wird ein **Prototyp** entwickelt und Feedback eingeholt, das sofort in eine Nachfolgeversion einfließt. Diese **Feedbackschleifen** (Iterationen) werden so oft wiederholt, bis der Auftraggeber zufrieden ist.[8]



[8] Im Kapitel 2.7 „Wiederholungsanweisungen steuern den Programmablauf“, wird Ihnen diese Problematik mehrfah begegnen.

Vorteile

- Der Einsatz ist möglich, auch wenn die Größe des Gesamtprojektes bei Projektstart noch unüberschaubar ist.
- Der Einsatz ist auch bei häufigem Kundenfeedback, auch verbunden mit Änderungswünschen möglich. Nach jeder Iteration ist eine Abnahme durch den Kunden erwünscht.

Nachteile

- Ein isolierter iterativer Ansatz führt dazu, dass erst recht spät ein marktfähiges Produkt entstehen kann.
- Ein mehrmaliges Durlaufen der Entwicklungsphasen bedingt, dass das Projekt zeitlich und finanziell schwer planbar ist.

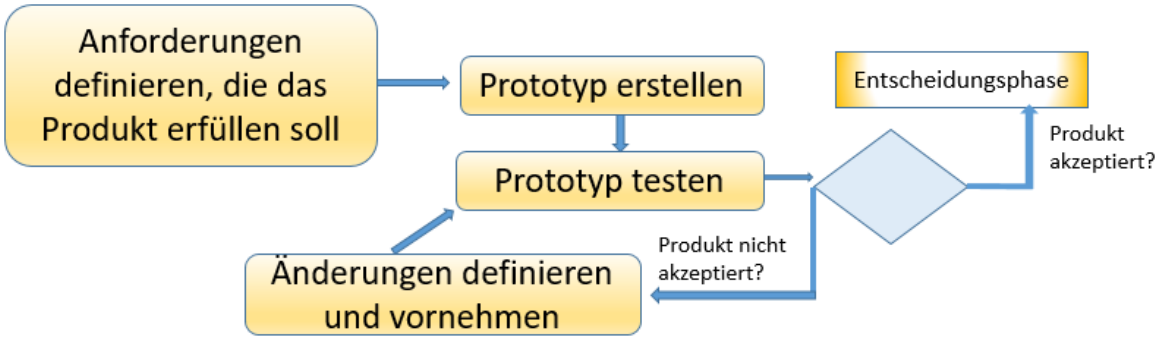
Im Gegensatz zum klassischen Wasserfallmodell, in dem das ganze Projekt in einzelnen Phasen über Meilensteine kaskadenförmig das Projektende erreicht, ermöglichen iterative Konzepte mittels Definition von kleineren überschaubare Häppchen, frühzeitig Prototypen, welche dem Kunden präsentiert werden können.

1.2.4 Inkrementelle Verfahren: Vom Kleinen zum Großen

Stellen Sie sich vor, ein Team von Bauingenieuren erhält den Auftrag, eine neue Umgehungsstgraße durch z. T. unwegsames Gelände zu bauen. Arbeitet das Team iterativ, wird wohl zuerst einmal ein Pfad für Fußgänger und eine provisorische Straße für Baufahrzeuge angelegt. In einer weiteren Phase wird der Straßenverlauf z. B: unter Umweltaspekten getestet. Vielleicht wird der ein oder andere Straßenabschnitt bezüglich des Verlaufes korrigiert, bevor die Baufirma anfängt, die Straße zu verbreitern, einen Belag aufzubringen, Leitplanken zu installieren, Ampelanlagen zu bauen usw. Jede Straßenversion ist ein Stück ausgereifter, aber grundsätzlich kommt der Autofahrer -wenn auch beschwerlich - bereits ab der ersten Iteration von A nach B.

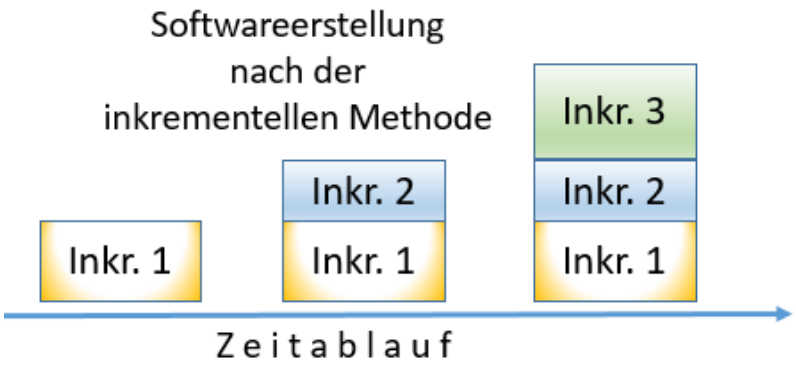
Entscheiden Sie sich hingegen für eine inkrementelle Arbeitsweise (von lateinisch incrementare ‚vergrößern), bauen Sie die Straße abschnittsweise – Sie kennen das vom klassischen Autobahnbau. Zwar ist jeder Teilbereich für sich sehr früh nutzbar, aber mit dem Auto zum Endpunkt fahren können Sie erst, wenn alle Etappen fertig sind.

Das Prototypverfahren als Teil eines inkrementellen Verfahrens kann wie folgt schematisiert werden.



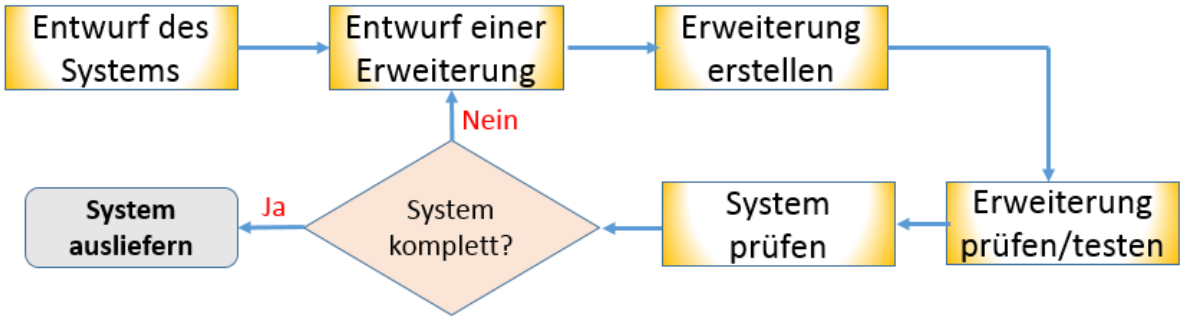
Das **inkrementelle Vorgehensmodell** beschreibt einen Prozess der kontinuierlichen Verbesserung, der häufig in kleinen oder sogar kleinsten Schritten vollzogen wird. Somit wird nicht am Gesamtsystem gearbeitet, sondern Schritt für Schritt in Zyklen (Iterationen) gearbeitet.

Folgendes Schaubild verdeutlichet, wie mit einem i. d. R. kleinen Kernsystem (Inkrement 1) begonnen, und diese Basis schrittweise erweitert wird.



Auf diese Art und Weise kann dem Kunden ein fertiggestelltes Teilsystem zur Verfügung gestellt werden. Die aus der erstmaligen Fertigstellung resultierenden Erfahrungen können für weitere Teilsysteme genutzt werden.

Die Ablaufprozesse für die Softwareerstellung zeigt folgendes Schaubild.



Vorteile

- Kundennähe während des gesamten Entwicklungsprozesses
- Kundenwünsche können gut eingearbeitet werden, was die Kundenakzeptanz erhöht.
- Lauffähige Software ist relativ schnell zu erstellenr.
- Frühe Inkremente können als Prototyp für spätere Inkremente dienen, weshalb Risiken leichter zu beherrschen sind.

Nachteile

- Schwierige Kalkulation der Gesamtkosten
- Einen Gesamtüberblick ist bei der Planung des ersten Inkrements ist schwierig. *Man sieht man den Wald vor lauter Bäumen nicht mehr!*

Aufgaben/Arbeitsaufträge zur Wiederholung und Vertiefung

1. Softwareerstellung ist ein komlexer Prozess. Hier spielt der Begriff *Implementierung* eine wichtige Rolle. Definieren Sie diesen Begriff und stellen die Phasen der Implementierung zusammen.
2. Am Anfang jeder Softwareerstellung steht ein zu lösendes Problem. Im Laufe dieser ersten Phase werden *Istanalyse, Sollkonzeptentwicklung, Durchführbarkeitsstudie* und *Projektplanung* erstellt. Erklären Sie die kursiv ausgegebenen Begriffe.
3. Genauso wie ein Fahrzeughersteller Anforderungen an die Sicherheit im Unfallverhalten seiner Fahrzeuge stellt, muss auch die Softwareerstellung bestimmte Ziele im Auge haben. Arbeiten Sie Qualitätsfaktoren (Anforderungen) heraus, die heutige Softwareprodukte erfüllen sollten.
4. Beschreiben Sie die Unterschiede der beiden Begriffe: *Softwaredesign* und *Programmierung*.
5. Zukunftsforscher gehen davon aus, dass Berufe im Bereich Softwareentwicklung im Hintergrund von Industire 4.0 große Chancen bieten. Sollten Sie sich für einen Beruf in diesem Bereich entscheiden ist es erforderlich, sich mit folgenden beiden Ansätzen näher befassen:

   - der modellbasierte Ansatz mit Agilen Methoden und
   - die Entwicklungsmethode des „**DevOps**“ (Development & Operation)

   a) Beschreiben Sie die obigen genannten Entwicklungsmethoden.
   b) Kennzeichnen sie die wesentlichen Punkte, die sich von den beschriebenen Modellen unterscheiden.
   c) Arbeiten Sie Vor- und Nachteile heraus.

2 Strukturiert Programmieren

Getränkeautomaten – sei es für durststillende Getränke oder für Leergut – findet man überall. Der Unterschied zwischen einem Getränkeautomaten oder anderen technischen Geräten und funktionsfähigen Computern ist, dass Automaten lediglich vorher festgelegte Aktionen ausführen, Computer aber Handlungsanweisungen ganz unterschiedlichster Art abarbeiten können.

2.1 Von der Problemstellung über den Algorithmus zum Programm

Generell kann man sagen, dass Computerprogramme – zusammen mit der Hardware - definierte Probelme lösen sollen.

Bevor aber überhaupt ein Computer zum Einsatz kommt, sollte die Arbeit mit einer **Problemstellung** beginnen. Hier wird in allgemeinen Formulierungen beschrieben, was ein Computer überhaupt tun soll. Danach wird in einem weiteren Schritt die Problemstellung in einer **Problemanalyse** konkretisiert. Erst danach erfolgt die eigentliche Programmierarbeit am Gerät, nämlich die Übertragung der Ergebnisse der Problemanaylse in ein maschinenverständliches Konzept, die **Codierung.**

Schrittfolge der Softwareerstellung

1. Schritt: Problemstellung

Soll ein Softwareprodukt entwickelt werden, so muss in einer Problemdefinition festgelegt werden, was das System überhaupt lösen soll. Dies geschieht in unserer eigenen Sprache, unabhängig von der Computerwelt.

Während aus der Problemdefinition klar hervorgeht, wozu man ein Programm benötigt, sollte in der Forderungsanlyse beschrieben werden, dass zum Beispiel eine ansprechende graphische Oberfläche gewünscht wird, das System mit anderen Programmen zusammenarbeiten soll und zwischen mehreren Computern ein Datenaustausch erwünscht wird.

2. Schritt: Problemanalyse[9]

Um eine vorgegebene Problemstellung programmtechnisch lösen zu können, ist es erforderlich, die gewünschten Ergebnisse (Ausgabedaten) und die dazu vorhandenen Eingabedaten genau zu analysieren.

[9] Weitere Informationen finden Sie unter „Das EVA-Prinzip als Grundprinzip der Datenverarbeitung“, Wolfgang Braun, Grundlagen der Informatik, Technische Grundlagen, BildungsverlagEINS, S. 20 f., Februar 2017

Die **Problemanaylse** soll auf folgende Fragen eine Antwort geben:

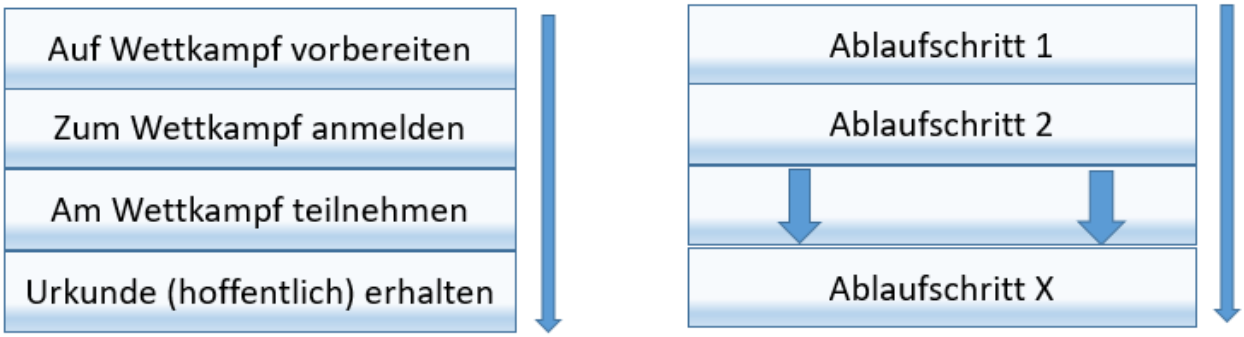
- Was soll ausgegeben werden?
  Zusammenstellung der **Ausgabedaten**.
- Welche Daten müssen eingegeben werden?
  Zusammenstellung der **Eingabedaten**.
- Wie sollen die Eingabedaten verarbeitet werden?
  Eindeutige Darstellung des gewählten **Lösungswegs.**

3. Schritt: Grafische Darstellung von Algorithmen

a) Struktogramm

Ein Struktogramm zeigt in grafischer Form die logische Reihenfolge der einzelnen Arbeitsschritte. Es ist ein nach DIN 66261 genormtes Hilfsmittel zur Planung, Entwicklung und Dokumentation von Programmstrukturen.

Ein linearer Verlauf wird z. B. durch eine untereinanderstehende Folge von Vierecken dargestellt.

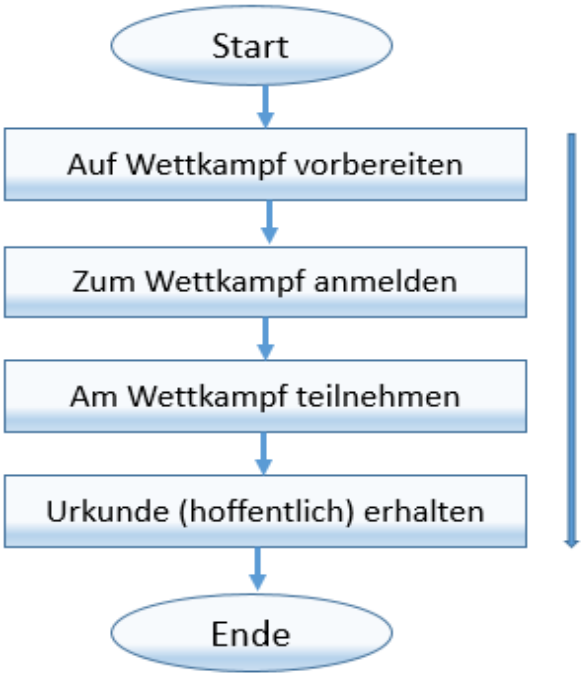


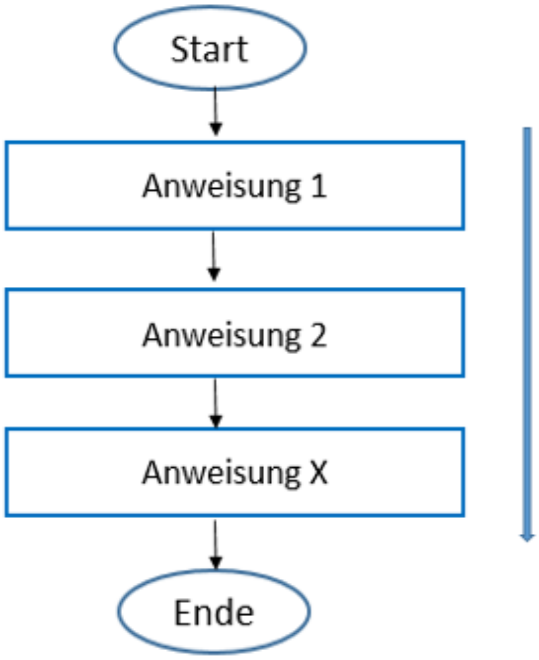
b) Programmablaufplan

Ein weiteres Hilfsmittel der grafischen Umsetzung logischer Abläufe in ein Programm, ist eine Entwurfstechnik, die schon lange angewandt wird, ein **Programmablaufplan**[10] (PAP). Symbole für verschiedene Arten von Ablaufschritten (z. B. Rechtecke, Rauten), sind durch Ablauflinien miteinander verbunden. Diese bestimmen die Reihenfolge, in der die Operationen ausgeführt werden sollen.

Folgende Darstellung zeigt die altenrative Darstellung des Problems in einem PAP.

[10] Die Normen und Regelungen dieser Entwurfstechnik finden Sie in der DIN-Norm Nr. 66001 von 1982


Start
Auf Wettkampf vorbereiten
Zum Wettkampf anmelden
Am Wettkampf teilnehmen
Urkunde (hoffentlich) erhalten
Ende


Start
Anweisung 1
Anweisung 2
Anweisung X
Ende

Struktogramm und Programmablaufplan[11] sind grafische Hilfsmittel zur Veranschaulichung von Abbläufen. Mit ihnen lässt sich die Programmabfolge ohne Bindung an eine konkrete Programmiersprache visualisieren. Der Ablauf kann bereits in dieser Phase der Programmentwicklung auf inhaltliche Richtigkeit hin überprüft werden. Die lineare Ablauffolge enspricht dem oben skizzierten Wasserfallmodell.[12]

4. Schritt: Codierung, d. h. Übertragung in eine Programmiersprache.

In diesem Buch wird die Programmiersprache **Java** unter der Oberfläche von **Eclipse** eingesetzt.

5. Schritt: Programmtest

Der so erstellte Programmcode wird vom Computer und/oder in Form eines Schreibtischtestes auf syntaktische (formale) und/oder semantische (logische) Fehler getestet.

6. Schritt: Dokumentation

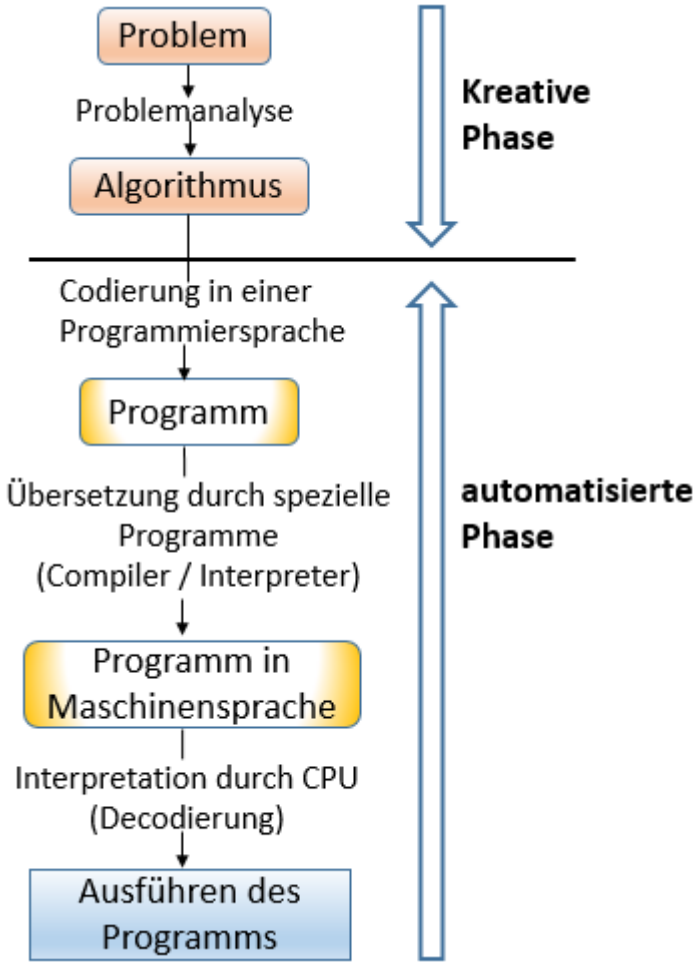
Während der gesamten Softwareentwicklung wird jede einzelne Entwicklungsstufe dokumentiert, was Wartung sowie Weiterentwicklung der Software vereinfacht.

---

[11] Sicher denken Sie: Wieseo soll ich mir die Unterschiede zwischen Struktorgramm und PAP merken, die sind doch so gering! Ja, aber liebe Leser, bitte warten Sie bis zu dem Kapitel „Kontrollstrukturen“; dann sieht alles ganz anders aus.

12 vgl. Kapitel 1.2.2, Softwareerstellung nach dem Wasserfallmodell

Ablaufsteuerung der beschriebenen Arbeitsschritte



2.2 Grundlegende Eigenschaften von Java

[[Einstieg / Aufhänger]]

Warum Programmieren mit Java?

Der Grund ist ganz einfach:

Java läuft überall, in PKWs, Waschmaschinen, Verkehrsampeln, Sicherheitssystemen, im Mikrowellengerät, … unter Windows oder Mac … Es gilt:

„write once run anywhere

Kurz: Java ist objektorientiert und plattformunabhängig.

Frage: Was hat Java mit einer indonesischen Insel zu tun?

Antwort: Nichts!

Java ist nichts anderes als die Bezeichnung einer in den USA populären Kaffeesorte, welche bei vielen Programmierern neben der Tastatur als „Dauerbegleiter“ steht.

Besonderheiten bei der Übersetzung und Ausführung von Java Programmen

Java ist eine Mischung aus **interpretierter** und **compilierter** Sprache und vereinigt die Vorteile beider Konzepte.

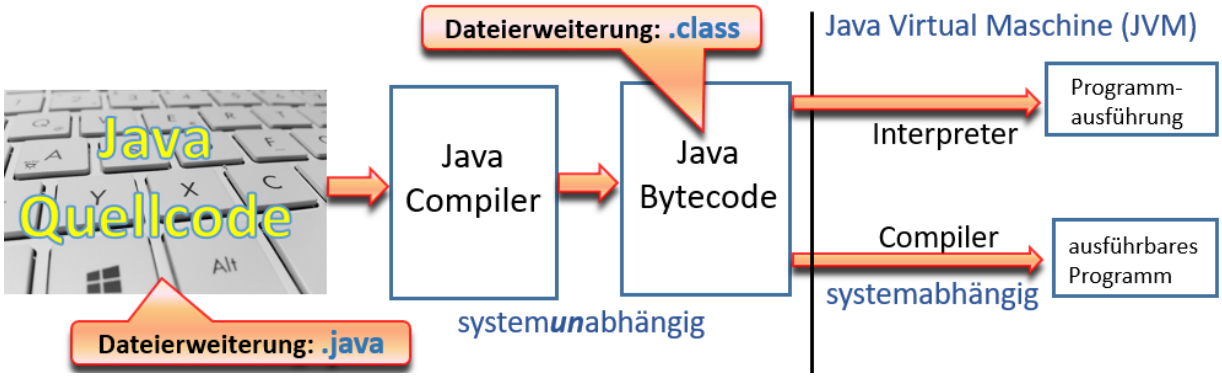
Wie versteht mein Computer das von mir geschriebene Programm?

Zur Lösung der gestellten Ausgangsfrage gibt es verschiedene Konzepte.

Gemeinsamkeit aller Programmiersprachen ist, dass der Programmcode in die Sprache der Maschine übersetzt werden muss. Ein solches Übersetzprogramm ist der **Compiler**. Er übersetzt ein Quellprogramm in ein Zielprogramm.
Verwandt mit einem Compiler ist ein **Interpreter**, der ein Programm nicht in die Zielsprache übersetzt, sondern Schritt für Schritt direkt ausführt.

Java kann beides!

Der Java-Quellcode wird zunächst, von einem **Java-Compiler** übersetzt. Die Besonderheit liegt nun darin, dass der Übersetzer keinen Maschinencode erzeugt, sondern einen plattformunabhängigen **Binärcode** (Bytecode). Dieser Bytecode (Zwischencode) kann unabhängig vom Betriebssystem und Prozessor ausgeführt werden, vorausgesetzt es existiert ein **Java-Interpreter** für die entsprechende Rechnerplattform. Man bezeichnet den Java-Interpreter auch als **Java Virtual Machine.** Diese virtuelle Maschine simuliert einen fiktiven Mikroprozessor, der Befehle des Java-Bytecodes empfängt und verarbeitet wie ein realer Mikroprozessor.



Sie können den Begriff **Maschinensprache** wörtlich nehmen. Es handelt sich um Anweisungen, die ein Computer direkt ausführen kann, den meisten Menschen aber wie „Böhmische Dörfer“ vorkommt, da alle Anweisungen in einer Folge von Dualzahlen

erstellt sind. Eine weitere Besonderheit bei der Ausführung von Java Programmen ist die Art und Weise wie sie gestartet werden können.

Programme der meisten anderen Sprachen können nur auf eine Art gestartet bzw. ausgeführt werden. Bei Java jedoch hat der Anwender, zwei Möglichkeiten:

- **Ausführen als Applikation:** Das Java-Programm wird wie ein „normales“ Programm lokal auf dem jeweiligen Rechner gestartet und interpretiert.
- **Ausführen als Applet:**[13] Ein Applet ist ein spezielles Java-Programm, das mit einem WWW-Browser aus dem Internet geladen wird. Typischerweise sind Java-Applets in HTML (Hyper Text Markup Language) eingebettet. HTML ist die Sprache, mit der WWW-Seiten geschrieben sind. Derjenige Rechner, der die WWW-Seite mit dem Applet bereitstellt, wird **Server** genannt, der Rechner, der das Applet lädt und ausführt, **Client**.

2.3 Projekterstellung mit Eclipse

Das Software-Entwicklungswerkzeug Eclipse ermöglicht dem Benutzer bei der Erstellung von Java-Problemlösungen zwei Vorgehensweisen:

1. Arbeiten mit der von Windows bekannten bunten Welt der Fenster, Schaltflächen, Textfelder, Farben u. v. m., also dem Einsatz einer **GUI** = **G**raphical **U**ser **I**nterface, oder
2. zeilenweise Ein- Ausgabe der Ergebnisse als Texte in verschiedenen Fenstern. Das Ausgabefenster wird Konsolenfenster, kurz **Konsole** genannt.

Beide Möglichkeiten werden in diesem Buch vorgestellt, wobei sich der Autor entschieden hat, zuerst mit der Eclipse-Konsole zu arbeiten.

Eclipse ist ein freies **Softwareentwicklungswerkzeug**, das es seit dem Jahre 2002 gibt. Es handelt es sich um **Open Source**, also Software, die nach Belieben weitergegeben werden kann, ohne dass Lizenzgebühren anfallen. Eclipse bietet dem Benutzer eine **IDE** (integrated development environment = integrierte Entwicklungsumgebung, bestehend aus …

- Editor (Erstellung von Programmtexten)
- Compiler (Übersetzungsprogramm)
- Debugger (Fehlersuche)

Genug der Theorie: Es kann losgehen, aber wohl wissend, dass Eclipse nicht nur ein Softwareprodukt ist, sondern ein für Autofreunde in den 30er Jahren ein Pkw-Statussymbol von Peugeot. Weltweit ist noch die Existanz von 34 dieser Peugeot-401-Eclipse-Modellen bekannt.

[13] Applets sind nicht Gegenstand dieses Buches

2.3.1 Grundlagen der Eclipse-Bedienung

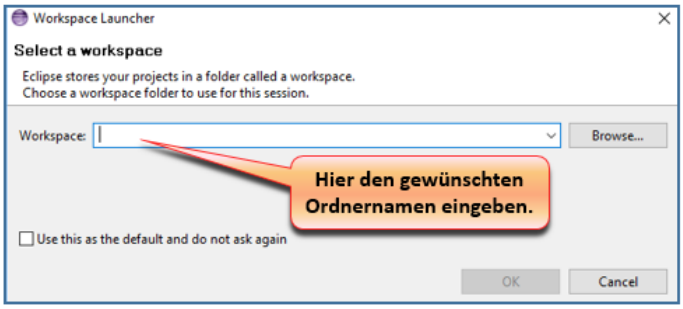
Eclipse besteht aus verschiedenen Teilen (Komponenten), von denen die folgenden die wichtigsten sind:

- das **Workbench** (Hauptfenster mit mehreren Unterfenstern und Editoren)
- das **Workspace** (Projektordner, der alle Dateien enthält)

Starten Sie Eclipse und gehen Sie wie folgt vor:

Immer, wenn folgendes Zeichen erscheint ► ist eine Aktion, meist ein Mausklick vorzunehmen.
((Ende Hinweis oder ähnliches))

► Wählen Sie einen Ordner (Workspace), in dem das Projekt gespeichert werden soll.[14]
► Bestätigen Sie mit OK.
► Legen Sie ein neues Projekt an: *File / New / Java Projekt*



((Workspace_1.TIFF))

Es öffnet sich das Fenster: *New Java Projekt.* Hier ist der neue Projektname einzugeben, z. B. *MeinErstesProjekt*

Es ist ein „Miniatur-Programm“ zu erstellen, welches am Bildschirm den Text:

```
Dies ist mein erstes Projekt
```

ausgibt.

Java unterscheidet streng zwischen Groß- und Kleinschreibung. Somit gilt es, **Namenskonventionen** zu beachten. Die erste Regel lautet:

Programm- bzw. Klassennamen sind groß zu schreiben. Der Klassenname ist mit dem Programmnamen identisch.

[14] Im Menü *File/Switch Workspace* können Sie das Arbeitsverzeichnis jederzeit ändern.

Mit den Besonderheiten von Klassen werden wir uns später beschäftigen. Vorerst genügt es zu wissen, dass eine Klasse so etwas wie ein Bauplan für Objekte ist. Klassen beschreiben Gemeinsamkeiten einer Menge von Objekten mit denselben Eigenschaften (Attributen), demselben Verhalten (Operationen) und denselben Beziehungen. [15]

- ► Geben Sie im leeren Textfeld den Projektnamen ein, hier: *MeinErstesProjekt.*
- ► Die Optionsfelder *Use an execution environment JRE* und *Create separate folders for sources and class files* müssen aktiv sein.

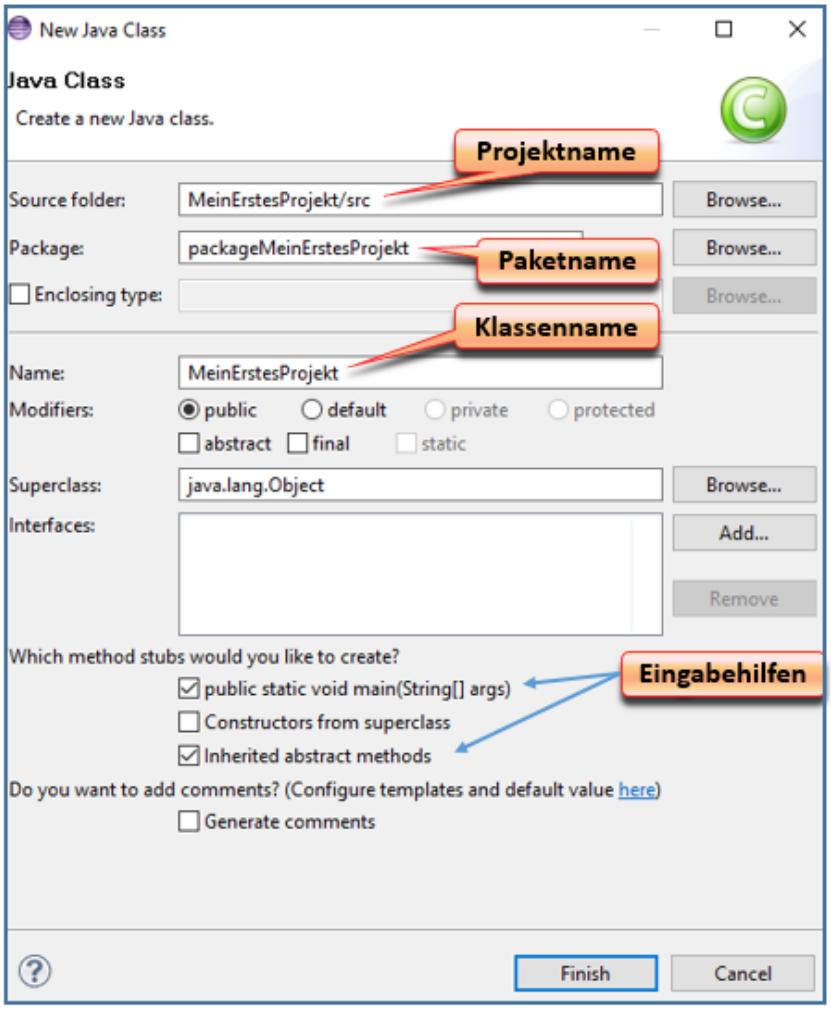
Jetzt ist die Klasse zu bestimmen.

- ► WählenSie: *File / New / Class*
- ► Geben Sie den Klassennamen ein, hier: *MeinErstesProjekt*

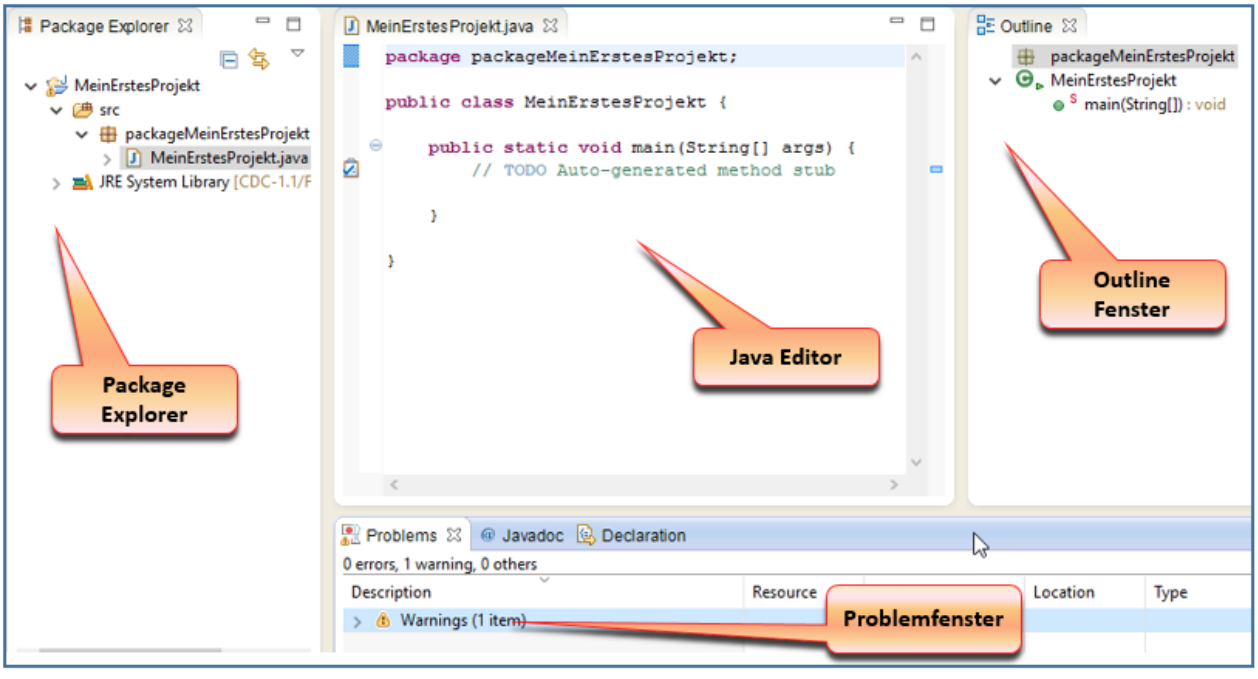
Üblicherweise besteht ein Javaprogramm aus mehreren Dateien. Um diese besser überschauen und verwalten zu können, werden sie in einem **Projekt** zusammengefasst, wobei die einzelnen Teillösungen in Form von Java-Klassen in einem **Paket** organisiert sind.

- ► Geben Sie im leeren Textfeld mit der Beschriftung *Package* einen *packageNamen* ein, wobei zu empfehlen ist, das Wort *package*, dem Projektnamen voranzustellen, z. B. *packageMeinErstesProjekt*.
- ► Geben Sie im noch leeren Textfeld mit der Beschriftung *Name* folgende Namensbezeichnung: *MeinErstesProjekt.*
- ► Wählen Sie die Option (wenn nicht bereits voreingestellt) Modifiers: *public.*
- ► Wählen Sie die beiden Auswählmöglichkeiten (Eingabehilfen)
  public static void main(String[] args) und *Inherited abstract methods.*
- ► Bestätigen Sie auf der Schaltfläche: *Finish.*

[15] Näheres finden Sie ab Kapitel 5 „Objektorientierte Analyse und objektorientiertes Design“, S. xyx



Jetzt erscheint folgender – je nach Installation auch anders darstellbarer – Bildschirm, von Eclipse als *Workbench* bezeichnet.



Wichtige Elemente im Überblick

Package Exporer
Er zeigt alle Projekte und Dateien des aktuellen Workspace an.

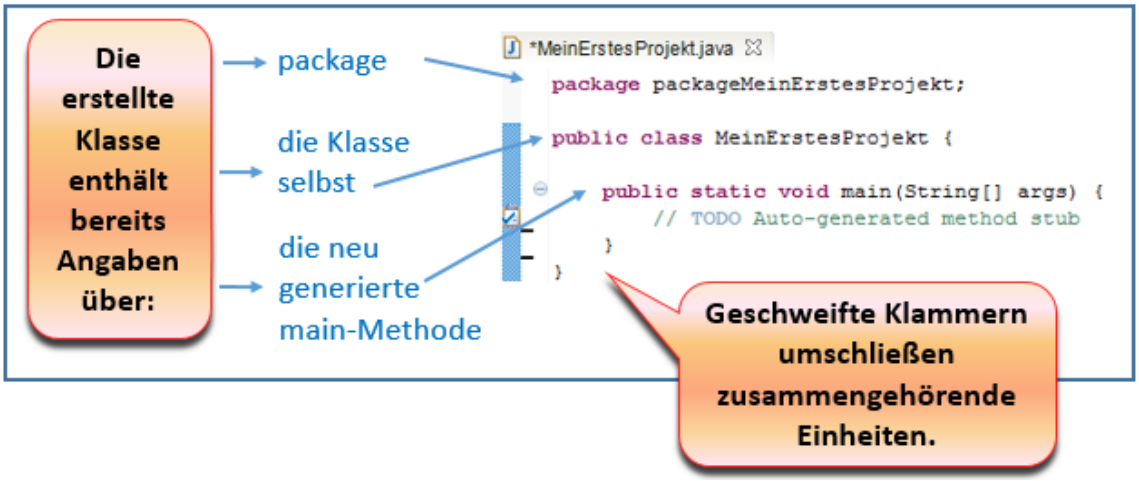
Java Editor
Er dient zur Eingabe des Quelltextes.

Outline-Fenster
Dieses Fenster dient zur Navigation im Source-Code. Es werden die Typen (Klassen, Interfaces), Methoden usw. angezeigt, doch davon später mehr.

Problem-Fenster
Werden Fehler festgestellt, werden diese hier angezeigt. Durch einen Klick auf die Meldung gelangen Sie direkt zum fehlerhaften Code.

Folgende Code-Zeilen wurden vom System als Voreinstellung erzeugt. Sie brauchen sich im Moment darüber überhaupt nicht zu kümmern.

```
package packageMeinErstesProjekt;
public class MeinErstesProjekt {
   public static void main(String[] args) {
     // TODO Auto-generated method stub
 }
}
```

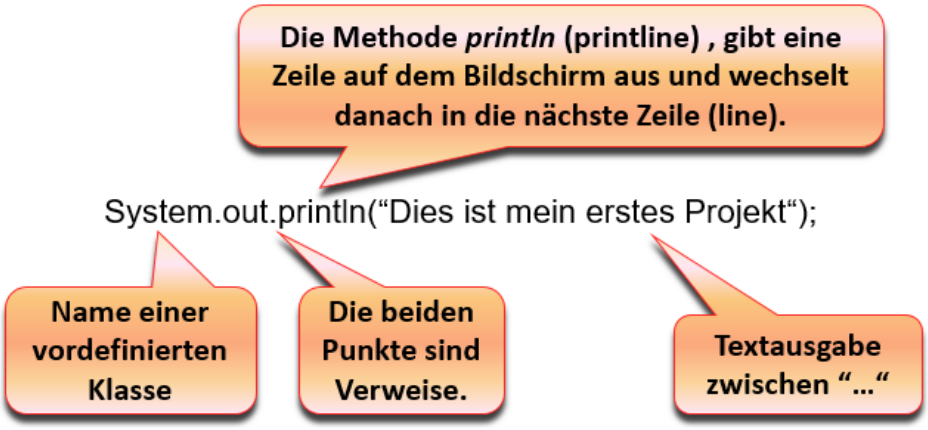


Beginnen wir mit dem „legendärsten“ Programmbeispiel überhaupt: „Hallo Welt“, ein Miniaturprogramm, das nur einen Text, z. B. `Dies ist mein erstes Projekt` ausgeben soll.

Fügen Sie in die Methode `main()` hinter die letzte automatisch generierte Zeile folgende Zeile ein:

```
System.out.println(„Dies ist mein erstes Projekt“);
```

Erklärungen



Was ist ein Verweis?

Wenn Sie jemandem Ihre verwandtschaftlichen Beziehungen zu einer bestimmten Person erklären wollen und sagen …

```
"Das ist die Tante von der Cousine von dem Opa von der Schwester
von der Mutter von der Tochter meines Bruders.",
```

… dann ist das zwar sehr schlechtes Deutsch, aber es erklärt, was ein Verweis ist. Obiger Erklärungswortschwall würde in Java so zu notieren sein:

```
Bruder.Tochter.Mutter.Schwester.Opa.Cousine.Tante
```

Und nun zu dem Schlüssewort: `void`:

`void` bedeutet so viel wie "nichts, leer, Lücke", und signalisiert, dass die Methode `main()` nichts zurückgeben soll. Oft erwartet ein Programmierer von einer Methode einen Rückgabewert. Z. B. sollte eine Methode, die eine Zahl quadriert, das Ergebnis zurückgeben, wobei der Mathelehrer erwartet von einem Schüler erwartet, dass er eine Antwort gibt. Man übergibt der Methode die Zahl (der Mathelehrer stellt dem Schüler die Aufgabe), und bekommt das Ergebnis (die Antwort) von der Methode zurück.[16]

Das erste Programm auf einen Blick

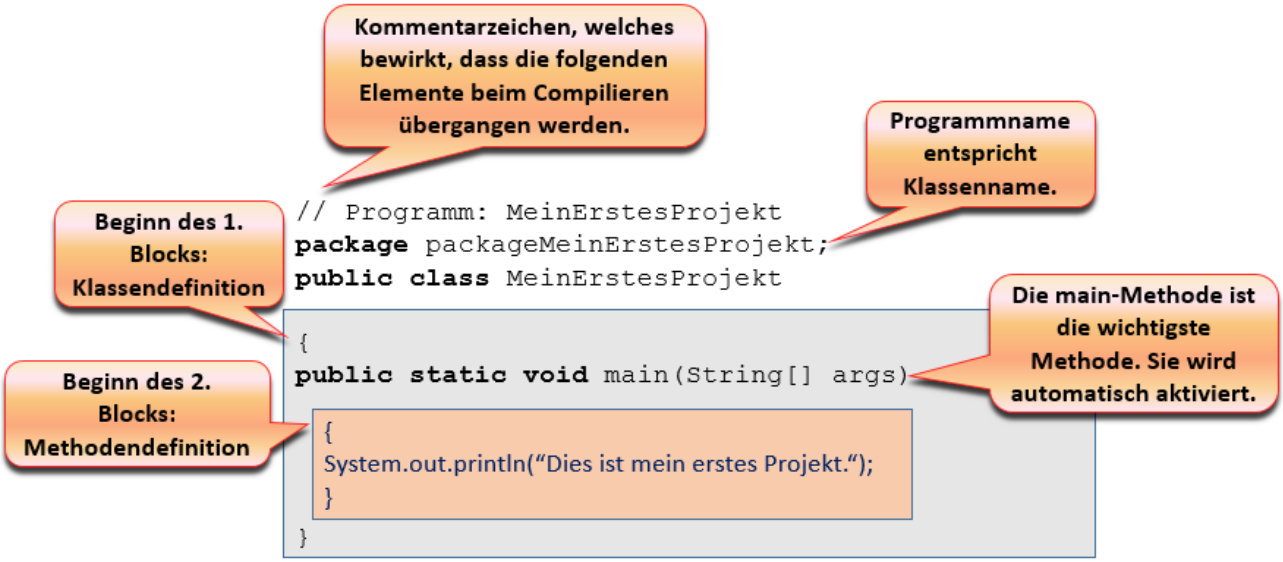
```
package packageMeinErstesProjekt;
public class MeinErstesProjekt
{
  public static void main(String[] args)
  {
  System.out.println("Dies ist mein erstes Projekt.");
  }
}
```

[16] Die Begriffe Methode, Klasse u.v.m. werden in Kapitel 5.2.2 „Von der Klassenbildung zum Objekt“ noch genauer erklärt.

► Starten Sie das Programm (Anwendung ausführen) mittels Betätigung der Schaltfläche `Run`.

Haben Sie im Fenster `Window/PreferencesGerneral/Workspace` das Kontrollkästchen *`Bulid automatically`* aktiviert, wird bei jedem Speichern das Projekt aktualisiert und übersetzt (compiliert).

Schematischer Überblick



2.3.2 Erklärungen zur Anatomie eines Programms

`public class MeinErstesProjekt`	Hier wird eine Klasse mit dem Namen `MeinErstesProjekt` definiert. `class` ist ein Schlüsselwort für die Klassendefinition.
`{ }`	Durch die gescheifeten Klammern weiß der Computer, wo die Definition der Klasse beginnt und wo sie endet. Alles, was innerhalb der beiden geschweiften Klammern steht, gehört zu einer Klasse. Klammern können auch geschachtelt vorkommen. So kann innerhalb eines Blockes ein neuer Block eingebettet werden.
`public`	Das Schlüsselwort `public` besagt, dass diese Klasse öffentlich sichtbar ist (im Gegensatz zu `private`).
`public static void main(String[ ] args)`	Hier wird eine Methode mit dem Namen *`main`* in der Klasse *`MeinErstesProjekt`* definiert. Es wird eine Methode aufgerufen, ohne dass vorher ein Objekt der Klasse erzeugt werden muss, d. h., diese Methode wird automatisch beim Programmstart ausgeführt.
`System.out.println ("Dies ist mein ers-`	Dieses Statement ruft die integrierte **Methode** `println` des **Objektes** `out` in der **Klasse**

`tes Projekt.");`	`System` mit dem **Argument** `"Dies ist mein erstes Projekt.“` auf und produziert eine Datenausgabe im Konsolenfenster. Statements enden immer mit einem Semikolon.

`out.print(x)` gibt x auf dem Konsolenfenster aus.
`out.println(x)` wie oben, danach beginnt aber eine neue Zeile.
(ln = line)

2.3.3 Kommentare erhöhen die Benutzerfreundlichkeit

…