

# Kapitel 1

---

## Einführung

### Inhaltsverzeichnis

1	Einleitung .....	2
1.1	<i>Kurze Geschichte des Software Engineering</i> .....	2
1.2	<i>Definition Software Engineering</i> .....	3
1.3	<i>Die professionelle Disziplin Software Engineering</i> .....	4
1.3.1	Code of Ethics .....	4
1.3.2	Computing Curriculum Software Engineering .....	5
1.3.3	Software Engineering: Body of Knowledge .....	7
1.4	<i>Perspektiven des Software Engineering</i> .....	7
1.4.1	Formale Aspekte des Software Engineering .....	7
1.4.2	Technische Aspekte des Software Engineering .....	7
1.4.3	Ingenieur Aspekte des Software Engineering .....	8
1.4.4	Gestalterische Aspekte des Software Engineering .....	8
1.4.5	Ökonomische Aspekte des Software Engineering .....	8
1.5	<i>Grundlegende Fragestellungen der unterschiedlichen Leser</i> .....	9
1.5.1	Studenten .....	9
1.5.2	Entwickler .....	10
1.5.3	Projektleiter .....	11

# 1 Einleitung

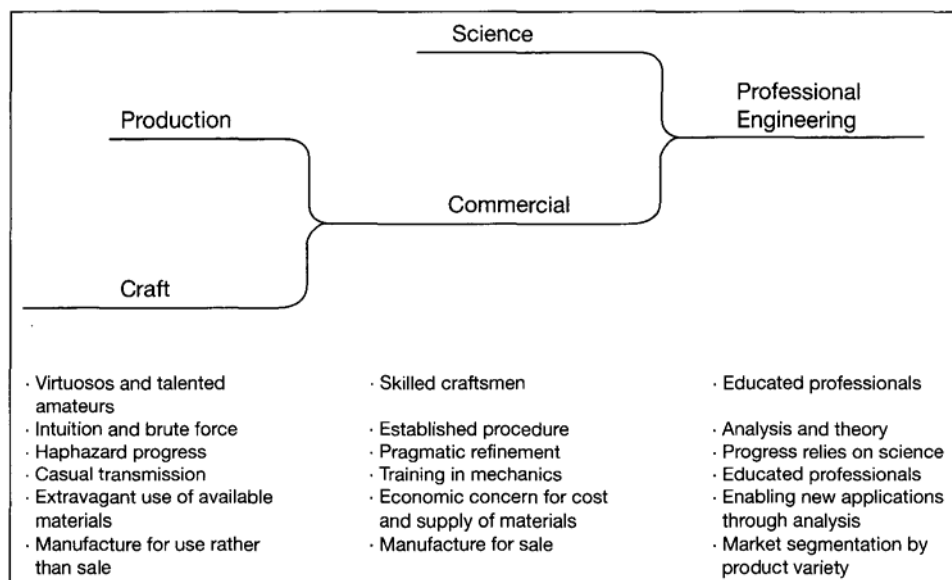
Dieses Kapitel erläutert zuerst den Begriff Software Engineering und seine Geschichte. Anschliessend werden wichtige grundlegende Aspekte besprochen (Definition, Standards, Perspektiven).

## 1.1 Kurze Geschichte des Software Engineering

Software Engineering ist eine junge Ingenieurdisziplin. Dieser Umstand begründet die oftmals unzureichenden Methoden und Techniken, welche zur Erstellung von Software eingesetzt werden. Während in anderen Disziplinen schon seit Jahrzehnten oder Jahrhunderten auf einer professionellen Ebene gearbeitet und geforscht wird, befindet sich der Bereich der gezielten Forschung und Weiterentwicklung von Software Engineering noch im Anfangsstadium.

Mary Shaw zeigt in [Shaw96] die Stufen der Entwicklung zur professionellen Ingenieurarbeit. Am Anfang steht das Handwerk mit der Produktion von Waren durch besonders talentierte Amateure. Diese bewirken zusammen das Entstehen einer eigenen Wirtschaft

für diesen Bereich. Die Produktion wird systematisiert und rationalisiert, um für den entstehenden Markt konkurrenzfähige Produkte anzubieten. In dieser Phase werden die Produkte bereits durch geübte Arbeiter gefertigt Parallel dazu entwickelt sich die Wissenschaft, welche durch Forschung in diesem Bereich zur Innovation beitragen will. Die Kombination von Forschung und Wirtschaft ergibt den Kern einer Ingenieurdisziplin. Diese ist vor allem durch die Analyse bestehender Strukturen und Fakten gekennzeichnet. Basierend auf den Erkenntnissen dieser Analyse kann eine Weiterentwicklung vorangetrieben werden, welche zu einer Segmentierung des Marktes durch neue weiterentwickelte Produkte führt. Durchgeführt werden die Arbeiten von professionell ausgebildeten Personen. Eine übersichtliche Darstellung dieser Entwicklung zeigt *die untenstehende Abbildung*.



**Abbildung 1.1:** Entwicklung einer Ingenieurdisziplin (aus [Shaw96] S. 8.)

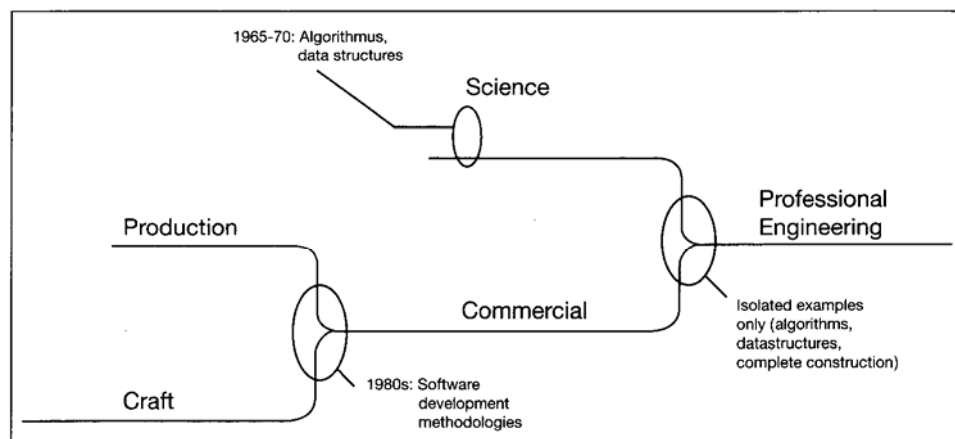
Die oben beschriebene Entwicklung lässt sich auch auf dem Gebiet des Software Engineering nachvollziehen. Von Beginn der Nutzung von Computern bis Anfang der 80er Jahre wurde

unter Software Engineering nicht viel mehr als das Programmieren der verfügbaren Recheneinheiten verstanden. Je nach Aufgabenstellung und zu benutzender Technik wurde von einem Programmiererteam ein Programm erstellt. Im wissenschaftlichen Bereich wurden bis zu dieser Zeit hauptsächlich Algorithmen und Datenstrukturen untersucht, welche direkt Eingang in die Programmiersprachen fanden und darin umgesetzt wurden.

Der Begriff Software Engineering wurde Ende 1967 von einer Forschungsgruppe der NATO geformt. Auf den Software Engineering-Konferenzen der NATO 1968 in Garmisch und 1969 in Rom wurden erstmals in Anlehnung an andere Ingenieurdisziplinen Software-Programme als Industrieprodukt bezeichnet. Es wurde gefordert, Software Engineering nicht als Kunst zu sehen, sondern als ingenieurmässige Tätigkeit anzuerkennen.

Ab Mitte der 60er und in den 70er Jahren kam der Begriff der „Software-Krise“ auf. Dieser bezog sich auf die sich nicht ändernde schlechte Qualität der erzeugten Software-Systeme. Als erste Reaktion wurden in der Praxis erprobte Vorgehensmodelle publiziert.

Ab den 80ern wurden einzelne Softwareentwicklungsmethoden, welche sich in der Praxis bewährt hatten, formalisiert und einem breiteren Publikum zugänglich gemacht. Der Übergang von Wissenschaft und Wirtschaft zu einer professionellen Ingenieurdisziplin ist noch nicht vollzogen. Es gibt nur vereinzelte gut analysierte Beispiele, auf denen aufbauend Ingenieure ausgebildet werden können, die Software Engineering tatsächlich als Ingenieur Tätigkeit betreiben. Abbildung 1.2 zeigt die Entwicklung von Software Engineering im Vergleich zu oben gezeigtem Schema.



**Abbildung 1.2:** Entwicklung von Software Engineering (aus [Shaw96] S. 12.)

## 1.2 Definition Software Engineering

Es existiert eine ganze Menge an Definitionen zum Begriff Software Engineering. An dieser Stelle werden nur einige wenige stellvertretend für viele andere zitiert. Die erste Definition von Barry Boehm streicht vor allem den Aspekt des Wissenstransfers von Wissenschaft in Richtung Praxis hervor:

“The practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate, and maintain them“ [Boeh79]

Demnach kann es Software Engineering erst wirklich geben, wenn alle Praktiken ausreichend wissenschaftlich definiert und überprüft wurden.

Eine sehr ausführliche Definition bietet Helmut Balzert für den Begriff Softwaretechnik als Synonym für Software Engineering an:

Softwaretechnik“: Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden, Konzepten, Notationen und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen. Zielorientiert bedeutet dies die Berücksichtigung z.B. von Kosten, Zeit, Qualität.“ [Balz98]

Das IEEE (Institute of Electrical and Electronic Engineers) schlägt eine sehr knappe, aber exakte Definition vor:

“The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.“ [IEEE90]

Entsprechend dieser Definition darf alles als Software Engineering betrachtet werden, was ein systematisches Vorgehen im Bereich der Softwareentwicklung, dem Betrieb und der Wartung von Software vorweisen kann. Wesentlich an dieser Definition ist noch die Forderung, dass das Vorgehen messbar sein muss, damit eine Verfolgung und Evaluierung des Vorgehens zum Zwecke der Verbesserung erfolgen kann.

### 1.3 Die professionelle Disziplin Software Engineering

Im letzten Jahrzehnt wurden nicht zuletzt durch grosse Organisationen wie IEEE oder ACM starke Bemühungen in Richtung einer Etablierung einer professionellen Disziplin Software Engineering unternommen. In letzter Zeit gab es vor allem drei Produkte als Ergebnis dieser Bemühungen: Ethische Richtlinien (Code of Ethics) für Software-Ingenieure ([ACM99]), ein Studienplan für ein Bakkalaureat Software Engineering ([ACMO3]) und eine Software Engineering-Wissensbasis (Software Engineering Body of Knowledge, [SWEBO3]).

#### 1.3.1 Code of Ethics

Der Code of Ethics wurde in Zusammenarbeit zwischen den beiden renommierten Organisationen im Bereich der Computertechnik IEEE und ACM erstellt.

Laut dem Code of Ethics sollen sich Software-Ingenieure verpflichten, aus Analyse, Entwurf, Implementierung, Testen und Wartung von Software eine nützliche und respektierte Profession zu machen. In Übereinstimmung mit deren Bekenntnis zu Gesundheit, Sicherheit und dem Wohlergehen der Öffentlichkeit folgen sie diesen acht Prinzipien:

1. **Öffentlichkeit:** Software-Ingenieure sollen in Übereinstimmung mit dem öffentlichen Interesse handeln.
2. **Kunde und Arbeitgeber:** Software-Ingenieure sollen in ihren Handlungen das Interesse des Kunden und des Arbeitgebers in Übereinstimmung mit dem öffentlichen Interesse bestmöglich wahren.
3. **Produkt:** Software-Ingenieure sollen sicherstellen, dass ihr Produkt und damit verbundene Änderungen die höchstmöglichen professionellen Standards beachtet.

4. **Urteilsvermögen:** Software-Ingenieure sollen Rechtschaffenheit und Unabhängigkeit bei ihrem professionellen Urteil bewahren.
5. **Management:** Manager und Leiter von Software-Ingenieuren sollen sich einem ethischen Vorgehen im Management von Softwareentwicklung und -wartung verschreiben und diesen fördern.
6. **Profession:** Software-Ingenieure sollen die Integrität und Reputation der Profession in Übereinstimmung mit dem öffentlichen Interesse fördern.
7. **Kollegen:** Software-Ingenieure sollen ihren Kollegen gegenüber gerecht und unterstützend wirken.
8. **Selbst:** Software-Ingenieure sollen ein lebenslanges Lernen anstreben und den ethischen Ansatz in der Praxis fördern.

### 1.3.2 Computing Curriculum Software Engineering

Ebenfalls aus einer Zusammenarbeit zwischen IEEE und ACM wurde ein Vorschlag für einen Studienplan für ein Bakkalaureat Software Engineering erarbeitet. Diese Arbeit ist Teil eines größeren Projekts, in dem verschiedene Studienpläne im Bereich der Computertechnik erarbeitet werden. Ziel ist es, über eine Akkreditierung einen einheitlichen Ausbildungsstandard für Software Engineering an den Universitäten zu schaffen.

Der Vorschlag gliedert den Studienplan in folgende Wissensbereiche:

1. **Grundlagen der Computertechnologie:** Grundlagen der Informatik, Konstruktionstechnologien, Konstruktionswerkzeuge, formale Konstruktionsmethoden
2. **Mathematische Grundlagen & Grundlagen des Ingenieurwesens:** mathematische Grundlagen, Grundlagen des Ingenieurwesens, wirtschaftliche Grundlagen des Ingenieurwesens
3. **Professionelle Praktiken:** Gruppendynamik und -psychologie, Kommunikationstechniken, Grundlagen einer Profession
4. **Softwaremodellierung & Analyse:** Grundlagen der Modellierung, Modelltypen, Grundlagen der Analyse, Grundlagen der Anforderungsanalyse, Anforderungsfindung, Spezifikation und Dokumentation von Anforderungen, Validierung von Anforderungen
5. **Softwareentwurf:** Entwurfskonzepte, Entwurfsstrategien, Architekturentwurf, Entwurf von Anwenderschnittstellen, Detailentwurf, Entwurfswerkzeuge und Evaluierung des Entwurfs
6. **Softwareverifikation & -validierung:** V&V Terminologie und Grundlagen, Reviews, Testen, Anwenderschnittstellen testen und evaluieren, Problemanalyse und -bericht
7. **Softwareevolution:** Evolutionsprozess, Evolutionsaktivitäten
8. **Softwareprozess:** Prozesskonzepte, Prozessimplementierung
9. **Softwarequalität:** Konzepte und Kultur der Softwarequalität, Standards der Softwarequalität, Prozesse der Softwarequalität, Prozesssicherung, Produktsicherung
10. **Softwaremanagement:** Managementkonzepte, Projektplanung, Personal und Organisation im Projekt, Projektkontrolle, Software-Konfigurationsmanagement

Weiters wird empfohlen, dass sich Studenten im Laufe des Studiums auf einen oder mehrere der folgenden Systemtypen spezialisieren sollen: Netzwerkzentrierte Systeme,

Informationssystem und Datenverarbeitung, Finanzsysteme und e-Commerce-Systeme, fehlertolerante Systeme, Hochsicherheitssysteme, sicherheitskritische Systeme, eingebettete Systeme und Echtzeitsysteme, biomedizinische Systeme, wissenschaftliche Systeme, Telekommunikationssysteme, avionische Systeme und Fahrzeugsysteme, industrielle Prozesskontrollsysteme, Multimedia, Spiele und Unterhaltungssysteme, Systeme für kleine und mobile Platt- formen, Agenten-basierte Systeme.

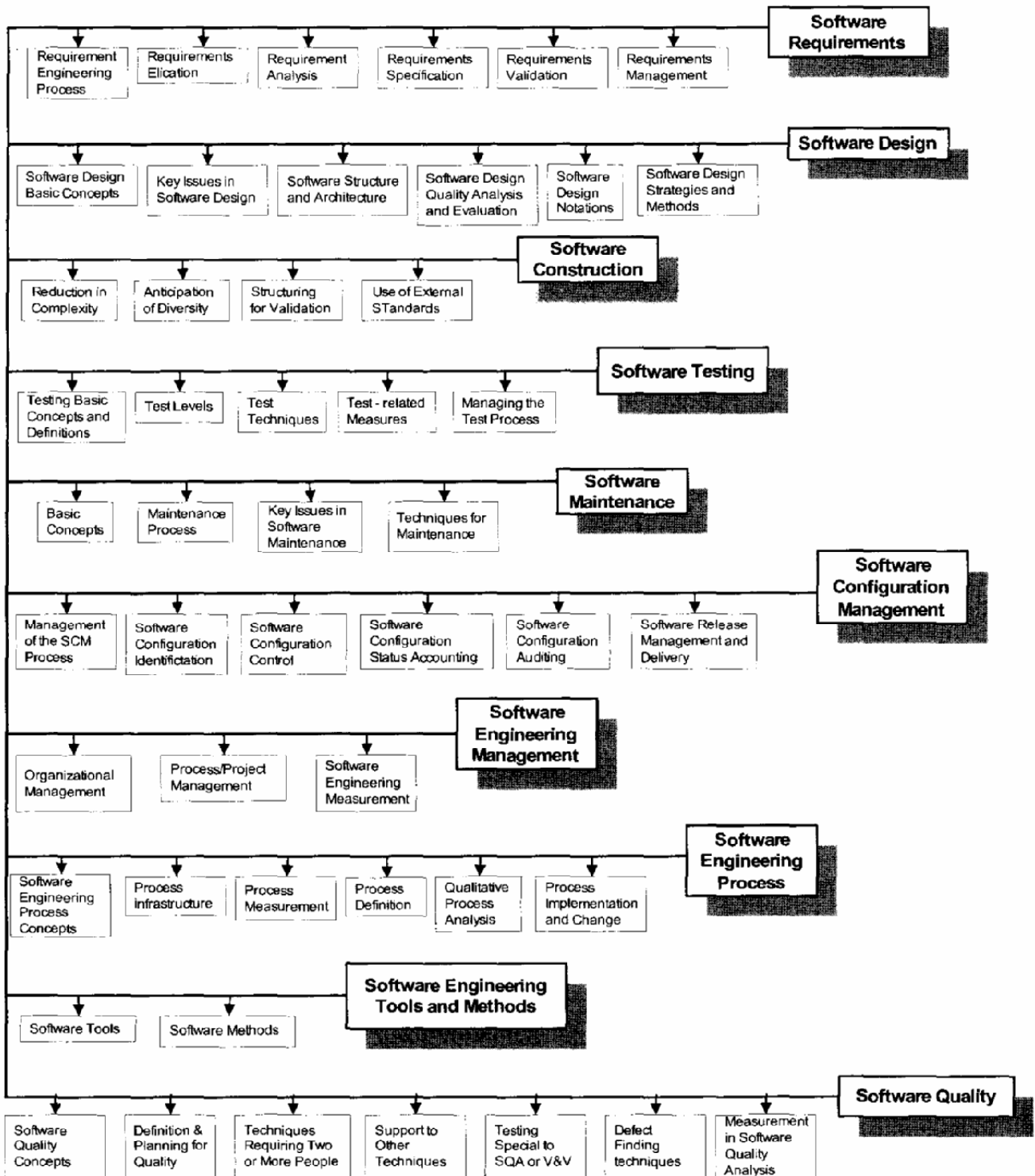


Abbildung 1.3: SWEBOK Knowledge Areas

### 1.3.3 Software Engineering: Body of Knowledge

Dieses Projekt wird von mehreren internationalen und nationalen Organisationen und Firmen getragen. Ziel des Projekts ist es, jenes Wissen darzustellen, welches erstens zur Profession Software Engineering zugerechnet wird und zweitens international als anerkannt betrachtet werden kann. Der Aufbau des Body of Knowledge erfolgte stark inkrementell. Es wurden zahlreiche Experten in die Feedbackzyklen eingebunden, um einen möglichst breiten Konsens zu erreichen.

Das Wissen wurde ursprünglich vor allem aus der Literatur zusammengetragen. Dabei dienten in erster Linie internationale erfolgreiche Lehrbücher zum Thema Software Engineering als Grundlage. Ein wesentlicher Mangel an dem Projekt ist, dass nur englischsprachige Quellen verwendet wurden.

Die in SWEBOK definierten Wissensgebiete (Knowledge Areas (KA)) für Software Engineering und ihre Unterteilung in Untergebiete sind in Abbildung 1.3 dargestellt.

## 1.4 Perspektiven des Software Engineering

Die Beschäftigung mit Software Engineering umfasst sowohl in der Praxis als auch in der Forschung mehrere Bereiche, deren Zusammenspiel es zu einem komplexen Ganzen werden lässt.

### 1.4.1 Formale Aspekte des Software Engineering

Als wesentliches Fundament jeder Erzeugung von Software sind natürlich die formale Logik und ihre unmittelbare Anwendung in Programmiersprachen und Compilern zu betrachten. Formale Elemente finden aber zum Beispiel in Form von Grammatiken auch in anderen Bereichen ihren Eingang und haben wesentliche Bedeutung (z.B. Schemen in XML oder IDL (Interface Definition Language) zur Festlegung von Schnittstellen). Auch die formale Spezifikation von Software (z. B. mit der Spezifikationssprache Z) basiert auf formalen Konstrukten.

Nicht zu vergessen ist die Bedeutung von formalen Beweisen im Bereich der Verifikation von Software. Der vollständige Beweis der Richtigkeit von Software ist aufgrund des Aufwandes aber nur in Bereichen ökonomisch sinnvoll, in denen die Wahrscheinlichkeit eines Auftretens von Fehlern z.B. zum Schutz von Menschenleben auf realistisch nahe null reduziert werden muss (z.B. Medizin, Luftfahrt, Anlagensteuerung und -sicherung wie in Atomkraftwerken).

Aufgrund der spezifischen Ausrichtung dieses Buches auf administrative Software im weitesten Sinn, spielen formale Aspekte in der weiteren Betrachtung von Software Engineering keine tragende Rolle.

### 1.4.2 Technische Aspekte des Software Engineering

Die jeweils aktuelle Technologie bestimmt ganz wesentlich den Rahmen des Machbaren. Nicht zuletzt scheitern zahlreiche Projekte an Über- oder Unterschätzung der verwendeten Technologie. Überschätzung im Sinne, dass die Technologie nicht genügend Unterstützung für ein Vorhaben liefert und damit das Projekt entweder gar nicht oder nur unter erheblichen Mehraufwand realisierbar ist. Unterschätzung im Sinne, dass die Technologie ob ihrer Komplexität oder Neuheit nicht beherrscht wird und daher mehr Probleme verursacht als löst. Eine überlegte Auswahl der zu verwendenden Technologie und deren ziel-führenden Einsatz ist damit einer der wesentlichen Eckpfeiler jedes Softwareentwicklungsprojekts.

### 1.4.3 Ingenieuraspekte des Software Engineering

Die realistische Planung eines Softwareentwicklungsprojekts und damit ein möglichst vorhersagbarer Projektverlauf ist ein wesentliches Qualitätskriterium. Der Anspruch jeder Ingenieurdisziplin ist die Verfügbarkeit von Methoden, diese Planung und die ebenfalls erforderliche Überprüfung der Realisierung nach Plan durchführen zu können. Der Anspruch an jeden Ingenieur im Einzelnen sollte der Wille sein, diese Methoden auch anzuwenden und verantwortungsvoll zu agieren, auch wenn Probleme auftreten und auch als solche erkannt werden.

Ein weiterer Aspekt eines professionellen Ingenieurwesens ist ein gewisses Berufsethos. Dieser sollte jeden Ingenieur dazu anhalten, nach bestem Wissen und Gewissen möglichst hochwertige Produkte herzustellen, die den Ansprüchen des Kunden und der Anwender nach Massgabe des Machbaren bestmöglich genügen und dabei Schutzbedürfnisse von Einzelpersonen, Gruppen oder der ganzen Gesellschaft nicht verletzen.

### 1.4.4 Gestalterische Aspekte des Software Engineering

Im Zuge eines Softwareentwicklungsprojekts treten zahlreiche gestalterische Aspekte auf, welche beachtet werden müssen, um die Akzeptanz des Systems schlussendlich gewährleisten zu können. Schon am Beginn hat die konkrete Formulierung und Strukturierung der Anforderungen wesentlichen Einfluss auf die endgültige Gestalt des Systems. Die weitere Verfeinerung und Planung der Umsetzung der Anforderungen in der Analyse und dem Entwurf prägen die Gestalt des Systems immer weiter.

Die bekanntesten und auch am öftesten bewusst wahrgenommenen gestalterischen Aufgaben in der Softwareentwicklung betreffen die Anwenderschnittstelle sowie Anwenderdokumentation in Form von Handbüchern und der Online-Hilfe.

### 1.4.5 Ökonomische Aspekte des Software Engineering

Die Entwicklung und der Einsatz von Software bzw. Systemen mit Softwareanteil muss wie jede andere technische Entwicklung natürlich auch einer Kosten-Nutzen-Rechnung standhalten. Die Bewertung des Nutzens von Software ist zurzeit noch ein schwieriges Unterfangen. (Wie bewertet man beispielsweise den Nutzen eines reines Informationssystems?). Im Bereich der Kosten unterscheidet sich die Planung von den tatsächlichen Kosten oft massgeblich. Eine praktikable ökonomische Bewertung von Softwareentwicklungsprojekten ist damit mehr als schwierig, wenn gleich auch sehr wichtig. Wesentliche Fragen der ökonomischen Sicht wären beispielsweise:

- Soll ein System mit bestimmten Anforderungen neu gebaut oder zugekauft oder ein bestehendes Systems adaptiert werden?
- Rechnet sich der Einsatz von moderner (teurer) Technologie (vor allem auch angesichts der dadurch zu erwartenden Risiken)?
- Rechnet sich die Ablösung eines bestehenden Systems durch ein neues angesichts der Erstellungskosten, Kosten der Inbetriebnahme und der zu erwartenden Effizienzsteigerungen im Produktivbetrieb?
- Rechnet sich der Zukauf von Komponenten und die Integration in ein eigenes (gerade in Entwicklung befindliches oder bereits laufendes) System?

## 1.5 Grundlegende Fragestellungen der unterschiedlichen Leser

Dieser Abschnitt versucht, zentrale Fragen des Leserkreises zu beantworten. Dazu unterteilen wir den im Vorwort angesprochenen Leserkreis in drei Untergruppen: Studenten, Entwickler und Projektleiter. Die wesentlichen Fragen und Verweise zu den Teilen, in denen die verschiedenen Aspekte der Probleme behandelt werden, sind im Folgenden aufgeführt.

Die Fragestellungen von Studenten beziehen sich auf die Natur von Softwareprojekten und eine Begründung des Nutzens von Entwicklungsmethoden und -prozessen im Allgemeinen. Die zentralen Fragen von Entwicklern betreffen vermehrt den Nutzen des hier vorgestellten Vorgehens und der hier vorgestellten Methoden. Für Projektleiter werden wichtige Fragen bezüglich Team-, Projekt- und Qualitätsmanagement beantwortet.

### 1.5.1 Studenten

Studenten besitzen oft eher wenig Erfahrung mit der Herstellung von Software. Dementsprechend grundlegend sind auch ihre Fragen in Bezug auf das Software Engineering.

#### Was ist ein Softwareentwicklungsprojekt?

Viele Studenten haben im Laufe ihres Studiums das eine oder andere kleine Programm geschrieben. Manche benötigen diese Programme zur erfolgreichen Absolvierung einer (Programmier-)Übung, andere versuchen sich mit nützlichen Makros den studentischen Alltag zu erleichtern. Für derartige kleine und überblickbare Aufgaben ist keinerlei Dokumentation oder Form von Methodik notwendig. Der Student hat eine Idee und setzt diese direkt in ein Programm um. Dieses wird nach verfügbarer Zeit und persönlichem Qualitätsanspruch (z.B. nützliche Fehlermeldungen bei Falscheingaben) so lange verbessert, bis es brauchbar ist.

Bei der Erstellung von grösseren Programmen, an denen mehrere bis sehr viele Personen beteiligt sind, und welche bis zu mehrere hundert Einzelfunktionen aufweisen (z. B. eine moderne Textverarbeitung), kann nicht jede handelnde Person nach eigenem Ermessen so lange etwas programmieren, bis vielleicht das gewünschte Ergebnis herauskommt.

Vielmehr muss die Programmierung einer solch umfangreichen Software genau organisiert, koordiniert und laufend unterstützt werden. Der Programmierer erhält so während der gesamten Dauer der Programmierung Rückmeldungen über seine Arbeit und kann gezielt zum Projektfortschritt beitragen. Die bevorzugte Organisationsform für die Software-Erstellung in Arbeitsgruppen mit mehreren Personen ist ein Projekt.

#### Was geschieht in einem Projekt?

Bei grossen Programmen ist das erste wesentliche Problem, herauszufinden, was dieses Programm überhaupt können soll. In der Analyse werden diese so genannten Anforderungen mit den zukünftigen Anwendern besprochen und festgehalten.

Um die Anforderungen in ein funktionierendes Programm umzusetzen, muss der Aufbau des Programms genau geplant werden. Dies geschieht im so genannten Entwurf. Ohne Entwurf wäre es für den einzelnen Programmierer unmöglich, die Programmierung richtig zu erledigen, da er keine Übersicht über das gesamte Programm erhalten könnte (genauso wenig ist es einem Tischler möglich, ohne Werkzeichnung einem Werkteil die richtige Form und Abmessungen zu geben und die Bohrlöcher für die Verschraubung an die richtige Stelle zu setzen).

Bei der Programmierung, meist als Implementierung bezeichnet, werden verschiedenste Werkzeuge eingesetzt, um die Produktivität zu steigern und die Fehlerrate zu senken bzw. die Fehlerfindung zu erleichtern (der Tischler verfügt mittlerweile auch über wesentlich mehr

Hilfsmittel als nur eine Säge und einen Hammer). Der Entwicklung solcher Werkzeuge wird grosse Beachtung geschenkt, da sie wesentliche Auswirkungen auf eine mögliche Produktivitätssteigerung haben. Bei der Programmierung leistet methodisches Vorgehen einen ebenso wichtigen Beitrag wie die richtigen Werkzeuge selbst (ein Tischler zeichnet sich Schnittlinien vor, um die richtige Form zu schneiden, oder beginnt mit der Verleimung erst, wenn alle Einzelteile fertig sind).

Während und nach der Programmierung muss das fertige Programm ausreichend getestet werden, um seine Fehlerfreiheit zu garantieren. Bei der Durchführung von Tests von grossen Programmen ist ein systematisches Vorgehen von besonderer Bedeutung, da sonst zahlreiche Fehlerquellen übersehen werden könnten. Der einzelne Programmierer ist nicht mehr in der Lage, alle Auswirkungen seines Programmtails auf das gesamte Programm abzuschätzen. Im Rahmen von ausführlichen Tests werden Fehler entdeckt und deren Verbesserung veranlasst.

Im Gegensatz zu dem Makro eines Studenten wird das Programm nicht mehr von derselben Person programmiert und verwendet. Daher muss das fertige Programm vor der Inbetriebnahme noch bei den zukünftigen Benutzern ordnungsgemäss installiert und entsprechende Schulungen müssen durchgeführt werden.

### **Wozu brauche ich einen dokumentierten Softwareentwicklungsprozess?**

Bereits aus dieser vorangegangenen kurzen Darstellung der Softwareentwicklung im grösseren Rahmen lassen sich einige Tätigkeiten erkennen, welche durchgeführt werden müssen, um ein funktionierendes und brauchbares Produkt zu erhalten. Diese Tätigkeiten müssen ausführlich beschrieben und untereinander abgestimmt werden, damit nicht jeder Entwickler dieses Vorgehen neu definiert, was unweigerlich zu Chaos führen würde. Das konkrete Vorgehen in einem Projekt, d.h. die Abfolge aller Tätigkeiten zur Erzeugung der Software, wird Softwareentwicklungsprozess genannt. In einem formal definierten Softwareentwicklungsprozess werden alle diese Tätigkeiten beschrieben und miteinander in Verbindung gesetzt. Die Notwendigkeit der Verwendung eines Prozesses wird im Kapitel 3 erläutert.

### **1.5.2 Entwickler**

Entwickler haben bereits vielfältige Erfahrungen bei unterschiedlichen Projekten gesammelt. Für sie ist vor allem wichtig zu erfahren, ob ihre Vorgangsweise bei der Entwicklung von Software bisher optimal war oder verbessert werden könnte.

### **Was ist mindestens erforderlich, um ein vernünftiges Produkt zu erstellen?**

Um ein gutes Produkt zu erhalten, können verschiedene Massnahmen bei seiner Erzeugung getroffen werden, um die Qualität sicherzustellen oder im Vergleich zu früheren Produkten zu steigern. Bei der Produktion von Software, an der mehr als eine Person beteiligt ist und/oder der Aufwand eine Personenwoche überschreitet, ist ein strukturiertes und methodisches Vorgehen entscheidend.

Strukturiertes Vorgehen garantiert die Beachtung aller wichtigen Punkte, die für ein funktionierendes Endprodukt relevant sind, und stellt sicher, dass dieses Endprodukt auch den ursprünglichen Anforderungen entspricht. Ohne ein strukturiertes Vorgehen werden wichtige Tätigkeiten ausgelassen oder in einer falschen Reihenfolge ausgeführt, was aufgrund fehlender oder falscher Informationen zu verminderter Qualität führt.

Eine methodische Vorgehensweise garantiert ein einheitliches Vorgehen. Die Zwischenprodukte sind untereinander abgestimmt, sodass ein Produkt stets eine gute

Grundlage für ein Folgeprodukt bildet und diese nahtlos aneinander anschließen. Bei einem methodischen Vorgehen sind die Produkte meist so gestaltet, dass sie keine redundanten Informationen enthalten und somit Inkonsistenzen weitgehend vermieden werden können.

Strukturiertes und methodisches Vorgehen ermöglicht innerhalb einer Projektgruppe Kommunikation ohne Missverständnisse, da alle Beteiligten über dieselben Tätigkeiten und Produkte sprechen.

Dieses Buch präsentiert in der Praxis erprobte Methoden gemeinsam mit einem Entwicklungsprozess als strukturiertes Vorgehen. Der Prozess und seine Grundlagen werden im Kapitel 3 einführend dargestellt, die Methoden sind Thema der Kapitel 8 bis 13.

## Was ist der Unterschied zwischen strukturierten und objektorientierten Ansätzen?

Grundsätzlich können sowohl strukturierte als auch objektorientierte Methoden zu einem erfolgreichen Softwareprojekt führen. Eine objektorientierte Methode allein ist kein Garant für einen Erfolg. Die erfolgreiche Anwendung einer Methode, welchem Paradigma sie auch folgt, ist stets von den handelnden Personen abhängig.

Eine objektorientierte Methode bietet heutzutage aufgrund mehrerer Voraussetzungen eine besonders gute Basis für erfolgreiche Softwareentwicklung: Der objektorientierte Ansatz lässt sich bei Verwendung entsprechender Technologien (einer objektorientierten Programmiersprache) bei allen Tätigkeiten eines Projekts verfolgen. Es kommt zu einer einheitlichen Betrachtung vom Problem bis zur Lösung, und es müssen keine Abstraktionslücken oder Methodenbrüche künstlich behoben werden. Die Prinzipien der Objektorientierung (Datenkapselung, Vererbung, Polymorphismus usw.) bieten auch eine Grundlage für eine qualitativ gute Implementierung, sofern sie konsequent angewendet werden. Die Aufteilung des Systems in Objekte bzw. deren Klassen lässt sich für eine gute Strukturierung nutzen und führt zu einem besser wartbaren Code als bei strukturierten Ansätzen.

## Wie wird ein Softwareentwicklungsprozess systematisch verbessert?

Ein Softwareentwicklungsprozess lebt von der fortlaufenden Weiterentwicklung und der ständigen Anpassung der verwendeten Methoden an sich ändernde Bedingungen. Wird ein Prozess nicht weiterentwickelt und angepasst, so kann er möglicherweise nicht mehr sicherstellen, dass Projekte ökonomisch Gewinn bringend und qualitativ verkaufbar realisiert werden können.

Da jegliche Veränderung im Prozess oder der Wechsel von Methoden mit einem Aufwand aufgrund notwendiger Schulungen und der völlig normalen Startprobleme mangels Erfahrung verbunden ist, sollten Prozesse so lange wie möglich beibehalten werden können und für eine möglichst grosse Bandbreite an Projekten einsetzbar sein.

Für die Einführung von neuen Abläufen, Technologien und Methoden im Rahmen des Softwareentwicklungsprozesses gibt es in einem Unternehmen vielleicht eigens dafür bestimmte Personen oder eine Abteilung. Solche Änderungen sollten in diesem günstigen Fall keineswegs vom Entwickler selbst vorgenommen werden, da dadurch Standards verletzt werden und die erstrebte einheitliche Softwareentwicklung unmöglich gemacht wird.

### 1.5.3 Projektleiter

Projektleiter haben genug Erfahrung, um ein Projekt erfolgreich durchführen zu können. Gerade in einer sich so rasch weiterentwickelnden Disziplin wie dem Software Engineering ist

es aber auch für sie wichtig, immer über aktuelle Entwicklungen informiert zu sein und ihre wertvollen praktischen Erfahrungen mit wichtigem theoretischen Knowhow zu ergänzen.

## Warum brauche ich vorgegebene Entwicklungsmethoden und einen Entwicklungsprozess?

Jeder Projektleiter sollte in der Lage sein, aufgrund seiner Erfahrungen ein Projekt zu führen und es zu einem guten Abschluss zu bringen. Dennoch bringt jedes Projekt neue Herausforderungen mit sich, die neue Entscheidungen erfordern. Bei all jenen Dingen, bei denen der Erfahrungsschatz eines Projektleiters nicht ausreicht, können ein definierter Prozess und die Anwendung von Methoden hilfreich sein.

Das Anwenden eines Prozesses hat den Vorteil, dass das Vorgehen für Aussenstehende (z. B. den Kunden oder zukünftige Mitarbeiter) und auch für die Mitarbeiter nachvollziehbar ist. Dies stärkt das Vertrauen in das Vorgehen und erspart lange Diskussionen über den Sinn oder Unsinn von Tätigkeiten oder Produkten.

Eine standardisierte Methode und die damit verbundene Notation nimmt zahlreiche Dokumentations-, Arbeits- und Prozessentscheidungen ab, die sonst immer wieder getroffen werden müssen (z. B. wie sieht ein Klassendiagramm aus oder wie kann man die Anforderungen brauchbar dokumentieren). Sind diese Dinge festgeschrieben, bleibt dem Projektleiter viel mehr Zeit für andere Entscheidungen, die meist wesentlich wichtiger sind (z.B. Lösung technischer oder organisatorischer Probleme).

Je detaillierter und umfangreicher die Beschreibung der Methode ist, desto mehr Anpassungen und Abstriche sind bei einem konkreten Projekt möglich<sup>1</sup>. Es können für bestimmte Projekte (z. B. kleine Projekte) Produkte verkleinert, ausgelassen oder modifiziert werden, ohne den Erfolg des Projekts zu gefährden. Somit kann eine gut beschriebene Methode für viele Projekte unterschiedlicher Grösse und Art verwendet werden. Dies ermöglicht es, dass der durch die Methode gesetzte Standard bei den meisten Projekten weitgehend beibehalten werden kann. Je öfter dieselbe Methode eingesetzt werden kann, desto höher wird — aufgrund der Erfahrung mit der Methode — die Produktivität sein.

## Wie kann ich überprüfen, ob ich auf dem richtigen Weg bin?

Die Grundlage für ein Projekt bildet stets ein Projektplan. Dieser gibt Termine vor, zu denen Tätigkeiten gestartet und beendet werden sollten. Nach dem Abschluss von besonderen Tätigkeiten (meist Tätigkeiten, welche einen grösseren Arbeitsschritt oder eine Phase beenden) muss überprüft werden, ob die Ziele gemäss des Projektplans erreicht worden sind. Dazu gibt es formale Verfahren, in denen die Qualität von Produkten und Prozessen überprüft werden kann (z. B. Reviews, Inspektionen usw.).

Aufgrund des Projektfortschritts kann es in einem Projekt zu Terminänderungen kommen. Der Projektplan wird entsprechend der jeweils neu entstehenden Situation angepasst. Auch die Termine für Überprüfungen müssen mit verändert werden. Um zwischen den grossen Überprüfungen nicht die Übersicht zu verlieren, sollte der Projektleiter von den Entwicklern regelmässig Berichte verlangen, in denen ihre erledigten Tätigkeiten beschrieben sind und Probleme aufgeführt werden.

## Welche Auswahl an Produkten, Prozessen und Personen ist für mein Projekt wesentlich?

Jedes Projekt erfordert einen darauf zugeschnittenen Entwicklungsprozess. Der Entwicklungsprozess definiert die daran beteiligten Personen, Prozesse und Produkte, die für eine erfolgreiche Durchführung des Projekts notwendig sind. Die Auswahl dieser Faktoren ist konkreten Projekttyp abhängig. Dies hat einen wesentlichen Einfluss sowohl auf die Anzahl der

notwendigen Personen, Prozesse und Produkte als auch auf deren konkrete Auswahl bzw. Gestaltung.

Die Bestimmung von Projekttypen und die Auswirkungen eines konkreten Projekttyps auf Personen, Prozesse und Produkte werden in Kapitel 2 dargestellt.

### **Wie misst man die Qualität eines Softwareentwicklungsprozesses?**

Die Qualität eines Softwareentwicklungsprozesses kann an der Qualität seiner Produkte und dem dafür benötigten Aufwand gemessen werden. Um die Qualität von Produkten vergleichend bestimmen zu können, müssen dafür geeignete quantitative Methoden geschaffen werden. Diese so genannten Metriken können für die Beurteilung der Produkte und des Erzeugungsprozesses verwendet werden.

Eine weitere Möglichkeit der Beurteilung eines Prozesses ist eine qualitative Beurteilung in Form von Audits. In solchen Überprüfungen wird der Prozess von eigens dafür geschulten Prüfern in Gesprächen mit der Unternehmensführung und auch mit den Mitarbeitern analysiert. Dabei können mögliche Schwächen aufgedeckt, vor allem aber auch ein angestrebtes Qualitätsniveau des Softwareentwicklungsprozesses bestätigt werden. Solche qualitativen Beurteilungen finden auch im Rahmen von Zertifizierungen statt. Diese bescheinigen einem Unternehmen einen bestimmten Qualitätsstandard, auf den sich unter anderem Kunden verlassen und berufen können.

In Abhängigkeit von der festgestellten Qualität eines Entwicklungsprozesses kann der bestehende Prozess unter Berücksichtigung von erwünschter Produktivitätssteigerung und geänderten Rahmenbedingungen verbessert werden.