Software Engineering

Was ist Software

Software ist nicht nur ein Programm sondern auch alle dazugehörigen Informationen, welche von Computern verarbeitet werden. Des Weiteren zählen auch die Dokumentationen und die Konfigurationsdaten, welche für ein ordnungsgemäß, funktionierendes Programm benötigt werden aus als Software. Somit Zählen zugehörige Informationen und Materialien zur Unterstützung von Installation, Programmausführung, zur Reparatur und zur Erweiterung des Programms als Software.

Was beinhaltet Software

- Software beinhaltet das ausführbare Programm und zugehörigen Daten
- ggf. zugehörige Konfigurationsdateien
- die Systemdokumentation auch Systemanforderungen genannt
- die Benutzerdokumentation oder Handbuch / Handbücher
- selbst die zugehörige (Hersteller-) Website über das eigentliche Programm, um sich beispielsweise über Probleme zu informieren oder um Updates herunterladen zu können.

Unterscheidung von Software

generische Produkt

Dabei handelt es sich Software, welche (meist) in großen Mengen benutzerunabhängig hergestellt hergestellt wird, beispielsweise Anwendungssoftware von Herstellern wie Microsoft, Acrobat oder Spielherstellern. In USA auch als shrink-wrapped software“ bekannt, da diese in der Vergangenheit, als ein im Karton, eingeschweißtes Produkt mit allerhand Zubehör verkauft wurde. z.B. Microsoft Office, Open Office, Acrobat Reader, usw. Diese Software wird und wurde für den Markt als „fertiges“, ganzes Produkt verkauft.

Sonderanfertigung / Individualsoftware

Die Individualsoftware, welche als Auftragssoftware genau für die Bedürfnisse eines Benutzers oder Unternehmens entwickelt wurde. Zum Beispiel SAP Software Pakete welche genau für den Betriebsablauf für ein bestimmtes Unternehmen wie beispielsweise für Lufthansa, Deutsche Bank oder NSA entwickelt wurde.

Dabei sind die Grenzen nicht immer erkennbar, da beispielsweise ERP-Software (Enterprise-Resource-Planning Software) wie SAP R/3 oder heute SAP S/4HANA, Oracle ERP Cloud oder auch Microsoft Software aus einer Kernkomponente besteht, welche als generisches Produkt angesehen werden kann und dann an die Arbeitsabläufe eines bestimmten Unternehmens angepasst wird. Die Anpassung macht dann aus einer generischen Software und hinzufügen von bestimmter Schnittstellen eine Sonderanfertigung.

Eigenschaften von Software

Software ist im Gegensatz zu Hardware „einzigartig“, da es keine echten physischen Grenzen gibt. So nutzt sich Software beispielsweise im Gegensatz zu Hardware nicht ab, es gibt zwar keine Ersatzteile und trotzdem muss die Software ständig aktualisiert und gewartet werden. Diese Aktualisierungen geben der Software die Möglichkeit, mit Änderungen in der Umgebungen umzugehen. Zum Beispiel 1999/2000 die Umstellung von DM auf Euro. Dadurch wird sichergestellt, dass die Software nicht veraltet. Aber auch Kompatibilitäts-Anpassungen dass die Software auch auf neuerer Hardware (z.B. schnellere CPU, größerer Speicher, andere Architektur) lauffähig bleibt. Andernfalls altert die Software und kann auch veralten und verliert Ihren Wert und Zweck.

Qualität von Software ist schwer zu „messen“ im Gegensatz zu Hardwarekomponenten. Wir definieren beispielsweise Hardware über diverse Werte, wie Stromverbrauch, Kapazität oder Geschwindigkeit. Und können diese dann vergleich und Ihr ein Preis-Leistungsverhältnis zuordnen. Wie definiert man die Qualität von Software? Kann man Software überhaupt messen oder vergleichen? Sind Dinge, wie die Anzahl von Codezeilen dafür ausschlaggebend? Oder Messen wir immer nur die Hardware auf der die Software läuft?

Gute Software zeichnet sich durch guten Code aus. So ist ein Code, mit vielen unnötigen Sprüngen oder Abfragen langsamer. Darum ist ein durchdachtes Konzept so wie ein gutes (Code-) Design sehr wichtig. Strukturierter und durchdachter Code ist qualitativ hochwertiger Code.

Lebenszeit von Software

Während Hardware ständig altert und darum auch öfter mal ausgetauscht werden muss, ist die Lebenszeit von Software unbestimmt. Je besser Software gewartet und angepasst wird um so länger lebt diese. Darum altern Betriebsystem im Gegensatz zur Hardware im Vergleich nur halb so schnell. Im Schnitt, überlegt ein aktuelles Betriebsystem zwei Computer, bevor es ein Versions-Upgrade auf eine neue Version gibt. Anwendersoftware lebt dagegen noch ein vielfaches länger.

Vielleicht kennen Sie es ja selbst, dass in der Regel Hardwareänderungen / Hardwareaufrüstung wie beispielsweise Festplatten (Umrüstung von HDD auf SSD), Monitore (17″ auf 21″) oder Drucker häufiger gewechselt werden als ein Betriebsystem.

Und Anwendungssoftware wie beispielsweise Microsoft Office Pakete, welche zusammen mit dem Betriebsystem Windows 7 erworben wurden noch heute auf Ihrem Windows 8 oder Windows 10 Betriebsystem laufen.

Die Software-Krise

Der Begriff „Software Crisis“ oder Software-Krise wurde in den 50er und 60er Jahren geprägt und bezieht sich hierbei auf mehrere Probleme:

- Die Preise für neuere und bessere Hardware sanken, im Gegenzug stiegen aber die Kosten für die Entwicklung von Software
- Programme konnte nicht rechtzeitig fertiggestellt werden, da die Anpassungen an neue

Hardware nicht im Budget lagen
- Die ausgelieferte Software war oft unvollständig und enthielt zu viele Fehler

Dazu kamen dann noch technologische Probleme wie:

- Mangel an geeigneten Programmiersprachen
- Mangel an Erfahrung bei der Umsetzung
- Mangel an Entwicklungsumgebung / Entwicklungssoftware

Software Engineering

Software Engineering bezieht die disziplinierte Anwendung von technischen, wissenschaftlichen und mathematischen Prinzipien zur wirtschaftlichen Herstellung von qualitativ hochwertiger Software. Dabei bezieht sich Qualität auf den Grad, in dem ein Produkt seinen Anforderungen oder den Bedürfnisse des Anwenders entspricht.

Ich beziehe mich dabei auf die Aussagen von W.S. Humphrey, aus dem seinem Buch The Software Engineering Process: Definition and Scope; ACM SIGSOFT Software Engineering Notes, Vol. 14, Issue 4, 1989

Der Begriff Software Engineering entstand durch das systematische, disziplinierte, quantifizierbare Herangehen an die Entwicklung einer Anwendung, den Betrieb und die Wartung von Software. Also unter der Anwendung von Ingenieurwissen eine Software zu entwickeln.

Der Duden deklariert Informatik wie folgt: Der Begriff Software-Engineering steht auch für die Auffassung, dass die Erstellung, Anpassung und Wartung von Programmsystemen kein “künstlerischer”, sondern vorwiegend ein ingenieurmäßig verlaufender Prozess ist.

Bereiche der Software-Entwicklung

- Software Anforderungen: Die Anforderungen definieren, was von den Systemen erwartet wird.
- Software-Design: Wie ist das System aufgebaut?
- Softwaretest: Die systematische Identifizierung (und Beseitigung) von Fehlern.
- Software-Wartung: Anpassung an Veränderungen und Fehlerbehebung
- Software Konfigurationsverwaltung: Verwaltung verschiedener Versionen und Konfiguration einer Software.
- Software Engineering-Prozess: Definition und Verbesserung von Softwareentwicklungsprozessen.
- Software-Engineering-Tools und -Methoden
- Software Qualitätssicherung:Überprüfung auf korrektes Verhalten

System und Software Engineering

Systembezogene Aktivitäten, z.B. die Definition des Gesamtsystems Ziele und Anforderungen, Zuordnung von Systemfunktionen zwischen Hardware und Software, Hardware- / Software-Schnittstellen definieren, so wie Akzeptanztests sind wesentlicher Bestandteil des Software Engineering, aber auch Teil von System Engineering. Die Grenzen verwischen hier stark. Da Software

Engineering ein Teil des Systems Engineering ist.

Erfolgsquoten bei der Entwicklung

Die Standish Group, Boston, Massachusetts verkündete im April 23, 2009: „Die diesjährigen Ergebnisse zeigen einen deutlichen Projektrückgang.

32% aller Projekte waren erfolgreich (wurden pünktlich fertiggestellt, lagen im Budget und beinhalteten alle erforderlichen Eigenschaften und Funktionen).

44% aller Projekte wurden angefochten, da diese was zu spät, über dem Budget und/oder nicht mit all denen vom Kunden geforderten Eigenschaften und Funktionen ausgeliefert wurden.

24% aller Projekte sind fehlgeschlagen, das heisst, sind vor der Fertigstellung storniert, nie an den Kunden ausgeliefert worden oder wurden nie benutzt.

Diese Zahlen bedeuten einen starken Abwärtstrend bei der Erfolgsquote aus vorangegangenen Studien sowie eine signifikante Steigerung der Anzahl von Fehlern in Software.“

Scheitern von Softwareprojekten

Softwareprojekte scheitern aus verschiedenen Gründen. Beispielsweise gilt ein Softwareprojekt als fehlgeschlagen, sobald das Projekt nicht pünktlich an den Kunden ausgeliefert wurde oder nicht mehr in dessen Budget liegt.

Gründe für ein Scheitern sind dass die Anforderungen und Systemabhängigkeiten im Vorfeld nicht genau definiert wurden. So sind beispielsweise Anpassungen an die Anforderungen der Software während der Entwicklung sehr viel einfacher als Änderungen der Hardware. Software muss für Hardwareänderungen und Hardwareprobleme geeignet sein. So darf eine Software nicht plötzlich falsch rechnen, nur weil weniger Speicher vorhanden ist.

Ein weiteres bekannte Problem, welche oft zum Scheitern von Softwareprojekten verantwortlich ist, ist der Mangel an Werkzeugen, Mangel an Erfahrung bei der Umsetzung, Mangel an ausgebildeten Entwicklern, so wie falsche Planung schon bei Beginn des Projektes.

Häufig werden auch die komplexen und innovativen Hardwaresysteme auf denen die Software laufen soll unterschätzt, wodurch der straffe Zeitplan zur Erstellung der Software nicht eigehalten werden kann. Beispiele dafür wären die verspäteten Softwareauslieferungen für den Airbus A380, der Boeing 787 oder das Apple iPhone. Beim Software Engineering geht es darum, „die richtig, funktionale Software“ zu entwickeln und nicht darum 100mal eine Software für den A380 zu entwickeln.

Fünfzehn Prinzipien der Softwareentwicklung

1. Qualität hat immer höchste Priorität
2. Hochwertige Software ist immer möglich

3. Geben Sie den Kunden früh und kontinuierlich Einsicht in sein Produkt
4. Beseitigen Sie mögliche Probleme, bevor diese in die Anforderungen eingehen und im Code landen.
5. Überlegen Sie sich auch immer alternative Designs und Ansätze
6. Verwenden Sie ein geeignetes Prozessmodell
7. Verwenden Sie verschiedene Sprachen für verschiedene Probleme (Assembler und C sind hardwarenahe und schnell, Java hingegen plattformunabhängig)
8. Minimieren Sie die intellektuelle Distanzen im Team
9. die verwendende Technologie steht vor der Wahl der Entwicklerumgebung. Bevor Sie eine Entwicklerumgebung verwenden, sollten Sie sich mit dieser vertraut machen und verstehen.
10. Machen Sie es richtig, bevor Sie es verbessern
11. regelmässig des Code durchsehen und zu verstehen ist meistens effektiver als den Code zu testen.
12. Gutes Management ist wichtiger als gute Technologie. Ihr Führungsstil muss an die aktuelle Situation angepasst werden.
13. Der Umgang mit ihrem Personal ist der Schlüssel zum Erfolg
14. Bleiben Sie immer skeptisch! Nur weil jeder etwas gut findet, oder jeder so macht, muss es nicht gut oder richtig sein.
15. übernehmen Sie die Verantwortung!

Zusammenfassung

Beim Software-Engineering geht es um das richtige Entwerfen von Software und nicht um das Runterschreiben von Code-Zeilen für Software.