

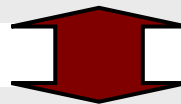


Perspektive auf Software-Entwicklung im Grundstudium

- ❑ Softwareentwicklung =
 Programmierung =
 Lösung eines kleinen und wohldefinierten Problems
- ❑ Problemdefinition =
 Übungsaufgabe (oder Aufgabe im Praktikum)
- ❑ Programm wird von einem einzigen Programmierer entwickelt
- ❑ Problem kann in begrenzter Zeit gelöst werden (1-2 Stunden)
- ❑ Beherrscht werden müssen:
 - » Programmiersprache
 - » Programmierwerkzeuge (Editor, Compiler, Debugger)
 - » Entwurf und Codierung von Algorithmen

Software-Entwicklung in der Realität

- ❑ Softwaresysteme sind sehr groß
 - » SAP-R3 (Standardsoftware für betriebswirtschaftliche Anwendungen) besteht aus mehreren Megazeilen (106) Code
 - » AXE-10 (Vermittlungssystem von Ericsson) umfasst etwa 10 Megazeilen
- ❑ Softwaresysteme leben sehr lange
 - » BS 2000 Betriebssystem (Siemens) wird seit den 70er Jahren benutzt
- ❑ Softwaresysteme werden von sehr großen Teams entwickelt
 - » Mehrere Tausend Entwickler arbeiten an SAP-R3
- ❑ Anforderungen sind ständigen Änderungen unterworfen
 - » Änderungen werden nicht immer nachgezogen
- ❑ Alle Softwaresysteme enthalten Fehler
- ❑ Trotzdem ist Korrektheit (und Zuverlässigkeit) absolut essenziell,
 - » z.B. Telekommunikationssysteme (Ausfallzeit: wenige Minuten pro Jahr)
 - » z.B. Raumfahrt-Steuerungssoftware (potenzieller Verlust von Milliarden €)



- ❑ Vorlesungen im *Hauptstudium*:
Betriebssysteme, Compilerbau, Datenkommunikation und verteilte Systeme, Deklarative Programmierung, Informationssysteme, Muster- und Spracherkennung, Parallele Programmierung, Computergraphik, Eingebettete Systeme, Computerunterstütztes Lernen und Wissensstrukturierung, Medieninformatik, Softwaretechnik

Prominente Software-Katastrophen: Fiscus (1)

- ❑ *Föderales integriertes standardisiertes computergestütztes Steuersystem*
 - » Ziel: einheitliche Software zur Datenverarbeitung in 650 deutschen Finanzämtern
 - » Bund-Länder-Projekt ⇨ 17 Auftraggeber
 - » Projektbeginn: 1991
 - » kalkulierte Kosten: 170 Millionen €

Prominente Software-Katastrophen: Fiscus (2)

- ❑ „Ergebnisse“ nach 13 Jahren Entwicklungszeit
 - » 1,6 Millionen Codezeilen
 - » 50.000 Seiten Dokumentation
 - » Zwei rudimentäre Anwendungen zur Erhebung von Grunderwerbssteuer sowie zur Eintreibung von Bußgeld
 - » Tatsächliche Kosten (geschätzt): 900 Millionen €



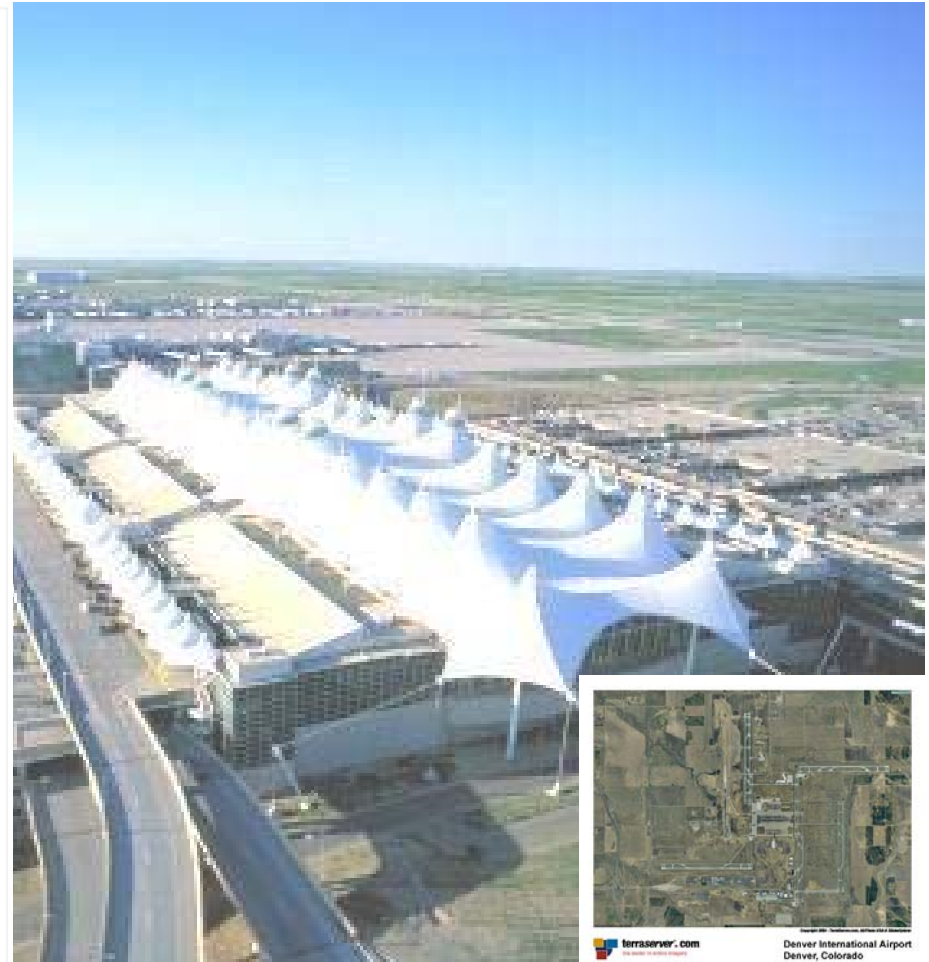
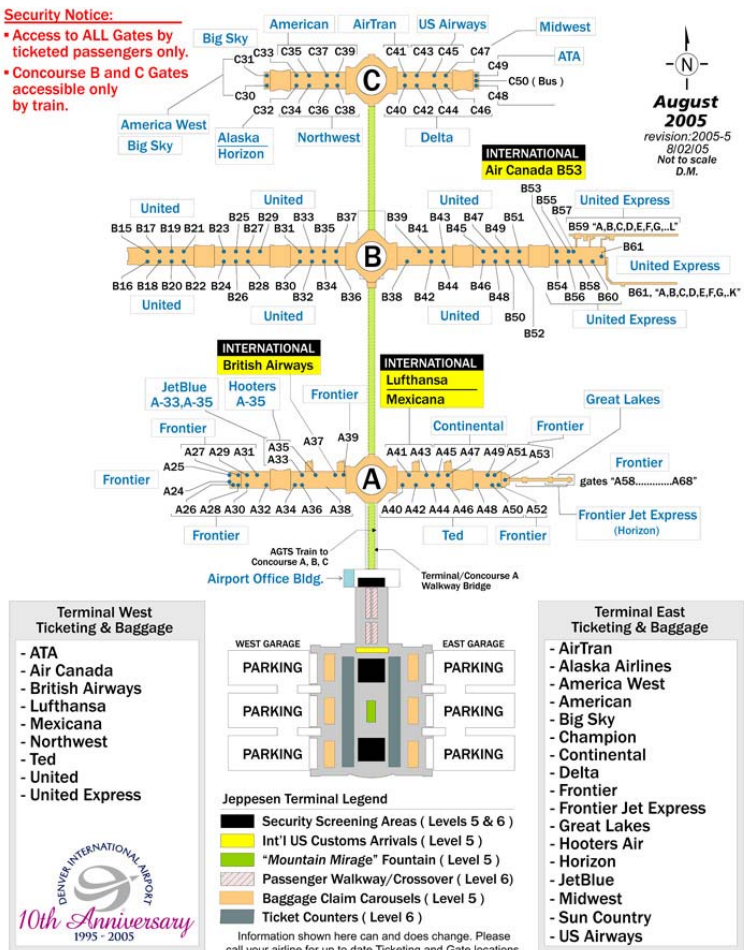
❑ Ursachen

„... zu wenige Experten ..., die in der Lage gewesen wären, Vorgaben der Steuerverwaltung in IT-taugliche Spezifikationen übersetzen zu können.“

Quellen:

- c't 23/2004, S. 218-223
- <http://www.heise.de/newsticker/meldung/63945>

Prominente Software-Katastrophen: DIA (1): Denver International Airport



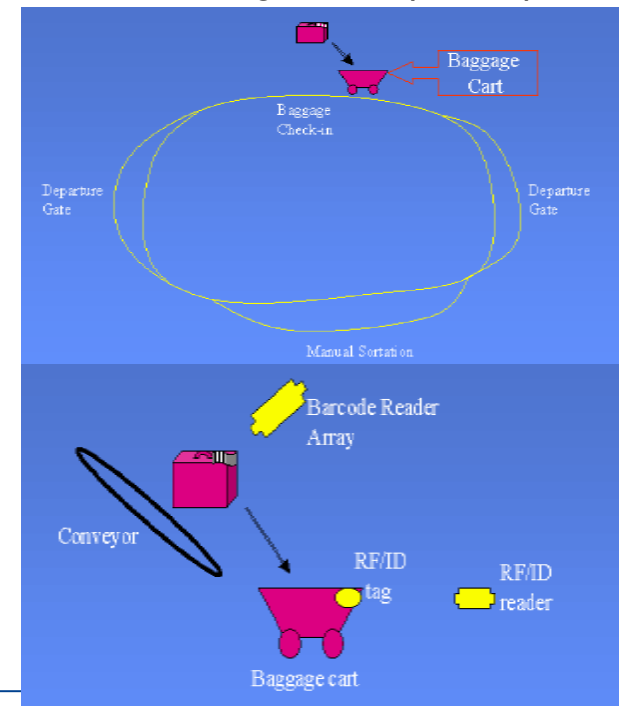
Eröffnung am 28.02.1995 (16 Monate verspätet)

Kosten des flughafenweiten Gepäcktransportsystems: \$ 311 Mio. (\$ 118 Mio. teurer)

Prominente Software-Katastrophen:

DIA (2): Transportsystem

- ❑ Entwicklung eines automatischen Gepäcktransportsystem für die Fluggesellschaft „United Airlines“ durch die Fa. Boeing Airport Equipment
- ❑ Ausweitung dieses Projekts auf flughafenweites System nach zweijähriger Entwicklungszeit
 - » Vollautomatische Beladung und Entladung für alle Fluggesellschaften (weniger Kosten/Zeit)
 - » Überwindung sehr langer Transportwege und Vermeidung langer Standzeiten der Flugzeuge
 - » Ziel: Null-Wartezeit (Gepäck und Passagier in gleicher Zeit zwischen Flügen transportiert)
- ❑ Größenordnungen
 - » 300 Rechner in 8 Kontrollräumen
 - » 15 Mio. Fuß Glasfaserkabel Netzwerk
 - » ca. 27 km Gleisanlage
 - » ca. 3500 standardisierte Waggon
 - » 56 Barcode-Lesegeräte für Gepäckstück-Etiketten
 - » Positionsbestimmung und Steuerung über Funk



Prominente Software-Katastrophen:

DIA (3): Probleme bei Einführung/Betrieb des Transportsystems

❑ Probleme

- » Datentyp-Konvertierungsprobleme bei Kombination verschiedener Programmiersprachen
- » Echtzeitsteuerung überlastet das zu langsame 10 Megabit Netzwerk → nur langsame Bedienung möglich
- » Fehlerhafte Steuerung (z.B. Waggonen fliegen aus Kurven, Zusammenstöße, orientierungslose Geisterfahrten, Waggonstaus) trotz Prototyp der BAE.

❑ Stand bei Eröffnung des Flughafens

- » Verspätete und verteuerte Fertigstellung (Verlust \$ 0.5 Mio. pro Tag)
- » Keine flughafenweite Lösung (Separate Systeme für jedes der drei Terminals)
- » Netzwerk durch 100 Megabit Netzwerk ersetzt
- » Nur die Hälfte aller 84 Gates werden versorgt (Zusätzlich existiert ein traditionelles Transportsystem mit manueller Sortierung zum Preis von \$ 71 Mio. und ein Backup-System)

❑ Ursachen / Lehren

- » Unterschätzung der Komplexität: Lösung für eine Fluggesellschaft ausgeweitet auf flughafenweite Größe, Systemkomponenten stets an Belastungsgrenze eingeplant
- » Unzureichendes Projektmanagement (wegen Projektgröße und -Komplexität)
 - Gebäudeplanung i.W. vor Planung des umbauten Transportsystems abgeschlossen
 - Nach Ausweitung keine Änderung des Fertigstellungsdatums
- » Hohe Zahl verspäteter Änderungsanforderungen aller Fluggesellschaften
- » Fehlende Testphase

Prominente Software-Katastrophen: Weitere Beispiele

- ❑ **Raketenkreuzer USS Yorktown, September 1997**
 - » Eingabe einer 0 in das Steuerungssystem des 'smarten' Raketenkreuzers führte zur Division durch 0 und damit zum Absturz aller LAN-Konsolen und des Remote-Terminals.
Das Antriebssystem des Schiffes war für mehrere Stunden tot.
 - » Ursache: Die Eingabe wurde nicht auf gültige Werte überprüft (Überlauf in Datenbank des Windows-NT Systems).

- ❑ **Mars Polar Lander, Dezember 1999**
 - » Das Mars-Landefahrzeug stürzte 40 m oberhalb der Oberfläche mit 80 km/h auf den Mars. Durch die Erschütterung beim Ausfahren der Landebeine hatten die Sensoren reagiert, was von der Software als Bodenberührung interpretiert wurde und zu einem Abschalten der Bremsraketen führte.
 - » Ursache: Die Software wurde falsch entworfen und schlecht getestet. Bei dem einzigen Test waren 3 der 4 Sensoren falsch verdrahtet.
 - » Kosten: \$ 165 Millionen

Quelle: [Homepage Ingolf Giese: <http://www-aix.gsi.de/~giese>]

Literaturquellen:

- ❑ Vorlesung „Softwarefehler in der Logistik am Beispiel des Denver Gepäcktransportsystems“
Prof. Thomas Huckle, Institut für Informatik, TU München
- ❑ Schloh: Analysis of the DIA baggage system
- ❑ de Neufville: The Baggage System at Denver: Prospects and Lessons
- ❑ Montealegre et.al.: BAE Automated Systems
- ❑ Donaldson: A case narrative of the project problems with the Denver Airport