Sicherheit und Zuverlässigkeit in der Software-Entwicklung

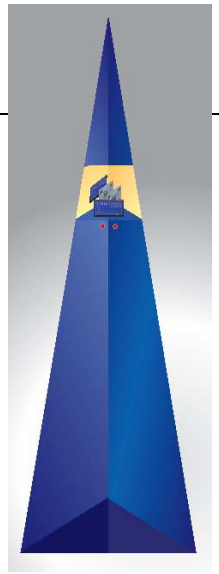
Sergio Montenegro

sergio@first.fhg.de

Holger Schlingloff

Holger.Schlingloff@first.fhg.de

Sichere Softwareentwuicklung



Fraunhofer FIRST Institut
Rechnerarchitektur
und Softwaretechnik

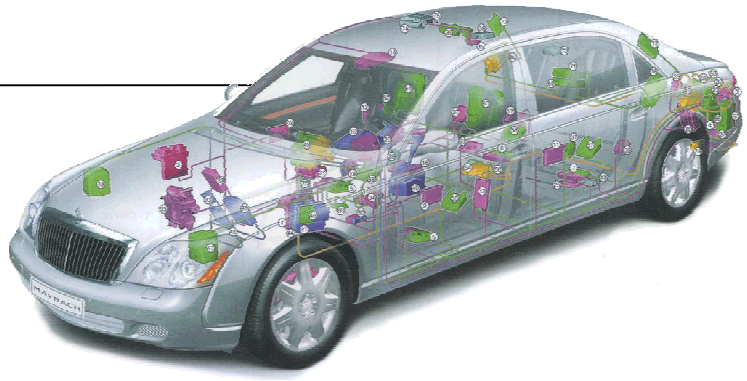
Warum funktionieren so viele komplexe Systeme nicht verlässlich?

Weil sie komplex sind!

Und die komplexeste Komponente ist...

So komplex, dass NIEMAND sie wirklich kennt oder versteht!

Software ist...



- **überall**
  integraler Bestandteil von Geräten / technischen Systemen

- **groß und komplex**

SAP/R3:	> 80.000.000 LOC (Zeilen Programmcode)
Microsoft Windows XP:	> 45.000.000 LOC (NT 2000: 20 M LOC)
Linux	> 30.000.000 LOC (Jun 2001 RH-7.1, RH-6.2: 15M)
Bank-Applikation:	> 3.000.000 LOC
Automobil:	> 2.000.000 LOC (High Dependability??)
Mobilfunktelephon:	> 1.000.000 LOC

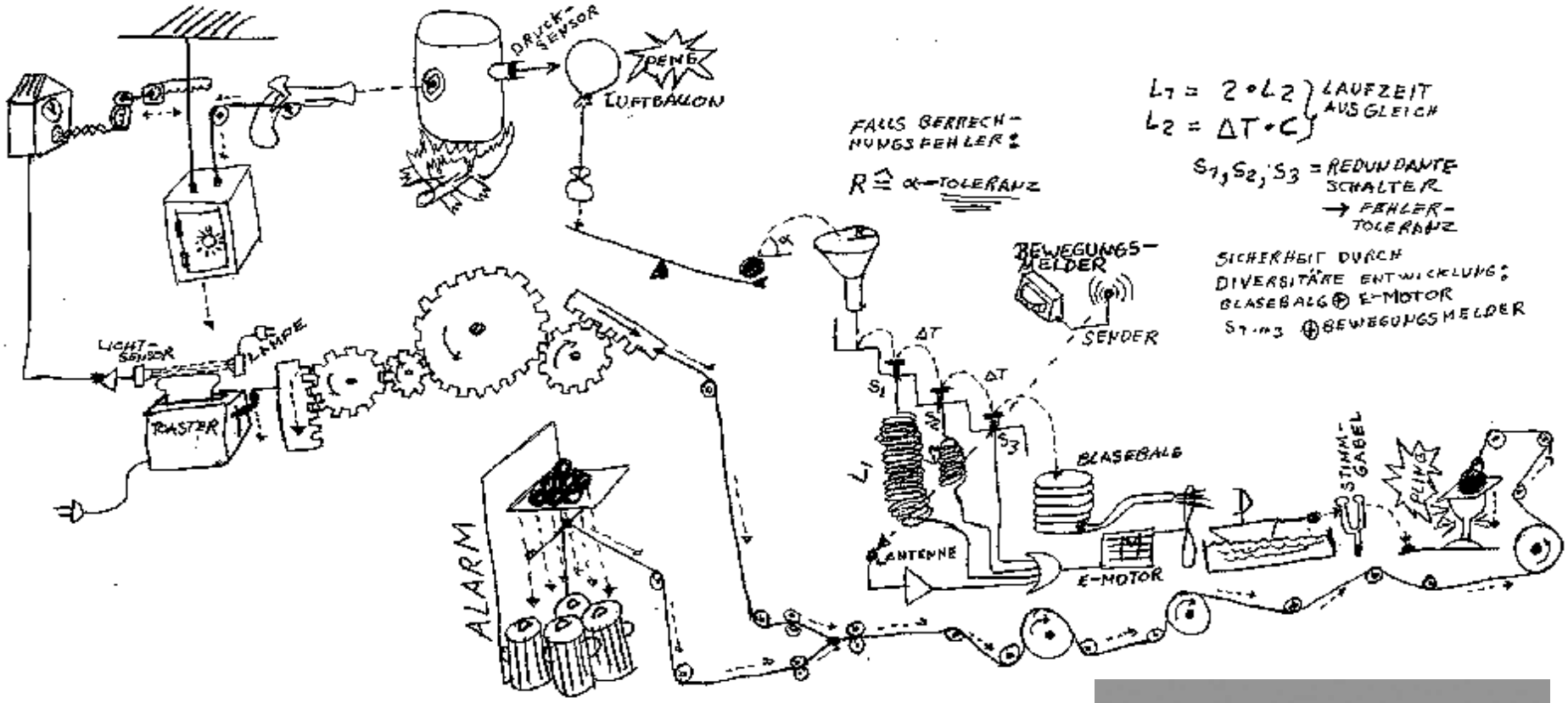
- **sehr vielgestaltig**

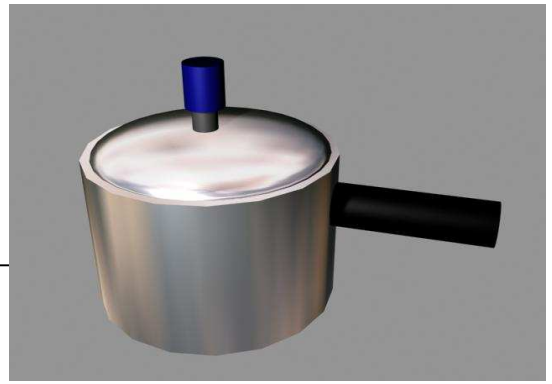
- **langlebig und anpassbar**

Software wird immer mehr zur qualitätsbestimmenden Komponente komplexer Systeme.

Quelle: nach SQ-Lab, FHG FIRST, 2003

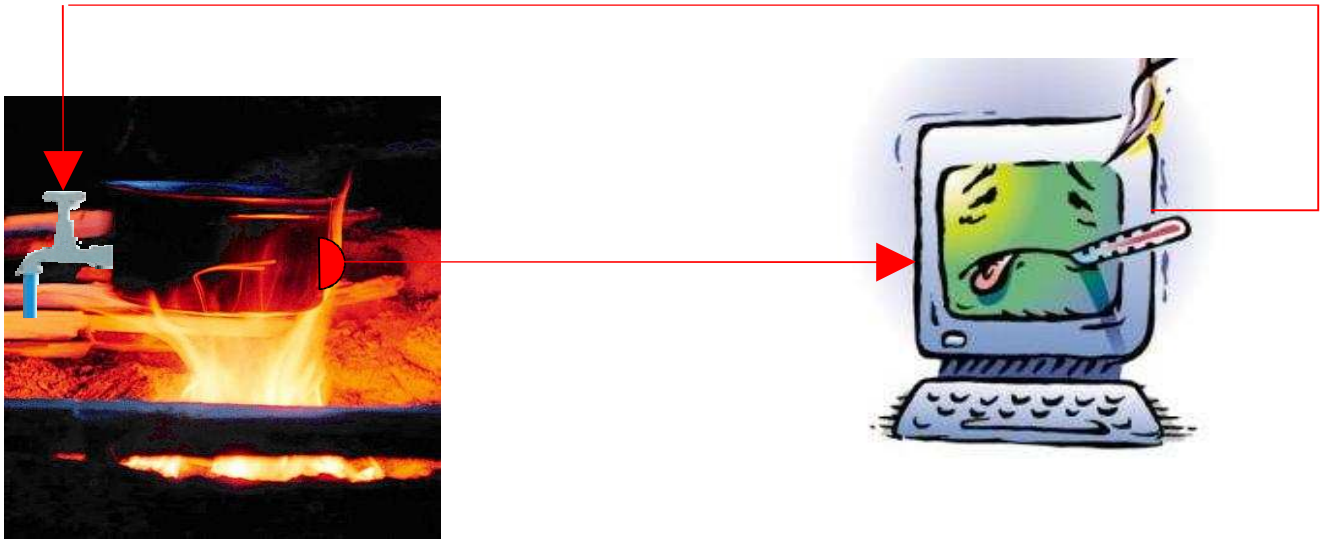
Einfachheit gegen Komplexität





FIRST
Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

Eine einfache Steuerung?



1. Elektronischer Drucksensor -> Elektronik -> elektrischen Ventil

2. Drucksensor -> Protokoll -> Input Driver -> Interrupt erzeugt -> das laufende Programm wird unterbrochen ->
Betriebssystem rettet den Zustand -> aktiviert den Interrupthandler -> Interrupthandler aktiviert das passende Programm ->
Programm wird initialisiert -> es ruft den Gerätetreibe zum Lessen ..... -> Berechnen -> Öffnen-Befehl zum Driver -> Protokoll -> Ventil -> Elektronik... ->
irgendwann später wird das unterbrochene Programm wieder aktivieren.

Total: Hunderttausend Lines of code (OS. Treiber u.a.),
mehr als tausend Prozessorbefehle ausgeführt,
mehr als eine Million Transistoren.

Evolution of a Programmer

High School/Junior High

```
10 PRINT "HELLO WORLD"
20 END
```

First Year in College

```
program Hello(input, output)
  begin
    writeln('Hello World')
  end.
```

Senior Year in College

```
 (defun hello
  (print
    (cons 'Hello (list 'World))))
```

New Professional

```
#include <stdio.h>
void main(void)
{
  char *message[] = {"Hello ", "World"};
  int i;
for(i = 0; i < 2; ++i)
    printf("%s", message[i]);
  printf("\n");
}
```

Seasoned Professional

```
#include <iostream.h>
#include <string.h>
class string
{
private:
  int size;
  char *ptr;
public:
  string() : size(0), ptr(new char('\0')) {}
string(const string &s) : size(s.size)
  {
    ptr = new char[size + 1];
    strcpy(ptr, s.ptr);
  }
~string()
  {
    delete [] ptr;
  }
friend ostream &operator <<(ostream &, const string &);
  string &operator=(const char *);
};
ostream &operator<<(ostream &stream, const string &s)
{
  return(stream << s.ptr);

}
string &string::operator=(const char *chrs)
{
  if (this != &chrs)
  {
    delete [] ptr;
    size = strlen(chrs);
    ptr = new char[size + 1];
    strcpy(ptr, chrs);
  }
  return(*this);
}
int main()
{
  string str;
str = "Hello World";
  cout << str << endl;

  return(0);
}
```

Master Programmer

```
[
uuid(2573F8F4-CFEE-101A-9A9F-00AA00342820)
]
library LHello
{
    // Bring in the master library.
    importlib("actimp.tlb");
    importlib("actexp.tlb");
// Bring in my interfaces.
    #include "pshlo.idl"
[
    uuid(2573F8F5-CFEE-101A-9A9F-00AA00342820)
    ]
    cotype THello
      {
      interface IHello;
      interface IPersistFile;
      };
};
[
exe,
uuid(2573F890-CFEE-101A-9A9F-00AA00342820)
]
module CHelloLib
{
// Some code-related header files

    importheader(<windows.h>);
    importheader(<ole2.h>);
    importheader(<except.hxx>);
    importheader("pshlo.h");
    importheader("shlo.hxx");
    importheader("mycls.hxx");
// Needed typelibs
    importlib("actimp.tlb");
    importlib("actexp.tlb");
    importlib("thlo.tlb");
[
    uuid(2573F891-CFEE-101A-9A9F-00AA00342820),
    aggregatable
    ]
    coclass CHello
      {
      cotype THello;
      };
};
#include "ipfix.hxx"
extern HANDLE hEvent;

class CHello : public CHelloBase
{
public:
    IPFIX(CLSID_CHello);
CHello(IUnknown *pUnk);
    ~CHello();
HRESULT  __stdcall        PrintSz(LPWSTR pwszString
);

private:
    static int cObjRef;
};
```

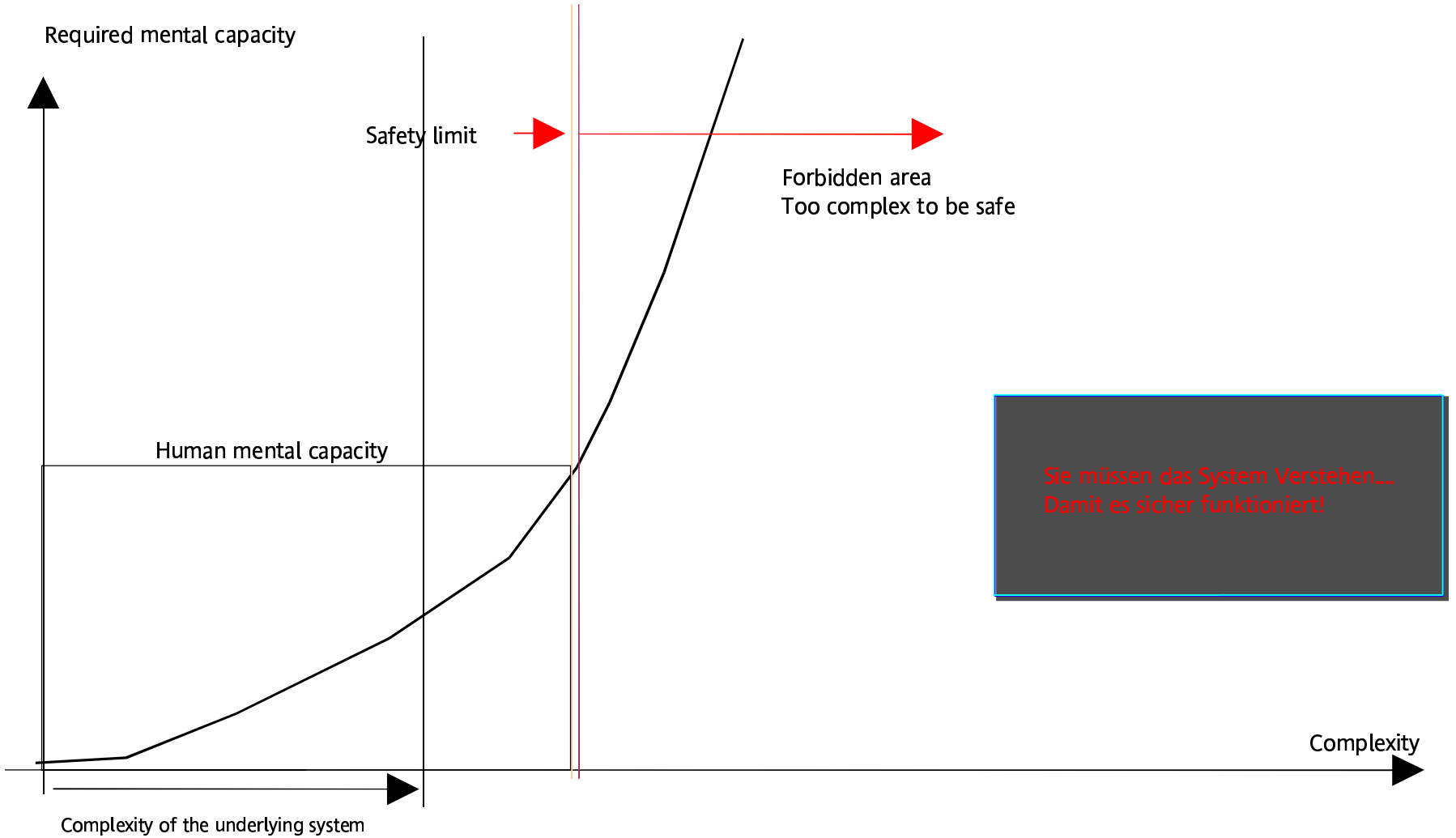
```
#include <windows.h>
#include <ole2.h>
#include <stdio.h>
#include <stdlib.h>
#include "thlo.h"
#include "pshlo.h"
#include "shlo.hxx"
#include "mycls.hxx"
int CHello::cObjRef = 0;
CHello::CHello(IUnknown *pUnk) : CHelloBase(pUnk)
{
    cObjRef++;
    return;

}
HRESULT  __stdcall  CHello::PrintSz(LPWSTR pwszString)
{
    printf("%ws\n", pwszString);
    return(ResultFromScode(S_OK));
}
CHello::~CHello(void)
{
// When the object count goes to zero, stop the server.
cObjRef--;
if( cObjRef == 0 )
    PulseEvent(hEvent);
return;
}
#include <windows.h>
#include <ole2.h>
#include "pshlo.h"
#include "shlo.hxx"
#include "mycls.hxx"
HANDLE hEvent;
int _cdecl main(
int argc,
char * argv[]
) {
ULONG ulRef;
DWORD dwRegistration;

CHelloCF *pCF = new CHelloCF();
hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
// Initialize the OLE libraries.
CoInitializeEx(NULL, COINIT_MULTITHREADED);
CoRegisterClassObject(CLSID_CHello, pCF, CLSCTX_LOCAL_SERVER,
                      REGCLS_MULTIPLEUSE, &dwRegistration);
// Wait for an event to stop.
WaitForSingleObject(hEvent, INFINITE);
// Revoke and release the class object.
CoRevokeClassObject(dwRegistration);
ulRef = pCF->Release();
// Tell OLE we are going away.

CoUninitialize();
return(0);
}
extern CLSID CLSID_CHello;
extern UUID LIBID_CHelloLib;
CLSID CLSID_CHello = { /* 2573F891-CFEE-101A-9A9F-00AA00342820
    0x2573F891,
    0xCFEE,
    0x101A,
    { 0x9A, 0x9F, 0x00, 0xAA, 0x00, 0x34, 0x28, 0x20 }
};
```

```
UUID LIBID_CHelloLib = { /* 2573F890-CFEE-101A-9A9F-00AA00342820 */
    0x2573F890,
    0xCFEE,
    0x101A,
    { 0x9A, 0x9F, 0x00, 0xAA, 0x00, 0x34, 0x28, 0x20 }
};
#include <windows.h>
#include <ole2.h>
#include <stdlib.h>

#include <string.h>
#include <stdio.h>
#include "pshlo.h"
#include "shlo.hxx"
#include "clsid.h"
int _cdecl main(
int argc,
char * argv[]
) {
HRESULT       hRslt;
IHello               *pHello;
ULONG         ulCnt;
IMoniker *    pmk;
WCHAR         wcsT[_MAX_PATH];
WCHAR         wcsPath[2 * _MAX_PATH];
// Get object path.
wcsPath[0] = '\0';
wcsT[0] = '\0';
if( argc > 1) {
    mbstowcs(wcsPath, argv[1], strlen(argv[1]) + 1);
    wcsupr(wcsPath);
    }
else {

    fprintf(stderr, "Object path must be specified\n");
    return(1);
    }
// Get print string.
if(argc > 2)
    mbstowcs(wcsT, argv[2], strlen(argv[2]) + 1);
else
    wcscpy(wcsT, L"Hello World");
printf("Linking to object %ws\n", wcsPath);
printf("Text String %ws\n", wcsT);
// Initialize the OLE libraries.
hRslt = CoInitializeEx(NULL, COINIT_MULTITHREADED);
if(SUCCEEDED(hRslt)) {
hRslt = CreateFileMoniker(wcsPath, &pmk);
    if(SUCCEEDED(hRslt))
      hRslt = BindMoniker(pmk, 0, IID_IHello, (void **)&pHello);

if(SUCCEEDED(hRslt)) {
// Print a string out.
      pHello->PrintSz(wcsT);
Sleep(2000);
      ulCnt = pHello->Release();
      }
    else
      printf("Failure to connect, status: %lx", hRslt);
// Tell OLE we are going away.
    CoUninitialize();
    }
return(0);
}
```

Wo steckt das Problem?

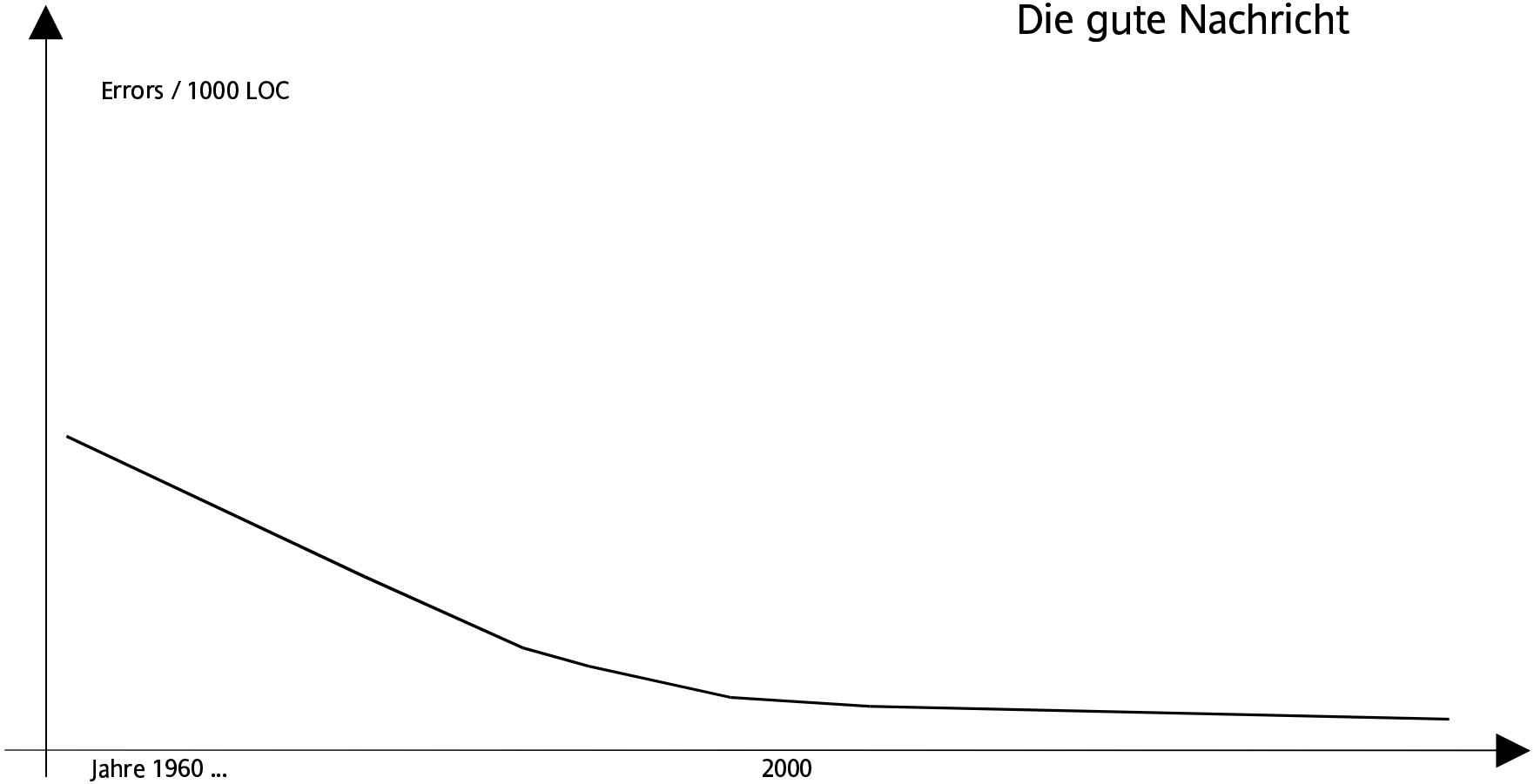
Zu viel des Gutes...: Leiber einfacher
noch besser? lieber gut als besser..
Wie ein Topf wo man alle zutaten rein tut die gut sind...

Komplexität ist der Feind der Sicherheit

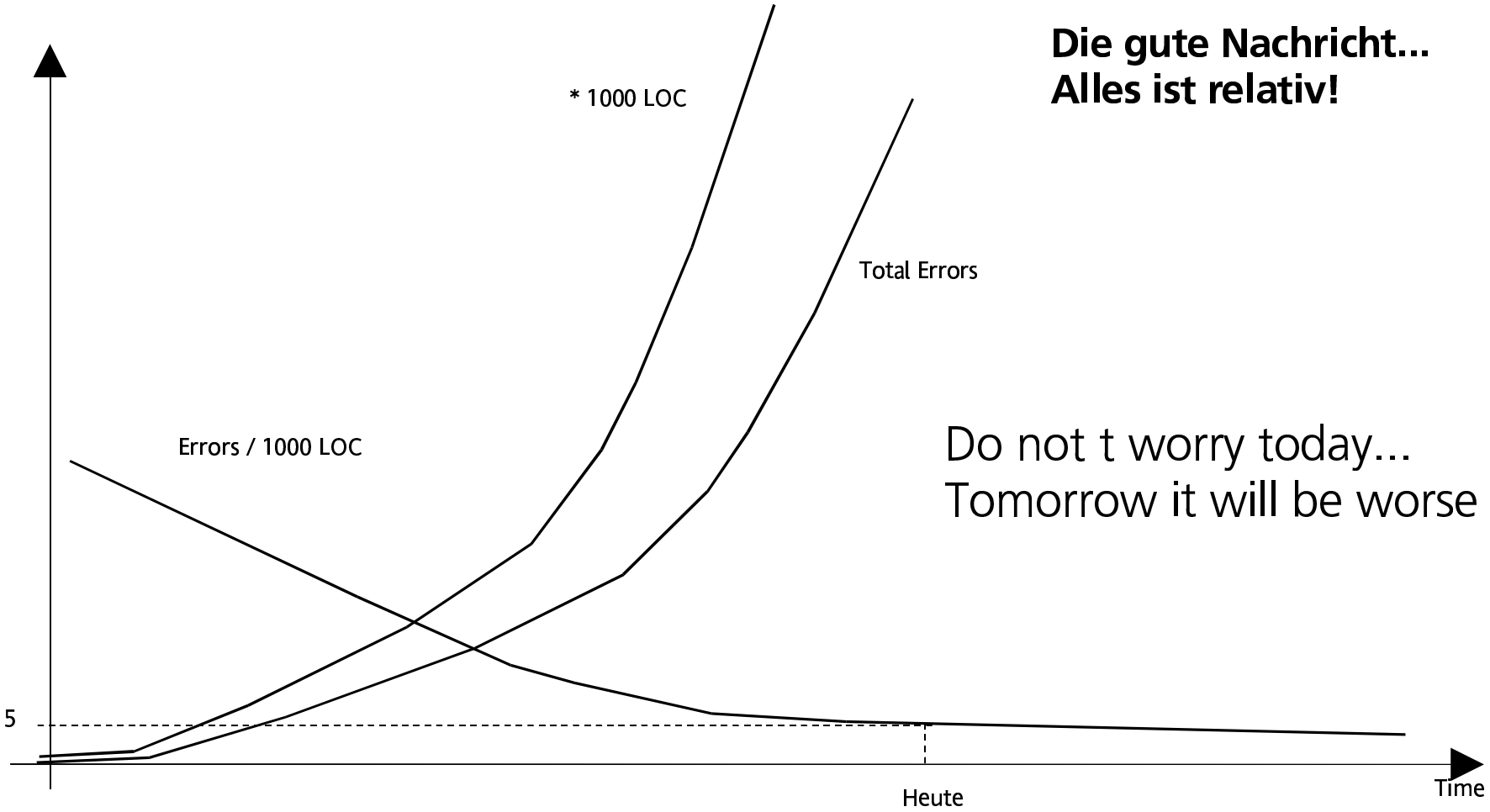


Fraunhofer FIRST Institut
Rechnerarchitektur
und Softwaretechnik

Komplexität ist der Feind der Sicherheit



Komplexität ist der Feind der Sicherheit

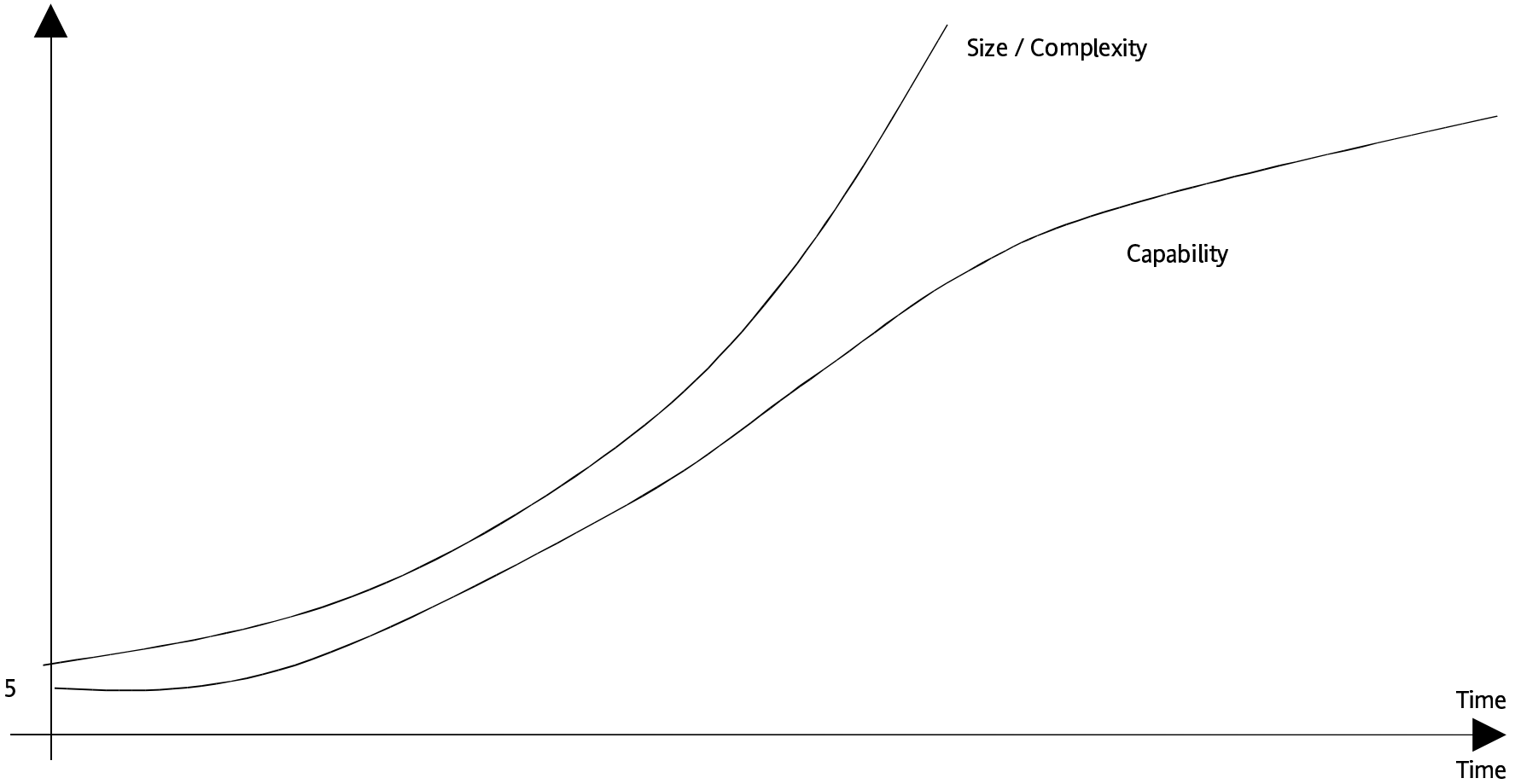


Die gute Nachricht...
Alles ist relativ!

Do not t worry today...
Tomorrow it will be worse

Fraunhofer FIRST Institut
Rechnerarchitektur
und Softwaretechnik

Entwicklung eines Softwarepackets



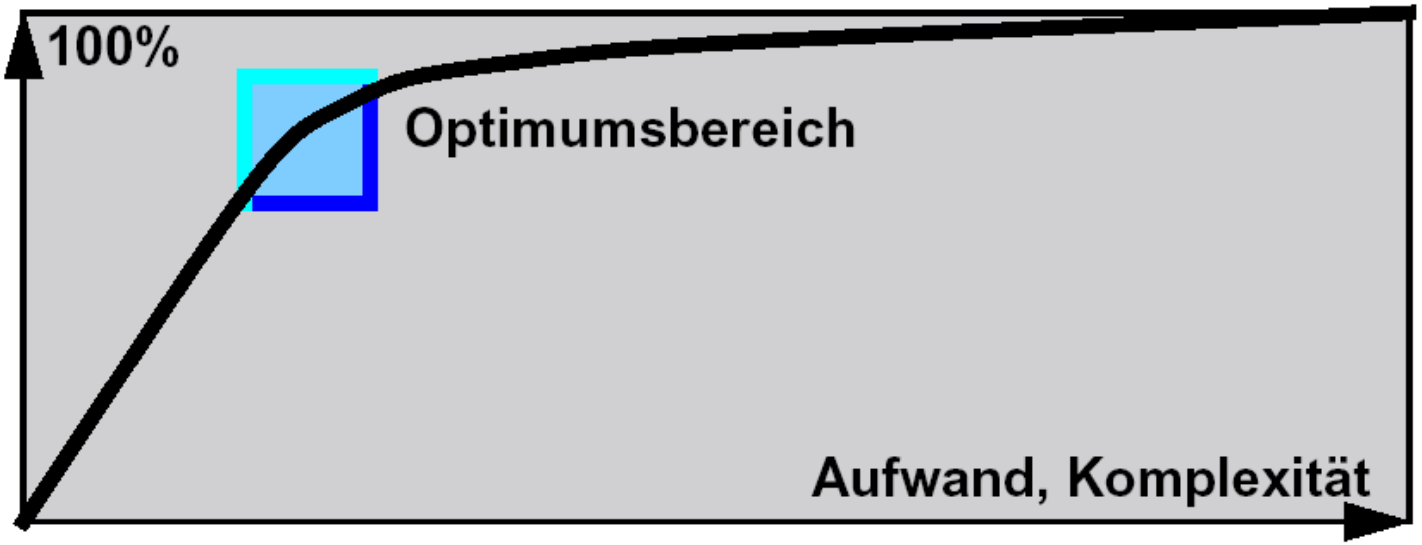
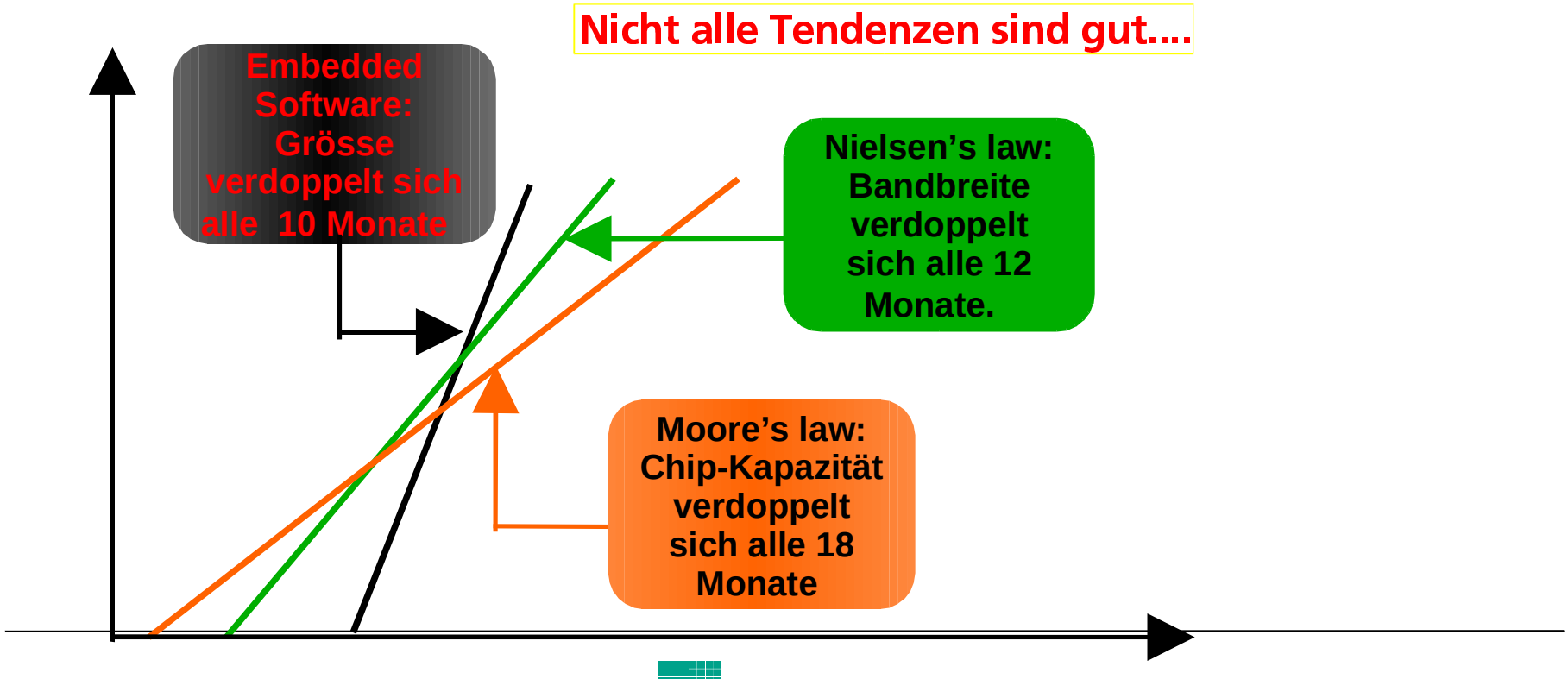


Abbildung 3-1: Aufwand gegen Leistung

Wichtige Tendenz



Quelle: nach ITEA 2004

Der Beitrag vom Betreibssystem

Es bestimmt (in Groben) wie wird das System programmiert!

....

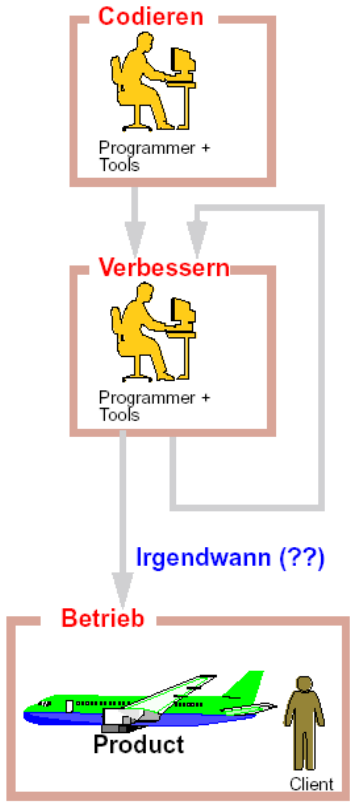
Oder Sie müssen Umwegen nehmen um das OS zu befriedigen

Alle (SW) Entwickler haben einen Interface zum OS

-> ALLE sollen es vollständig verstehen! (auch die nicht SW-leute)

OS API == Event Horizont?

Wie wird Software (nicht) Gemacht (1)

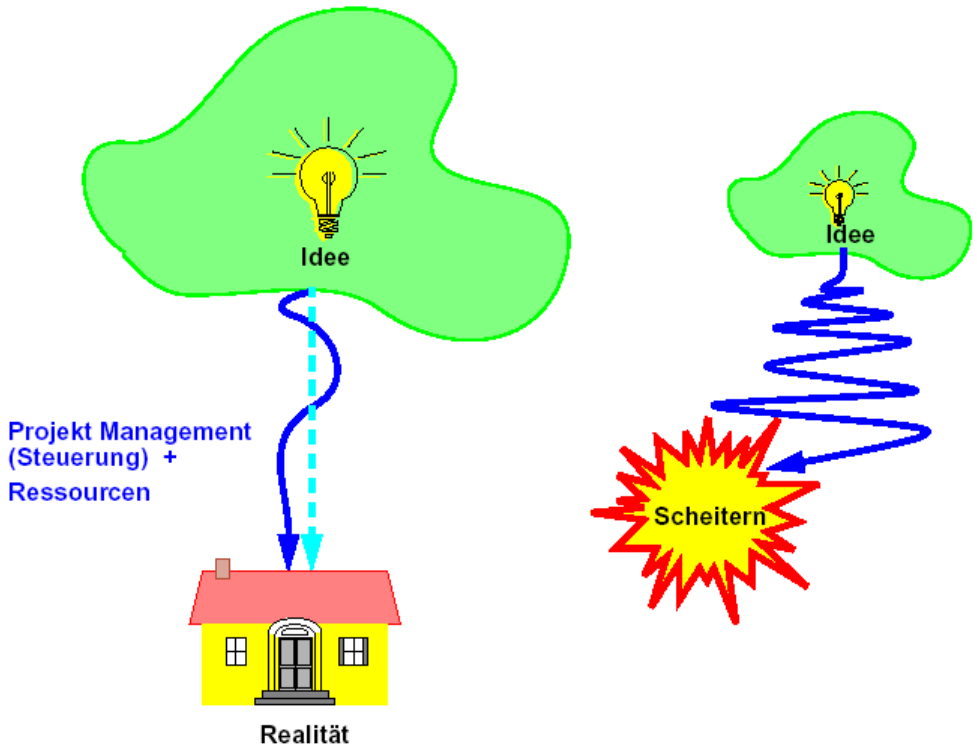


Computer analyst to programmer:
"You start coding. I'll go find out what they want"

Codiere-und-Verbessere

FIRST
Fraunhofer Institut Rechnerarchitektur und Softwaretechnik

Wie wird Software Gemacht (2)



Klares Ziel
Planung
Organisation
Feste, klare Konzepte
Disziplin
Qualifikation (Kompetenz)
Ressourcen

Erfolg

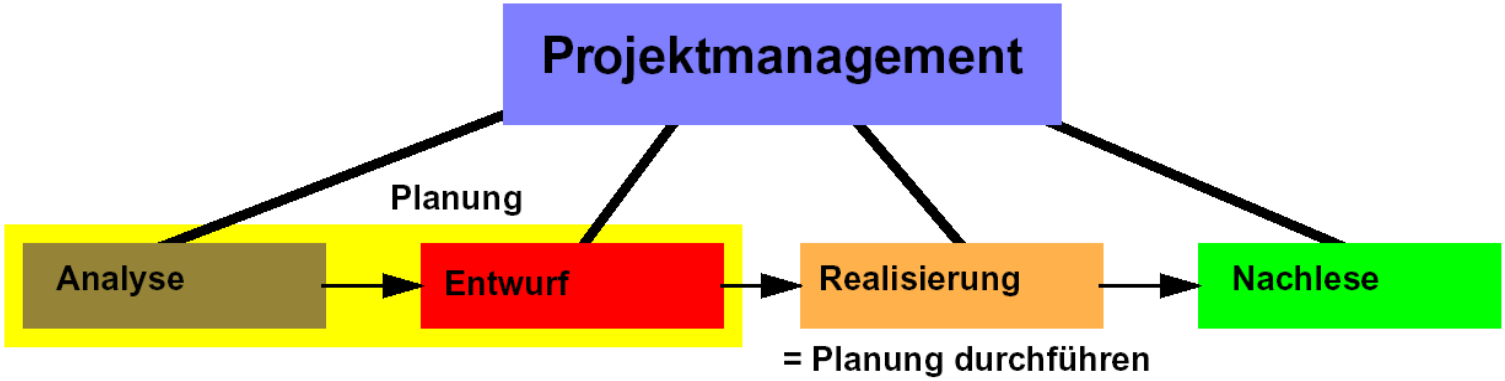
Ungenaue Planung, Kostenschätzung
Falsche Zeitpläne
Unklare Ziele
- unvollständig, unpräzise
- nicht meßbare Schritte
- öftere Zielwechsel
- kein Konsens

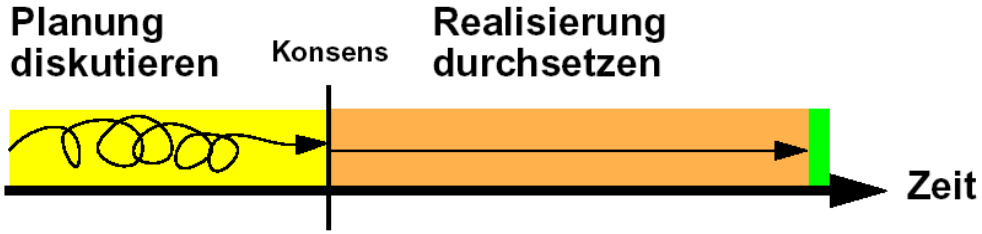
Falsche Organisation
- unscharfe Aufgaben
- unscharfe Kompetenzen
- Unfähigkeit

Mangelhafte Abwicklung
- Zweck der Arbeiten unklar
- kein Entscheidungsraum
- Ist/Soll Analyse unmöglich

Scheitern

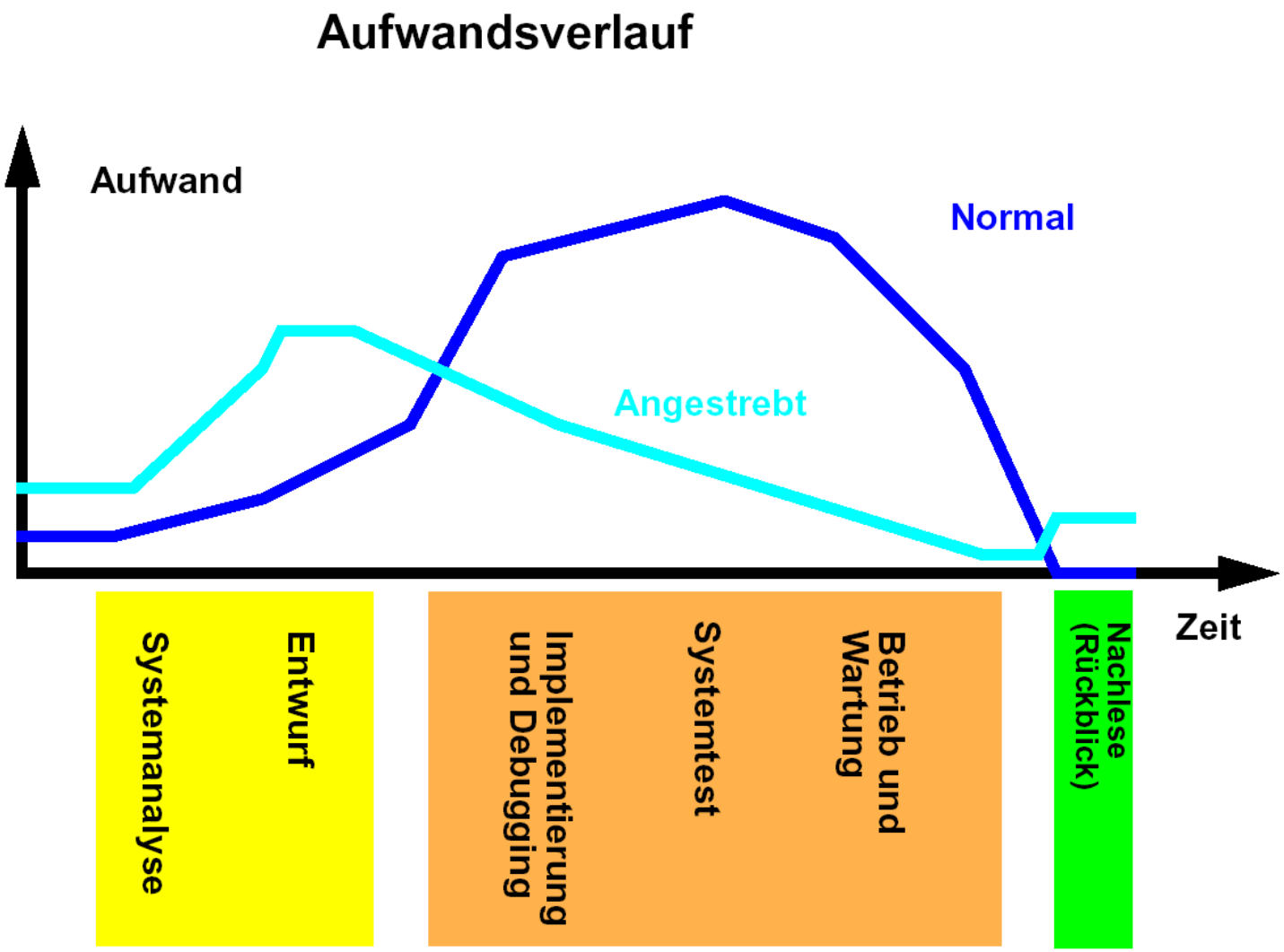
Wie wird Software Gemacht (3)



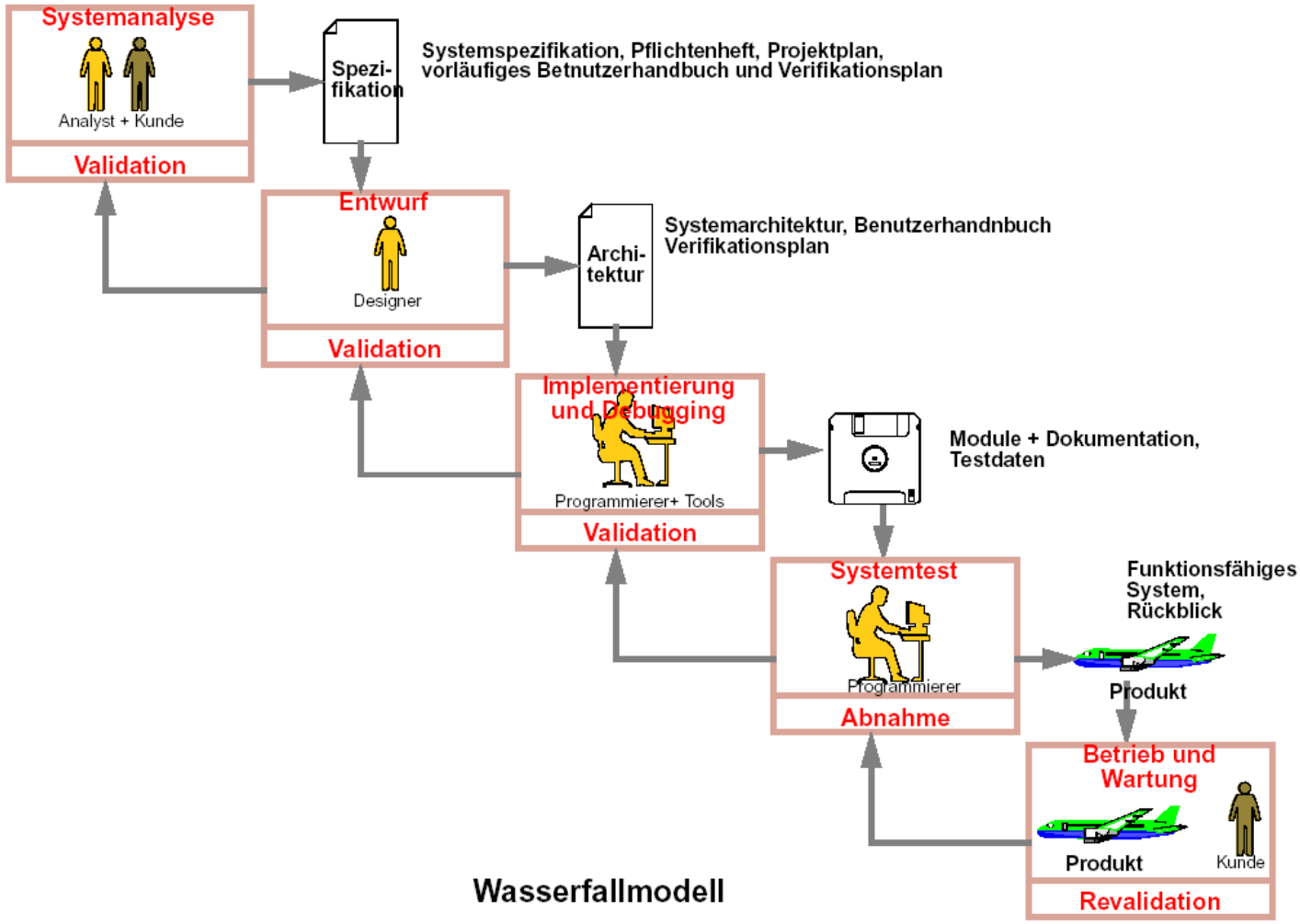


FIRST
Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

Wie wird Software Gemacht (4)

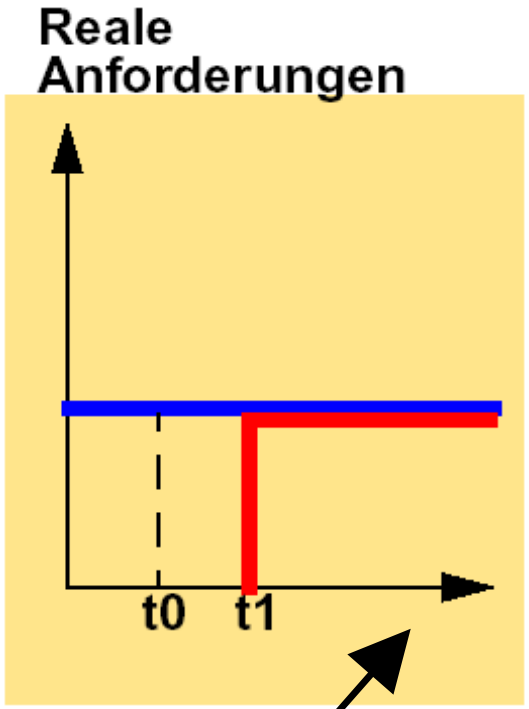


Wasserfall modell



Wasserfallmodell

1. Fehler: Denken am Anfang weiss man bereits alles

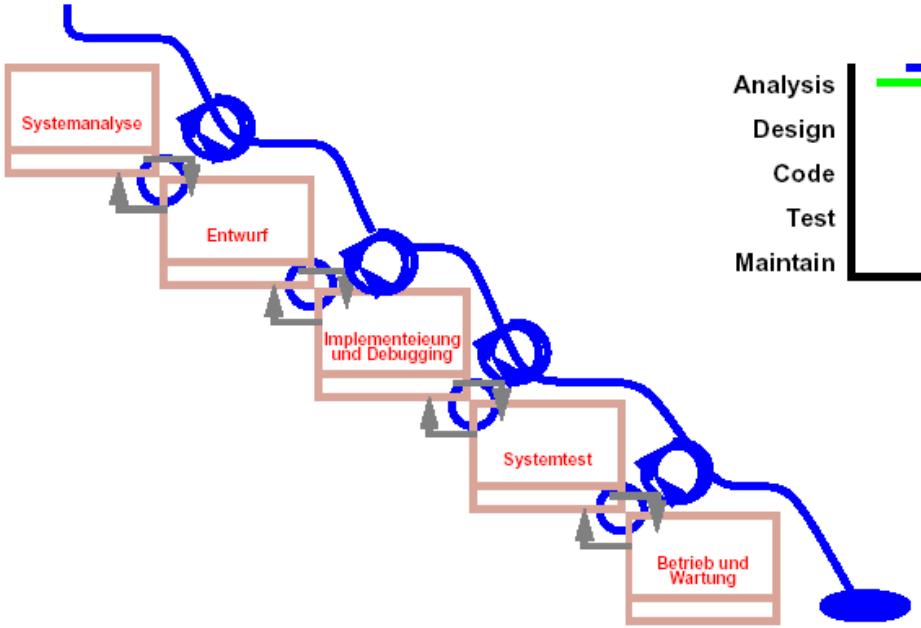


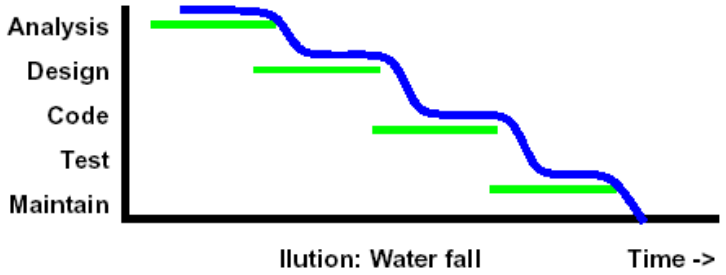
Was der Benutzer braucht

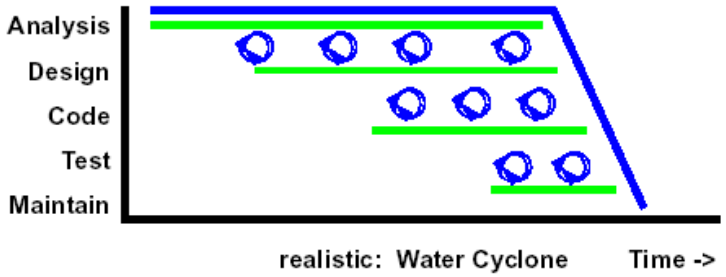
Was das System machen kann.

Requirements werden festgelegt

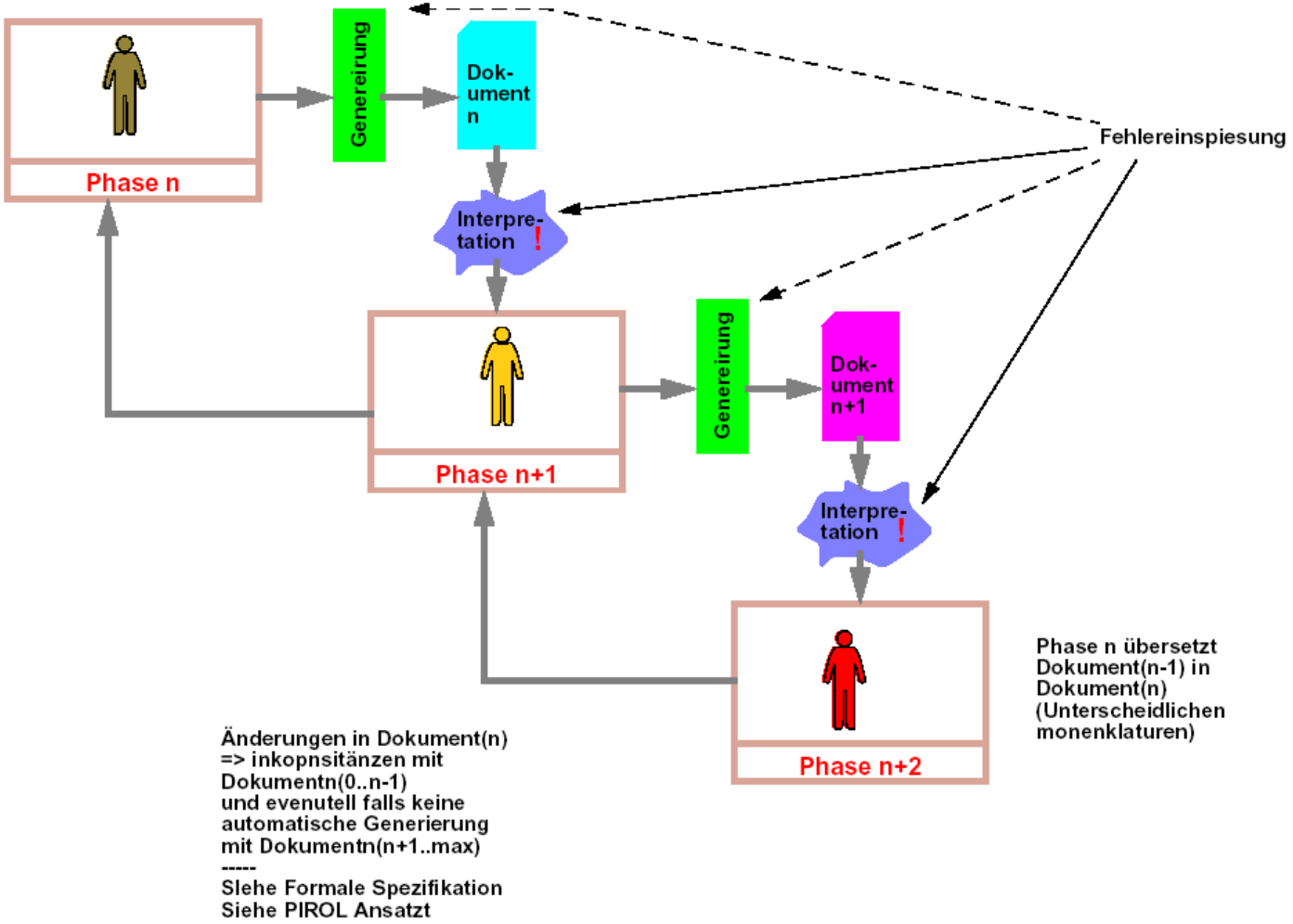
2. Fehler: Denken man hat saubere Übergänge



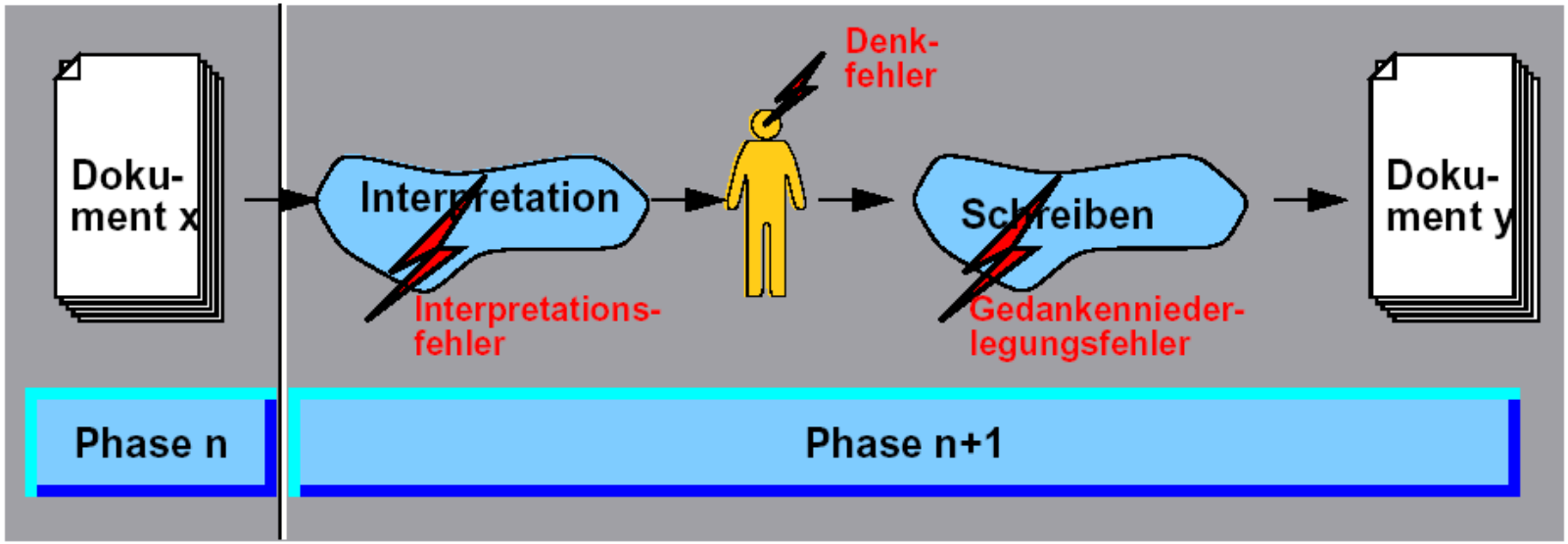




3. Fehler: Fehler Entstehung und Übertragung

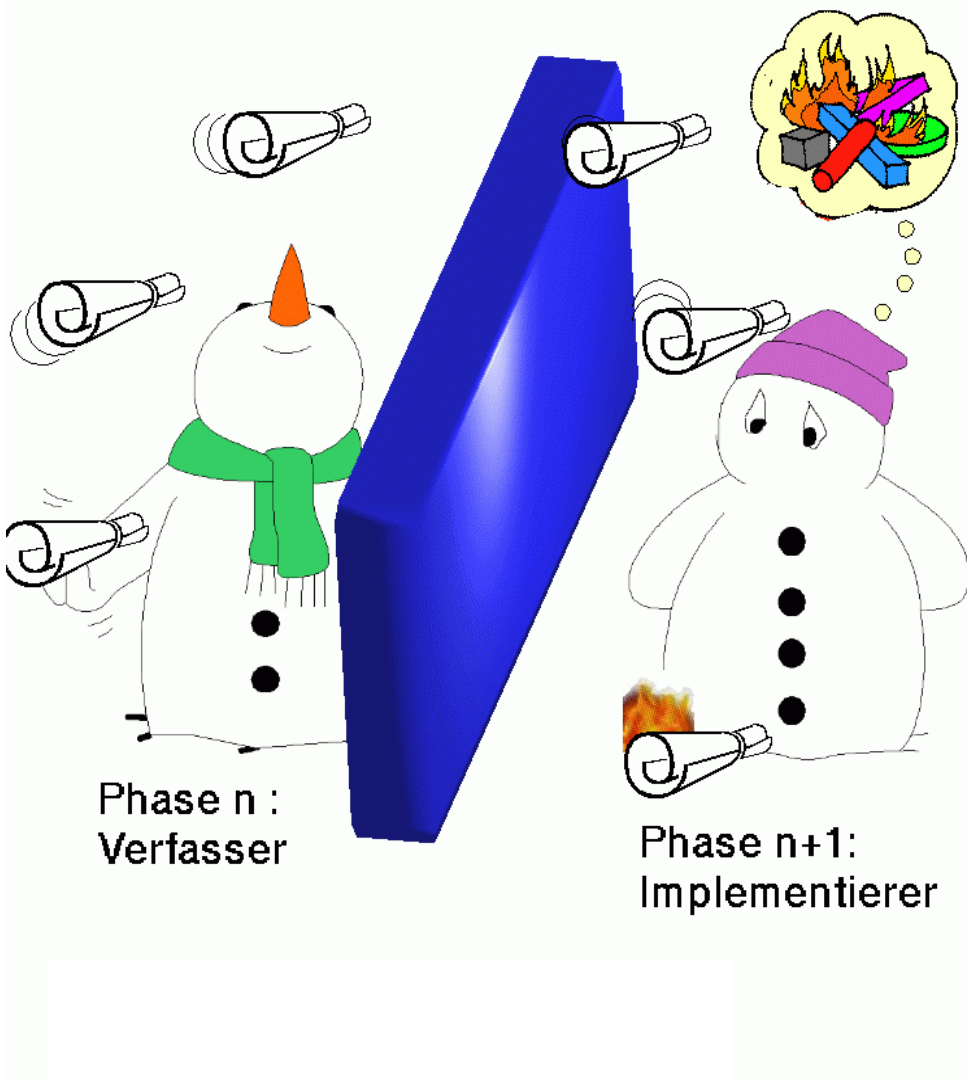


Fehlerenstehung



Entwicklungsfehler

Fehlerenstehung



Zeitaudehnung

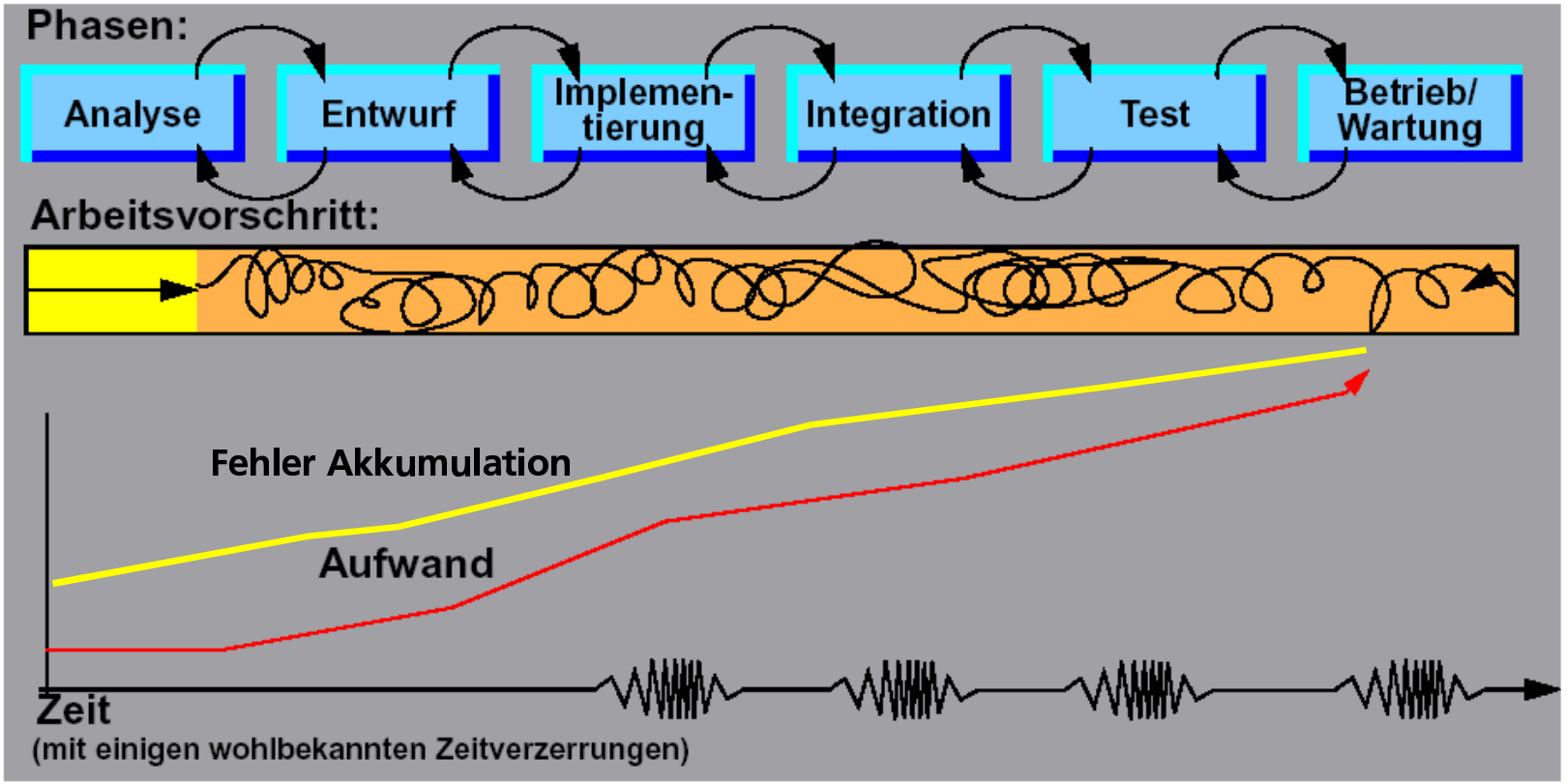
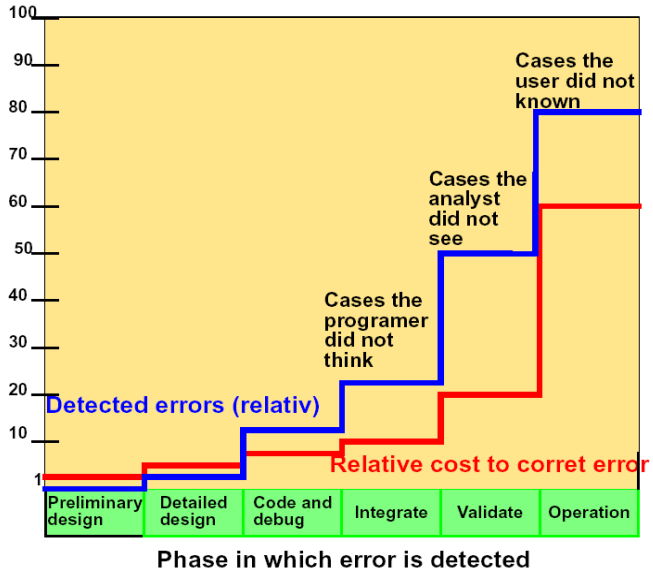
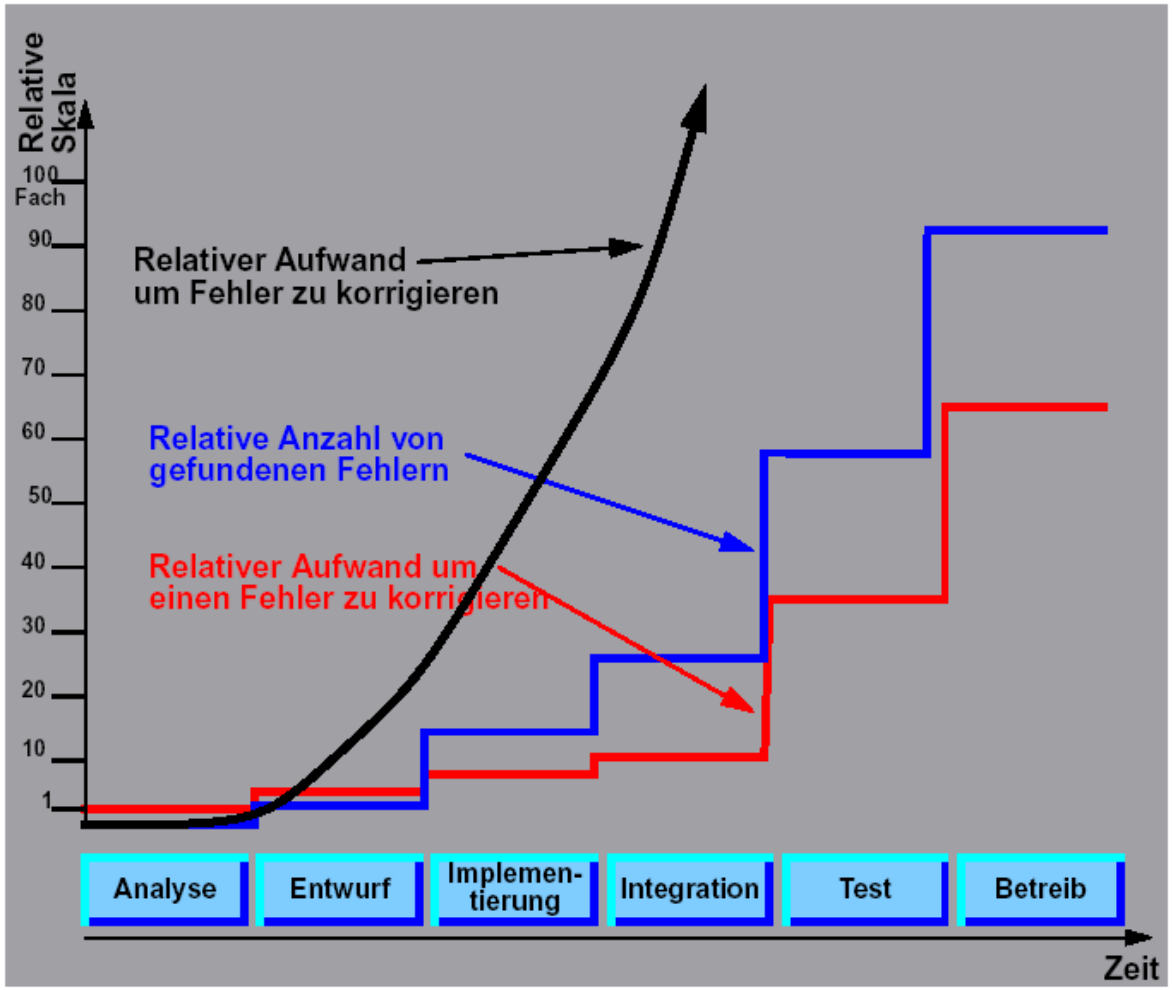


Abbildung 1: Entwicklungsphasen

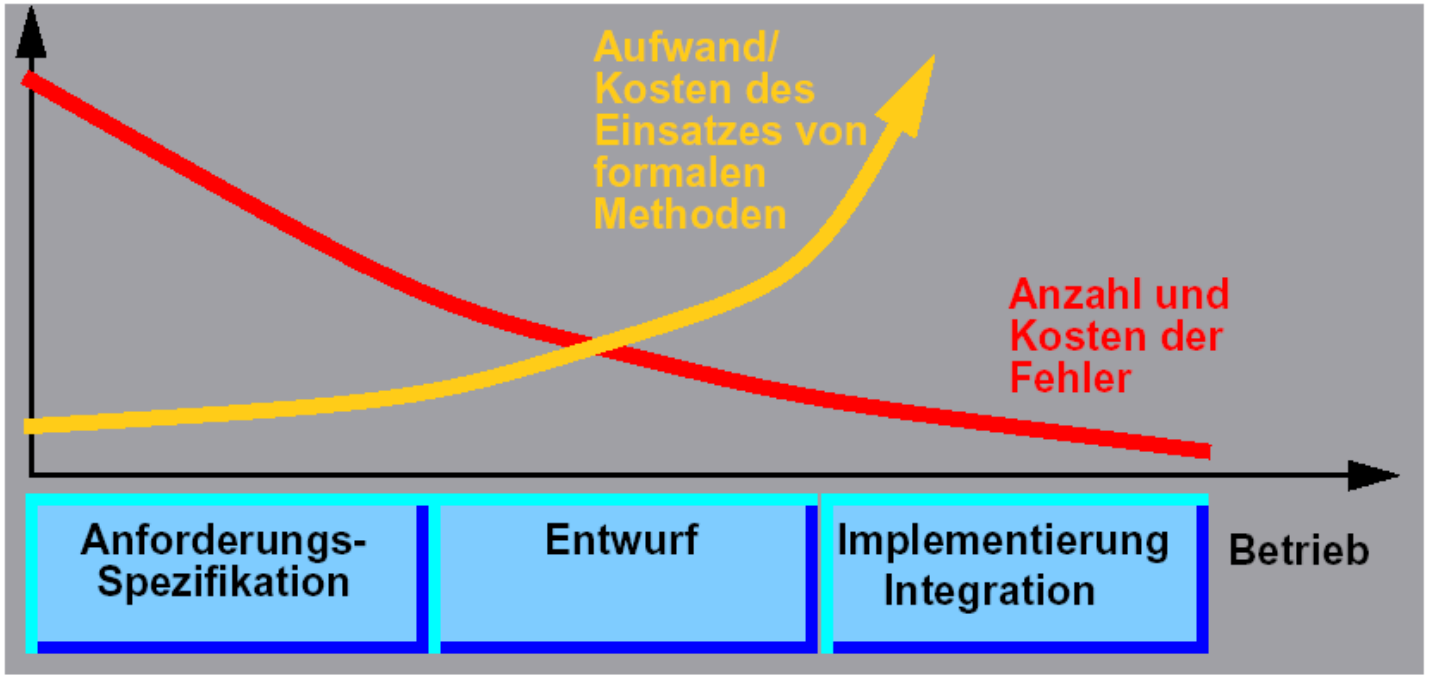




FIRST

Fraunhofer Institut Rechnerarchitektur und Softwaretechnik

```
--- sqrt ----------    ; Name der Operation
| x?   : Realzahl     ; "?" gekennzeichnet eine Input Variable
| a!   : Realzahl     ; "!" gekennzeichnet eine Output Variable
-----------
| x? >= 0             ; Vorbedingung: Die Input Variable x?
|                     ;                darf nicht negativ sein.
| a! * a! = x?        ; Die Output Variable a! multipliziert
|                     ;     mit sich selbst ergibt die Input Variable x?
--------------------
```


Aufwand/
Kosten des
Einsatzes von
formalen
Methoden
Anzahl und
Kosten der
Fehler
Anforderungs-
Spezifikation
Entwurf
Implementierung
Integration
Betrieb