© J. Ludewig 2020

Lasst, die ihr eintretet,
alle Hoffnung fahren!

Dante Alighieri,
Die Göttliche Komödie,
Inferno III, 9 (Das Höllentor)

Das kleine SE-Testament

Was wir im Software Engineering geerbt haben und was wir weiter vererben

Jochen Ludewig

dann und wann noch: Universität Stuttgart

Folien zum Vortrag „Kleines SE-Testament“ am 27.1.2020 bei der INFOGEM AG in Baden/AG.

© Jochen Ludewig. Alle Rechte beim Verfasser

© J. Ludewig 2020

Was können Sie erwarten / müssen Sie befürchten?

- eine zu lange Vorrede
- ein Gruppenbild und seine Geschichte(n)
- was wir haben oder gern hätten
- und dann (endlich) ein paar Schlussbemerkungen

Achtung:

Heute keine Künstliche Intelligenz,
kein Deep Learning,
keine Blockchain.

Einfach Software Engineering.

© J. Ludewig 2020

Hintergrund

Die INFOGEM AG hat mich gebeten, zum heutigen Jubiläum einen Vortrag beizusteuern, der an meine Beiträge in früheren Jahren anknüpft.

Da wären zu nennen:

- zwei *INFOGEM*-**Jubiläen 1997** und **2003**, beide am 27.1. im Kurtheater Baden, bei denen mir die Rolle des Conférenciers angetragen wurde.

  1997 habe ich nachdrücklich vor dieser Wahl gewarnt; 2003 habe ich die Teilnehmer mit Mussorgskis *Bilder einer Ausstellung* beschallt.

  Die bisherige Abfolge der Jubiläen erschwert eine Prognose.
  Bitte halten Sie sich den 27. Januar in den Jahren 2030 bis 2038 frei.

- **25 Werkstätten** zu den Dauerthemen des Software Engineerings, veranstaltet in Gerzensee, Thun und Brunnen von der Technischen Rundschau (1990 bis 1999) und der INFOGEM AG (bis 2016)

  Da habe ich jeweils am Ende *nicht*, wie es mein Auftrag war, die Beiträge zusammengefasst und gepriesen, sondern die Referenten verspottet.

Mich erneut als Referent einzuladen, war also **außerordentlich mutig!**

MANAGING SOFTWARE QUALITY-ENGINEERING SUCCESS

Conference Registration

January 27th, 1997
Kurtheater Baden
Switzerland

Please fax to
++41-56-222 00 38

or mail to

Monika Peterhans
INFOGEM AG, Rütistrasse 9, CH-5401 Baden

Admission fee:

- ❑ normal sFr. 360.- plus 6.5% VAT
- ❑ member of SAQ sFr. 280.- plus 6.5% VAT
- ❑ accommodation desired

name: _____

company: _____

department: _____

street: _____

zip code / city: _____

telephone: _____

telefax: _____

email: _____

date: _____

signature: _____

MANAGING SOFTWARE QUALITY-ENGINEERING SUCCESS

January 27, 1997
One Day Conference
Kurtheater Baden
Switzerland

Intended for

Managers and Professionals
in Software and Quality
Related Activities.

featuring

Tom DeMarco, USA
Norbert Fuchs, Austria
András Pataricza, Hungary
Lucy Suchman, USA
Niklaus Wirth, Switzerland

chaired by

Jochen Ludewig, Germany

organized by

INFOGEM AG
Informatiker Gemeinschaft für Unternehmensberatung

with support of Swiss Association
for the Promotion of Quality, SAQ

MANAGING RISK - MASTERING QUALITY

One day conference

January 27, 2003
Kurtheater Baden, Switzerland

intended for
managers and professionals in system and software engineering and in quality related activities

featuring
Kent Beck, USA
Tim Lister, USA
François de Nazelle, France
Peter Prónay, Slovakia
Georges Schlegel, Switzerland

chaired by
Jochen Ludewig, Germany

sponsored and organised by
INFOGEM AG
Informatiker Gemeinschaft für Unternehmensberatung

© J. Ludewig 2020

Warum *Kleines SE-Testament*?

François Villon (1431 bis ca. 1462) war ein Gauner und Totschläger, der nur mit viel Glück mehrmals dem Scharfrichter entkommen ist – und ein großer Dichter.

In seinen Balladen-Sammlungen **Kleines Testament** und **Großes Testament** hat er eine **Bestandsaufnahme** seiner aktuellen Situation geliefert – sehr drastisch und frei von jener Krankheit, die man heute „Political Correctness“ nennt.

Ich stehle ihm den Titel **„Kleines Testament“** mit dem Zusatz „**SE**“ für Software Engineering“. Und wie bei ihm geht es weniger um einen konkreten Nachlass, vielmehr um eine **Bestandsaufnahme** des Inventars, hier natürlich des SE-Inventars. Und wie jedes Testament ist dies eine sehr persönliche Betrachtung.

Gegenstand dieses Testaments sind die **Erkenntnisse und Errungenschaften**, die von einer Generation zur anderen vererbt werden. Aus meiner Sicht sind hier drei **Generationen** zu unterscheiden:

- meine Eltern, die keinen Computer angefasst haben
- wir, die wir die Rechner (frühestens) im Studium kennenlernten
- unsere Kinder, die mit den Kisten aufgewachsen sind.

Das sind die vor 1920, vor 1950 und vor 1980 geborenen.

© J. Ludewig 2020

Was haben wir geerbt?

Vor etwa vierzig Jahren hat unsere Generation (die Vorfünfziger) die Arbeit auf den Software-Baustellen übernommen. Was hat sie vorgefunden, übernehmen können?

- Die **von-Neumann-Architektur**
- Die **Theorie** zur Software
  (z. B. der Algorithmen-Begriff, die Chomsky-Hierarchie der Sprachen, das Halte-Problem, die Graphentheorie usw.)
- Erfahrungen beim Versuch, **große Software-Systeme** zu bauen
- eine lange **Reihe schlechter Gewohnheiten** bei der Codierung (wie Code ohne klare Strukturen, ohne sinnvolle Kommentare, mit vielen Clones durch Cut & Paste)
- unglaubliche **Ignoranz** gegen alles, was nicht Code ist
  (z. B. gegen die Spezifikation, die systematisch und rational gewählten Testdaten, den seriösen Projektplan).

Daraus ergaben sich die Themen, an denen sich viele aus unserer Generation abgearbeitet haben.

© J. Ludewig 2020

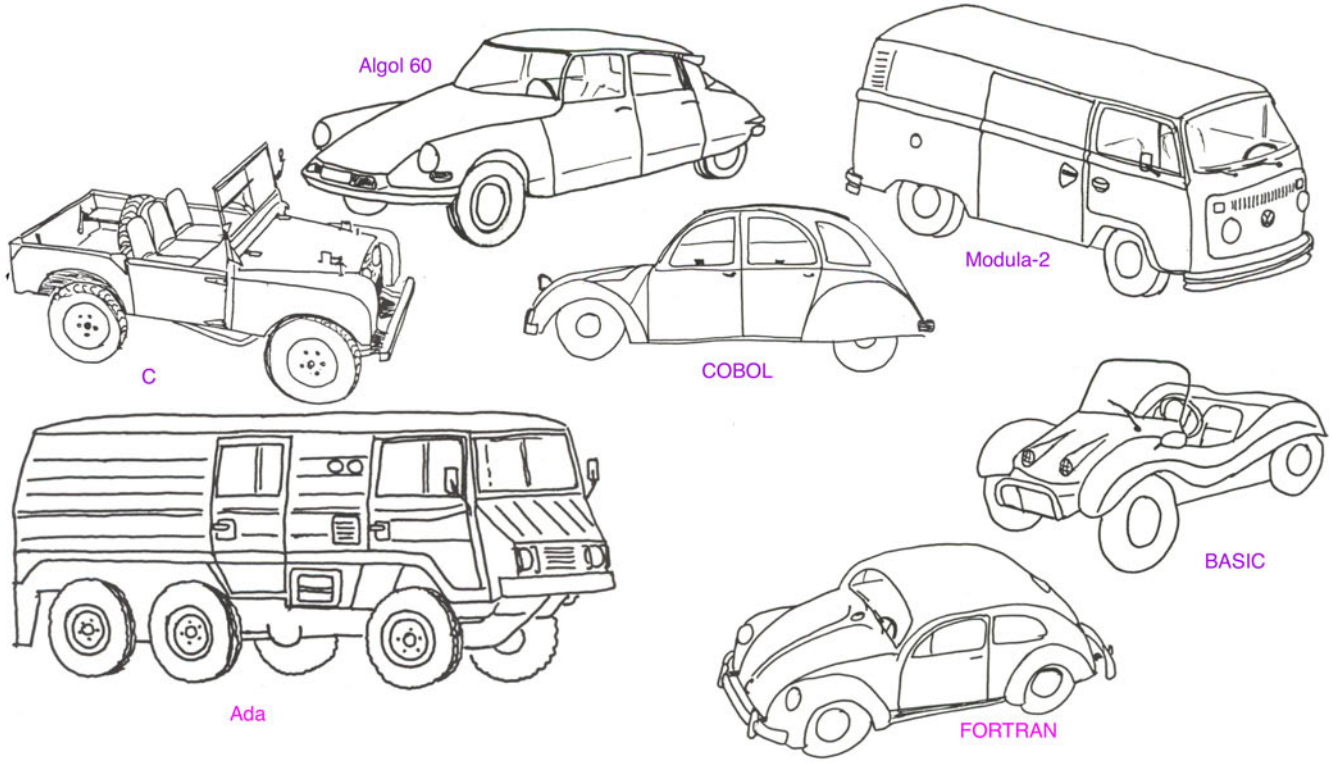
Wo war unsere Arbeit erfolgreich?

In den letzten vierzig Jahren sind entstanden:

- brauchbare **Programmiersprachen**
- **Information Hiding**, objektorientierte Programmierung
- **Komfortable Interaktion** zwischen Benutzer/Entwickler und System
- Sehr kluge, hoch effiziente **Algorithmen**
- **Life-Cycle-Begriffe**, **Prozessmodelle** (wie Scrum)
- definierte **Rollen** und **Dokumente**
- **Management-Techniken** (wie Kanban)
- Ansätze zur **Quantifizierung**, **Metriken** (z. B. Technical Debt)
- **Werkzeuge**, z. B. für den Unit-Test

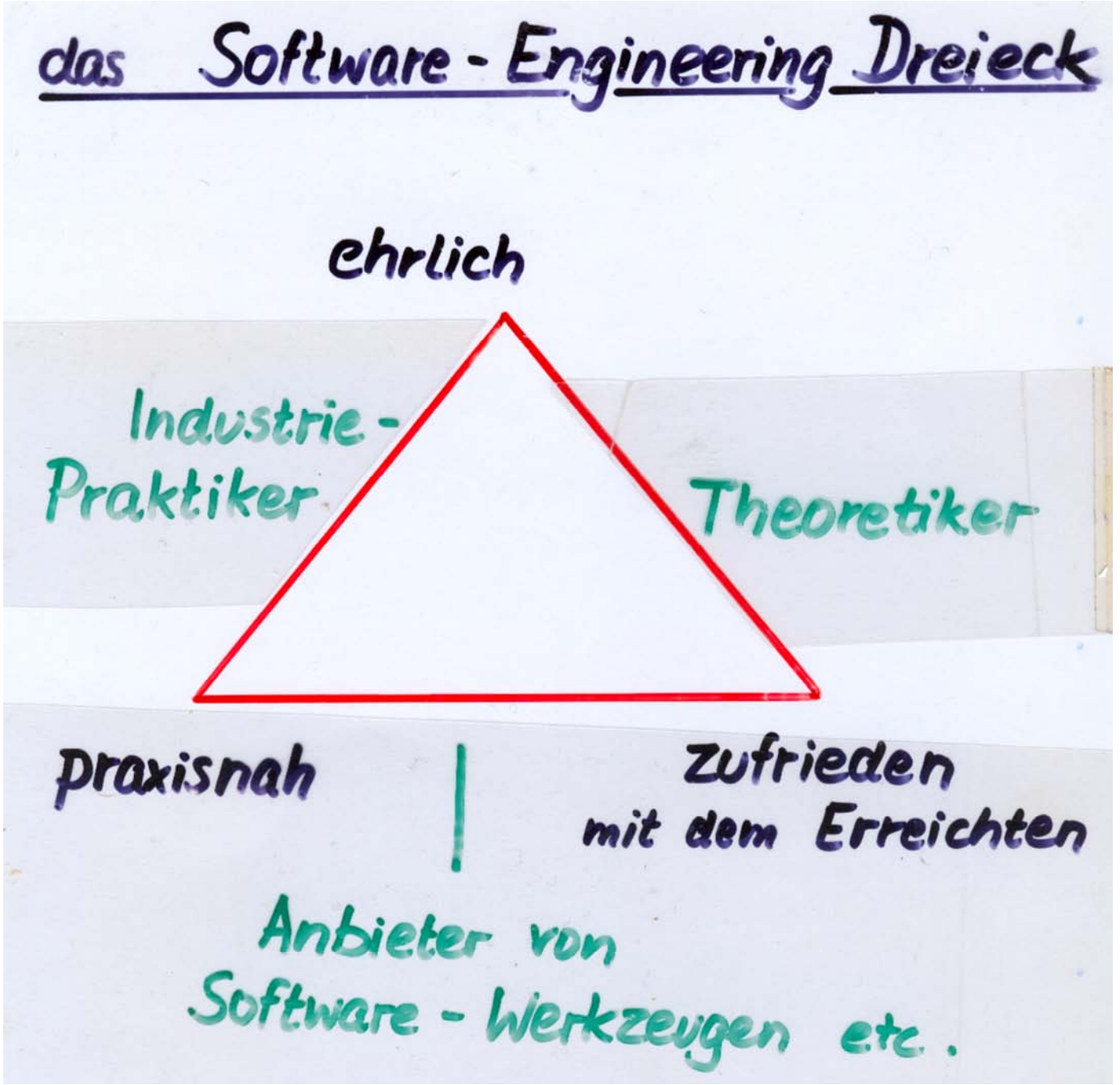
Das alles (und noch viel mehr) bildet unseren Nachlass.

© J. Ludewig 2020



© J. Ludewig 2020

das Software - Engineering Dreieck



© J. Ludewig 2020

In welcher Form steht dieses Erbe zur Verfügung?

Die **Erkenntnisse und Lösungen** findet man als Texte in **Fachzeitschriften** und **Tagungsbänden**, mit einiger Verzögerung in den **Lehrbüchern**.

Das ist wunderbar. (Erst 1979 kam das erste deutsche **SE-Lehrbuch** auf dem Markt!)

Aber wir haben nicht nur bedrucktes Papier und LED-Bildschirme.
Als Informatiker genießen wir das phantastische Privileg, viele der **Protagonisten unseres Fachs** persönlich kennenzulernen.

Kein Physiker hat die Chance, Newton, Maxwell und Einstein zu treffen, womöglich noch auf einer einzigen Konferenz.

Aber wir Software-Ingenieure können noch (fast) jedem unserer Vordenker die Hand schütteln.

Wenigstens gab es diese Möglichkeit 2001 auf dem großartigen **Symposium der Software-Pioniere**, im alten Bundestag in Bonn.
Der Sponsor war s d & m (Ernst Denert).

© J. Ludewig 2020

ganz links: **Denert** (Veranstalter); ganz rechts **Broy** (wissensch. Leitung)
kniend: **deMarco**, **Wirth**, **Guttag**, **Jackson**, **Brooks**, **Parnas**, **Chen**, **Kay**
stehend: **Boehm**, **Dijkstra**, **Bauer**, **Gamma**, **Dahl**, **Nygaard**, **Bayer**, **Hoare**

© J. Ludewig 2020

Zu (fast) jeder Person auf dem Foto kann man aus dem Stehgreif eine Laudatio verfassen. Diese (und einige andere) Leute haben das Software Engineerung vorangebracht.

Großer Dank an unsere Koryphäen!
Sie haben das Rad der SE-Geschichte gedreht.

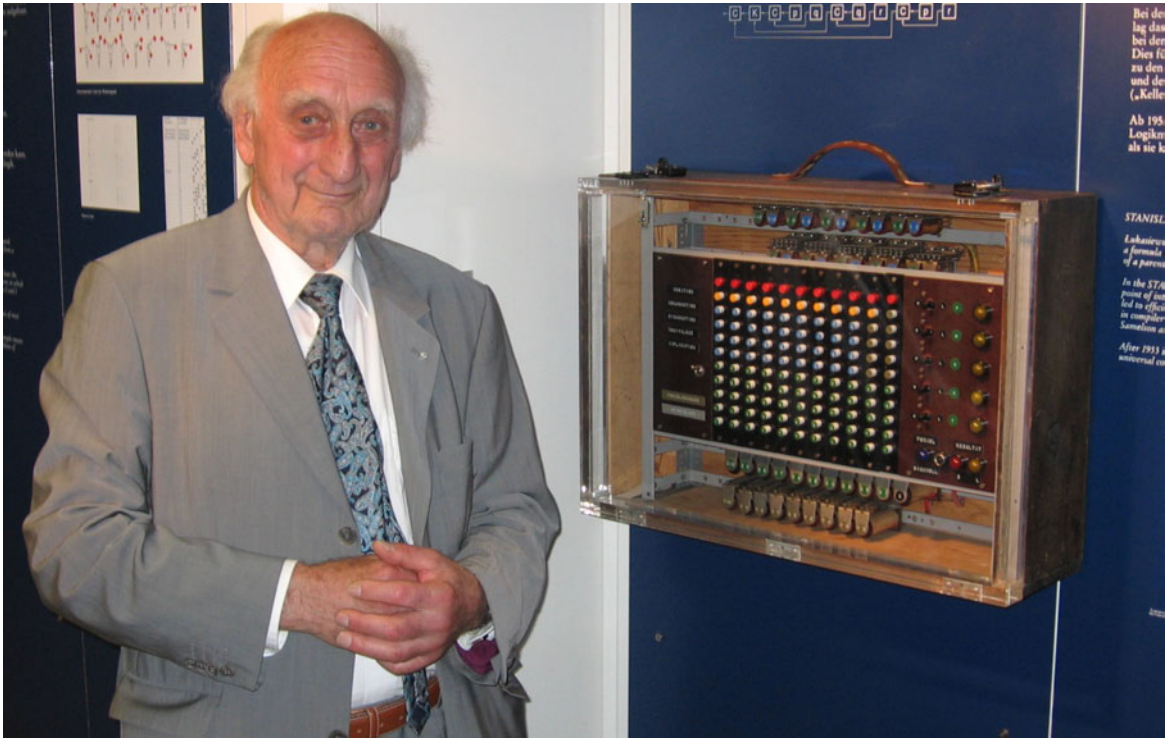
Und großer Dank an die Veranstalter dieses Symposiums!

Nur wenige haben in Bonn gefehlt; mir fallen ein

- **Donald Knuth** (The Art of Computer Programming)
- **Ken Thompson**, **Dennis Ritchie**, **Linus Torvald** (UNIX/LINUX)
- **Adele Goldberg** (Smalltalk).

Und natürlich sind nicht mehr alle am Leben (z. B. **Alan Perlis**)

© J. Ludewig 2020



Der Vater des Software Engineerings, Prof. Friedrich L. Bauer, mit seinem Logikrechner „Stanislaus“ in der von ihm gestalteten Informatik-Sammlung des Deutschen Museums in München (2004)

© J. Ludewig 2020

Was fehlt?

Einige Themen hat unsere Generation leider nicht entscheidend weitergebracht:

- eine wirklich brauchbare, auch nach ästhetischen Gesichtspunkten akzeptable und weltweit verbreitete Programmiersprache
- All-in-One-Dokumente
- eine graphische Notation für Software, wie sie ansatzweise in UML definiert ist. Aber UML ist bereits zu Code-nah.
- formale, trotzdem gut lesbare Schnittstellendefinitionen
- verifizierte Bibliotheken
- Rollen und Verantwortlichkeiten
- wohldefinierte Qualifikationen für Software-Ingenieure
- klare Haftungsregeln für Software

Und das alles bitte international und, wenn möglich, standardisiert!

© J. Ludewig 2020

Und was wünsche ich mir sonst?

- Die Fachleute der **SE-Theorie** und der **SE-Praxis** müssen viel enger **zusammenarbeiten**, so, wie es Chemiker und Ingenieure seit Menschengedenken tun. Die weiterhin bestehende Distanz ist einfach dumm.
  - Die Forschung muss Resultate liefern, die funktionieren.
  - Die Praktiker müssen die Resultate einsetzen.
- Dazu muss wesentlich mehr als bisher **quantitativ** gearbeitet werden: Welchen Aufwand hat die Entwicklung benötigt? Was kostet die Wartung? Wieviele Anforderungen wurden verändert? Wieviele Fehler wurden entdeckt? Und nicht zuletzt: was ist die Software wert?
- In einigen Firmen und Institutionen wird Software noch immer als Anhängsel der Hardware betrachet. Dabei steckt heute der Wert vorwiegend in der Software.
  Und: SE ist weit mehr als Programmierung! **Begreift es doch endlich!**
- ProjektmanagerInnen, die Projekte geräuschlos durchführen und abschließen, sollten weit höhere Wertschätzung erfahren als ihre Cowboy-Kollegen.
- Kurz: Software Engineering braucht eine förderliche **Kultur**.

© J. Ludewig 2020

Ceterum censeo ...

Unser aller Feind ist die **Dummheit** (meine, unsere). Sie ist eine starke und gefährliche Macht, wie Gotthelfs schwarze Spinne, mit der die Dummen paktieren.

Dummheit ist **unwiderlegbar** und **unbesiegbar**. Im SE wie in der Politik.

Denn gegen Bosheit kann man argumentieren, gegen Dummheit nicht.

Widerpart der Dummheit ist die **Weisheit**, die sich u. a. durch **hohe Qualität** zeigt.

Unser Ideal ist Software hoher Qualität, über lange Zeit eingesetzt und gewartet. Solche Software-Qualität ist **möglich**, **nötig** und **kostengünstig**.

Und als Zugabe: Sie ist **Balsam für die Seele**.

Es ist eine **Freude, gute Software zu entwickeln** (und weiterzuentwickeln).

Es ist eine **Freude, gute Software zu benutzen**.

Lassen Sie sich dieses Vergnügen nicht entgehen!

© J. Ludewig 2020

Der blinde Arzt

Stellen Sie sich einen blinden Arzt vor, der zudem keine Lust hat, seinen Patienten zuzuhören.

Kann er ihnen trotzdem sinnvolle Ratschläge geben?

Freilich kann er.

Hören Sie auf mit dem Alkohol und dem Rauchen !

Nehmen Sie ein paar Kilo ab !

Treiben Sie Sport, nicht zu viel, aber regelmäßig !

Hören Sie auf damit, herumzurasen zwischen Ihrem Haus und Ihrem Büro und Sitzungen und Techtelmechteln und ..., machen Sie Pause und sehen Sie ein, dass Sie bei weitem nicht so wichtig sind, wie Sie glauben (oder glauben, sein zu müssen)!

Und, oh Wunder, er hat in den meisten Fällen recht, den Patienten ginge es wirklich besser, wenn sie sich nur an die Ratschläge hielten.

© J. Ludewig 2020

Der blinde Berater

Stellen Sie sich einen blinden Software-Berater vor, der zudem keine Lust hat, seinen Kunden zuzuhören.

Kann er ihnen (Ihnen?) trotzdem sinnvolle Ratschläge geben?

Freilich kann er.

Hören Sie auf, Programme zu schreiben, ohne eine anständige Spezifikation, einen soliden Entwurf zu haben !

Kümmern Sie sich um das Configuration Management !

Sammeln Sie Daten, nicht zu viele, aber regelmäßig !

Hören Sie auf damit, herumzuwuseln zwischen Hunderten chaotischer Klassen, Prozeduren, Modulen, Subsystemen, Interfaces und ..., machen Sie Pause und sehen Sie ein, dass Sie bei weitem nicht so schlau sind, wie Sie glauben (oder glauben, sein zu müssen)!

Und, oh Wunder, ...