



**Antworten auf meine kleine Umfrage  
zur Zukunft des Prüfens und Testens  
im Jahr 2010, 2020, 2035**

HOCHSCHULE BREMEN  
UNIVERSITY OF APPLIED SCIENCES

# Eure / Ihre Antworten

- Vielen Dank
- Ich hatte gar nicht mit so vielen Antworten gerechnet (erhofft schon)!
- Die Antworten sind nun in der Reihenfolge des "Eingangs" in meine Mailbox aufgeführt!
- Viel Spaß beim Lesen!

# Tilo Linz, imbus

2010:

test generierung / model driven testing  
certified tester expert level operativ

2020:

Wir besitzen und nutzen (statistische) Modelle, um den optimalen Testaufwand zu bestimmen

2035

Entwickelt wird nur noch in China und Indien. Die Tester sind billige Europäer.

imbus TestBench Ultra 9.0 ist Weltmarktführer.

Buch »Basiswissen Softwaretest« (3. Auflage) ist ein begehrtes Sammlerstück (wir schreiben an 30. Auflage)

# Jochen Ludewig, Uni Stuttgart

2010

5 a sind so gut wie nichts. In 5 a werden wir Mühe haben zu erkennen, was sich verändert hat. (Fragen wir doch mal: Was hat sich seit 2000 verändert?) In 5 a wird alles so sein wie jetzt; es gibt ein paar neue Bücher, und ein paar Leute werden behaupten, dass das Problem nun endlich erkannt sei. Genau wie heute.

Vielleicht (hoffentlich!) ist erstmals ein Mensch ins Gefängnis gekommen, weil er schlampige Software gemacht oder zugelassen hat. Aber wahrscheinlich geht auch das nicht so schnell.

# Jochen Ludewig, Uni Stuttgart

2020

In 15 a wird die Zahl der Leute, die eine eigentliche Informatik-Ausbildung haben, gewachsen sein, von ca. 20 % vielleicht auf 40 %, höchstens 50 % derer, die von ihrer Tätigkeit her eigentlich eine solche Ausbildung brauchen.

Durch eine veränderte Rechtssprechung wird es eine schärfere Haftung geben, die eine bessere QS unvermeidlich macht. Darum wird es in 15 a weit mehr als heute definierte Prüfprozesse geben, teilweise rein bürokratisch, teilweise auch mit praktischen Folgen, also mit sinnvollen Reviews und systematischen Tests. Und natürlich wird der Markt mehr Werkzeuge anbieten, die das unterstützen.

# Jochen Ludewig, Uni Stuttgart

2035

In 30 a werden unsere Systeme kaum noch im heutigen Sinne programmiert, sondern weit mehr komponiert sein. Dafür wird ein beträchtlicher Teil der Leistungsgewinne in der Hardware draufgehen. Die Komponenten werden mit erheblicher Redundanz arbeiten und damit viele der Fehler, die sie weiterhin enthalten, abfangen. Damit wird der Begriff der Korrektheit obsolet, wir werden ihn durch die Zuverlässigkeit ersetzen. Die Prüfung wird damit zu einer Spezialisten-Aktivität, denn es wird ein (einklagbares) Merkmal der Komponenten sein, wie zuverlässig sie sind. Prüfungen finden dann ganz überwiegend auf der Komponentenebene statt, nach klaren Normen und Gesetzen. Das Zeitalter der Programmier-Künstler (die wir heute noch ausbilden) geht langsam zu Ende.

# Tim Koomen, sogeti NL

2010

The offshoring of IT will continue. The offshore countries will certainly develop good test practices and offshoring of testing will often be chosen. Offshoring requires more formality in the process, so structured testing will get more focus (again). Onshore testing will shift towards acceptance testing, test coordination but also towards very early testing (onshore) of prototypes and such (where testers and developers work very closely).

# Tim Koomen, sogeti NL

2020

10 years is already hard to predict ... But I think IT will continue to grow (rapidly), will become more pervasive (software in everything). This will be very complex, with interfaces between many, many things. Luckily, developers will have many building blocks (services, components, packages). Testing will shift from mostly functional to half functional/half non-functional testing, and will have to move earlier in the development life-cycle: the "end-test" as main test will cease to exist, as it's way to costly to detect the majority of defects in the final software.

# Tim Koomen, sogeti NL

2035

26 years ago the first book on software testing (Myers, '79) was published. These principles are still valid today... Maybe I'm conservative, but I wonder if it will change all that much in the next 30 years. I don't believe in the notion that you can compare IT to mass-production (where you don't test anymore, but just take a few samples every now and then).

# Dr. Ernest Wallmüller, IT Quality Group CH

2010

ES PASSIEREN IMMER mehr Zwischenfälle mit Softwarefehlern zb. Automotiv-Bereich

die Anstrengungen in Sachen Verification & Validation werden verdoppelt. SPICE/ISO 15504 wird durch CMMI abgelöst!!!

2020

erster bemannter Flug zum Mars der Amerikaner wird wegen eines Softwarefehlers abgebrochen.

NASA bildet alle Programmierer in PSP, TSP und TPI aus.

2035

Roboter im Haushalt und in der Arbeitswelt sind stark verbreitet.

Wegen Softwarefehlern entsteht eine Revolte unter den Robotern, die zu erheblichen Störungen in der Gesellschaft führen.

# Klaus Peter Löhr, FU Berlin

2010

Verbesserung von Test Tools. Testen als integrierter Bestandteil agiler Entwicklungsmethoden a la XP. Zunehmende praktische Bedeutung von Model Checking und ähnlichen Ansätzen.

2020

Im Gefolge fortgeschrittener Akzeptanz von deklarativen Sprachen - insbesondere ausführbaren Spezifikationssprachen - weite Verbreitung von High-Level Testing.

2035

[Da traue ich mir kein Urteil zu ;-]

# Hans Schäfer, Software Test Consulting N

## Allgemein

Testen verschwindet nicht. Im Gegenteil, es wird mehr werden. Die Ursache ist mehr Komplexität und mehr Globalisierung, Outsourcing etc. Der Fokus des Testens wird mehr und mehr die Beschaffung von Information über das zu testende Objekt.

Ich denke auch, dass es mehr normal wird, das Prüfen und Testen wie bei anderen Ingenieurdisziplinen zu betreiben. Das bedeutet vor allem, dass man früher daran denkt, es mehr systematisch betreibt und mit Ressourcen ausstattet.

# Hans Schäfer, Software Test Consulting N

2010

Es wird sich durchsetzen, dass auf jeden Fall ein gewisser Komponententest durchgeführt werden sollte und mitgeliefert wird. Werkzeuge wie Junit werden mehr und mehr zum Standard bei der Entwicklung von Komponenten. Auf der anderen Seite sehe ich keine Tendenz dazu, dass dieser Test auch den notwendigen Umfang erreicht, die notwendige Gründlichkeit, denn die tonangebenden Mitarbeiter werden auch weiterhin nur sehr wenig im Testen ausgebildet. Ich denke, dass trotz der Initiativen wie ISTQB und dem Buch "Basiswissen Softwaretest" das Fachgebiet weiterhin ein Stiefkind sein wird.

# Hans Schäfer, Software Test Consulting N

2020

Testwerkzeuge werden in Entwicklungswerkzeuge eingebaut und gehören einfach dazu. Ein rudimentärer Test wird automatisch erzeugt und mit dem System weitergeführt. Schnittstellen werden mehr und mehr standardisiert und durch Testwerkzeuge oder Standard-Testsuiten unterstützt. Statische Analyse oder mehr robuste Programmiersprachen oder Umgebungen werden sich mehr durchsetzen.

Ob in der Ausbildung so viel geschehen wird, will ich nicht voraussagen.

# Hans Schäfer, Software Test Consulting N

2035

TJA.... Vor dreißig Jahren, 1975, hatte das Uni-Rechenzentrum seine Eingabe mit Lochkarten, fuhr die Eisenbahn mit Dampflokomotiven aber der Fahrkartenverkauf ging weit schneller als heute, und ich hatte eine Amateurfunklizenz. Ich konnte damals nicht voraussehen, dass ich heute einen Rechner habe, der weit mehr kann als das Uni-Rechenzentrum und fast alle Kommunikation per Handy und Internet geht. Ich habe damals Relais getestet, als ungelernter Mitarbeiter. Die Relation zum Testen heute? - weiterhin ungelernte Mitarbeiter (Programmierer), aber eine ganz andere Arbeitssituation. ICH WEISS NICHT! (Außerdem bin ich da 83 und wohl unter der Erde).

## J. Siedersleben, FH Rosenheim

- Prüfen und Testen ist ja viel mehr als nur die Prüfung auf Korrektheit. Ich erwarte/erhoffe große Fortschritte bei der kontinuierlichen Prüfung nicht-funktionaler Eigenschaften (Performance, Stabilität, Wartbarkeit, Betreibbarkeit).
- Ich erwarte/erhoffe Fortschritte bei der Spezifikation von Schnittstellen, mit deren Hilfe man Testfälle erstellen kann und deren Vollständigkeit bewerten kann (die Überdeckungsmessungen sind besser als nichts, sagen aber nicht besonders viel aus).
- Ich erwarte/erhoffe viel in Bezug auf die objektive Bewertung von Systemen. Dies wäre hilfreich bei Streitigkeiten über den erreichten Projektstand, bei der Bewertung von teilfertigen Leistungen und letztlich vor Gericht.
- Ich erwarte/erhoffe wenig von der Generierung von Testfällen. Generierung kann hilfreich sein, aber die kniffligen Sachen werden wir uns immer selbst ausdenken müssen.
- Ich überlasse Ihnen die Zuordnung zu den Zeitabschnitten (ich bin kein Hellseher).

# Martin Pol, polteq NL

2010

development testing shifting to development workbenches and package development,  
system testing towards outsourcing suppliers,  
acceptance testing reduced and exclusively to systems management  
testing roles shifting towards monitoring and control of outsourcing

2020

testing is only controlling business processes,  
testing owns the QA-role

2035

testing is comparable to testing video, DVD, cars, etc. by ourselves (the end-users) now, all other tests are included in the production processes

# Karol Frühauf, INFOGEM CH

wie war es doch ... ich bin sehr schlecht in Voraussagen, vor allem, wenn sie die Zukunft betreffen. Deshalb mache ich keine Prognosen, sondern versuche mit *worst* (Befürchtung) und *best case* (Hoffnung) zu antworten.

Hinweis: Für mich ist Prüfen der Oberbegriff von Reviewen und Testen.

2010

Ich befürchte, es wird so aussehen wie heute.

Ich hoffe, dass automatischer Unit Test für die Absolventen irgendeiner Informatik-Ausbildung so selbstverständlich ist wie das Zähneputzen. Reviews in irgendeiner Form auch.

# Karol Frühauf, INFOGEM CH

2020

Ich befürchte, nicht viel anders als heute.

Ich hoffe, dass Software-Entwicklung kein *single*, kein *pair* sondern *team programming* ist. Das Team sitzt in einem (virtuellen) Raum und erarbeitet die Lösung in dem alle Aspekte ausdiskutiert werden, laufend mit einer cleveren Notation auf einer hohen Abstraktionsebene für alle sichtbar fest gehalten werden und unmittelbar auf eine ganze Menge syntaktischer und einige semantische Mängel hin automatisch geprüft werden. Reviews erübrigen sich, außer zu Zwecken der Abnahme. Hauptthema der Prüfung ist die Suche nach den Fehlern, die man bei der Integration macht, vermutlich mit Tests.

# Karol Frühauf, INFOGEM CH

2035

Ich befürchte, ganz anders als heute. Wegen der Mobilität und der allgegenwärtigen, kontinuierlichen Kommunikationsbereitschaft müssen gegenseitige Störungen ausgeschlossen werden, damit meine Herzfrequenz beim Arzt ankommt, beim richtigen Arzt ankommt und nicht als Blutdruck ankommt. Und zwar immer. Mit unseren Prüftechniken hätten wir Mühe, wirtschaftlich zu prüfen, ob dies (immer) zu trifft. Also müssen andere Techniken her.

Ich hoffe, wir müssen nicht mehr prüfen, weil man inzwischen heraus gefunden hatte, wie man Fehler vermeidet und dies auch praktiziert. Ausnahmslos.

# Bernd Hindel, ASQF

hier meine (nicht unbedingt ernst zu nehmenden) Visionen

2010

Komponententests sind größtenteils automatisiert, incl. Testfallfindung  
Integrationstest und Systemstests werden von Arbeitskräften aus  
Niedriglohnländern durchgeführt.

2020

die meisten SW-Tests werden von BST ("Bachelor of Software Testing")  
durchgeführt. Ein Berufsstand, der eine zweijährige Ausbildung nach dem  
Abitur oder der Mittleren Reife voraussetzt. Hauptsächlich in  
Niedriglohnländern etabliert. Es gibt Zertifikate für SW mit unterschiedlichen  
Testgraden an der Software-Tankstelle (70% Oktan, 80% Oktan, 90% Oktan,  
100% Oktan) (Oktan = ok für Tests gegen Anforderungen)

2035

Es gibt nur noch selbst heilende SW, Allianz als Marktführer bei SW-  
Versicherungen

# Rudolf van Megen, SQS

2010

Der Sinn von besserem Testen (Mikro- und Makroökonomisch) wird erkannt sein. Der neueste Standish Report von 2004 ist alarmierend genug.

2020

Die großen Zusammenhänge zwischen Development und Testen werden erkannt worden sein. Traceability und Manageability ist bis dahin ein *must* und nicht mehr *nice to have* - und zwar für alle.

2035

Die Komplexität der Systeme ist inzwischen so groß, dass alle Welt versuchen wird, Fehler zu vermeiden und so früh wie möglich zu finden.

# Ina Schieferdecker, Fraunhofer FOKUS

2010

noch immer werden alle sagen, wie wichtig Testen ist, aber nicht die notwendigen Ressourcen (Zeit, Werkzeuge, Personal) zur Verfügung stellen und sich mit im Wesentlichen mit Unit-Level Tests beruhigen und begnügen. Es wird aber ein Umdenken stattfinden, da einige größere Systemausfälle wegen fehlerhafter Software zu verzeichnen sein werden - auch in der Ausbildung rückt das Thema immer mehr in den Blickwinkel.

# Ina Schieferdecker, Fraunhofer FOKUS

2020

Parallel dazu haben wir Tester aber die Technologie vorangebracht und können sie dann bei Bedarf aus der Tasche ziehen und anwenden. Sowie so wird nun im Wesentlichen modell-basiert entwickelt - und getestet - in einem durchgängigen Prozess: es gibt dabei aber vielfältige Prozesse, die auf den Bedarf der Technologien, Systeme, etc. angewendet werden

2035

Wir haben endlich den SSTV - den Software und System Test Verein, der unabhängig, systematisch und automatisiert Software und Software-basierte Systeme welcher Art auch immer, prüft, Gütesiegel vergibt und darüber Qualitätsstandard durchsetzt.

# Eike Riedemann, Uni Dortmund

2010

Die automatisierte, modellbasierte Testfallgenerierung ist State-of-the-Art. Die Semantik von UML 4.1 ist daher so präzise festgelegt worden, dass Testfälle (fast) automatisch erzeugbar sind, d.h. der *Model Maturity Level (MML) 4* ("precise models") ist erreicht.

2020

Die EU hat eine Software-Test-Richtlinie erlassen. Danach muss für ausgelieferte Software offengelegt werden, welche Tests mit welcher Qualität (z.B. Überdeckungsrate) absolviert wurden. Einige Firmen haben den *Model Maturity Level 5* ("models only") erreicht, d.h. der Code wird automatisch aus den präzisen Modellen erzeugt, die z.B. mit UML 9.0 beschrieben sind. Tester müssen also nur noch die Modelle gegenüber den Kundenanforderungen validieren.

# Eike Riedemann, Uni Dortmund

2035

Die Orientierung an der *Model Maturity* ist aus der Mode gekommen, stattdessen wird der *Requirements Maturity Level* betrachtet. Stufe 5 ("requirements only") wird angestrebt. Es gibt erste erfolgreiche Prototypen zur automatischen Codegenerierung und Testdatengenerierung aus quasi natürlichsprachlichen Anforderungsdefinitionen, die in URL 2.0 (Unified Requirements Language 2.0) beschrieben sind. Die Entwickler von entsprechenden Tools sind Informatiker mit Zusatzqualifikation *Certified Tester*, die mit Linguistikern und KI-Experten zusammenarbeiten.

# Martin Glinz, Uni Zürich CH

Generelle Tendenz:

1. Wir werden massive Fortschritte bei der Automatisierung von Prüfverfahren erzielen; manuelles Testen wird nur in Nischenmärkten überleben.
2. Wir werden uns an Software-Tote gewöhnen wie wir uns an die Verkehrstoten gewöhnt haben. Einzelne, besonders verantwortungslose Entwickler werden zwar bestraft werden (so wie im Straßenverkehr ein Raser, der Menschen auf dem Gewissen hat), aber allgemeine, durchgreifende (und wirksame!) Maßnahmen werden bei der Software ebenso wenig ergriffen werden, wie im Straßenverkehr, da zu aufwendig und zu unbequem.

# Martin Glinz, Uni Zürich CH

Generelle Tendenz:

3. Es entwickeln sich drei parallele Märkte:

a) einer für Billigsoftware, die genau so weit geprüft wird, wie die Prüfungen automatisierbar sind;

b) einer für open source Software, die recht zuverlässig ist, weil die Defekte durch Gebrauch der Software in einer großen Entwicklergemeinde erkannt und eliminiert werden;

c) Teure Software für kritische Systeme, die neben den automatisierten Tests zusätzlich auch sorgfältig manuell getestet/inspiziert wird. Automatisiertes statistisches Testen wird diesen Markt langfristig bedrängen und ihn wahrscheinlich nur im Luxussegment, wo für Handarbeit beliebige Preise gezahlt werden, überleben lassen.

# Martin Glinz, Uni Zürich CH

Generelle Tendenz:

4. Das Problem der nicht adäquaten Anforderungen ist hart und geht nicht weg. Es kann durch automatisierte Gewinnung von Anforderungen aus der Benutzung von Prototypen nur teilweise gelöst werden. Damit sind auch die Grenzen der Testautomatisierung abgesteckt: wo die Anforderungen falsch sind, nützt der automatisierte Test, ob der Code die Anforderungen erfüllt, nichts.

2010

Automatisiertes Modultesten à la JUNIT entwickelt sich zum Standardverfahren.

# Martin Glinz, Uni Zürich CH

2020

**Technisch:** Es entstehen neue Testverfahren, die auf Data Mining und maschinellem Lernen basieren und in der Lage sind, fehlerhafte bzw. verdächtige Stellen im Code zu entdecken. Diese werden insbesondere die manuellen Inspektionen konkurrenzieren und teilweise verdrängen. Die aus dem Modultest heute bekannten Automatisierungsverfahren werden ausgedehnt auf Schnittstellentests und dringen in den Systemtest vor. Für kritische Systeme werden Verfahren entwickelt, bei denen die Anforderungen parallel zum Code mitgeführt und im Betrieb der Software fortlaufend überprüft werden.

**Gesellschaftlich:** In der Rubrik "Vermischtes" der Zeitungen werden neben den Verkehrstoten auch die Softwaretoten gemeldet. Nur in besonders krassen Fällen gibt es öffentlichen Aufschrei und Bestrafung der Schuldigen (genau wie heute im Straßenverkehr).

# Martin Glinz, Uni Zürich CH

2035

**Technisch:** Automatisiertes, statistisches Testen ist zum Standardverfahren geworden. Die notwendigen Benutzerprofile werden mit Verfahren des Data Mining und maschinellen Lernen gewonnen. Die notwendigen formalen Anforderungen werden automatisiert aus der Arbeit von Benutzern mit Prototypen gewonnen. Es werden Systeme entwickelt, welche sich nicht nur fortlaufend selbst überprüfen, sondern bei Abweichungen auch selbst reparieren.

**Gesellschaftlich:** für Normalsterbliche ist nur noch automatisch getestete Software (oder durch eine Benutzergemeinde durch intensiven Gebrauch getestete open source Software) bezahlbar. Manuell getestete / inspizierte Software gilt als Luxus und wird zu entsprechenden Preisen gehandelt.

# Mario Winter, FH Köln

2010

unterliegt Software der Produkthaftung wie alle anderen industriell produzierten Güter auch; damit wird die Qualitätssicherung vollwertiges Mitglied im Kanon der Softwaretechnik.

2020

sehe ich so langsam meiner Pensionierung entgegen und blicke auf eine Konsolidierung der Methoden und Techniken der Qualitätssicherung zurück. Es gibt ein weltweit anerkanntes Curriculum in Sachen QS, und viele der ehemals analytischen Verfahren sind nun bereits konstruktiv berücksichtigt und somit aus dem QS-Kanon "verschwunden". QS betrachtet zunehmend die "politisch-sozialen" Auswirkungen und Möglichkeiten der Systeme.

# Mario Winter, FH Köln

2035

sind die konstruktiven Methoden so ausgereift, dass die QS sich nur noch auf die Validierung und Verifikation der Modelle, nicht aber mehr auf die Software an sich bezieht. Vielleicht gibt es gar keine SW-Entwicklung im eigentlichen Sinne mehr, da diese von den Systemen selbst "übernommen" wird.

# A. Spillner, Hochschule Bremen

2010

Softwareentwicklung erfolgt in Komponenten, Produktlinien (konstruktive Wiederverwendung!).

Zu jeder Komponenten werden die Testfälle mitgeliefert

- als Nachweis des sorgsam durchgeführten Tests und
- als zusätzliche Spezifikation!

Es ist kein Grund erkennbar, warum die Testfälle nicht mitgeliefert werden sollten, es sei denn, es wurde nicht ausreichend getestet!

- Teilweise (bei open source - Projekten) schon heute so!

SW-Produkt = Software + Testware!

# A. Spillner, Hochschule Bremen

2020

Es wird weiterhin neue Konzepte und Methoden für die Entwicklung von Software geben. Es sollte gleich an die Testbarkeit mit gedacht werden! Auswirkungen auf den Test sind zu berücksichtigen, Vor- und Nachteile sind abzuwägen (Objektorientierung brachte viele Vorteile, aber auch (Test-)Nachteile!)

Einbau von Plausibilität (nicht nur bei Übergabe von Parametern) sondern auch in den "Berechnungen" - was wir Menschen auch machen (Bei Ergebnissen fragen wir: »Kann das stimmen?« leider zu nehmend aus der Mode gekommen bzw. es kann kaum noch einer! Überschlagsrechnung!)

# A. Spillner, Hochschule Bremen

2035

Zurück zur Einfachheit!

Kompliziert ist etwas nur solange man es nicht (völlig) versteht!

Software sollte angesehen werden können, dass sie funktioniert!

Beispiele (aus der Vergangenheit / heute):

- Oberon Betriebssystem!
- Toll-Collect - Lösung  
(für die Hauptfunktion einfachste Lösung von 3 Bremer Schülern!)

Warum soll ein Telefon Fernsehbilder empfangen?

Handy mit DVB-T-Empfang

Der japanische Elektronik-Multi MEC hat den Prototypen eines Handys mit DVB-T-Empfänger vorgestellt. 11.07.03

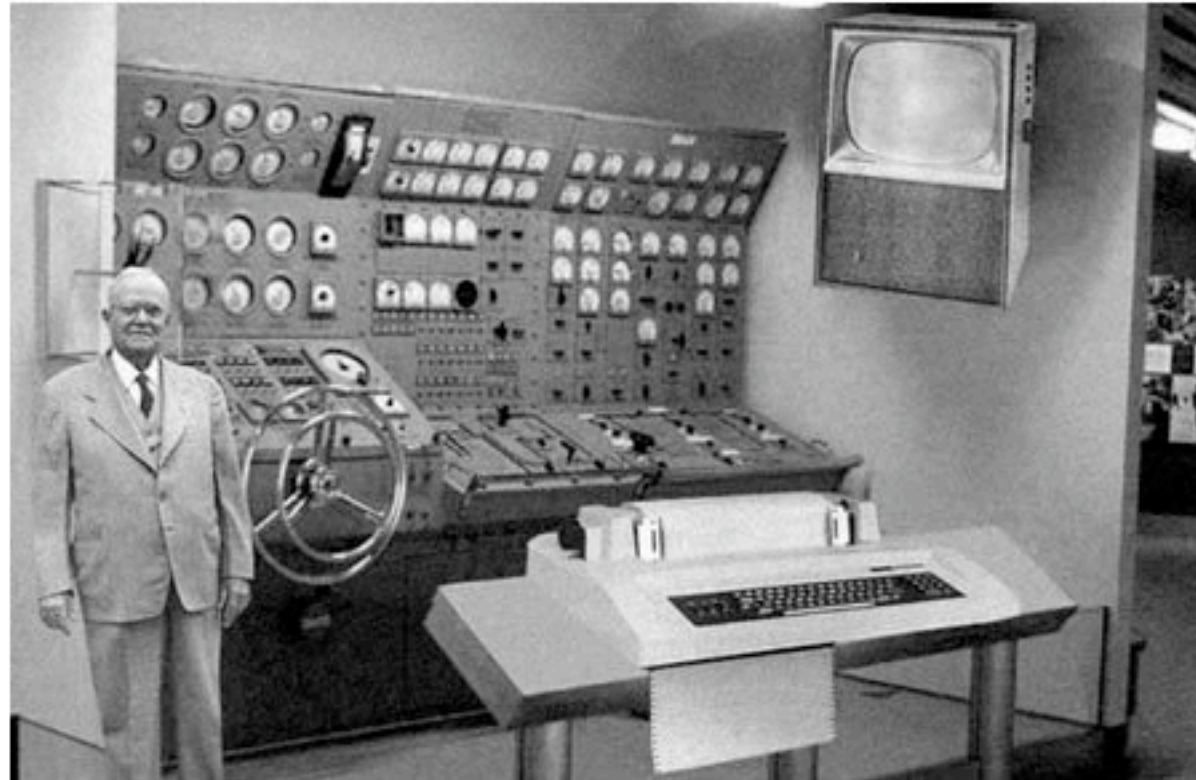


# Prognosen sind risikoreich

- Zum Trost all denjenigen, die sich getraut haben, in die Zukunft zu schauen!
- Computervision des Jahres 1954 für 2004  
(kommt aus dem Internet - also ohne Gewähr!)

[http://www.computerbase.de/news/hardware/komplettsysteme/2004/dezember/computervision\\_jahres\\_1954\\_2004/](http://www.computerbase.de/news/hardware/komplettsysteme/2004/dezember/computervision_jahres_1954_2004/)

# 1954: Vorschau auf 2004



*Scientists from the RAND Corporation have created this model to illustrate how a "home computer" could look like in the year 2004. However the needed technology will not be economically feasible for the average home. Also the scientists readily admit that the computer will require not yet invented technology to actually work, but 50 years from now, scientific progress is expected to solve these problems. With teletype interface and the Fortran language, the computer will be easy to use.*