## Beta, Service Pack 2

Ich verstehe ja die Einwände, daß unendliches Testen nicht bezahlbar ist.
Und das ist es wirklich nicht.
Jedoch darum geht's mir ja gar nicht.

Es ist nur der **total falsche Ansatz, Software so zu entwickeln**.
Seit Jahrzehnten über Assembler, Cobol, PL, C ... (ich kenne 16) eingefahren.
Obwohl **immer wieder** die Erfahrung gemacht wurde,
daß das sehr oft, zu oft, in einem **Chaos** endete.
Nämlich daß eine 10 Jahre alte Software tot ist, nicht mehr pflegbar ist.
Und damit auch das Wissen, das darin steckt, tot ist.

Das Rad wird wieder neu erfunden - mit gleichem Angehensstil,
mit gleichen Verfahren wieder dieselbe Tour.
Das Programm schnell mal eben 'hinrotzen' zwecks purem Geldverdienens.

Es macht sich einfach keiner? Gedanken,
wie dieser immer wiederkehrende Vorgang radikal verändert werden kann.
Einfach durch radikal neue Verfahren der Software-Entwicklung.
Es wurde allerdings bereits wenige male durchexerciert.
Uhd damit ist erwiesen, daß es auch funktioniert.
Und zudem Riesenvorteile hat.

Ein **Beispiel** aus meinem Berufsleben.
Es sollten Geräte produziert werden, die fehlerfrei sein **mussten**.
Und zwar 6 Millionen Geräte pro Jahr. Über 5 Jahre. Verteilt auf die gesamte Welt.
Mit eingebrannter Software, die man also nicht mal so eben über Internet austauschen (ServicePack-3) konnte.

Die Lieferungen:
Hier mal 20.000 Geräte auf einem Containerschiff,
daß erst Wochen später zum Ausladen kam.
Oder mal ein Schwung im Flieger - nach Timbuktu.

Wonach die Geräte erst dann ausgeliefert wurden.

Kann sich einer das **Chaos** vorstellen, da 'mal eben' alle Geräte zu öffnen,
um ein Bauteil auszutauschen? Hilfe, wo sind die überhaupt.
Gleich alle ins Meer kippen und neue liefern wäre preiswerter gewesen.

Das Team war mit 7 Kern-Leuten besetzt, einer davon war ich (> Foto, letzte Seite).
Und es gab keinen Streit, aber eine sehr gute Zusammenarbeit,
weil einfach die (Taiwan-)Chinesen ganz genau wussten um was es geht.
Sie waren/sind derart clever -und auch sehr gut ausgebildet-, daß sich hier der 'Westen' eine ganz dicke Scheibe abschneiden könnte.
Gegen 'die' haben wir -speziell in D- keine Chance.

Mit sauberer Methodik bei der Entwicklung ist es uns gelungen,
das Produkt -und natürlich das Programm- fehlerfrei zu bauen. In einem Jahr.
Ich habe noch das Dankeschreiben.
**Es geht also. Wenn man nur will - und kann.**

**Es ist nämlich total falsch**,
**anzunehmen, daß durch schnelle Entwicklung der üblichen Bananensoftware ein dauerhaftes Geschäft zu machen ist.**

Windows bildet da eine Ausnahme, einfach **deswegen weil es so weit verbreitet ist**, **und die Welt darauf angewiesen ist**, gezwungen wird, damit umzugehen.
M$ kann sich so etwas genau deswegen erlauben, eine oder zwei oder drei 'Service-Packs' herauszuschieben, mit neuen Bananen.
Ein ServicePack ist nämlich nichts anderes als ein Austausch des fast gesamten Systems.
Und ***jetzt erst*** **festgestellte 'Sicherheitslücken'** wären bei sorgfältigem, bedachtem Vorgehen gar nicht erst aufgetreten. Plumpe Ausreden.
**Wenn andere die Macken finden, dann kann sie auch der Hersteller finden.**
Abgesehen davon, daß es solche Lücken in einem System schon mal gar nicht zu geben hat. Falscher Ansatz.

Eine **Horrovorstellung**, daß weltweit von simplen Kiddis Unfug getrieben werden kann, und dadurch Schäden entstehen, die unvorstellbar sind!
**Noch viel schlimmer ist allerdings, das zu wissen, und es nicht prinzipiell zu ändern.**

Das ist vorsätzliche Schädigung. Man **kann** es nämlich ändern.
Zwar eventuell mit einem Hauruck und totalem Umdenken, aber es geht.

**Stattdessen (noch) eine Beta-Version? Noch ein ServicePack?**
**Immer noch nix gelernt?**

Wie war die Rede mit den 23 Zillionen* Fliegen, die sich nicht irren können?

## Warum ist die bekannte Mehode der Entwicklung denn nun falsch?

**1)** Der Kunde kann nie, weder jetzt noch später, sicher sein, daß er ein unter allen Umständen funktionierendes System erhält.
Aber er -und sein Abnehmer- ist davon abhängig.

Man stelle sich folgende **Situation** vor.
Das Verteilerzentrum eines Logistikunternehmens.
3 Uhr nachts, 50 Lastwagen an der Rampe zum Beladen mit auszuliefernden Waren.
Mit Ware, die zu verderben droht (Fisch, Früchte, ...).
Und dann macht das EDV-System die Grätsche, sprich es verreckt.
Ein Katholik müsste für die 'frommen Sprüche' mindestens dreimal zur Beichte.

Der Maschener (HH) Verschiebebahnhof oder der Flughafen Frankfurt wären auch Kandidaten für ein solches Horror-Scenario.

Noch Fragen zur Beta-Version dann?

**2)** Die **wirklichen** Kosten der Software-Entwicklung werden auf den Kunden abgeladen (siehe 1). Zudem trägt **der Kunde allein** das Risiko.

* Zillion: eine Zahl mit beliebig vielen wertdarstellenden Nullen (vor dem Komma).

**3)** Die **'Pflege'** (anderes Wort für 'Fehlerbehebung') fehlerhafter Software verschlingt **regelmäßig mehr Aufwand als die eigentliche Entwicklung.**
Viel mehr, sehr viel mehr.
Verkorkste Programm können u.U. gar nicht mehr gewartet werden.
Die können nur noch weggeschmissen werden.
Wehe dem Software-Unternehmen, daß dann in der Pflicht ist mit Garantie und so.

Auch ich habe so ein Projekt empfohlen zu beerdigen. Man hat getobt.
Aber es nach zwei weiteren kostbaren Jahren eingesehen. Und dann neu gebaut.
Die zwei Jahre noch Personal dafür durchgeschleppt, nämlich die, die davon noch wenigstens etwas Ahnung hatten. Die ursprünglichen Erbauer waren längst weg.

**Neue Programme**, die wirklich neuen Umsatz und Gewinn gebracht hätten, konnten dabei gar nicht mal gebaut werden, weil die alten Fehler nie zuende kamen.
Jeden Tag eine neue Blase. Panik pur.

**Ist das etwa billiger?**

**4)** Die bekannten Software-Methoden haben die Branche erwiesenermaßen in die Sackgasse getrieben.
Nämlich dank nicht nur **möglicher** unsinniger Verfahren (Kiddi-Angriffe, **die bekannt sind**),
sondern auch noch zusätzlich **fehlerhafter Programmierung** an sich,
weltweit unbezifferbaren Schaden zu stiften.

**5)** Die bisherigen etablierten Entwicklungsverfahren **behindern gar weiteren Fortschritt**, der durch andere Verfahren, durch andere Ansätze, sinnvoller möglich wäre. Auch daran sollte man denken.

**Folgerung:**
Es **ist** möglich, fehlerfreie und sichere Software/Programme zu bauen, das ist bewiesen.
Wenn man es nicht tut, dann halst man sich, bzw dem Abnehmer, eine Menge Säcke voller Flöhe auf.
Diese Schnulli-Programme bringen sogar einen kurzfristigen u.U. immensen finanziellen Erfolg (M$). Jedoch zum Schaden der Anwender.

**Mit fehlerfreier Software wäre der Erfolg allerdings noch weitaus größer.**
Und zudem müsste der Kunde nicht bangen - und die Kohlen aus dem Feuer klauben.

Da frage ich mich, warum das so ist.
**Die offensichtliche Lösung: Schnelles Abzocken.**
(Dazu wäre noch mehr zu sagen).

**Gute, fehlerfreie Software ist nämlich eben *nicht* zu teuer (** > MacIntosh**).**
**Sie ist im Endeffekt die einzig richtige Lösung.**

**Eine Beta-Version**
sollte fehlerfrei und wasserdicht sein.
Und lediglich einge Polituren am Erscheinungsbild noch benötigen.

**Dazu muss allerdings ein komplettes Umdenken**
**bei der Entwicklung stattfinden.**

Development Facility
http://www.ust.edu.tw/english/english_621.htm