

Never Touch a Running System

(Wir warten erst, bis nichts mehr geht)

Benedikt Stockebrand

03. Juni 2003

Zusammenfassung

Kaum eine Bauern- oder besser gesagt Sysadmin-Regel wird so oft und so fatal mißverstanden wie „Never Touch a Running System“. Der Vortrag untersucht die Herkunft und ursprüngliche Bedeutung der Regel, ihre Anwendbarkeit auf heutige IT-Umgebungen, gängige Fehlinterpretationen, ihren bewußten Mißbrauch und wie ein Systemadministrator mit diesem Phänomen umgehen kann.

1 Woher kommt die Regel „Never Touch a Running System“?

Auf der Suche nach der Herkunft der Regel gab nur Fehlschläge: Das Jargon File brachte nichts zutage, meine älteste Fachliteratur war genauso wenig hilfreich und auch wenn Google die üblichen „ungefähr 1660“ Hits fand, war doch nichts relevantes—außer der Vortragsankündigung—dabei.

Persönlich war ich davon etwas überrascht.

2 Was war ursprünglich ihre Intention?

Auch die nächste Frage, was die ursprüngliche Aussage der Regel war, bringt eine Überraschung mit sich: Es gibt eine Reihe unterschiedliche Zusammenhänge, in denen die Regel jedesmal eine andere Bedeutung hat:

Warte bis nach Betriebsschluß. Wer schon einmal mit „klassischen“ Produktivsystemen zu tun hatte, die nur zu „Geschäftszeiten“ laufen müssen, wird sofort an diese Interpretation denken. Generell gilt, daß man Arbeiten an einem Produktivsystem in möglichst unkritische Zeitfenster legen sollte—so es denn noch solche Zeitfenster gibt.

Wer zum Beispiel die Server der Finanzbuchhaltung unmittelbar vor dem Zahltag stilllegt, wird sich bei seinen Kollegen nicht beliebt machen.

Das ist nach meinem Wissen die ursprüngliche Aussage der Regel.

Erst denken, dann handeln. So ziemlich jeder, der schon einmal für die Administration von betriebsrelevanten Systemen verantwortlich war, kennt das Phänomen: Irgend etwas geht immer schief und wenn wichtige Services betroffen sind, geht alles durcheinander und vor allem der Adrenalinspiegel in ungeahnte Höhe. In dieser Situation ist es entscheidend, mit sauberen Fehlerbehebungsmethoden das Problem unter Kontrolle zu bringen und sich nicht zu blindem Aktionismus verleiten zu lassen; sonst wird der Schaden nur noch größer.

Ein Beispiel, das ich von einem Kollegen gehört habe, betraf ein RAID5-Plattensystem. Eine Platte war ausgefallen, das Array machte sich durch eine eingebaute Hupe lautstark bemerkbar. Der Servicetechniker, der die defekte Platte austauschen sollte, zog während der Kunde sich noch über besagte Hupe beschwerte versehentlich die falsche Platte aus dem laufenden Array (und besagter Kollege durfte anschließend ein Disaster Recovery durchführen)...

Auch wenn dieser Punkt offensichtlich wichtig ist, bleibt er doch eher ein Thema, das unter dem Stichwort „handhabbare Systeme“ in der Systemarchitektur und im Prozessmanagement eine zentrale Rolle einnehmen sollte. Deshalb betrachten wir ihn im Rahmen dieses Vortrags nicht weiter.

Tanz immer nur auf einer Hochzeit. Im Umgang mit Produktivsystemen ist es wichtig, sich immer auf ein einzelnes Problem zu konzentrieren und nicht unnötig an mehreren Fronten gleichzeitig zu arbeiten.

Vor längerer Zeit hatte ich einmal das Vergnügen, ein angeblich gehacktes System untersuchen zu dürfen. Ergebnis: Die externen „Application Admins“, die auf dem System dummerweise Root-Rechte hatten, hatten nicht nur an ihrer Anwendung herumgeschraubt, sondern nebenher für Root eine „richtige“ Shell eingerichtet: `/bin/bash`. Das System war aber kein Linux, sondern ein Solaris, so daß man sich anschließend als Root mangels existierender Shell nicht mehr einloggen konnte.

Neben der persönlichen Arbeitsweise betrifft dieses Problem auch die Betriebsorganisation: Nur wer Aufgaben der Reihe nach abarbeiten kann, statt im Interrupt-Betrieb seine Zeit zu verlieren, wird auch Qualität liefern.

Unnötige Risiken sind genau das: Unnötig. Es kommt gelegentlich vor, daß an einem laufenden System Arbeiten durchgeführt werden, die keinerlei Nutzen bringen, aber möglicherweise zu erheblichen Problemen führen können.

Auch wenn die meisten Sysadmins das klassische Beispiel in dieser Kategorie, das Flashen von PC-BIOSsen, verstanden haben, gibt es genug Stellen, wo dieses Verhalten außer Kontrolle gerät: Unter Linuxern ist es oft der neueste Kernel, der unbedingt installiert werden muß, ein namhafter Unix- und Hardware-Hersteller erwartet von Kunden mit Problemen regelmäßig zuerst die Installation der aktuellen „Recommended Patches“ (obwohl sie keine „Required Patches“ sind), bevor er überhaupt vertragsgemäß anfängt, Probleme zu suchen und eventuell zu beheben und eine andere große Softwarefirma überredet ihre Kunden regelmäßig im Abstand von zwei bis drei Jahren, neue, teilkompatible Versionen ihrer „Betriebssysteme“ und „Büroanwendungsprogramme“ zu kaufen und zu installieren.

Die letzten beiden Beispiele zeigen, daß das Problem nicht allein in der Systemadministration, sondern nur zusammen mit dem übergeordneten Management lösbar sein kann.

3 Warum ist sie heute so gefährlich?

Wo sind nun die Probleme mit diesen Interpretationen der Regel? Soweit sind sie doch eigentlich sehr schlüssig. Der Untertitel des Vortrags verrät das Problem: „Never Touch“ führt allzu oft dazu, daß potentiell kritische Probleme erst behoben werden, nachdem sie zu einem Schaden geführt haben.

3.1 Verschleppte Aufgaben

Die wenigsten Sysadmins haben heute so viel Langeweile, daß sie in blinden Aktionismus verfallen. Weit verbreitet ist aber das Problem, daß aus Zeitnot wichtige aber nicht unbedingt eilige Arbeiten immer wieder hinausgeschoben werden, bis sie dann nicht nur wichtig sind, sondern auch eilig werden—weil eben „nichts mehr geht“. An dieser Stelle verkommt die Regel zu einer Ausrede—es läuft, also kann ich die Aufgabe, das ich ja durchaus sehe, erstmal liegenlassen.

Mit der üblichen Arbeitsüberlastung führt das aber dazu, daß die Aufgabe dauerhaft liegenbleibt. Dann gibt es mehrere Möglichkeiten, wie es weitergeht:

1. Die Aufgabe kocht kurz danach hoch und wird unter Druck doch noch nachgearbeitet. Das ist der harmloseste Fall.

Wer also den Virenschanner zu spät aktualisiert oder einen Security Fix für seinen Apache nicht zügig einspielt, wird möglicherweise diese Situation erleben.

2. Unangenehmer wird es, wenn die Aufgabe in Vergessenheit gerät und es erst eine ganze Weile später zu einer Störung kommt. Dann ist die Ursachenbestimmung deutlich aufwendiger und die Störungsbeseitigung dauert entsprechend länger.

Wer erstmal Routing-Einträge, den Hostname oder IP-Filterregeln „von Hand“ eingetragen hat, ohne sie in der Boot-Konfiguration nachzutragen, oder noch besser, sie zwar eingetragen, aber nicht getestet hat, wird bei der Fehlersuche nach einem späteren Reboot seine Freude haben—es sei denn, es erwischt einen ahnungslosen Kollegen.

3. Wenn nun eine ganze Reihe solcher Altlasten zusammenkommen, verlängert sich nicht nur die Ursachenbestimmung, sondern es kommt auch noch zu Abhängigkeiten, die erhebliche Schwierigkeiten bei der Problembehebung verursachen können.

Bei einem DoS-Angriff auf den zentralen Webserver kann das heißen, daß man zuerst einmal einen aktuellen Kernelpatch gegen eine neue DoS-Vulnerability einspielt, ohne daß es Erfolg hat. Der Austausch des Apaches, der noch nicht aktualisiert wurde, weil ein Dutzend verschiedener Module in genau ausgetüftelter Reihenfolge eincompiliert werden müssen und das beim letzten Mal schon eine Woche mühsames Ausprobieren und Sourcen-Lesen

gekostet hat, bringt außer Zeitverlust auch keine Besserung. Kann ja auch nicht—wenn sich das Netzkabel mit der abgebrochenen Arretierzunge, das ein Kollege im Patchfeld mit Klebeband festgemacht hat, gelockert hat.

Wenn das Problem aber doch ein Sicherheitsloch im Uralt-Kernel war, aber der eingesetzte und hochgradig Compile-getunte Apache mit einem aktuellen Kernel nicht läuft und vielleicht die Hardware sich mit dem aktuellen Kernel nicht verträgt, führt die Fehlerbehebung im schlimmsten Fall dazu, daß erstmal neue Hardware angeschafft werden muß. Bei einem PC ist das noch unkritisch, bei einem analogen Fall mit einer Sun E 3500 aufwärts tun alleine schon die Lieferfristen weh.

Wirklich kritisch wird die Konstellation, wenn eine unternehmenskritische Server-Application in einer Uraltversion auf einer SUN Sparc Station 20 unter Solaris 2.5.1 läuft, zu der es nur Clients unter Windows 95/98 und NT gibt und dann dank einer anderen Anwendung die Clientrechner zwingend auf Windows 2000 oder .NET umgestellt werden müssen, wofür es nur eine neue Client-Version gibt, die eine neue Server-Version voraussetzt, die nur unter Solaris 8 aufwärts läuft.

4. In der nächsten Stufe wird die Aufgabe an allen Fronten einfach umgangen. Das führt nicht nur zum allseits beliebten „Das haben wir schon immer so gemacht“, sondern macht es extrem schwierig, das Problem später endgültig aus der Welt zu schaffen.

Ein klassisches Beispiel hier ist die nicht durchgezogene Einführung einer flächendeckenden DNS-Infrastruktur. Da behelfen sich manche Leute mit `scp`-verteilten `/etc/hosts`, die fast zwangsläufig bei diversen Problemen „von Hand optimiert“ werden, was dazu führt, daß z.B. der Loghost für jeden Server in der `/etc/hosts` fest per IP-Adresse eingetragen ist—natürlich immer unter dem Namen `loghost`. Wenn dann irgendwann die Ausfälle durch versehentlich doppelt benutzte IP-Adressen überhand nehmen oder aus anderen Gründen der Einsatz von DNS unumgänglich wird, muß diese Altlast, natürlich unter Zeitdruck, geradegezogen werden. Ohne dabei die eine oder andere Maschine versehentlich zeitweise außer Gefecht zu setzen, ist das in einem größeren Umfeld keine „Herausforderung“ mehr, sondern ein Himmelfahrtskommando.

5. Es gibt noch eine letzte Variante, das Endstadium dieser Entwicklung: Wird ein System höchstens symptomatisch betreut, entwickelt es durch kurzsichtig und ohne Rücksicht auf den Gesamtzusammenhang durchgeführte Flickschusterei immer mehr Ungereimtheiten, die es einem neuen Sysadmin unmöglich machen, sich in das System zuverlässig einzuarbeiten. Wenn die Betreuung des Systems irgendwann immer schneller wechselt, weil niemand sich mehr damit herumärgern will, oder das System auch nur lange genug nicht mehr angefaßt wird, ist niemand mehr in der Lage, es zuverlässig zu betreiben geschweige denn wieder in einen sauberen Zustand zu überführen. Wird ein System im Rahmen eines Projektes unter Zeitdruck von Externen aufgesetzt, kann es schon rechtzeitig zur Inbetriebnahme diesen Zustand erreichen.

Ich habe in der Vergangenheit mehrere solche Zombie-Systeme näher kennengelernt (als mir lieb war), kann aber aus Vertraulichkeitsgründen keine Beispiele nennen. Nur so viel: Systeme in diesem Stadium sind nicht mehr zu retten, sie müssen durch einen sauber hochgezogenen Nachfolger ersetzt werden. Das möglichst problemarme Umschalten auf den Nachfolger ist eine der letzten großen Herausforderungen der IT-Welt.

In allen angeführten Fällen hat ein „Never Touch a Running System“ dazu geführt, daß hinausgeschobene Aufgaben, die sich eben doch nicht von selbst erledigt haben, zu teilweise erheblichem Mehraufwand und vor allem zu unnötigen Betriebsstörungen geführt haben.

Langfristig führt das dazu, daß die Systemadministration immer aufwendiger wird, deshalb immer mehr Arbeiten liegenbleiben und so ein Teufelskreis entsteht, der dazu führen kann, daß eine ganze IT-Umgebung in sich kollabiert.

3.2 Kommunikations- und Managementprobleme

Dann gibt es zwei Probleme, die fast ausschließlich in Verbindung mit nicht funktionierender Kommunikation zwischen Technik und nicht-technischem Management auftreten.

Manager denken „in anderen Dimensionen“ als der durchschnittliche Sysadmin: Zuallererst einmal verwalten sie Ressourcen wie Geld und Manpower. Diese Ressourcen versuchen sie möglichst profitabel einzusetzen; das ist im Kern ihr Job. Deshalb priorisieren sie Aufgaben mit dem Ziel, mit ihren beschränkten Ressourcen möglichst viel zu erreichen. Um die Prioritäten festzulegen, müssen sie aber wenigstens im Ansatz verstehen, warum Aufgaben wichtig sind—und das in vielen Fällen, ohne die technischen Hintergründe zu verstehen.

Außerdem sind viele Manager einmal mit Hilfe des sogenannten „Eisenhower-Diagramms“ darauf getrimmt worden, Aufgaben liegenzulassen, die nicht unmittelbar akut sind. Das hilft vielleicht, um sich in einer akuten Krise auf die unmittelbar überlebenswichtigen Probleme zu konzentrieren, aber einen Weg aus dieser Krise heraus zeigen sie nicht auf. Stattdessen führen sie zu einer gewissen Kurzsichtigkeit, die Gerstner [Ger02] im Zusammenhang mit der Veröffentlichung der nächsten Quartalszahlen als Quartalskurzsichtigkeit bezeichnet, und die letztlich immer nur darauf ausgerichtet ist, akute Probleme kurzfristig unter den Teppich zu kehren.

Im Zusammenhang mit Software-Projekten haben Yourdon [You97] und Roberts und Roberts [RR00] die Auswirkungen dieses Management-Stils untersucht; ihre Ergebnisse sind auch für Sysadmins sehr interessant. DeMarco [DeM93] beschreibt in aller Deutlichkeit, was bewußte Ressourcenverknappung für Folgen hat.

Für einen Sysadmin hat diese Denkweise zwei potentielle Auswirkungen:

Dringend nötige Neuerungen, die kurzfristig Geld kosten, werden gerne mit der Begründung „das ist bisher gelaufen und es ist eure Aufgabe dafür zu sorgen, daß das so bleibt“ abgeblockt. Dabei gibt es zwei Gründe, warum es in der IT keine „endgültigen Lösungen“ geben kann: Die Anforderungen an Systeme nehmen im allgemeinen ständig zu und die Technik entwickelt sich noch immer rasant weiter.

Ähnlich sieht es aus, wenn „proaktive“ Arbeiten nicht genehmigt werden, weil sie zu Ausfällen führen könnten oder eine Downtime zwingend voraussetzen. Diese Variante ist besonders deshalb so kritisch, weil sie später oft zu schwerwiegenden Ausfällen führt und in der Folge die Suche nach einem Schuldigen regelmäßig bei den Sysadmins endet.

Es gibt vermutlich Ausnahmefälle, in denen ein Management absichtlich versucht, die eigene Technik durch so ein Verhalten zu sabotieren—vielleicht, um sich eine Ausrede zu verschaffen, sich des ungeliebten Kindes IT durch Outsourcing zu entledigen. Aber nach meiner Erfahrung ist in den meisten Fällen einem Management nicht bewußt, oder besser gesagt nicht von der Systemadministration bewußt gemacht worden, wie kritisch die Situation tatsächlich ist. Damit werden andere Probleme außerhalb der Systemadministration höher priorisiert und langfristig verkommt die IT-Landschaft.

4 Wie wird „Never Touch“ mißbraucht?

Es ist schon schlimm genug, daß „Never Touch“ inzwischen so oft mißverstanden wird. Noch problematischer ist, daß es gelegentlich auch bewußt mißbraucht wird.

4.1 Historische Fehlentscheidungen

Aus Angst, für eine tatsächliche oder vermeintliche Fehlentscheidung verantwortlich gemacht zu werden, kommt es vor, daß „suboptimale“ Lösungen vehement verteidigt werden.

Vor fünf bis zehn Jahren war der Einsatz von NIS+ statt NIS sicherlich in vielen Fällen sinnvoll. Aber heute ist LDAP+TLS+PAM sinnvoller, insbesondere in heterogenen Umgebungen. Wer erklärt der Geschäftsführung, daß der Aufwand, den man mit NIS+ vor einigen Jahren betrieben hat, inzwischen überholt ist?

4.2 Nach mir die Sintflut

Insbesondere externe Mitarbeiter, die wissen, daß sie die Folgen einer verschleppten Aufgabe nicht mehr selbst ausbaden müssen, lassen gelegentlich langfristig wichtige Aufgaben liegen. In vergleichsweise harmlosen Fällen will man nur nicht mit kurzfristig unproduktiven Aktivitäten aufpassen. Es soll aber auch vorkommen, daß dem Nachfolger Zeitbomben hinterlassen werden mit dem Ziel, sich in Richtung der Geschäftsführung zu profilieren, weil „bei mir das alles problemlos funktioniert hat.“

4.3 Machtpolitik

In eine ähnliche Richtung gehen machtpolitische Grabenkämpfe. Die sind mit „technischer“ Logik nicht zu verstehen, aber trotzdem sehr real:

„Nein, ein Backup Domain Controller unter Linux kommt mir nicht ins Haus, das funktioniert sowieso nicht. Außerdem müßte ich damit ja Macht an die Unix-Fraktion abgeben. Und überhaupt, anschließend wird dann auch noch Exchange entsorgt und am Ende bleibt mir nur noch die Betreuung von 08/15-Desktop-PCs—oder sollen die etwa auch auf Linux umgestellt werden?“

„PC-Server kommen mir nicht ins Haus, die sind für den Einsatz in „Mission Critical“-Umgebungen nicht zu gebrauchen. Außerdem könnte mir dann das Hardware-Budget gekürzt werden, und wenn das Controlling sich dann noch einmischt, wo Linux/i386 statt Solaris/SPARC (oder AIX/RS6000 etc.), dann kann ja jeder kommen. . .“

„Bleibt mir weg mit eurem komischen Postfix (oder Exim (oder Qmail (oder . . .))), mein Sendmail läuft und solange ihr die Finger davon laßt, Sorge ich schon dafür, daß das auch so bleibt.“

Ich denke, diese Beispiele brauchen keine weitere Erläuterung.

4.4 Erhalt der eigenen Existenzberechtigung

Eine letzte Variante ist, ein System so verkommen zu lassen, daß es nur noch von einer einzigen Person einigermaßen betrieben werden kann. Das betrifft hauptsächlich externe Mitarbeiter, die sich unersetzlich machen wollen, mir ist aber mindestens ein Fall bekannt, wo ein interner Mitarbeiter diese Masche so erfolgreich angewandt hat, daß er am Ende gekündigt wurde, nachdem er wiederholte Aufforderungen ignoriert hat, seine Arbeit zu dokumentieren.

5 Was kann ein Sysadmin dagegen tun?

Gibt es Möglichkeiten, sich gegen diese Phänomene zu wehren? Teilweise, wobei viel von der Unterstützung von Seiten des Managements abhängt.

5.1 Subversive Eigeninitiative—eine Nichtlösung

Was für Möglichkeiten gibt es, dringend nötige Arbeiten durchzuführen, wenn das Management sich auf die Position „Das ist bisher gelaufen und es ist eure Aufgabe, dafür zu sorgen, daß das so bleibt“, oder äquivalent „Es gibt keine Probleme, nur Herausforderungen“ zurückzieht und Änderungen am System verbietet?

Wer diese Situation zum ersten Mal erlebt, wird mit hoher Wahrscheinlichkeit versucht sein, die Arbeiten „auf eigene Verantwortung“ durchzuführen. Dabei wird er lernen, daß er damit das Verhältnis zwischen ihm und dem Management nicht gerade fördert. Wenn es Probleme gibt, wird er dafür zur Rechenschaft gezogen, im schlimmsten Fall bis hin zur fristlosen Kündigung und möglicherweise straf- und zivilrechtlichen Konsequenzen—der Perl-Guru Randy Schwartz durfte etwas ähnliches vor einigen Jahren erleben. Selbst wenn es klappt, wird es nachher heißen „das war ein völlig unnötiges Risiko, das wäre auch so weitergelaufen“.

Wer nun die Änderungen unbemerkt durchführt, wird sich damit erstens unglaublich machen, weil es ja offensichtlich auch ohne sie alles läuft, und zweitens das Desaster nur vor sich herschieben: Wenn dann irgendwann wirklich gar nichts mehr geht, wird man ihm auch noch Inkompetenz vorwerfen.

Schließlich gibt es die Variante, sich zwar außerhalb der bezahlten Arbeitszeit schlau zu machen, damit man wenn es soweit ist schnell und kompetent handeln kann, aber nichts an den produktiven Systemen zu tun. Das kostet viel Zeit und möglicherweise noch mehr Geld für Bücher, Software, Hardware und im schlimmsten Fall den Scheidungsanwalt. Wer das versucht, sollte sich deshalb vorher überlegen, wie viel Engagement ihm diese Maßnahme wert ist.

Ich halte diese Form von Eigeninitiative für bedenklich, denn sie geht ganz auf Kosten der Sysadmins. Mir ist ein Fall bekannt, wo das Management absichtlich darauf hingesteuert hat, daß ein Sysadmin sich auf eigene Kosten in seiner Freizeit umfangreich weiterbildet, sprich auf eigene Kosten und im Urlaub zu Fachtagungen fuhr. Das fing ursprünglich als Ausnahmereaktion an, wurde dann aber sehr schnell als „normale Vorgehensweise“ etabliert. Der private abendliche Besuch eines SAGE@GUUG-Vortrags oder der lokalen Unix/Linux User Group kostet kein nennenswertes Geld und schon gar keinen Urlaub, aber die Fahrt zur CeBIT oder Systems einschließlich Übernachtung vor Ort, um sich aktuelle Backup-Lösungen anzusehen, oder eine einwöchige Schulung beim Hersteller sprengen jeden privat tragbaren Rahmen.

5.2 Kampfrhetorische Selbstverteidigung

Wenn in einer Diskussion mit „Never Touch a Running System“ rabulisiert wird, kann man manchmal schon mit einem „Dann müssen wir eben warten, bis es kracht“ oder alternativ auf neudeutsch mit „An Ounce of Prevention is Worth a Pound of Cure“ ein rudimentäres Problembewußtsein schaffen.

Wer in einem „hochgradig politisierten Umfeld“ mit solchen Situationen konfrontiert wird, sollte darüber hinaus einen Blick in entsprechende Literatur werfen; mir persönlich hat ein Buch von Ruede-Wissmann [RW93] insofern geholfen, daß ich inzwischen wenigstens im Nachhinein merke, wenn ich über den Tisch gezogen worden bin—was nicht heißt, daß ich gegen einen Köhner auch nur die Spur einer Chance habe.

5.3 Selbstorganisation, Dokumentation und Kommunikation mit dem Management

Um anstehende Aufgaben sinnvoll zu organisieren, ist ein Problem Ticket System essentiell wichtig. Ob man dabei eine kommerzielle Lösung, eine Zettelwirtschaft mit A5-Zetteln oder ein großes Whiteboard eingesetzt, ist dabei zweitrangig und stark abhängig von den lokalen Gegebenheiten. Wichtig ist, daß irgendwo die lange liegengelassenen Aufgaben zu erkennen und wiederzufinden sind. Ein ständig wachsender Stapel von Altlasten wird damit wirkungsvoll dokumentiert, besonders wenn man bei regelmäßigen Meetings mit dem Management damit zeigt, daß der Rückstau kontinuierlich wächst. Ein Manager, der außerdem auch noch aufgefordert wird, die Aufgaben zu priorisieren, wird eher akzeptieren, daß er selbst etwas gegen die Situation tun muß (wobei Ausnahmen die Regel bestätigen). Limoncelli und Hogan [LH02] haben einige hilfreiche Tips zu der Thematik.

Der Umgang mit versteckten Abhängigkeiten ist deutlich problematischer. In manchen Fällen hilft hier eine Tabellenkalkulation. Alternativ läßt sich Software mißbrauchen, die GANTT-Charts erzeugt; es muß nicht unbedingt für teures Geld MS Project sein, für diesen speziellen Fall reicht z.B. möglicherweise auch `mrproject` oder ähnliches. Das ist allerdings mit meßbarem Aufwand verbunden und dadurch potentiell kontraproduktiv. Außerdem ist es unwahrscheinlich, daß diese Abhängigkeiten sinnvoll verwaltet werden, wenn die Ressourcen fehlen, sie zu bearbeiten. Ich persönlich vertrete eher den Standpunkt, daß eine Policy, die versucht, frühzeitig Upgrades durchzuführen, und mehr Wert auf dauerhafte Lösungen als auf Flickschusterei legt, dieser Thematik

eher gerecht wird und das Verwalten von Abhängigkeiten höchstens dann hilft, wenn man seinem Management die Situation klarmachen will.

Um Probleme sauber zu lösen, muß aber der erwähnte Teufelskreis durchbrochen werden—wer in kurzfristigen Problemen versinkt, kann sich keine grundlegenden Gedanken machen, wie man die Situation wieder in geregelte Bahnen lenkt. Ohne Unterstützung durch das Management ist das im besten Fall schwierig. Möglicherweise lassen sich kurzfristige Maßnahmen des Managements aber ausnutzen, wenn sie zeitweise den nötigen Freiraum verschaffen, um grundlegende Aufgaben endlich anzugehen.

5.4 Offene Konfrontation mit dem Management

Bleibt die Frage, wie man mit einem Management umgeht, das überlebenswichtige Maßnahmen, wie zum Beispiel die Anschaffung einer adäquaten Backup-Lösung, regelmäßig verweigert.

An dieser Stelle bleibt meiner Meinung nach kurzfristig nur die Möglichkeit, auch nach juristischen Maßstäben sauber zu dokumentieren, daß man das Management über die Probleme und vor allem ihre wirtschaftliche Tragweite informiert und zum Handeln aufgefordert hat. Es ist an diesem Punkt entscheidend wichtig, daß sich die Dokumentation im Zweifelsfall auch vor Gericht verwenden läßt—insbesondere für freie Mitarbeiter ist das unter Umständen existenzentscheidend. Diese Maßnahme, konsequent durchgeführt, rüttelt in vielen Fällen ein Management wach—spätestens, wenn man nach einer Empfangsbestätigung für seine Dokumentation fragt.

Wenn auch das tatsächlich nichts hilft, kann man noch versuchen, die Situation weiter zu eskalieren—was aber im allgemeinen langfristig das Verhältnis zwischen Sysadmin und Management sehr belastet. Wenn auch das nichts ändert, bleibt nach meinem Verständnis nur noch der „geordnete Rückzug“, die „Suche nach neuen Herausforderungen“ oder, platt gesagt, ein möglichst schneller Jobwechsel.

Zum Glück sind derart pathologische Fälle selten; spätestens das große Dotcom-Sterben dürfte hier die Masse solcher Umgebungen ihrem unvermeidlichen Schicksal zugeführt haben.

6 Das Selbstverständnis in Projekten

Weit unspektakulärer als die Entscheidungsschlacht mit dem Management, dafür aber um so weiter verbreitet, sind Probleme, die aus dem Selbstverständnis von Projekten entstehen, die neue Systeme in die Welt setzen. Auch wenn das mit der klassischen Systemadministration nichts oder nur ganz am Rande zu tun hat, sind doch Sysadmins am Ende diejenigen, die in der IT eines Unternehmens eine gewisse Kontinuität auch über einzelne Projekte hinaus bewahren und oft genug die einzigen sind, die diese Probleme potentiell verhindern können.

Es ist meine persönliche Meinung, daß der weitverbreitete Zyklus Hauruck-Projekt, Stillhalten bis der Leidensdruck nicht mehr zu ertragen ist und ein neues Hauruck-Projekt, in sich fundamental kontraproduktiv ist. Aber im Kontext dieses Vortrags haben zwei Phänomene, die sich unmittelbar aus diesem Projektverständnis ergeben, besondere Bedeutung für die Sysadmins, die während des Projekts und vor allem danach mit dem System involviert sind. Beide sind Folgen der Annahme, daß nach der Inbetriebnahme an dem System keine wesentlichen Änderungen mehr nötig sind und es beliebig lange unverändert weiterläuft.

6.1 Wozu eine Testumgebung?

Wenn nach der Inbetriebnahme keine wesentlichen Änderungen mehr anfallen, wird nach dem Projekt eine Testumgebung nicht gebraucht. Vor der Inbetriebnahme kann auf der zukünftigen Produktivumgebung entwickelt, getestet und installiert werden. Also kann man hier viel Geld sparen.

Erste Folgen dieser Einstellung zeigen sich erst, wenn das Projektteam sich wieder aufgelöst hat, das Einspielen von Betriebssystempatches zum ersten Betriebsausfall geführt oder die neu

installierte Austausch- oder Erweiterungsmaschine mangels entsprechender Doku nach mehreren Wochen immer noch nicht richtig läuft.

Wenn dann irgendwann in ferner, ferner Zukunft, also in zwei bis drei Jahren, einmal die benutzte Hardware, das Betriebssystem oder die zugrundeliegende Datenbank dank abgelaufenem Support auf einen neuen Stand gebracht werden muß, steht eine schwierige Entscheidung an: Lassen wir das System so, wie es ist, und verzichten auf Herstellersupport oder riskieren wir einen existenzgefährdenden Betriebsausfall, wenn wir ohne vorhergehende Tests ein Upgrade versuchen?

Ich behaupte nicht, daß in allen Fällen eine Testumgebung in voller Größe ständig bereitgehalten werden muß. Aber ohne den Zugriff auf eine Testumgebung kann ein zuverlässiger Betrieb langfristig nicht sichergestellt werden.

6.2 Die Lebenserwartung eines Systems

Wer schon einmal einer Geschäftsführung gesagt hat, daß ein fünf Jahre altes System reif für den Gnadenstrom ist, kennt vielleicht das entsetzte „aber das ist doch erst fünf Jahre alt, wir sind davon ausgegangen, daß das mindestens zehn Jahre läuft!“

Unabhängig von der Ursache für diese gängige Fehleinschätzung sind die Auswirkungen doch fatal. Deshalb sollte man schon während des Projekts die Lebenserwartung der benutzten Komponenten dokumentieren: Wie lange wird für die Hardware noch die Ersatzteilversorgung sichergestellt sein? Was ist mit Security Patches für das Betriebssystem und andere sicherheitskritische Komponenten? Was ist mit dem immer wieder gerne diskutierten Herstellersupport?

Eine solche Aufstellung ist auch für das Management sehr wichtig: Erstens kann es sich damit überlegen, wann welche Investitionen in der Zukunft nötig werden. Zweitens ist ein RoI¹ von fünf Jahren bei einer Lebenserwartung des Gesamtsystems von drei bis vier Jahren ein Signal, das so ziemlich jeder Manager versteht.

7 Fazit

Im Verlauf dieses Vortrags ist hoffentlich deutlich geworden, daß „Never Touch a Running System“ keine Universalweisheit der Systemadministration ist, sondern in vielen wichtigen Fällen hochgradig kontraproduktiv ist.

Weiter sollte der Vortrag einige Hinweise gegeben haben, warum die Kommunikation zwischen Technik und Management manchmal so schwierig ist und wie man als Techniker dazu beitragen kann, sie zu verbessern.

Schließlich sind in heute typischen IT-Landschaften Sysadmins die einzigen, die in der Technik eine langfristige Kontinuität bewahren können. Auch wenn das normalerweise nicht explizit als Aufgabe in einer Stellenbeschreibung genannt wird, ist diese Funktion existentiell wichtig.

Literatur

- [DeM93] Tom DeMarco. Lean and mean. In *[DeM95]*, chapter 1. Dorset House, 1993.
- [DeM95] Tom DeMarco. *Why Does Software Cost So Much?* Dorset House, 1995.
- [Ger02] Louis V. Gerstner. *Who Says Elephants Can't Dance?* Harper Collins, 2002.
- [LH02] Thomas A. Limoncelli and Christine Hogan. *The Practice of System and Network Administration*. Addison-Wesley, 2002.
- [RR00] Sharon Marsh Roberts and Ken Roberts. Do I want to take this crunch project? In *[WBK00]*, pages 25–42. Dorset House, 2000.

¹Return of Investment, sprich der Zeitpunkt, bis zu dem man die Investitionen wieder hereingeholt hat.

- [RW93] Wolf Ruede-Wissmann. *Satanische Verhandlungskunst—und wie man sich dagegen wehrt*. Wilhelm Heyne Verlag, 1993.
- [WBK00] Gerald M. Weinberg, James Bach, and Naomi Karten, editors. *Amplifying Your Effectiveness*. Dorset House, 2000.
- [You97] Edward Yourdon. *Death March—The Complete Software Developer’s Guide to Surviving “Mission Impossible” Projects*. Prentice Hall, 1997.

Über dieses Manuskript

Dies ist das Manuskript eines Vortrags, den ich am 03. Juni 2003 vor der Karlsruher Ortsgruppe der SAGE@GUUG gehalten habe. Es greift einige Fragen auf, die im Anschluß an meinen Vortrag „Nach dem Boom: Ein Gesundheitscheck für IT-Systeme“ beim GUUG-Frühjahrsfachgespräch 2003 aufgekomen sind.

Über den Autor



Der Autor ist Dipl.-Inform. und freischaffender Systemarchitekt im Unix- und TCP/IP-Umfeld.

Er unterstützt IT-Projekte dabei, Software auf real existierender Hardware in real existierenden Rechenzentren in einen effizienten und zuverlässigen Betrieb zu nehmen, bringt die Infrastruktur von Rechenzentren auf den Stand der Technik und führt vor allem die IT-Bausünden der New Economy in die betriebswirtschaftliche Realität.

Wenn er nicht gerade tauchen geht oder mit dem Fahrrad Kontinente sammelt, ist er unter stockebrand@guug.de, me@benedikt-stockebrand.de und <http://www.benedikt-stockebrand.de/> zu erreichen.