

IT-Profil

Name: Winfried Huber
Adresse: Illerweg 6, 82140 Olching
Telefon: +49 (0) 8142 13038
Fax: +49 (0) 8142 18612
Mobil: +49 (0) 172 9449946
WWW: Winfried.Huber@huber-und-boehm.de
<http://www.huber-und-boehm.de/>

Jahrgang: 1957
EDV-Erfahrung seit: 1984
Fremdsprachen: Englisch
Fachlicher Schwerpunkt: Systemprogrammierung Unix/Windows
Datenbanken, Datensicherung,
Versionierung, Koppelung Unix/Host/Vista/XP/NT/W2K
Integration, Test

Einsatzort: Großraum München
Hardware: Sparc, Intel, HP, Alpha, VAX, PDP
Betriebssysteme: Solaris/Unix/Linux/HPUX/AIX, Windows, VMS, RTE
Programmiersprachen: C/C++, div. Assembler, FORTRAN
Scriptsprachen: Perl, shell, sed, awk etc.
Datenbanken: Oracle 7/8/9/10, SQL, ProC, OCI, BLOBs, Image/1000
Datensicherung: TSM
Datenkommunikation: TCP/IP, VoIP, SNA, LU6.2, Winsock
Stand des Profils: April 2010
Branchen: Banken, Industrie, Energieversorgung, Wehrtechnik



Besondere Erfahrungen im Bereich:

- Hochverfügbarkeit, Reduktion der down time
- Synchronisation von QS-Umgebungen mit der Produktion
- Koppelung von Rechnern per ssh-Tunnel

- Anwendungs- und Systemprogrammierung
- Design, Entwicklung, Test, Einführung und Betrieb
- Rettung notleidender Softwareprojekte

- Automatische Fernüberwachung von und automatische Softwareverteilung auf großen, heterogenen Netzen (mehrere Hosts, 650 Unix/NT/W2K-Server, 45.000 Clients)
- Erfassung, Aufbereitung und Auswertung von SNA-Ausfalldaten von 8.500 SB-Geräten mit dem Ziel, die SNA-Anbindung stabiler zu machen

- Simulatoren zur Ausbildung von Schiffsbesatzungen (Radarnavigation, Bekämpfung feindlicher Fahrzeuge)
- Interprozesskommunikation, ssh-Tunnel, Firewalls
- Simulation komplexer technischer Vorgänge in Kernkraftwerken anhand von Ursachen-Folgendiagrammen

Einige Projekte:

- Projekt:** 01.2010 – 03.2010 Wehrtechnik
Weiterentwicklung Fleetwork Trainer
- Situation:** Ein von mir entwickelter Simulator sollte weiter entwickelt werden
- Ziel:** Der Simulator sollte auf Linux unter aktueller Hardware laufen und um Sprechfunk (VoIP) und Flaggensignale erweitert werden
- Lösung:** Migration des Projekts auf Linux, Einteilung der Schiffe in zwei Gruppen (rot und blau) mit der Möglichkeit, denen unterschiedliche Flaggensignale nach dem internationalen Signalhandbuch und den NATO-Erweiterungen zu zeigen. Die entsprechenden Mannschaften bestätigen dem Ausbilder per Mausclick, dass sie ihre Flaggensignale verstanden haben. Alle Aktionen werden aufgezeichnet, so dass sie in einem späteren Debriefing an einem Beamer besprochen werden können.
Über Headsets können die rote und die blaue Gruppe getrennt untereinander sprechen („Seefunk“), der Ausbilder kann sich auf die rote oder die blaue Gruppe aufschalten.
- Umgebung:** Linux, C, X11R7, VoIP, mumble
-
- Projekt:** 05.2009 – 12.2009 Bank
Automatisierte Infrastrukturüberwachung
- Situation:** Neue, scheidende oder den Aufgabenbereich wechselnde Mitarbeiter machen Änderungen an mehreren Stellen erforderlich (Eintrag ins LDAP, in net groups, Aufnahme von ssh keys etc.). Dabei kommt es immer wieder zu Inkonsistenzen.
- Ziel:** Etwaige Inkonsistenzen sind zu vermeiden oder wenigstens zu melden
- Lösung:** Automatische Überwachungen stellen sicher, dass die Einträge konsistent sind (z.B. Übereinstimmung der zur Anmeldung an den Maschinen erforderlichen Einträge in net groups). Falls nicht, so erfolgt eine Meldung direkt am Terminal, per Mail und/oder ein Eintrag im Quality Center.
- Umgebung:** Solaris, LDAP, perl, shell
-
- Projekt:** 02.2009 – 04.2009 Bank
Automatisierte, zentrale Erfassung von Testergebnissen
- Situation:** Beim Software Configuration Management fallen dezentrale Testergebnisse an (durch Installationstests und regelmäßige, automatische Tests)
- Ziel:** Diese Testergebnisse sollen zentral erfasst werden
- Lösung:** Durch eine Schnittstelle werden diese Testergebnisse korrekt den einzelnen Produkten und Servern zugeordnet und zentral im Quality Center abgelegt
- Umgebung:** Solaris, perl, shell

Projekt: 02.2009 – 04.2009 Bank
Automatisierte Unterstützung zur Behebung von Fehlkonfigurationen

Situation: Zum automatisierten Paketbau aus einem Versionierungssystem heraus ist es nötig, Eigentümer und Gruppen von Dateien im gerade neu zu erstellenden Paket richtig zu setzen. Das kann bei fehlerhafter Konfiguration des Buildservers oder falschen Angaben im Paket scheitern.

Ziel: Der zugrunde liegende Fehler und Lösungsweg soll ermittelt und an die Verantwortlichen und die Leidtragenden gemeldet werden.

Lösung: Ein gescheiterter Paketbau wird auf seine Fehlerursache untersucht, die zur Behebung nötige Information aus den entsprechenden Quellen beschafft (z.B. LDAP) und per Mail an die betroffenen Personen bzw. Postfächer gemeldet. So erspart sich der Entwickler die Analyse des Fehlers und das Ermitteln, wer den Fehler beseitigen kann. Der Systemadministrator muss sich nicht erst die zur Fehlerbehebung nötigen Informationen manuell (und fehlerträchtig) zusammen suchen. Alle Beteiligten sind auf den benötigten Kenntnisstand und wissen, dass das auch für die anderen Beteiligten gilt.

Umgebung: Solaris, LDAP, perl, shell

Projekt: 09.2008 – 11.2008 Bank
Automatisierte Erstellung von Solaris-Patches

Situation: Der Kunde betreibt ein Versionierungssystem. Aus diesem heraus werden automatisiert Solaris-Pakete erstellt.

Ziel: Der Upgrade eines Solaris-Paketes auf eine neuere Version bedingt eine nicht immer hinnehmbare down time. Um diese zu verringern oder ganz auszuschließen sollten automatisch Solaris-Patches generiert werden.

Lösung: Der Paketbaumechanismus holt sich die aktuellen Quelltexte aus dem Versionierungssystem und erstellt daraus die neue Version des Paket. Diese wird dann wegen der Reproduzierbarkeit wieder in das Versionierungssystem eingestellt.
Diesen Mechanismus habe ich dahingehend erweitert, dass er sich auch die letzte freigegebene Vorgängerversion des Paket holt und die beiden gegeneinander vergleicht. Daraus wird dann automatisch ein Patch erstellt, mit dem die Vorgängerversion des Paket auf die aktuelle Version upgegradet werden kann. Nur die geänderten Dateien werden in den Patch eingepackt, die gleich gebliebenen Dateien nicht. Dabei wird zwischen relevanten und nicht relevanten Änderungen unterschieden, weil das Versionierungssystem beim Auschecken der Dateien Kommentare anpasst, die aber vollkommen irrelevant sind. Zwei shell scripts können als identisch angesehen werden, wenn sie sich nur durch in Kommentar gesetzte, automatisch generierte, Header unterscheiden.
Man erhält also als Ergebnis nicht nur ein neues Paket sondern zusätzlich einen Patch, der das Delta zwischen dem aktuellen Paket und der Vorgängerversion abbildet. Der Entwickler muss dabei nicht angeben, was sich geändert hat – das wird automatisch ermittelt.

Umgebung: Solaris, perl, pkgadd/pkgmk-Family

- Projekt:** 08.2008 – 09.2008 Bank
Automatisiertes Aufsetzen von Solaris-Zonen
- Situation:** viele einzelne Server im Entwicklungsumfeld – hohe Kosten
Das manuelle Aufsetzen von virtuellen Maschinen unter Solaris („Zonen“) ist aufwändig und fehlerträchtig. Regelmäßig passieren Fehler, die erst später auffallen und mit hohem Aufwand manuell analysiert und behoben werden müssen.
- Ziel:** Konsolidierung auf wenige Server, die virtuelle Server (Solaris-Zonen) bereit stellen. Effizientes automatisiertes Aufsetzen dieser Zonen
- Lösung:** Erstellung eines Solaris-Paketes. mit dem in definierter und reproduzierbarer Weise Solaris-Zonen gelöscht und neu aufgesetzt werden können. Man bekommt dadurch gewissermassen eine „jungfräuliche“ Maschine mit allen Basiskomponenten per Knopfdruck bereit gestellt und hat so für Tests und Entwicklung eine definierte Ausgangsumgebung.
- Umgebung:** Solaris, shell, perl
-
- Projekt:** 06.2008 – 07.2008 Bank
Übersichtliche Darstellung des Softwarestandes
- Situation:** Es war schwierig, bei den vielen Entwicklungs-, IT-, QSU- und Produktionsservern die Übersicht über den Softwarestand zu behalten
- Ziel:** Übersichtliche Darstellung des IST-Zustandes mit Hervorhebung etwaiger Abweichungen
- Lösung:** Erstellung eines Datenaggregationstools, welches die benötigten Informationen von den einzelnen Server per ssh-Tunnel an eine zentrale Datensammelstelle zur Auswertung schickt. Dort werden die Daten übersichtlich aufbreitet und als HTML-Seiten bereit gestellt. Man sieht mit einem Blick, welche Packages wo installiert sind, ob sie korrekt installiert sind und ob es die richtigen Versionen sind. Abweichungen werden farbig dargestellt, damit man sie mit einem Blick bemerkt.
- Umgebung:** Solaris, perl, HTML
-
- Projekt:** 06.2008 – 07.2008 Bank
trouble shooting
- Situation:** abnorm hoher Ressourcenverbrauch führt zu Abstürzen
Die von einem anderen Kunden gewünschte Änderung einer Fremdanwendung hatte schlimme Auswirkungen auf den Betrieb beim Kunden: Extrem hoher Speicherverbrauch von Anwendungsprozessen führte zunächst zum Swappen gerade nicht aktiver Prozesse, im weiteren Verlauf zu drastischen Performance-Einbußen durch thrashing und später zum Crash der Maschine
- Ziel:** Diagnose, Sofortmaßnahmen zur Sicherstellung des Betriebs
- Lösung:** Erstellung eines Tools, welches die Speichersituation der Maschine überwacht und bei sich abzeichnender Knappheit angemessene Maßnahmen ergreift: Die Client-Prozesse der Anwender wurden nach ihrer Inaktivität sortiert und die am längsten inaktiven Prozesse werden getötet. Weil diese bei Bedarf transparent nachgestartet werden hat das abgesehen von einer minimalen Verzögerung keine nachteiligen Auswirkungen für die Anwender. Als Kriterium für die Inaktivität wurden die Zeitstempel der file descriptor verwendet, so dass jeglicher I/O, Datenbank, Anwendung, Netzwerk etc. betrachtet wird. So kann vermieden werden, dass möglicherweise zeitaufwändige Aktionen abgebrochen werden.
- Umgebung:** Solaris, PVCS Dimensions, perl

Projekt: 02.2008 – 05.2008 Bank
trouble shooting

Situation: dead locks legen Fremdhersteller-Anwendung lahm

Ziel: Diagnose, Abhilfe

Lösung: Erstellung eines Tools, welches die dead locks erkennt und Informationen aus der Anwendung und mehrerer Datenbanken verknüpft, um eine Diagnosemöglichkeit zu bekommen (welcher Anwender hat die Blockade ausgelöst? Was hat er versucht, zu tun?) und konkrete Maßnahmen zu ermitteln, um das Problem für den Moment zu lösen (einen Client-Prozess abbrechen ist besser als warten, bis überhaupt nichts mehr geht).

Umgebung: Solaris, Oracle, PVCS Dimensions, SQL, perl

Projekt: 03.2007 – 01.2008 Bank
Migration von (PVCS Dimensions 9, Solaris 9, Oracle 9) nach (PVCS Dimensions 10, Solaris 10, Oracle 10), Einführung von Sun Cluster, Resource Groups, Zonen

Situation: PVCS Dimensions Version 9 lief auf einer Sun Fire 440, die dazugehörige Oracle-9 Datenbank auf einer anderen Sun Fire 440, beide Maschinen unter Solaris 9

Ziel: Migration auf PVCS Dimensions 10, Solaris 10, Oracle 10 und Sicherstellung ununterbrochener Verfügbarkeit durch Einführung eines Sun Clusters

Lösung:

- Sicherstellung der Robustheit des Paketbau-Mechanismus der Solaris-Pakete direkt aus PVCS Dimensions heraus
- Installation des gesamten Systems einschließlich der Anwendung und Datenbank durch Solaris-Pakete
- Automatisierte, parallelisierte Migration der Daten von PVCS Dimensions 9 nach PVCS Dimensions 10
- Sicherstellung hoher Verfügbarkeit durch Schwenkbarkeit der einzelnen Komponenten innerhalb des clusters: Die Datenbank in einer resource group und der Anwendung in einer Zone (=virtuelle Maschine). Das bringt enorme Vorteile:
 - bei geringer Last werden Anwendung und Datenbank auf einem cluster node gefahren. Der Verzicht auf die TCP/IP-Verbindung zwischen DB und Anwendung bringt eine hervorragende Performance
 - bei hoher Last werden Anwendung und Datenbank auf verschiedenen cluster nodes gefahren: Sowohl Datenbank als auch Anwendung haben den gesamten Arbeitsspeicher und alle CPUs zur Verfügung, und die Performance-Einbuße der internen TCP/IP-Verbindung hält sich durch die Verwendung privater Interfaces in Grenzen
 - Sind Wartungsarbeiten an einem cluster node notwendig, so kann durch das Evakuieren dieses cluster nodes die Verfügbarkeit aufrecht erhalten und dennoch ohne Druck die Wartung des abgeschalteten nodes durchgeführt werden
 - Sollten ernsthafte Probleme an einem cluster node auftreten, so wird dieser zunächst evakuiert, die Verfügbarkeit wird bei möglicherweise reduzierter Performance durch den anderen cluster node sichergestellt. Dann wird ein weiterer cluster node hinzu konfiguriert, die Last kann nun wieder verteilt werden. Dann kann ohne Druck die Behebung der Probleme des kranken cluster nodes angegangen werden.
 - Durch Aufnahme weiterer nodes in den Cluster kann die Performance weiter gesteigert werden

- Bereitstellung einer regelmäßig aktualisierten QS-Umgebung: Die QS-Datenbank wird aus der produktiven Datenbank geclont, die item library (deren Konsistenz zur Datenbank absolut notwendig ist) wird durch eine Synchronisation mit der TSM-Sicherung erreicht. Dieses Vorgehen belastet nicht die produktive Umgebung, und durch die Synchronisation der bereits vorliegenden, allerdings veralteten, item library mit dem im TSM festgehaltenen Referenz-Stand hält sich der Zeitaufwand in Grenzen – ein vollständiger Restore der item library aus dem TSM würde Tage dauern. So werden nur die tatsächlich benötigten Dateien aus dem TSM geholt und die nicht mehr gültigen Dateien aus der QS-Kopie der item library gelöscht.

Umgebung: PVCS Dimensions, Solaris, Oracle, TSM, perl, bash

Projekt: 09.2004 - 12.2005 Bank

Konzeption, Entwicklung und Durchführung der Konsolidierung aufbewahrungspflichtiger Transaktionsdaten ins TSM-Langzeitarchiv

Situation: Über viele Jahre hinweg waren Transaktionsdaten auf vielen Servern angefallen und vor Ort auf unterschiedlichen Datenträgertypen archiviert worden. Diese sollten in einem zentralen TSM-Archiv konsolidiert werden.

Die Daten waren dabei auf Vollständigkeit, Konsistenz und Redundanz zu überprüfen und in eine geordnete Verzeichnisstruktur zu überführen.

Lösung: Die Datenträger wurden vor Ort eingelesen und auf einem Filer bereit gestellt. Per perl script wurden die dann auf Vollständigkeit, Konsistenz und Redundanz überprüft und nach Absprache bereinigt. Gegebenenfalls wurde der Vorgang nach dem Einspielen weiterer Datenträger wiederholt.

Dann wurden die bereinigten Altdaten ins TSM übernommen. So konnte darauf verzichtet werden noch viele Jahre Hardware vorhalten zu müssen, mit denen die alten Bänder eingelesen werden können. Außerdem wurde so klar, ob die Altdaten auch wirklich vollständig sind – eine Suche nach verschollenen Datenträgern wird um so aussichtsloser, je mehr Jahre seitdem vergangen sind.

Umgebung: W2K, perl, TSM command line tools

Projekt: 07.2004 - 10.2004 Bank

Überprüfung von DB-Software bei neuen Anforderungen

Situation: Auf vielen Servern lief jeweils Software, die Daten in lokale Oracle-Datenbanken geschrieben hat. Diese Server waren auf die Standorte verteilt. Im Zuge einer Rezentralisierung sollten die Rechner in einem Rechenzentrum zusammen gefasst werden und gegen eine gemeinsame Datenbank laufen.

Lösung: Die genaue Analyse der bestehenden Software ergab Schwachstellen, die bei der Ausgangskonfiguration kein besonderes Risiko dargestellt haben, wohl aber nach der Rezentralisierung. Durch rechtzeitiges Redesign konnten diese Probleme ausgeschlossen werden.

Software: C, Oracle 9, Java, ProC, NT4.0, W2K, XP

- Projekt:** 04.2003 - 09.2003 Bank
Design und Entwicklung eines Passwort-Verteildienstes
- Situation:** Hart codierte oder bei der Installation fest gelegte Zugangskennungen für Datenbank-Accounts und Netzwerkressourcen sollen ersetzt werden durch im laufenden Betrieb zu ändernde Zugangskennungen. Berechtigte Anwendungen sollen diese veränderten Zugangskennungen automatisch erhalten, manipulierte Anwendungen sollen erkannt und abgewiesen werden. Ebenso sollen sicherheitskritische DLLs der Anwendungen gegen Manipulation geschützt werden können.
- Lösung:** Es wurde ein Dienst entwickelt, der berechtigten (d.h. frei geschalteten) Anwendungen die benötigten Zugangskennungen in verschlüsselter Form übermittelt. So können die Zugangskennungen regelmäßig verändert werden, manuell oder automatisch. Sie verteilen sich dann selbständig im Netz an die berechtigten Anwendungen (und nur an die!).
Dabei wird auch die Integrität der Anwendungen überwacht, um Manipulationen an sicherheitskritischen Anwendungsteilen auszuschließen.
- Software:** C/C++, MFC, PGP/gpg, NT4.0, W2K, XP

- Projekt:** 05.2003 Bank
Langzeitarchivierung
- Situation:** Der Kunde betreibt ein hochverfügbares Zahlungsverkehrs-System auf einem unter AIX laufenden HACMP-Cluster. Die von Mitarbeitern und Firmenkunden angestoßenen Vorgänge erzeugen Dateien, die im Zuge der weiteren Verarbeitung durch Pools (Verzeichnisse) wandern. Um die Nachvollziehbarkeit der Vorgänge sicherstellen zu können, sind diese Daten, nach Mandanten getrennt, der Langzeitarchivierung zu unterwerfen.
- Lösung:** Die Bewegungsdaten werden erfasst und mandantenspezifisch getrennt. Bei manchen dieser Daten ist der Mandant leicht zu erkennen, sonst aus der Datenbank zu holen. Aus Logs werden mandantenspezifische Auszüge erstellt. So entsteht für jeden Tag und für jeden Mandanten ein Archiv.
Die Originalversionen der Logs landen ebenso wie Logs, die keinem Mandanten zugeordnet werden können, in einem „Lumpensammler“-Archiv.
Diese Tagesarchive werden dann je nach der Archivierungsstrategie des einzelnen Mandanten über TSM archiviert. So können unterschiedliche Aufbewahrungsfristen eingehalten werden, ohne unnötige Kosten zu verursachen.
- Software:** perl, shell, TSM

- Projekt:** 09.2003 Bank
Optimierung der Backup-Strategie
- Situation:** Der Kunde betreibt ein hochverfügbares Zahlungsverkehrs-System auf einem unter AIX laufenden HACMP-Cluster. Jede Nacht wird zur betriebsschwächsten Zeit am frühen Morgen ein TSM-Backup angestoßen. Das System bleibt dabei aber verfügbar, die Anwendung wird nicht heruntergefahren. Die bestehende Sicherungsstrategie war auf Schwachstellen und Kostenoptimierung hin zu untersuchen.
- Lösung:** Die Analyse ergab eine Reihe von Schwachpunkten, die unnötige Kosten und unerwünschte Gefährdungspotentiale zur Folge hatten, z.B.:
- Das ORACLE-Online-Backup wurde erst nach dem file system backup durchgeführt, so dass jeweils die am Vortag erzeugten Datenbank-Dateien gesichert wurden mit der Folge, dass bei einem Wiederaufsetzen nach einem crash unnötigerweise die redo logs des gesamten Vortages eingespielt werden mußten, um die Datenbank auf den Zeitpunkt des nächtlichen Backups zu bringen. Dazu kommen dann noch die redo logs bis zum Zeitpunkt des Crashes.
 - Die via management class im TSM festgelegten Aufbewahrungsstrategien der Online-Backups und des Datenpools waren nicht aufeinander abgestimmt. So entstehen unnötige Kosten, denen kein Nutzen gegenüber steht.
 - Zusätzlich zu den Online-Backup-Dateien und den redo logs der ORACLE-Datenbank wurden auch noch die „lebendigen“ ORACLE-Datenbankdateien über das reguläre file system backup mit gesichert. Das bringt nur erhebliche Kosten, denen kein Nutzen gegenüber steht. Im Gegenteil, dieses Vorgehen ist auch noch massiv gefährlich: Spielt man während des Vorrollens der Datenbank mittels redo-logs das file system backup ein, so zerstört man die gerade eben mühsam wieder hergestellte Datenbank!
 - Um einen schnellen Wiederanlauf nach einer Störung sicherstellen zu können wurden mehrere Generationen der durch das Online-Backup der Datenbank erzeugten Tablespace-Dateien neben der TSM-Sicherung auch noch auf dem Plattenspeicher vorgehalten. Wegen der rasch ansteigenden Datenvolumina war die Notwendigkeit neue Platten anzuschaffen absehbar. Durch Kompression der Online-Backups nach der TSM-Sicherung konnte diese Neuanschaffung von Platten vermieden werden. Der bei Tablespace-Dateien erreichbare Kompressionsfaktor ist sehr hoch. Zwar muss immer noch genügend Plattenplatz für eine unkomprimierte Sicherungsversion vorgehalten werden, aber halt eben nur für eine Version, nicht für mehrere.
- Diese Schwachstellen konnten vollständig beseitigt werden.
- Software:** perl, shell, TSM

Projekt: 04.1999 bis 10.2002 Bank
Remote Service Agent

Situation: Der Kunde, eine Dachgesellschaft, erstellt Software für etwa 100 rechtlich selbständige Organisationen. Die Installation der Software und der Betrieb der Server wird von den autonomen Kunden selbständig durchgeführt. Die Dachorganisation kann aber nach Freischaltung durch den lokalen Administrator über ein internes Netz auf die Kundenserver zu Wartungs- und Diagnosezwecken zugreifen (per telnet und ftp).

Diese umständliche Prozedur (Freischaltung des Serverzugriffs erbitten durch Anruf beim lokalen Administrator) ist bei Einzelproblemen tragbar. Ist es aber nötig, auf viele oder gar alle Server zuzugreifen, so ist dieser Weg nicht tragbar.

Lösung: Entwicklung eines automatischen, bedienerlosen, hostgestützten Fernwartungssystems für ca. 650 Anwendungsserver (Unix, NT und W2K). Die Jobs werden dabei auf einem bestimmten Server („master server“) bei der Dachorganisation erstellt:

- Alle Dateien (Programme etc.) bereitstellen
- AutoAction Script schreiben: Das übernimmt die eigentliche Durchführung der Aktion auf dem Kundenserver. Werden mehrere AutoAction Scripts erstellt, so wird automatisch das Richtige ausgewählt.
- Alles in ein komprimiertes Archiv zusammenpacken
- Das Archiv muss nun mit einer elektronischen Signatur versehen werden

Der Job muss nun auf die Kundensysteme verteilt werden. Dazu wird er vom Master Server in eine Host-Datenbank hoch geladen. Dabei wird auch fest gelegt, für wen dieser Job bestimmt ist und wann er ausgeführt werden soll.

Die Kundensysteme schauen regelmäßig in der Hostdatenbank nach, ob ein Job für sie bereit gestellt wurde, laden den ggf. herunter und führen ihn aus. Rückmeldungen und Protokolle werden auf den Master Server übertragen, so kann der Erfolg der Aktion überprüft werden. Man arbeitet „unter voller Sicht“: Man sieht sofort, ob die Aktion auf allen Servern problemlos verlaufen ist. Nur die ganz wenigen Problem-Server werden dann per Hand überprüft, nur da wird die Freischaltung durch den lokalen Administrator benötigt.

Außer dem AutoAction Script kann noch ein FailureRevert Script in das Archiv eingepackt werden. Damit kann man etwa auf halbem Weg stecken gebliebene Änderungen wieder zurückdrehen, damit der Server wenigstens im vorherigen Zustand weiter läuft statt womöglich gar nicht.

Mein Beitrag zu diesem System war das komplette Design, die Entwicklung der auf den Servern laufenden Teile und die Koordination mit den Host-Entwicklern. Die Stärken des Systems treten vor allem bei Stichtagsumstellungen, bei Notfall-Wartungsmaßnahmen und bei kurzfristig notwendigen Überprüfungen hervor. Etwa, wenn sicherzustellen ist, dass auch wirklich alle 45.000 Clients mit einer bestimmten Softwareversion versorgt wurden, bevor man sich trauen kann, ein neues Hostprogramm frei zu schalten.

Sehr nützlich ist das System auch, wenn ein neuer Server installiert wird: Nach der Installation der Software werden automatisch die neuesten Patches aus der Hostdatenbank herunter geladen und installiert.

Software: C, perl, SNA

- Projekt:** 02.1998 bis 07.2002 Bank
Service Control Monitor (Überwachung von Servern)
- Problem:** Sicherstellung des reibungslosen Betriebs von Servern
- Lösung:** Design, Entwicklung und Betrieb eines Überwachungssystems, welches die auf einem Server installierten Softwarekomponenten im laufenden Betrieb überwacht und bei Abweichungen vom Sollzustand selbständig geeignete Maßnahmen ergreift, zumindest aber Alarm schlägt.
Es ist eine differenzierte zeitliche Steuerung implementiert. So sind manche Komponenten nur tagsüber zu überwachen, und manche Fehlerzustände werden erst dann kritisch, wenn sie über einen längeren Zeitraum anhalten. Zu diesem Zweck kann sich der Service control monitor dynamisch rekonfigurieren, um sich der momentanen Störungslage anzupassen.
Besonders bei sporadisch auftretenden Betriebsstörungen bieten die log files des Service control monitors eine erstklassige Diagnosemöglichkeit.
Beim Design des Service control monitors war auf besondere Effizienz zu achten. Häufig vorkommende Prüfungen sind daher integriert und laufen sehr effizient. Exotische Prüfungen können entweder als shell script Fragmente eingetragen oder gleich als externe Programme bereitgestellt werden, die der Service Control Monitor dann aufruft.
- Software:** C, Unix, SINIX, NT, W2K, perl
- Projekt:** 10.1997 (bis Ende 2005 Pflege und Erweiterungen nach Bedarf)
LogFile()-Library
- Ziel:** Effiziente und einfache Protokollierung
- Lösung:** Design und Entwicklung einer Library, die eine einfaches und effizientes Schreiben von log files ermöglicht: Die Anwendung braucht sich um das Handling der log files nicht zu kümmern, das macht die Library. Die log files werden der Größe nach begrenzt. Wird die vorgegebene Maximalgröße erreicht, so wird das log file gealtert und erhält einen Zeitstempel hinten an den Namen angefügt. Früher gealterte Log Files werden der Gleitlöschung unterworfen, damit die Platte nicht über läuft. Es wird auf den freien Plattenplatz geachtet: Wenn der zu gering wird, dann wird die Aufbewahrungsfrist der log files verkürzt, um wieder freien Plattenplatz zu schaffen. Obwohl mehrere Programme gleichzeitig in in ein- und derselben Datei protokollieren ist sichergestellt, dass alle Programme die Alterung eines log files automatisch erkennen, damit das Protokoll konsistent bleibt.
- Software:** C
- Projekt:** 03.1996 bis 12.2001 Bank
Softwareverteilung, automatische Installation
- Ziel:** Reibungslosen Software-Installation trotz heterogener Umgebung
- Lösung:** Aufbau und Betrieb einer Integrationstestumgebung, um Softwarepakete auf ihre Verträglichkeit in Bezug auf Installation und Betrieb zu testen. Entwicklung einer Installationsoberfläche, die die für die Installation benötigten Daten nach Möglichkeit selbst herausfindet und sonst erfragt. Diese Angaben werden dann einer rigorosen Verifikation unterzogen (z.B. Überprüfung der Kommunikation der ORACLE-Datenbanken untereinander für die Replikation), Prüfung der Stimmigkeit der Auswahl der Packages, Verträglichkeit der Versionen, ob die aktuellen Datenbank-Tabellen installiert sind und vieles mehr, bevor die eigentliche Installation erfolgt. Besonderer Wert wurde dabei gelegt auf die Reduzierung der down time und die Nachvollziehbarkeit.
Die UNIX-Packages waren teils vollständig von mir, teils habe ich nur die Installation dazu geschrieben, und manche Packages wurden komplett mitsamt der Installation angeliefert.

Software: Unix, Sinix, NT, W2K, C, perl, shell, pkgadd-Familie

Projekt: 03.2001 bis 10.2002 Bank
Erfassung von SNA-Ausfalldaten von SB-Geräten

Situation: Der Kunde betreibt ein komplexes Netz mit ca. 8.500 SB-Geräten. Diese schreiben log files, aus denen man den Zustand der SNA-Verbindung erkennen kann.

Lösung: Diese Daten auf den Kundensystemen erfassen und zentral verfügbar zu machen, um Auswertungen zu ermöglichen:

- Erkennen von Schachstellen im Netz (z.B. überlastete Router)
- Analyse von Wiederanlaufproblemen nach Netzzusammenbrüchen oder Wartungsfenstern
- Erkennen von Lastproblemen (zu lange Antwortzeiten/SNA-Timeouts)
- Erkennen problematischer Server
- Erkennen problematischer Konfigurationen oder Gerätekombinationen

Die Auswertungen können dazu in einer Vielfalt von Formen dargestellt werden, tabellarisch und grafisch.

Software: C, perl

Projekt: 10.1988 bis 05.1990 (in mehreren Teilstufen) Wehrtechnik

Fleetwork Trainer

Lösung: Ziel war die Erstellung eines Simulators zur Ausbildung von Schiffsbesatzungen in Bezug auf die Radarnavigation. Jede Besatzung sieht das eigene Radarbild und kann es durch Änderung der Radarreichweite etc. beeinflussen. Jede Besatzung steuert das eigene Schiff. Alle zusammen sollen in einer vorgegebenen Zeit ein Flottenmanöver durchführen, etwa bestimmte Formationswechsel. Der Ausbilder kann sich von seinem Arbeitsplatz aus die Bildschirme der einzelnen Schiffsbesatzungen anzeigen lassen und über eine Sprechverbindung mit den Besatzungen kommunizieren. Er kann die Übung jederzeit unterbrechen und beeinflussen, etwa den Standort eines Schiffes ändern. Ebenso kann der Ausbilder „verwaiste“ Schiffe fahren, damit mehr Schiffe an Übungen teilnehmen können als tatsächliche Crews vorhanden sind. Die gesamte Übung einschließlich aller Aktionen der einzelnen Besatzungen wird protokolliert und kann gemeinsam in einem Übungsraum über einen Beamer nachvollzogen werden, damit Fehler der Besatzungen in der Gruppe besprochen werden können. Bestandteil des Systems ist auch ein map editor, mit dem pseudorealistische Küstenlinien erstellt werden können. Genau modelliert dort, wo die exakte Küstenlinie benötigt wird, sonst anhand weniger Stützpunkte automatisch generiert.

Software: C, X10, X11R3 bis X11R5

Projekt: 01.1994 bis 11.1994 Wehrtechnik
actual speed tactical trainer

Situation: Ausgangslage war ein Not leidendes Softwareprojekt:

Der Kunde hatte einem inzwischen in Konkurs gegangenen ausländischen Softwarehaus einen Simulator für Kriegsschiffe entwickeln lassen, in dem der Einsatz der Waffensysteme trainiert werden sollte. Eine zentrale Komponente des Systems („postal service“), zuständig für den Nachrichtenaustausch der einzelnen Komponenten des Simulators, war dem Normalbetrieb zwar gewachsen, brach aber zusammen, sobald es im Vorfeld von Kampfhandlungen hektisch wurde und das Nachrichtenaufkommen daher steil anstieg (durch Betätigen von Schaltern, Tasten etc.). Das ursprüngliche Entwicklungsteam war nicht mehr greifbar, die Dokumentation rudimentär.

Lösung: Das erste Teilprojekt war eine rigorose statische Analyse des vorliegenden postal services, um die Schwachstellen aufzudecken: Designfehler, die Flaschenhälse bewirkten und so zu deadlocks führten. Ergebnis war eine Dokumentation mit den gewonnenen Erkenntnissen und einem Kapitel „Wege aus dem Sumpf“, in dem Lösungsvorschläge aufgezeigt wurden.

Zweiter Schritt war eine Neuentwicklung des postal services, der keine Schwachstellen mehr aufgewiesen hat und dadurch jedem Nachrichtenaufkommen gewachsen war. Erreicht wurde dies durch geschickten Einsatz von Interprozess-Kommunikation und dynamisch vergrößerbare Ringspeicher im shared memory mit Zugriffssteuerung durch Semaphore.

Software: C, Unix, IPC

Projekt: 10.1986 bis 06.1988 Energieversorgung
STEP-Graph

Situation: Um das Verhalten von Kernkraftwerken im laufenden Betrieb simulieren zu können, insbesondere bei Störungen, wird die Fortpflanzung von Störungen in der Anlage durch von Ingenieuren a priori erstellte Ursachen-Folgendigramme beschrieben. Um diese Diagramme maschinell weiterverarbeiten und zu einem zur Laufzeit des Reaktors ständig mit den aktuellen Daten zu fütternden Modell zu kommen werden diese Diagramme zunächst per Hand in einen Quelltext übersetzt, der einer speziell entwickelten Grammatik genügen muss. Das ist ein wegen der ungewohnten Grammatik zeitaufwändiger Prozess mit vielen Roundtrips (editieren, übersetzen, Fehler beheben, wieder übersetzen etc.).

Lösung: Zur Erleichterung dieser nervtötenden Arbeit wurde im ersten Schritt ein syntaxgesteuerter Editor geschrieben, der vor dem Start die Grammatik einliest und dann nur noch solche Texte als Eingabe akzeptiert, die der Grammatik genügen. In einem Menü werden die an der Stelle des Cursors zulässigen Terminalsymbole angegeben, über Syntaxfehler kann man den Cursor nicht hinweg bewegen. Wenn man den Cursor an das Ende des Textes bewegen kann, dann ist der Text bis dahin syntaktisch korrekt, wenn ein besonderes Terminalsymbol („Ende des Textes“) an dieser Stelle zulässig ist, dann ist der Text auch vollständig.

Damit war das Problem der Syntaxfehler gelöst, aber es war noch nicht sichergestellt, dass das Diagramm auch wirklich korrekt übertragen wurde, also nicht zum Beispiel ein Und-Gatter statt einem Oder-Gatter eingegeben wurde.

Dieses Problem wurde dadurch gelöst, dass in der Grammatik zusätzliche, besondere Regeln eingebaut wurden, durch die alleine durch das Eingeben des Textes auf einem Grafikbildschirm das entsprechende Diagramm gleichzeitig mit aufgebaut werden kann. Wenn das so entstandene Diagramm am Ende des Textes mit dem Quelldiagramm übereinstimmt, dann ist der Text korrekt eingegeben worden.

Diese automatisch erstellten Diagramme wurden dann auch ausgedruckt zur Rückdokumentation für den TÜV eingesetzt.

Software: C, FORTRAN, GKS, X11