Adrian Kühni
Talweg 29

8707 Uetikon am See

Schriftliche Arbeit im Rahmen der Ausbildung Projektleiter I SIZ / Projektassistent (IPMA Ebene D) bei KV Zürich Business School mit dem Titel

„Fortlaufende Qualitätskontrolle in einem IT[1]-Projekt".

Nummer und Volltext des Lernziels:

1.3.2a	Unter Berücksichtigung des Projektstands die geplanten Kosten-, Termin- und Qualitätsziele periodisch überprüfen und Ist/Soll-Abweichungen festhalten.

Datum der Erstellung: Dezember 04/Januar 05.

[1] IT: Information Technology (engl.) ➔ Informationstechnologie

Inhalt

Hiermit bestätige ich, dass diese Arbeit noch nie andersweitig eingesetzt oder veröffentlicht wurde.
Uetikon am See, 26.02.2010.

1 Prolog

1.1 Zusammenfassung der Arbeit

Der Qualitätssicherung in IT-Systemen kommt zentrale Bedeutung zu. Trotzdem sind sie oftmals ungenügend getestet. Warum? Testen ist enorm schwierig und aufwändig und erfolgt meist ohne klare Methodik/Systematik. Die risikogetriebene Testoptimierung bleibt unbeachtet und die fehlende Gewaltentrennung zwischen Ersteller und Tester der Systeme ist den Testaktivitäten und somit der Qualität abträglich.

Der Praxisfall zeigt, dass seine Testaktivitäten ihren theoretischen Grundlagen absolut genügen.
In zwei aufeinanderfolgenden Systemtestzyklen der Realisierungsphase konnten die Tester von damals 271 definierten Testfällen 37% resp. 84% durchführen. Davon waren 45% resp. 74% ***nicht*** erfolgreich. Fazit: Bei steigendem Systemumfang wegen massiven Realisierungsanstrengungen nimmt die Qualität dramatisch ab. Ein oft beobachteter Trend! Dies hat verschiedene Ursachen, die bewirken, dass entweder die Testfälle resp. die erwarteten Ergebnisse nicht mehr aktuell oder aber die Software tatsächlich fehlerhaft sind resp. ist. Vgl. dazu auch Kapitel 3.3.2.5 (Testbericht).
In vorliegendem Projekt führten mangelnde Anforderungsdefinitionen, unrealistische Realisierungsphasenpläne und eine praktisch vollständig fehlende Änderungsverwaltung (Change Management) zu den beunruhigenden Ergebnissen.
Testen lohnt sich. Nur dank systematischem Testvorgehen mit der geforderten Gewaltentrennung zwischen Softwareersteller und –tester kam der massive PM[2]-Handlungsbedarf bezüglich Qualitätssteigerung in allen Softwareerstellungsdisziplinen zu Tage.

1.2 Begründung für das Thema

Nebst der Budget- und Termineinhaltung ist die Zielerreichung in der verlangten Qualität der dritte Erfolgsfaktor für Projekte ([1] S. 232). Seit zwanzig Jahren beobachte ich im IT-Bereich immer wieder, dass Qualität ***der*** Erfolgsfaktor für IT-Projekte schlechthin ist. Dies wird ganz besonders verständlich, wenn wir die mannigfaltigen Einsatzgebiete (technischer) IT-Systeme in unserem Alltag betrachten. Hier seien stellvertretend für weitere der Verkehr in der Luft, zu Wasser, auf Schiene und Strasse, die Medizin, die (Nuklear-)Energiegewinnung und die Katastrophenfrühwarnsysteme erwähnt. Qualitative Mängel in solchen Systemen gefährden in hohem Masse Mensch, Tier und Umwelt.

Qualitätssicherung ist mein berufliches Betätigungsfeld. Mit dieser Arbeit will ich einen Theorie-/Praxisabgleich bezüglich fortlaufender Qualitätssicherung in einem konkreten, kommerziellen IT-Projekt herbeiführen.

1.3 Selbstreflexion

Das Verfassen der Arbeit hat Spass gemacht. Die Herausforderung war, das umfassende und komplexe Thema IT-Qualitätssicherung auf rund zehn thematisch relevanten Seiten in Theorie und Praxis darzustellen.

Die Aussagen verschiedener Quellen zur Qualitätssicherung sind ungeachtet der Projektart sinngemäss angewandt deckungsgleich.

Qualitätssicherung ist für mich die komplexeste und aufwändigste aber auch eine der faszinierendsten Disziplinen im IT-Wertschöpfungsprozess. Der Grad ihrer Wahrnehmung ist ein Mass für das Verantwortungsbewusstsein gegenüber dem Auftraggeber und sich selbst und – in entsprechenden Projekten – gegenüber Mensch, Tier und Umwelt schlechthin.

[2] PM: Project Management (engl.) ➔ Projektleitung als Aufgabe/Tätigkeit

2 Theoretische Grundlagen

2.1 Qualitätsbegriff

Qualität ist die Gesamtheit von Merkmalen eines Produkts resp. einer Dienstleistung bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen ([4] S. II-7). 100%-ige Qualität ist erreicht, wenn gilt: $Qualität_Q = Ergebnisse_E/Anforderungen_A = 1$, mit anderen Worten wenn die Ergebnisse mit den Anforderungen deckungsgleich sind.

2.2 Softwarequalitätsmerkmale

Wollen wir Qualitätssicherung in einem IT-Projekt gewährleisten, müssen wir uns zunächst fragen, welches die Qualitätsmerkmale von Software im Speziellen sind. Welches sind demnach die festgelegten und vorausgesetzten Erfordernisse, deren Erfüllung es zu überprüfen gilt? Es sind dies nach [2] S.10ff und [4] S. 23ff:

Merkmal	Hauptkomponenten
Funktionalität	Funktionelle Anforderungen, Angemessenheit und damit Brauchbarkeit
Benutzerkomfort	Einheitliche, einfache Bedienung, Erlernbarkeit, Hilfestellungen
Zuverlässigkeit	Robustheit, Fehlertoleranz, Sicherheit, Wiederherstellbarkeit
Effizienz	Antwortzeitverhalten unter verschiedenen Lasten
Wartbarkeit	Erweiter-, Installier- und Übertragbarkeit (Plattform), Parametriergrad
Testbarkeit	Simulation zeitlicher Abläufe, Steuerbarkeit von Systemzuständen

Tabelle 2.2.1: Softwarequalitätsmerkmale[3]

2.3 Testarten im IT-Wertschöpfungsprozess

Das gewählte Lernziel spricht von periodischer Überprüfung unter anderem der geplanten Qualitätsziele unter Berücksichtigung des jeweiligen Projekstands. Das trifft bei IT-Projekten ganz besonders zu, wie wir bei der weiteren Lektüre sehen werden. Je nach Projektstand in den PM-Phasen Planung (Konzeption) und Realisierung nehmen wir unterschiedliche Testaktivitäten vor, um die erwähnten Softwarequalitätsmerkmale auf ihre Erfüllung hin zu überprüfen. Vgl. dazu auch [1] S. 323, Abbildung 6.35.

Endanwender	**Abnahme**			**Fachl. Valid.**	**Abnahmetest**
Testteam				**Systemtest**	
Entw./Testteam			**Integrationstest**		
Entwickler		**Unittests**			
Testteam	**Reviews**				
Testling	Anforderungen	Komponenten	Anwendung Schnittstellen	Anwendung Schnittstellen	Anwendung Schnittstellen
Fokus	Verständlichkeit Vollständigkeit	Lauffähigkeit Funktionalität	Funktionalität Benutzerkomfort Testbarkeit	Funktionalität Benutzerkomfort Zuverlässigkeit Performance	Funktionalität Zuverlässigkeit Performance
Umgebung U		Entw.-U	Ziel-U-Testzelle	Ziel-U-Testzelle	Ziel-U-Produkt

Abbildung 2.3.1: Testarten im IT-Wertschöpfungsprozess

Fachl. Valid. steht für Fachliche Validierung und ist ein Abnahmetest in der Testumgebung der Zielplattform wie er [1] S. 246 fordert. Wir sprechen auch von einem Vorproduktionsbetrieb.

[3] Tabellen und Abbildungen ohne Quellennachweise stammen vom Autor.

2.4 RUP[4] und Testaktivitäten

Entwickler gestalten ihre IT-Projekte grundsätzlich auch in den vier während des Lehrgangs vermittelten Phasen Initialisierung, Planung (Konzeption), Realisierung und Abschluss. Weit verbreitet ist heute die Überlagerung der Phasen Konzeption und Realisierung mit den vier RUP-Phasen Inception (Eröffnung), Elaboration (Ausarbeitung), Construction (Konstruktion) und Transition (Übergabe):

RUP-Phasen:		**Inception**	**Elaboration**	**Construction**	**Transition**	
PM-Phasen:	Initialisierung	Konzeption		Realisierung		Abschluss

Abbildung 2.4.1: RUP-Phasen überlagern PM-Phasen Konzeption und Realisierung

Was ist das Spezielle daran? RUP teilt seine vier Phasen Inception, Elaboration, Construction und Transition je nach Projektklasse gem. der Multiprojektinitative (vgl. [1] S. 71) in 1 – n Iterationen (Iterations, Wiederholungen) auf, die sich im Sinne von umfassenden Arbeitspaketen aus dem Projektstrukturplan gem. [1] S. 178ff ableiten und die ***anhand von Risikobereichen*** gem. [1] S. 306 (technische Risiken) ***zu priorisieren*** sind. Ziel: das Projekt behandelt seine grössten Risiken am Anfang!

Jede dieser Iterationen durchläuft alle softwareentwicklungstechnischen Arbeitsschritte mit unterschiedlicher Gewichtung. Es sind dies:

- Requirements/Analysis ➔ Was haben wir zu bauen (Anforderungserhebung)?
- Design ➔ Wie bauen wir es (Architektur)?
- Implementation ➔ Wir bauen es (Konstruktion).
- Test ➔ Wir prüfen es (Qualtitätssicherung).

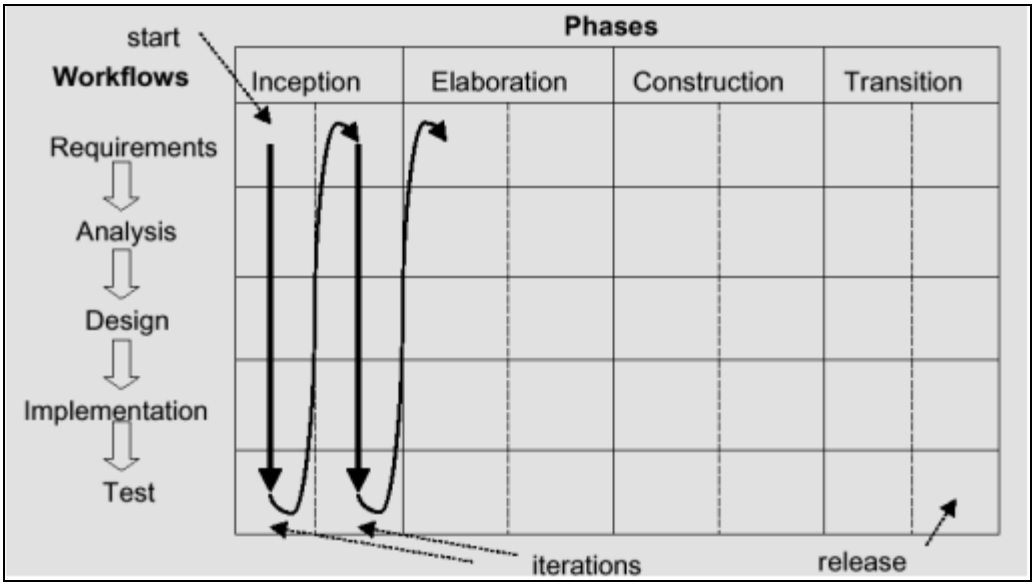


Abbildung 2.4.2: RUP-Phasen und –Iterationen[5]

Als letzte aufgeführte softwareentwicklungstechnische Aktivität steht Test. Was impliziert das? ***Jede RUP-Iteration endet mit einem Stück ausführ- und somit testbarer Software***, einem Release in der Fachsprache. In der Eröffnungsphase (Inception) ist es in der Regel ein technischer Prototyp zum Überprüfen der gewählten Architektur (Proof of Concept). Die letzte Iteration der Übergabephase (Transition) endet mit der Version 1.0 des Produkts, mit dem jungfräulichen System, das dem Endanwender erstmalig zur produktiven Nutzung übergeben wird. Und: ***das Ende einer jeden RUP-Iteration ist ein Projektmeilenstein!***

[4] RUP: Rational Unified Process®
[5] Quelle: International Business Machines Corporation IBM Software Group Rational Software

RUP eignet sich somit ausgezeichnet, um den Anforderungen des Lernziels nach periodischer Qualitätsüberprüfung anhand des jeweiligen Projektstands nachzukommen. Damit setzt Qualitätssicherung bereits ganz am Anfang des Projekts ein. Dank der risikogetriebenen Arbeitspaketpriorisierung erfolgt auch die Abarbeitung zugehöriger Testpakete risikogetrieben.

Abschliessend die Frage, wie viele Iterationen pro RUP-Phase vorzusehen sind. IBM Software Group Rational Software empfiehlt:

	Total	**pro RUP-Phase**			
Projektklasse[6]		Inception	Elaboration	Construction	Transition
Low	**3**	**0**	**1**	**1**	**1**
Typical	**6**	**1**	**2**	**2**	**1**
High	**9**	**1**	**3**	**3**	**2**
Very High	**10**	**2**	**3**	**3**	**2**

Tabelle. 2.4.1: Anzahl RUP-Iterationen pro Projektklasse[7]

2.5 Regressionstests

Tremp und Scheuring ([4] S. 52) schreiben: „Bei jeder Änderung des Systems ergibt sich das Problem bzw. das Risiko sogenannter Side-Effects (Aut.:[8] Seiteneffekte).“. Der IT-Fachbegriff meint das Phänomen, wenn an einer bestimmten Stelle eines Programms eine Änderung vorgenommen wird, die sich an anderen Stellen negativ auswirken. Dies hat einen massiven Einfluss auf die fortlaufende Qualitätssicherung in IT-Projekten, namentlich bei iterativer Entwicklung (➔ RUP). ***Es genügt wegen den Seiteneffekten nicht, am Ende einer Iteration nur den Produktefortschritt dieser Iteration zu testen. Vielmehr ist das Gesamtprodukt erneut umfassend zu prüfen***, und zwar ab einschliesslich Anforderungsreviews bis einschliesslich Fachliche Validierung gem. Kapitel 2.3 (Testarten im Wertschöpfungsprozess). Die Fachwelt spricht von Regressionstests. Fazit: Der Umfang der zu testenden Software und somit der Testaufwand nehmen mit fortschreitendem Projektverlauf massiv zu:[9]

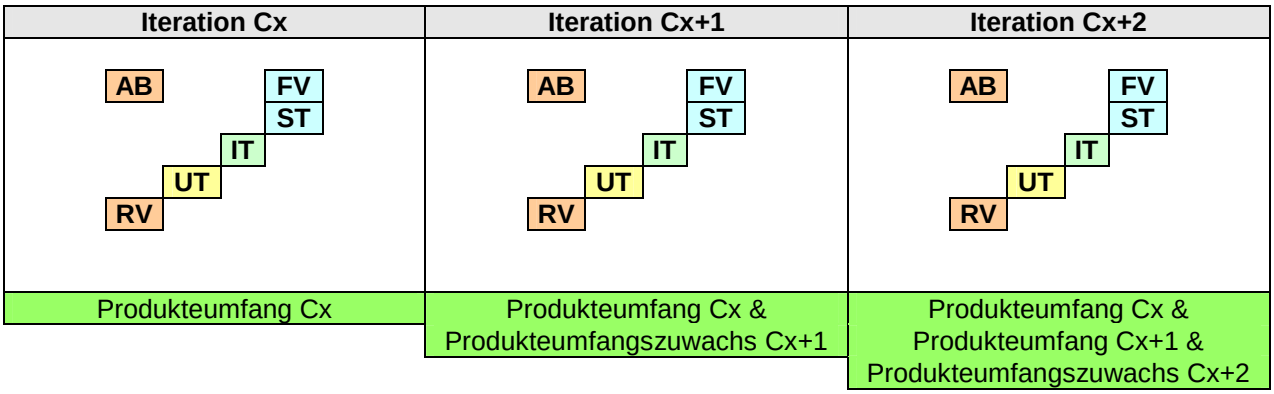


Abbildung 2.5.1: Produkte- & Testumfangszuwachs in Folgeiterationen

Der Testaufwand bestimmt sich folglich nach dem Gesamtvolumen des Projekts (Gestaltungsbereiche und –aufgaben gem. [1] S. 113) und nach der Anzahl RUP-Iterationen. Ferner sind in jeder Iteration korrigierte Fehler aus der Vorgängeriteration nachzutesten. Wir sprechen von Fehlernachtests gem. [2] S. 69. In der Praxis ist eine 100%-ige Testabdeckung wie beschrieben in [2] S. 124ff nicht erreichbar. Es gilt also, die durchzuführenden Testfälle risikogesteuert zu priorisieren, um so zum stichprobenartigen Testen überzugehen, was gem. [1] S. 233 im Projektmanagement legitim ist.

Sodann erlaubt erst der Einsatz von Testwerkzeugen, die notwendige Effizienz zur Einhaltung einer vertretbaren Testabdeckung zu erreichen ([4] S. 84ff).

[6] Projektklassen gem. [1, S. 71]
[7] Quelle: International Business Machines Corporation IBM Software Group Rational Software
[8] Aut.: = Anmerkung des Verfassers
[9] AB = Abnahme (Anforderungen), RV = Review (Anforderungen), UT = Unittest, IT = Integrationstest, ST = Systemtest, FV = Fachliche Validierung

2.6 Zusammengefasste Qualitätssicherungsanforderungen

Die unten stehende Tabelle trägt aus der Literatur zu unserem Lehrgang die zentralen Anforderungen an die Qualitätssicherung im Projekt zusammen und dient somit als Grundlage für die Analyse der Unterschiede und Probleme zwischen Theorie und Praxis wie sie die Disposition zu dieser Diplomarbeit vorsieht.

Stichwort	Anforderung
Gegenstand Projektdiagnose	Nebst Kosten und Terminen überwachen, Soll-/Ist-Vergleich zwischen Zielen und Ergebnissen ([1] S. 232 und Lernziel 1.3.2a) durchführen. IT: Anforderungen versus Release.
Periodische Überprüfung	An Projektmeilensteinen ([1] S. 233) prüfen. IT: Iterationsenden gem. RUP.
Opportune Projektdiagnose	Unter Berücksichtigung des Projektstands unterschiedliche Testaktivitäten (Lernziel 1.3.2a) durchführen.
Opportune Diagnosetechniken	Unter Berücksichtigung des Projektstands unterschiedliche Diagnosetechniken ([1] S. 240) anwenden.
Festhalten Soll-/Ist-Abweichungen	Mit Diagnoseprotokollen die Ergebnisse sichern ([1] S. 242.).
Gewaltentrennung	Keine Personalunion zwischen Entwickler und Tester ([1] S. 245) zulassen. Abnahmetests von Kompetenzträgern ([1] S. 246) durchführen lassen.
Formalisiertes Diagnosevorgehen	Strukturiertes, formalisiertes Abarbeiten eines Testkonzepts ([1] S. 245) durchsetzen.

Tabelle 2.6.1: zusammengefasste Qualitätssicherungsanforderungen

3 Praxisfall

3.1 Projekteckdaten

Thema	Inhalt
Auftraggeber	Kantonales Steueramt XY.
Ausgangslage	Bestehende, nicht integrierte Legacy[10]-Anwendung für Registerführung, Veranlagung und Bezug für Kapital- und Ertragssteuer juristischer Personen auf Basis von IMS-DB/DC™ und Adabas/Natural™. Bisherige Nutzungsdauer grösser 15 Jahre.
Gestaltungsbereiche	Register, Veranlagung, Bezug.
Ziele	Vollständige Ablösung mit integriertem System mit heute gängiger, zukunftsträchtiger Architektur. Ausmerzung fachlicher Mängel der Legacy-Anwendung.
Gestaltungsaufgaben	Vollständige Projektabwicklung nach den bekannten PM-Phasen mit RUP und sämtlichen softwareentwicklungsspezifischen Disziplinen. Zusätzlich Datenmigration auf IBM-DB2™.
Aufwand/Budget	Zirka 40 Personenjahre, 12 Mio. Schweizer Franken.
Projektklasse	Grossprojekt.
Termine/Meilensteine	Endtermin: 1.1.2006 Meilensteine: Inception 2 Iterationen Elaboration 3 Iterationen Construction 7 Iterationen (!) Transition 2 Iterationen (voraussichtlich)
Aktueller Projektstand	Ende Iteration Construction 2, rund 2000 Programmodule (Klassen) implementiert.
Projektaufbauorganisation	kundenseitig: Matrixprojektorganisation herstellerseitig: Reine Projektorganisation

Tabelle 3.1.1: Projekteckdaten

[10] legacy (engl.) ➔ Vermächtnis, Erbschaft, IT: Altanwendung

3.2 Testorganisation

Die Testaktivitäten im Projekt sind auf verschiedene Leistungserbringer aufgeteilt. Die farblich unterlegten Referenzen beziehen sich auf Kapitel 2.3.

Entwickler: Da sie mit ihrer Entwicklungsumgebung über die dafür notwendige Infrastruktur verfügen, nehmen sie die Einzel- oder **Unittests** der von ihnen erstellten Klassen vor. Ferner liegt ein Integrationstest zum Prüfen des Zusammenspiels der Komponenten auf einem lokalen Server in ihrer Verantwortung.

Deploymentteam[11]: Bei der Installation der Software auf einer spezifischen Plattform testet es rudimentär die grundsätzliche Lauffähigkeit des Systems. Die Fachwelt spricht von einem Smoke[12]-Test. Spillner und Linz meinen dazu: „Es wird nur geprüft, ob ein Systemabsturz oder offensichtliche Fehlerwirkungen auftreten. (...) Der Begriff Smoke-Test ist in Analogie zum Ausfall bei älteren elektrischen Geräten gewählt, bei denen Rauch im Fehlerfall aufsteigt. (...) Ein Smoke-Test wird oft als erster Test durchgeführt, um zu entscheiden, ob das Testobjekt die notwendige Reife hat, um mit den umfangreicheren Testverfahren geprüft zu werden.“ ([2], S. 123).

Testteam: Unsere Organisation verfügt über ein projektübergreifendes/autonomes Testteam mit zurzeit vier Personen, das dem Leiter des Teilprojekts „Test, Deployment, Infrastruktur“ rapportiert. Es befasst sich mit den **Reviews** der Anforderungsdokumente und dem **Systemtest** auf der Zielplattform. Es arbeitet vollkommen strukturiert nach einem Haupt- oder Mastertestplan über das Gesamtprojekt und nach einem Iterationstestplan pro Wiederholung. Die Testfälle leitet es mit anerkannten Technologien aus den Anforderungsdokumenten ab und priorisiert ihre Durchführung risikogetrieben.

Endanwender: Ihnen kommt die Aufgabe zu, die Anforderungsdokumente vor der Umsetzung fachlich abzunehmen (**Abnahme**).

Zudem nehmen sie am Ende einer Iteration die **Fachliche Validierung** auf der Zielplattform vor. Es handelt sich um eine iterative Abnahme des System.

Schliesslich werden sie am Ende der letzten Iteration der Übergabephase (Transition) einen umfassenden **Abnahmetest** des Gesamtsystems vornehmen.

Dritte: Externe Berater nehmen neuerdings **Audits** vor. Im Gegensatz zu Reviews überprüfen Audits keine (Realisierungs-)Dokumente sondern im Wesentlichen das PM-Vorgehen und die Einhaltung der Prozesse. Tremp und Scheuring dazu: „Ein Audit wird vom Management angeordnet. Das Ziel ist es, (...) die Einhaltung und Effizienz der vorgegebenen Qualitätsrichtlinien, Standards und Prozesse zu überprüfen.“ ([4], S. 46).

3.3 Aktivitäten Testteam im Projekt

3.3.1 Reviews

Das Testteam führt die formalen Reviews der Anforderungsdokumente durch, um sie auf ihre Verständlich- und Vollständigkeit hin zu überprüfen. Diese Arbeit geht meist Hand in Hand mit dem Erstellen der Testfälle für die funktionalen Systemtests (vgl. auch Kapitel 2.2, Softwarequalitätsmerkmale ➔ Funktionalität). Seine Beanstandungen hält es in einer Problemdatenbank zu Handen des Projektleiters fest. Mehrheitlich beurteilt es die von externer Stelle während den RUP-Phasen Inception und Elaboration erstellten Anforderungsdokumente als ungenügend bezüglich Verständlich- und Vollständigkeit sowie bezüglich Durchgängigkeit und Integrität der Anforderungen. Lenkungsausschuss und Projektleitung haben zu Projektbeginn beschlossen, die Anforderungserhebung extern durchführen zu lassen, weil sie interne Kapazitätsengpässe fürchteten.

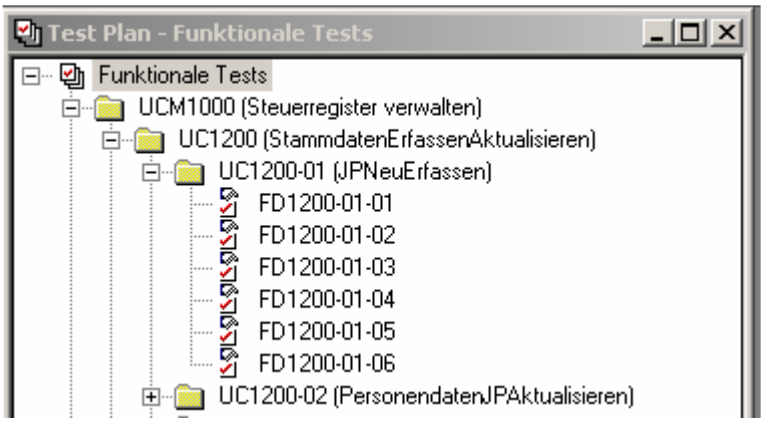
[11] deployment (engl.) ➔ Entfaltung, Entwicklung, IT: Produkteinstallation auf einer spezifischen Plattform
[12] smoke (engl.) ➔ Rauch

3.3.2 Systemtests

Klassischerweise besteht Testen aus dem Erstellen eines Testplans, dem Ableiten von Testfällen und -vorschriften (Testdrehbuch), dem Durchführen und Protokollieren der Tests, dem Festhalten von Fehlern und schliesslich dem Erstellen eines Testberichts über den gesamten Testzyklus für eine Iteration ([3], S. IV-12). Wir führen diese Aktivitäten hauptsächlich werkzeugunterstützt[13] durch und bewegen uns bezüglich Testausführung hin zur geforderten Testautomation.

3.3.2.1 Testplan

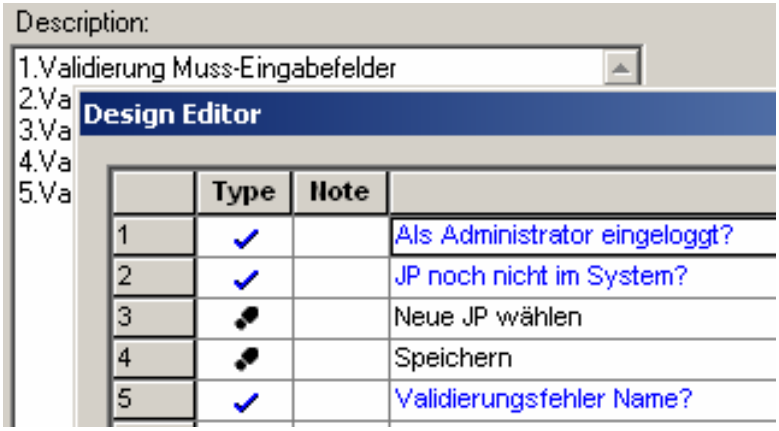
Der Testplan hält die geplanten Testfälle gegliedert nach Applikationsbereichen fest. Die Zuordnung zu den einzelnen Iterationen und somit zu den Testzyklen ist Aufgabe der Projektleitung in Absprache mit dem Endanwender. Typischer- und sinnvollerweise erfolgt die Zuordnung risikogesteuert.



(JP = Juristische Person)

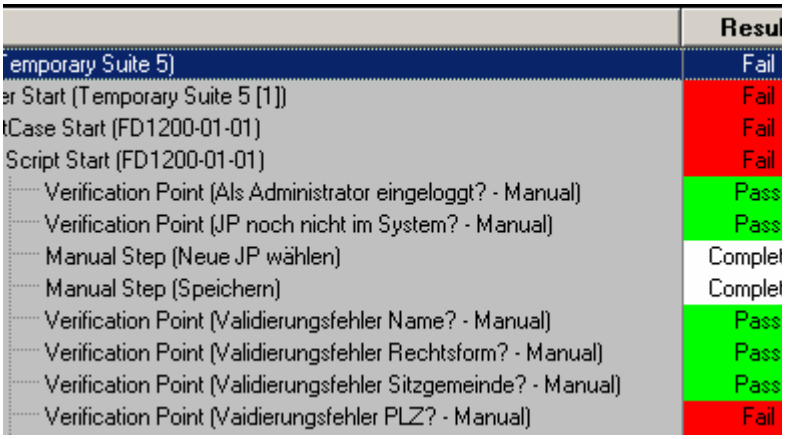
3.3.2.2 Testfälle und –vorschriften

Zu jedem Testfall existiert eine Testfallvorschrift, die sich als manuell oder automatisiert durchführbar implementieren lässt. Zeilen mit blauen Häkchen in der Spalte „Type“ sind Überprüfungs- oder Verifikationspunkte, die bei der Testdurchführung die Reaktion „Fail“[14] für nicht bestanden resp. „Pass“[15] für bestanden erhalten können. Zeilen mit dem Schuhsohlensymbol sind manuelle oder automatisierte Testschritte, mit denen eine bestimmte Aktion durchzuführen ist.



3.3.2.3 Testprotokoll

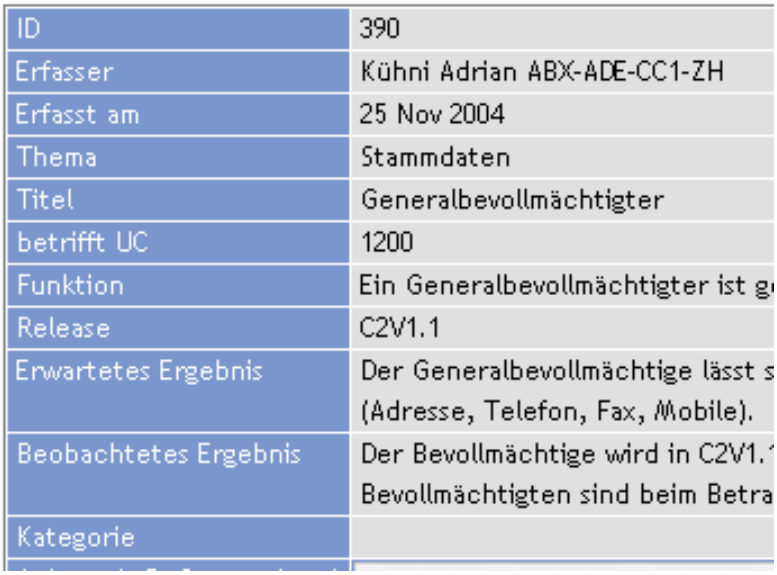
Bei der Testdurchführung entsteht ein Durchführungsprotokoll, das für jeden Verifikationspunkt den Rückgabewert ausweist. Für Testschritte weist das Protokoll aus, ob das System sie erfolgreich durchgeführt hat (Completed[16]) oder nicht. Ein Testfall als Ganzes gilt als bestanden, wenn alle Verifikationspunkte die Reaktion „Pass“ und alle Testschritte die Reaktion „Completed“ erhalten haben.



[13] Rational TestManager™ und Rational Robot™
[14] fail (engl.) ➔ fehlschlagen
[15] pass (engl.) ➔ durchkommen, bestehen
[16] complete (engl.) ➔ beendigen, vollenden, abschliessen

3.3.2.4 Fehlerrapporte

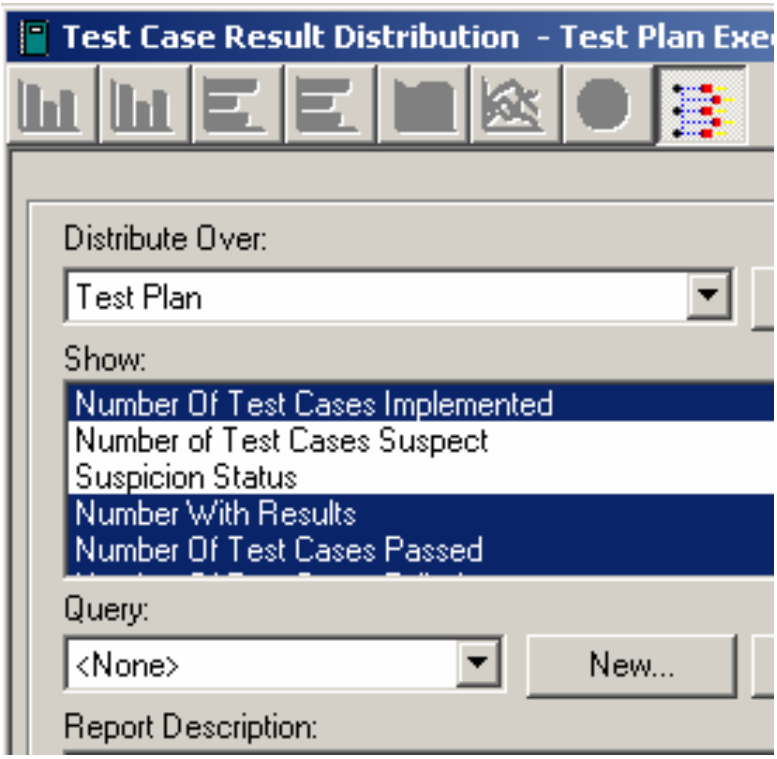
Die Rapportierung der Ergebnisse einzelner Testfalldurchführungen erfolgt leider manuell, das heisst, wir verfügen noch nicht über eine durchgängige Produktepallette für die Testdurchführung und die Fehlerrapportierung resp. Änderungsverwaltung (Fault Tracking resp. Change Management). Fehlerrapporte zeigen Abweichungen zwischen den erhaltenen Istwerten einer Testfalldurchführung und den erwarteten Ergebnissen gem. Testfalldefinition und dienen somit als Grundlage für den Fehlerbehebungsprozess und für den Fehlernachtest.



3.3.2.5 Testbericht

Am Ende eines Testzyklus, wenn die Testaktivitäten für eine bestimmte RUP-Iteration abgeschlossen sind, verfassen wir einen gesamthaften Testbericht zu dieser Iteration. Dazu nutzen wir die Statistik- und Auswertungsfunktionalitäten unserer Testwerkzeuge. Ihre Ergebnisse erweitern wir mit Anmerkungen in Prosa und erhalten so einen Gesamtbericht zu Handen des Projektleiters.

Ganz wichtig dabei sind auf den Ergebnissen basierende Trendanalysen bezüglich der künftig zu erwartenden Qualität und die Forschung nach Ursachen für allfällige Abweichungen zwischen erwarteter und erreichter Qualität.



Folgende Tabelle zeigt die Testfalldurchführungsstatistik für zwei RUP-Iterationen der Construction-Phase:

	Testfälle			
Testzyklus	vorgesehen	durchgeführt	bestanden	nicht bestanden
Nummer C1	271	102(37%)	56(55%)	49(45%)
Nummer C2	271	228(84%)	60(26%)	163(74%)

Tabelle 3.3.2.5.1: Testfalldurchführungsstatistik über zwei RUP-Iteratioen

- Nach Abschluss der Iteration C1 konnten wir von den vorgesehenen Testfällen gerade mal 37% durchführen. Ursache:

  Zu optimistische Planung für Iteration C1.

- Von den durchführbaren Testfällen haben nur etwas mehr als die Hälfte bestanden. Diese Qualität müssen wir als absolut ungenügend betrachten. Ursache:

  Die mangelhaften Anforderungsdokumente lassen die Definition gültiger Testfälle nicht zu und/oder sie verleiten die Entwickler zu unkontrollierter Kreativität in der Interpretation der Anforderungen.

- Nach Abschluss der Iteration C2 konnten wir dank massiven Realisierungsanstrengungen 84% der Testfälle durchführen. Dabei allerdings sank die Qualität massiv ab. 74% der durchgeführten Testfälle haben nicht bestanden. Ursachen:

  Die im Testzyklus zu Iteration C1 festgestellten Mängel sind nicht in einen sauberen Fehlerbehebungs- resp. Änderungsverwaltungsprozess eingeflossen (Bugfixing- resp. Change Management-Prozess).

  Vorgenommene Anpassungen seitens der Entwickler sind nicht in den Anforderungsdokumenten vermerkt. Folge: Wir haben unsere Testfälle in Unkenntnis der Änderungen nicht angepasst (➔ Kommunikation im Projekt).

  Der Produktefortschritt von Iteration C2 ist – aus den gleichen Gründen wie bei C1 – qualitativ mindestens ebenso schlecht wie das Resultat von C1. Die Probleme kumulieren sich.

Aufgrund dieser erschreckenden Resultate hat der Projektsteuerungsausschuss einen Entwicklungsstopp zwecks Bereinigung der Architektur und der Anforderungen veranlasst. Die Nichterfüllung des terminlichen Projektziels ist damit gegeben.

4 Analyse der Unterschiede und Probleme

Aufgrund meiner Aussagen und Erfahrungen halte ich anhand der Tabelle der zusammengefassten Testanforderungen gem. Kapitel 2.6 fest:[17]

Stichwort	Anforderung
Gegenstand Projektdiagnose	Nebst Kosten und Terminen überwachen, Soll-/Ist-Vergleich zwischen Zielen und Ergebnissen ([1] S. 232 und Lernziel 1.3.2.a) durchführen. IT: Anforderungen versus Release. ☺ Unsere Testaktivitäten stellen einen Vergleich zwischen den festgehaltenen funktionalen und nicht funktionalen Anforderungen sicher. Der vom Lernziel verlangte Soll-/Ist-Vergleich ist damit gegeben. (Zur Kosten- und Terminüberwachung macht diese Arbeit gem. Themenanmeldung keine Aussagen.)
Periodische Überprüfung	An Projektmeilensteinen ([1] S. 233) prüfen. IT: Iterationsenden gem. RUP. ☺ Wir nehmen Systemtests und die Fachliche Validierung am Ende jeder RUP-Iteration vor. Das Ende einer RUP-Iteration ist ein Projektmeilenstein. Die Forderung, an Projektmeilensteinen zu prüfen, ist somit erfüllt.
Opportune Projektdiagnose	Unter Berücksichtigung des Projektstands unterschiedliche Testaktivitäten (Lernziel 1.3.2a) durchführen. 😐 Grundsätzlich nehmen wir je nach Projektstand in den vier RUP-Phasen unterschiedliche Tests vor (Abnahme der Anforderungen, Review, Unittest, Integrationstest, Systemtest, Fachliche Validierung, Abnahmetest). In der Praxis tun sich die Endanwender mit der Abnahme der Anforderungen sehr schwer und der Integrationstest ist vernachlässigt. Zwar nimmt das Entwicklerteam Integrationstests auf einem lokalen Server vor, um das Zusammenspiel der Komponenten zu überprüfen. Ein solcher Test auf der produktionsnahen Zielumgebung hingegen fehlt.

[17] ☺ = erfüllt, 😐 = teilweise erfüllt, ☹ = nicht erfüllt

Opportune Diagnosetechniken	Unter Berücksichtigung des Projektstands unterschiedliche Diagnosetechniken ([1] S. 240) anwenden. ☺ Die Unittests der Entwickler basieren auf Codeebene. Sie prüfen ihre Klassen mit entsprechenden Werkzeugen Programmschritt um Programmschritt. Die Fachwelt spricht dabei vom White-Box-Testverfahren ([2], S. 124ff). Die Systemtests, die Fachliche Validierung und endlich der Abnahmetest sind geschäftsfallbasiert. Das Testteam und der Endanwender testen demnach nach dem Black-Box-Testverfahren (vgl. dazu [2], S. 98ff). Die Forderung nach unterschiedlichen Diagnosetechniken ist mit den beiden erwähnten Testverfahren erfüllt. Wir ergänzen sie zudem mit Reviews (Anforderungen) und mit Audits (Prozesse).
Festhalten Soll-/Ist-Abweichungen	Mit Diagnoseprotokollen die Ergebnisse sichern ([1] S. 242.). 😐 Wie gefordert halten wir sämtliche Befunde aus den Review- und Testaktivitäten in Protokollen fest. Leider fehlt dabei aber eine Schnittstelle zwischen den Testdurchführungswerkzeugen und der Protokollierung ihrer Ergebnisse. Wir stellen einen Medienbruch fest.
Gewaltentrennung	Keine Personalunion zwischen Entwickler und Tester ([1] S. 245) zulassen. Abnahmetests von Kompetenzträgern ([1] S. 246) durchführen lassen. ☺ Entwickler, Deploymentteam, Testteam, Endanwender und Dritte sind in die Qualitätssicherung eingebunden. Die Forderung ist erfüllt.
Formalisiertes Diagnosevorgehen	Strukturiertes, formalisiertes Abarbeiten eines Testkonzepts ([1] S. 245) durchsetzen. ☺ Testteam wie Endanwender arbeiten mit klar strukturierten Testplänen. Das Testteam entwickelt seine Testfälle nach anerkannten Technologien aus den Anforderungsdokumenten (Pfadanalyse, Äquivalenzklassenbildung, Grenzwertanalyse ([2], S. 98ff)). Der Endanwender testet geschäftsprozessorientiert. Vgl. dazu auch [2], S. 9 und 59.

Tabelle 4.1: Zielerreichung Qualitätssicherung

Wir erkennen, dass unsere Testaktivitäten den Vorgaben gemäss Lernziel und der in Kapitel 2 aufgearbeiteten Theorie absolut genügen.

Warum ist die Qualität des Produkts nach der zweiten Konstruktionsiteration trotzdem so schlecht?

- Das Grundproblem liegt in den Anforderungsdokumenten. Sie wurden im Auftrag unserer Organisation von externer Stelle (vgl. Kapitel 3.3.1 (Reviews)) zusammen mit dem Auftraggeber/Endanwender erstellt. Sowohl Ersteller als auch Auftraggeber waren resp. sind dabei überfordert. Unsere eigene Steuerfachexperten blieben vom Prozess ausgeschlossen.

- Die Befunde aus den Testteam-Reviews der Anforderungsdokumente verhallten praktisch ungehört.

- Die Iterationen der Konstruktionsphase waren zu ehrgeizig geplant. Als Folge davon wurden die Fehler/Probleme der Vorgängeriteration in der nächsten Iteration nicht bereinigt. Die Probleme kumulierten sich. Die Qualität sank.

- Das nach RUP geforderte Assessment[18] am Ende einer Iteration fand in der Inception- und in der Elaborationsphase nicht oder nicht ehrlich genug statt. Erkenntnisse späterer Assessments wurden vom Projektsteuerungsausschuss bis vor kurzem zu wenig wahrgenommen. Der Vorschlag eines Testers/Qualitätssicherers, eine nach RUP legitime Zwischeniteration zwecks Fehlerbereinigung einzuschieben, wurde als absurd abgetan.

- Der Fehlerbehandlungs- und Änderungsverwaltungsprozess (Fault Tracking und Change Management) ist nicht ausgereift. Vorgenommene Änderungen am Produkt fliessen nicht in die Anforderungsdokumente ein. So finden sie auch keinen Eingang in die Testfalldefinitionen. Produkt und Anforderungsdokumente klaffen je länger je mehr auseinander. Folge: Das Produkt an sich wird zum Testorakel ([2], 28ff), zu der Quelle also, aus der die zu erwartenden Reaktionen und Resultate des Systems zu Testzwecken abzuleiten sind. Spillner und Linz dazu: „Sind weder eine umfassende Benutzerdokumentation noch eine detaillierte Spezifikation oder Diagramme verfügbar, so muss das System selbst als Testorakel dienen. (...) Diese Möglichkeit des Testorakels ist sicherlich die schlechteste, da das Testobjekt selbst Grundlage ist und eine Kernforderung, die Festlegung der Sollwerte vor der Testausführung, nicht erfüllt ist.“ ([2], 29).

5 Schlussfolgerungen, Würdigung

Qualitätssicherung ist für mich eine der wichtigsten Disziplin in der Projektarbeit. Sie schützt sowohl den Auftraggeber als auch den Auftragnehmer.

Qualitätssicherung hat ganz zu Beginn eines Projekts einzusetzen und begleitet es bis zu seinem Abschluss.

Qualitätssicherung ist enorm schwierig und aufwändig. Gerade wegen Letzterem ist noch viel missionarische Arbeit für sie zu leisten.

Qualitätssicherung kann nur wirken, wenn Taten folgen.

[18] assessment (engl.) ➔ Abschätzung, Bewertung, Taxierung

6 Literaturverzeichnis

[1] Pfetzin K., Rohde A.: Ganzheitliches Projektmanagement, 1. Auflage, Verlag Dr. Götz Schmidt, Wettenberg bei Giessen, 2001

[2] Spillner A., Linz T.: Basiswissen Softwaretest, 2. überarbeitete Auflage, dpunkt.verlag GmbH, Bremen und Möhrendorf, 2003

[3] IFA Institut für Informatikausbildung Zürich [Hrsg.][19], Kursunterlagen zum Lehrgang SAQ/ISTQB®/ASQF® Cetrified (Software) Tester, Zürich, 2004

[4] Tremp H., Scheuring J., IT-Systeme prüfen, 1. Auflage, Compendio Bildungsmedien AG, Zürich, 2003

7 Urheberrechtshinweise

Rational Unified Process® (RUP) ist eingetragenes Warenzeichen der International Business Machines Corporation IBM Software Group Rational Software, Santa Barbara CA, U.S.A.

ISTQB® Certified Tester ist eingetragenes Warenzeichen des International Sofware Testing Qualification Board, Erlangen, Bundesrepublik Deutschland.

ASQF® –Certified-Tester ist eingetragenes Warenzeichen des Arbeitskreises Softwarequalität Franken e.V., Erlangen, Bundesrepublik Deutschland.

IMS-DB/DC™ ist eingetragenes Handelszeichen von International Business Machines Corporation IBM, Santa Barbara CA, U.S.A.

Adabas/Natural™ ist eingetragenes Handelszeichen der Software AG, Darmstadt/Eberstadt, Bundersepublik Deutschland.

IMS-DB2™ ist eingetragenes Handelszeichen von International Business Machines Corporation IBM, Santa Barbara CA, U.S.A.

Rational TestManager™ und Rational Robot™ sind eingetragene Warenzeichen der International Business Machines Corporation IBM Software Group Rational Software, Santa Barbara CA, U.S.A.

[19] [Hrsg.] = Herausgeber

8 Anhang I: Text Themenanmeldung

Diplom Projektassistent/in KV Zürich Business School
Leistungsnachweis: Schriftliche Arbeit zu einem Lernziel

Themenanmeldung

Name/Vorname des Kandidaten:	**K ü h n i** Adrian
Titel der Arbeit:	Fortlaufende Qualitätskontrolle in einem IT-Projekt
Nummer des Lernziels:	1.3.2a
Volltext des Lernziels:	Unter Berücksichtigung des Projektstands die geplanten Kosten-, Termin- und Qualitätsziele periodisch überprüfen und Ist/Soll-Abweichungen festhalten.
Kurzbegründung:	Nebst der Budget- und Termineinhaltung ist die Zielerreichung in der verlangten Qualität der dritte Erfolgsfaktor für Projekte. Seit zwanzig Jahren beobachte ich im IT-Bereich immer wieder, dass Qualität ***der*** Erfolgsfaktor für IT-Projekte schlechthin ist. Nach Jahren der Analyse und Programmierung habe ich mich beruflich auf professionelles, systematisches Softwaretesten und die dazu gehörende Teilprojektleitung ausgerichtet. Ziel meiner Arbeit ist der Theorie-/Praxisabgleich bezüglich fortlaufender Qualitätskontrolle in einem konkreten IT-Projekt, in dem ich zurzeit als Tester arbeite. Die Arbeit behandelt die Themen Budget- und Terminkontrolle nicht.
Disposition über Struktur und Inhalt:	**Teil 1:** Vermittelt die theoretischen Grundlagen bezüglich fortlaufender Qualitätskontrolle in IT-Projekten aufgrund der Lehrgangsunterlagen und weiterer Literatur. **Teil 2:** Beschreibt die Testaktivitäten in einem IT-Grossprojekt für die steuerliche Veranlagung juristischer Personen. Wir wickeln das vierzig Personenjahre starke Projekt nach dem Rational Unified Process (RUP) ab, der für die fortlaufende Qualitätskontrolle wegen des iterativen Vorgehens prädestiniert ist. **Teil 3:** Zeigt die Differenzen zwischen den theoretischen Ansätzen und dem Praxisbeispiel auf und sucht nach möglichen Gründen für Abweichungen, z.B. ungenügende Anforderungserhebung (Requirements Engineering) oder mangelhafter Änderungsprozess (Change Management). **Teil 4:** Weist die eigenen Schlussfolgerungen bezüglich dem Testprozess und den Entwicklungsaktivitäten aus.

Uetikon, 6.12.04.
A. Kühni.