Resilienz – "Joy on Error" Usability als Chance und Ressource

Kerstin Palatini
Hochschule Anhalt, FB Informatik,
Lohmannstraße 23,
06366 Köthen,
k.palatini@inf.hs-anhalt.de

Volkmar Richter
Hochschule Anhalt, FB Informatik,
Lohmannstraße 23,
06366 Köthen,
v.richter@inf.hs-anhalt.de

Abstract
Technische Artefakte realisieren ihre Funktion immer mehr mit Hilfe von Software. Sichere und gut handhabbare Produkte und Systeme werden gefordert, wobei allein durch die massive Ausbreitung und weltweite Vernetzung die Möglichkeiten für Kommunikationsfehler und auch die Fehlerwirkungen immens werden. Zudem wird die Nachverfolgbarkeit erschwert.
Im Folgenden wird die Notwendigkeit herausgearbeitet, der nicht erfüllbaren Forderung nach Fehlerfreiheit in der Useware-Entwicklung mit der Entwicklung von Resilienz zu begegnen. Hierbei wird auf die Möglichkeiten innerhalb des Usability Engineering Prozesses hingewiesen, der die Entwicklung von Resilienz mit der Forderung nach zuverlässigen und langlebigen Produkten von der Konzeption, über die Entwicklung bis hin zum Recycling bzw. der Weiterentwicklung verbindet und so z. B. bereits bei der Implementierung die Reaktion auf nicht geplante Ereignisse sorgfältig berücksichtigen kann. Resilienz muss demnach in den Betrachtungen zur Usability und zum Systemtest eine stärkere Rolle spielen, denn unklare Fehlermeldungen und Systemabstürze beeinträchtigen den Nutzer mindestens genau so wie eine umständliche Handhabung.

Keywords:
/// Resilienz
/// Usability
/// Fehlertoleranz
/// Softwareengineering
/// Test

1. Resilienz

Bei einer kritischen Betrachtung technischer Artefakte stellt man fest, dass selbst die gesellschaftlichen, sozialen Umwelten nicht fehlerfrei, nicht die besten aller denkbaren Welten sind und damit in Zusammenhang betrachtet werden müssen. Miniaturisierung, Vernetzung und der Ersatz von immer mehr Hardwarefunktionen durch Software lassen die Menge an eingesetztem Code, die Menge an Schnittstellen und damit die möglichen Schwachstellen immer mehr ansteigen. Technikdependenz und Technikpaternalismus, also Abhängigkeit von den softwarebasierten Lösungen und Bevormundung durch diese – oder deren Schöpfer – sind die Folge.

Da prinzipiell nicht alle Fehler durch Tests, und seien sie noch so intensiv, gefunden werden können, gewinnt die sinnvolle Behandlung von Ausnahmezuständen eine immer größere Bedeutung für die Überlebensfähigkeit und Resilienz der Artefakte. Fehlertoleranz, Fehlermanagement und Resilienz verbessern die Usability von technischen Artefakten wesentlich, deshalb müssen sie wesentlicher Bestandteil des Usability-Engineerings werden.

1.1. Begriff

Der Begriff Resilienz geht auf „resilire" (lat.) zurück und bedeutet so viel wie zurückspringen oder abprallen. Der englische Begriff „resilience", der mit Widerstandskraft, Spannkraft und Elastizität übersetzt werden kann, wurde ursprünglich in der Werkstoffkunde für Materialien verwendet, die nach erfolgter Belastung (z. B. Druck) in ihre ursprüngliche Ausgangsform zurück kehrten. Resilienz wird aus verschiedener Sicht definiert und erfuhr besonders in Entwicklungspsychologie und Pädagogik seit den 1980er Jahren eine besondere Beachtung. Die Begriffsbestimmung durch Corinna Wustmann ist allgemein anerkannt, da sie sowohl externale wie internale Kriterien berücksichtigt und Resilienz als „[…] Widerstandsfähigkeit gegenüber biologischen, psychologischen und psychosozialen Entwicklungsrisiken" bezeichnet. (Wustmann 2004, 18)

Im Zusammenhang mit immer häufiger auftretenden Störfällen und Krisen wird der Begriff der Resilienz auch zunehmend in anderen wie z. B. technischen Zusammenhängen diskutiert. Bezogen auf technische Systeme wird Resilienz mit Fehlertoleranz übersetzt, bzw. mit der Fähigkeit, ihre Funktionsweise auch aufrechtzuerhalten, wenn unvorhergesehen Eingaben oder Fehler in der Hard- oder Software auftreten.

1.2. Merkmale

„Das Konstrukt Resilienz ist ein dynamischer oder kompensatorischer Prozess positiver Anpassung bei ungünstigen

Entwicklungsbedingungen und dem Auftreten von Belastungsfaktoren. Charakteristisch für Resilienz sind außerdem ihre variable Größe, das situationsspezifische Auftreten und die damit verbundene Multidimensionalität." (Fröhlich-Gildhoff, Rönnau-Böse 2009, 13)

Resilienz ist eine zu beeinflussende und zu gestaltende Größe. Sie ist damit keine gegebene oder permanente unveränderliche Eigenschaft, also „eine variable Größe" (Wustmann 2004, 30).

Resilienz ist damit auch ein „dynamischer Anpassungs- und Entwicklungsprozess" (Wustmann 2004, 28).

1.3. Forschung

Ein Meilenstein in der Resilienzforschung ist die von der Psychologin Emmy Werner von der University of California Mitte der 1950er Jahre begonnene Studie über ca. 700 Kinder der Insel Kauai, die über mehr als 40 Jahre in ihrer Entwicklung beobachtet wurden. Mit ihrer „Kauai-Studie" (Emmy Werner 1971) leitete sie den Perspektivenwechsel in der Wissenschaft, weg von der Pathologie hin zur Resilienz, ein. Sie machte hierin auf die Möglichkeiten des „Gedeihens trotz widriger Umstände" aufmerksam und beschrieb Risiko- und Schutzfaktoren.

In der Resilienzforschung stehen folgende Aspekte auf der Agenda (s. a. Wustmann 2004, 19):

- günstige Entwicklung trotz ungünstiger Umgebungsfaktoren (Risiken, Prävention)
- beständige Kompetenzen zur Fehler- bzw. Stressbewältigung (Nachhaltigkeit)
- positive Bewältigung bzw. schnelle „Erholung" nach Störfällen (Krisenmanagement)

Es geht dabei insgesamt um die Erforschung und Entwicklung präventiver (Kompetenzerwerb, Bedingungen) und nachhaltiger (Erhalt, Management) Maßnahmen, die Resilienz befördern können und um die Beeinflussung von Risikofaktoren. Resilienz ist bedingt einerseits durch eine Risikosituation bzw. Risikofaktoren und andererseits deren positiver Bewältigung bzw. -beherrschung mittels geeigneter Ressourcen.

In den üblichen Baumstrukturen des Usability-Engineering (u. a. Nielsen 1994) und der Teststrategie (u. a. Spillner 2010) findet sich funktionaler Test rsp. Usabilitytest jeweils in den äußersten Zweigen, was den Schluss nahelegt, dass es sich in Wirklichkeit nicht um hierarchische Strukturen handelt. Usability - Korrektheit - Resilienz müssen in ihrem inneren Zusammenhang betrachtet werden.

2. Risikofaktoren für Resilienz technischer Systeme

Im Gegensatz zur Resilienz von Organismen, die per evolutionärer Prozesse genetisch vorprogrammiert ist (interessanterweise haben Pflanzen, die nicht vor ungünstigen Einflüssen weglaufen können eine wesentlich höhere Zahl genetischer Programme), muss die Resilienz technischer Artefakte implementiert werden. Dazu ist es notwendig, die speziellen Risiken von Software zu beachten. Bis auf das Risikopotential von Software als Unikat treffen die weiteren teilweise auch auf Hardware zu. Kontrollverlust und Unschärfen in der Kommunikation sind weitere Problemfelder.

2.1. Unikate

Software wird grundsätzlich als Unikat produziert. Sicher gibt es wenige Ausnahmen, in denen eine mehrfache Programmierung aus Sicherheitsgründen erfolgt, aber auch da handelt es sich um (mehrere) Unikate. Grundproblem ist, dass von dem einmal erstellten Urexemplar beliebig viele identische digitale Kopien herstellbar sind.

Während bei der Produktion von mechanischen, elektronischen und anderen Produkten immer wieder einzelne Produkte getestet werden, beschränkt man sich bei Software auf die Prüfung unverfälschten Kopierens. Somit kommen sehr schnell fehlerhafte Exemplare in Umlauf, da der Weg über die Kunden recht lang ist und die Kunden im Sinne unserer These „Joy on Error" sich inzwischen an fehlerhafte Software gewöhnt haben. Der Grundansatz der Softwaretechnik, Software wie jedes andere technische Produkt zu behandeln scheitert auch daran.

Kritisch ist in diesem Rahmen die Übertragung vieler bisher mechanischer oder elektronischer Verfahren auf Softwarekomponenten. Letztere sind zweifellos leichter anpassbar, flexibler, billiger, manchmal auch sexier, aber bringen das spezifische Fehlerpotential von Software mit ein.

Grund für die unterschiedliche Bewertung von Hard- und Softwaresystemen ist offensichtlich die Vermutung, dass Software als determiniertes Produkt, immer gleich reagiert (einschließlich evtl. vorhandener Fehler), während das technische System in der öffentlichen Wahrnehmung richtig als Zusammenspiel von toleranzbehafteten Bauelementen angesehen wird, die in ihren Eigenschaften auch altern und stochastisches Verhalten zeigen.

Während für technische Systeme Ausfallsicherungen, wie Mehrfachauslegung, Überdimensionierung und Sicherheitskonzepte (Fall Back, Reserveanlagen) eingesetzt werden und die Funktionalität durch Wartungs- und Prüfintervalle in weiten Bereichen garantiert werden kann, ist das bei Software kaum durchgesetzt. Ein Grund dafür ist der Unikatcharakter (es existieren nur identische Kopien) der zulässt, dass Prototypen bereits zur Auslieferung gelangen, da die Kopierkosten verschwindend gering sind und Patches schnell über das Internet verbreitet werden können. Insofern hat auch der hohe Innovationsdruck in der IT-Branche die Situation mitverschuldet. Nicht ganz unschuldig sind die Entwickler an dieser Fehleinschätzung, da sie oft nur den eigentlichen Implementierungsprozess als „Schöpfungsakt" im Gedächtnis behalten und die Meinung vertreten, dass dieses oder jenes „schnell mal geschrieben" werden kann.

2.2.
Entfremdung

Hauptursache der unbefriedigenden Situation sind jedoch die Entfremdungsprozesse. In der Anfangszeit der „Computerei" waren Konstrukteure, Programmierer, Nutzer in Personalunion vereint. Selbst als die Arbeitsteilung auch da Einzug hielt, war der Kontakt noch sehr eng. Erst in den 60er Jahren, als die betriebliche Informationsverarbeitung mit Massendaten und Mainframes in Serienproduktion (z. B. IBM 360) mit Betriebssystemen zum Einsatz kamen, entfremdeten sich die Beteiligten. Heute, in der PC- und Internetzeit ist der Kontakt nicht nur unmöglich geworden, die schiere Menge der Nutzer und der Angebote in weltweiter Vernetzung erschweren auch die Suche nach Problemen ungemein.

Das „web 3.0 – Internet der Dinge" das zurzeit propagiert wird, verschärft die Probleme durch die interne Kommunikation von Teilnehmern, die nicht menschlich sind (Übergang Human-Computer Interaction,-HCI zu Machine-to-Machine Kommunikation – M2M). Sensoren und Aktoren sind disloziert und entziehen sich der menschlichen Kontrolle immer mehr. Fehler werden dann nicht mehr schnell als solche erkannt, sondern können sich fortpflanzen, manchmal auch über längere Zeit verdeckt sein, da sie durch andere Fehlabläufe überdeckt werden (Maskerade von Fehlern). Damit sinkt die Zuverlässigkeit der Systeme allgemein und damit in Zusammenhang auch die Verlässlichkeit der gespeicherten Daten.

2.3.
Aging

"Programs, like people, get old. We can't prevent aging, but we can understand its causes, take steps to limit its effects, temporarily reverse some of the damage it has caused, and prepare for the day when the software is no longer viable. … (We must) lose our preoccupation with the first release and focus on the long term health of our products." (Parnas 1994)

Nicht erst seit Parnas ist der Begriff der Softwarealterung bekannt. Man unterscheidet zwischen der relativen Alterung durch die Entwicklung der Umgebung und der absoluten Alterung durch die bei notwendigen Änderungen oftmals entstehenden neuen Fehler. Beide Formen der Alterung sind in ihren konkreten Auswirkungen nicht vorhersehbar. Im Zusammenwirken verschiedener Komponenten kommt es außerdem schnell zu Inkompatibilitäten, wenn Versionen nicht die gleiche Schnittstellenspezifikation nutzen.

Der Ansatz „Resilienz von Software" erfordert, dass Software Fehler selbständig erkennt, darauf reagieren kann und sinnvoll und sicher, vielleicht mit eingeschränktem Funktionsumfang, weiterarbeitet. Das schränkt keineswegs die Forderung ein, die im Change-Mangement besteht, periodisch neue Versionen vorzubereiten und auszuliefern, erlaubt es aber, Software über den beabsichtigten Zeitrahmen noch eingeschränkt zu nutzen oder katastrophische Reaktionen zu vermeiden.

2.4.
Fehlerfreiheit

Neben der anerkannten Tatsache, dass es nicht möglich ist, softwaregestützte Systeme vollständig auf Fehlerfreiheit zu testen, muss natürlich auch gefragt werden, ob es überhaupt möglich ist, eine womöglich vorhandene Fehlerfreiheit in einer nicht fixen Umgebung langfristig zu garantieren.

In einem Beitrag hat Mascheck bereits die Notwendigkeit von Resilienz für langlebige technische Artefakte vorweggenommen:

„Durch die Anzahl der möglichen unterschiedlichen Zustände ist [aber] eine Schranke für die Informationsmenge gegeben, die im System gespeichert werden kann. Es ist also im Prinzip niemals möglich, die zukünftige Entwicklung auf irgendeinem Gebiet oder das Verhalten einer Maschine unter den Bedingungen des praktischen Einsatzes mit absoluter Sicherheit vorauszusagen. Durch die Unterscheidung von wesentlichen und unwesentlichen Zusammenhängen und die Einführung des Begriffs Zufall hat es der Mensch verstanden, trotz des Bestehens dieser objektiven Schranke in Wissenschaft und Technik große Fortschritte zu erzielen." (Mascheck 1986)

2.5.
Semantik

Allein die Tatsache, dass man über bestimmte Prozesse Informationen besitzt, bedeutet noch nicht, dass diese auch für jeden Zweck nutzbar sind. Der Prozess der Programmentwicklung ist ein fortschreitender Abstraktionsprozess, in dessen Verlauf viele anfangs vorhandene Informationen über die Daten verloren gehen. Ebenso bedeutet die Nutzung einen bestimmten Kommunikationsprotokolls nicht, dass der Entwickler dieses auch vollständig realisiert hat. Subsets von Standardprotokollen realisieren dann die für notwendig erachteten Teile des Protokolls, während auf andere keine oder unsinnige Reaktionen erfolgen. Gerade in der Kooperation zwischen verteilten Anwendungen kann das problematisch werden, wenn Komponenten ausgetauscht werden müssen, die gesamte Installation aber nicht erneuert werden soll oder kann. Nach Möglichkeiten die dadurch entstandene semantische Lücke durch Ontologien zu schließen, wird zurzeit gesucht und in ersten Machbarkeitsstudien am Beispiel eines Phasenumrichters für Fahrstühle (bereits) getestet. (Richter, V. & Rellin, S. 2011)

3.
Chancen durch Ressourcen

Wenn man unter Ressourcen allgemein Quellen, Mittel und Möglichkeiten versteht, die unser Handeln befördern, so kann man Resilienz einerseits als Ressource verstehen, andererseits benötigt Resilienz Ressourcen.

Ressourcen können in äußere und innere, bzw. strukturelle und persönliche unterschieden werden.

Zu den persönlichen Ressourcen gehört das Potenzial an mentalem und

emotionalem Wissen und Kompetenzen, die im Laufe eines Lebens erworben werden (s. auch: Fröhlich-Gildhoff, Rönnau-Böse 2009, 40 ff.: sechs Faktoren, die Resilienz unterstützen: Selbstwahrnehmung, Selbststeuerung, Selbstwirksamkeit, soziale Kompetenz, Umgang mit Stress und Problemlösen).

Zu den äußeren Ressourcen gehören Strukturen und Prozesse, die das Handeln zur Beförderung von Resilienz ebenso beeinflussen; diese sind z. B. betrieblicher, institutioneller, staatlicher aber auch globaler Natur.

Im Prozess zur Entwicklung und Erreichung von Usability, dem Usability-Engineering-Prozess, werden innere und äußere Ressourcen zur Entwicklung gebrauchstauglicher Artefakte genutzt. Gleichzeitig müssen diese Ressourcen erhalten und immer wieder neu aufgebaut werden im Sinne einer nachhaltigen Resilienz. Im Folgenden wird auf einige Möglichkeiten zur Förderung der Resilienz, in Bezug auf die Beeinflussbarkeit von inneren und äußeren Risikofaktoren durch Usability bzw. im Usability-Engineering-Prozess eingegangen.

3.1.
Ressource Mensch

Der Mensch als wichtigste Größe im Umgang mit Technik aber auch als unsicherer Faktor in Bezug auf seine funktionalen und ästhetischen Ansprüche steht in der Usability und im Usability Engineering im Mittelpunkt. Die subjektiven und emotionalen Bedürfnisse werden besonders im erweiterten Konzept der Usability, der User Experience (s. ISO 9241-210) berücksichtigt. Gleichzeitig wird damit schwer bzw. nicht quantifizierbaren Qualitäten eine Berechtigung einräumt. Dies wiederum vergrößert das Risikopotenzial der Gestaltung.

Im Usability-Engineering werden Versuche unternommen, durch interdisziplinäre Zusammenarbeit und in Berücksichtigung der Nutzerperspektive diese Risiken einzudämmen (s. a. user-centered, partizipatorisches Design), und Joy-of-Use bzw. positives Nutzungserleben zu ermöglichen. Im gemeinsamen Usability-Engineering-Prozess werden wichtige Ressourcen wie explizites und implizites Wissen in der Herstellung und im Umgang mit technischen Artefakten entwickelt.

3.2.
Hersteller- und Nutzerinteressen

Im Prozess der Entwicklung von Usability, der schon in der Phase der Konzeption gemeinsam mit dem Nutzer durchgeführt werden sollte, können Hersteller-, Entwickler- und Nutzerinteressen verbunden werden. Dem Prozess der Entfremdung (s. o. Risikofaktor Entfremdung) Hersteller-Nutzer bzw. Anwender kann entgegengewirkt werden.

Durch interdisziplinäre Zusammenarbeit im Usability-Engineering-Prozess mit ausgewiesener Nutzerzentrierung kann Komplexität besser beherrschbar gemacht werden, indem z. B. schon bei der Konzeption die von Entwicklern und Herstellern favorisierte Multifunktionalität von technischen Artefakten (s. Mobiltelefon) in optionale Angebote unter Berücksichtigung der gewünschten Hauptfunktionalität umgewandelt werden kann. Gut benutzbare und bedienfreundliche Produkte lassen sich besser verkaufen und die dazu erforderliche notwendige Vereinbarkeit von Hersteller- und Nutzerinteressen stellt sich als wichtiger wirtschaftlicher Faktor heraus.

3.3.
Fehlertest vs. Usabilitytest

Fehler in der Usability entpuppen sich häufig auch als technische System- bzw. Programmierfehler, z. B. wenn es Probleme mit der Bedienbarkeit eines (Fahrschein-) Automaten bzw. in der Navigation auf einer Website gibt.

Werden nun, wie laut ISO-Norm 9241-210 „Prozess zur Gestaltung gebrauchstauglicher Systeme“ gefordert, Tests in schon in sehr frühen Stadien anhand von Prototypen (z. B. Mock-Ups oder Struktur- bzw. Funktionsmodelle) durchgeführt, kann das zur Vermeidung ansonsten langfristig mitgeführter Fehler und damit zu geringerem Zeit- und Kostenaufwand in deren Behebung führen, besonders wenn auch Extremsituationen analysiert und berücksichtigt werden. Je mehr und besser Fehlertests in der Soft- und Hardwareentwicklung und Usability-Testverfahren miteinander verknüpft werden können, umso größer sind die Chancen einer nachhaltigen Resilienz.

3.4.
Fehlertoleranz im Usability-Engineering

Im Usability-Engineering werden in den Grundsätzen der Dialoggestaltung nach ISO 9241-110 sieben Grundeigenschaften technischer Systeme eingefordert. Neben Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Erwartungskonformität, Individualisierbarkeit, Lernförderlichkeit wird auch Fehlertoleranz eingefordert: Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand seitens des Benutzers erreicht werden kann.

Beispiel: Ein Eingabefeld erkennt eine fehlerhafte Eingabe automatisch und teilt dies dem Benutzer mit. Der Benutzer kann seine Arbeit fortsetzen. Der konstruktive Umgang mit Fehlern ist eine neue Anforderung an das Usability-Engineering. Resilienz kann Aging und semantischen Lücken entgegenwirken (s. o. Risikofaktoren Aging, Semantik), ihr kommt neben der Forderung nach grundsätzlicher Fehlervermeidung immer mehr Bedeutung zu.

4.
Fazit

Usabilty-Entwicklung und Maßnahmen zur Stärkung von Resilienz technischer Systeme stehen im direkten Zusammenhang: Usability und Usability-Engineering können technische Systeme nachhaltig stärken da Entwickler- und Nutzerperspektive verbunden und Fehler frühzeitig erkannt und behoben werden können. Grundsätzlich muss ein neues Verständnis

für auftretende Fehler und den Umgang mit ihnen entwickelt werden. Neben Vermeidungsstrategien (Prävention) muss ebenso Wert auf Management bestehender Krisen- bzw. Risikosituationen und auf Stabilisierung von Systemen (nachhaltiges Fehlermanagement) gelegt werden. Das kann durch einen konsequent durchgeführten Usability-Engineering-Prozess unterstützt werden. Dazu müssen auch parallel laufende Prozesse (Usability Test, Fehlertests) aufeinander abgestimmt werden.

Usability kann somit mit produktiven Fehlerstrategien zur Unterstützung von Resilienz ihren Wert als zentraler Wettbewerbsfaktor weiter erhöhen. Dazu muss das Resilienz-Engineering genau wie das Usability-Engineering grundsätzlicher Bestandteil jeder Entwicklung sein, da Resilienz als Eigenschaft nicht von selbst oder zufällig entsteht, sondern systematisch entwickelt werden muss.

Literatur

1. Fröhlich-Gildhoff K.; Rönnau-Böse M.(2009). Resilienz. München Basel: UTB Reinhardt-Verlag.
2. Mascheck, H.-J. (1986). Die Information als physikalische Größe. Vortrag unter http://www.h-j-mascheck.de/
3. Nielsen, J. (1994). Usability Engineering. Mountain View, California: Morgan Kaufmann
4. Richter, V. & Rellin, S. (2011). Ein System zur automatischen, ontologiebasierten Parametrierung von Feldgeräten. in 12. Nachwuchswissenschaftlerkonferenz mitteldeutscher Fachhochschulen. Wernigerode: Hochschule Harz
5. Schüttelkopf, E. M. (2008). Erfolgsstrategie Fehlerkultur. Wie Organisationen durch einen professionellen Umgang mit Fehlern ihre Performance optimieren. In: Ebner G., Heimerl P., Schüttelkopf, E. M. .Fehler – Lernen- Unternehmen: Wie Sie die Fehlerkultur und Lernreife Ihrer Organisation wahrnehmen und gestalten. Frankfurt (Main).
6. Spillner, A.& Linz, T. (2010). Basiswissen Softwaretest, Aus- und Weiterbildung zum Certified Tester Foundation Level nach ISTQB® -Standard. Heidelberg: dpunktverlag
7. Werner, E. E. (1971). The children of Kauai. A longitudinal study from the prenatal period to age ten. Honolulu: University of Hawaii Press
8. Wustmann C. (2004). Resilienz. Weinheim: Beltz.

Internetquelle

1. http://www.ursula-nuber.de/i/ursula_nuber_leseprobe_resilienz_schicksal.html [8.6.2011]