Konzeption von produktionsnahen Testumgebungen

Dr. Christian Brandes
christian.brandes@me.com

Zusammenfassung
Testumgebungen sind ein komplexes Thema, das in IT-Projekten häufig unteschätzt wird und so schnell zu einem Projektrisiko werden kann. Eine Ursache dafür ist der Mangel an Standards zur Beschreibung von Testumgebungen und die Unschärfe des vielgebrauchten Begriffs der *Produktionsnähe*.

Der vorliegende Artikel definiert erstmals den Produktionsnähe-Begriff, wodurch dieser zu einem zentralen Werkzeug für das Umgebungsmanagement wird, und stellt ein Schema vor, das zielgerichtet zu benötigten Testumgebungen führt und diese beschreibbar und vergleichbar macht.

Ausgangssituation
Gerade in großen Organisationen mit historisch gewachsenen heterogenen IT- und Umgebungs-Landschaften und fehlenden Vorgaben entwickeln sich Testumgebungen über die Jahre zu einem schwer durchschaubaren Dschungel. Das Thema selbst ist komplex und „zerfasert“ bei Diskussionen schnell in verschiedene Richtungen: Neben der verwendeten Hardware, Middleware und Software tauchen Aspekte wie Testdaten, Organisation, Datumssimulation, Benutzerkennungen, betriebliche Komponenten, Schnittstellen und Deployment auf.

Über allem schwebt dabei der pauschale Anspruch, dass Testumgebungen „so produktionsnah wie möglich“ sein sollen, um überhaupt aussagekräftige Testergebnisse zu ermöglichen. Was *genau* diese Forderung aber besagt - inwieweit etwa eine Umgebung A „produktionsnäher“ als eine andere Umgebung B ist - bleibt häufig nebulös. Eine Schattenseite davon ist, dass als Reaktion zuweilen versucht wird, Tests gleich in der produktiven Echtumgebung anzusiedeln. Das Motto „Nur das Echte ist das Wahre“ führt dann aber zu einem ineffizienten Testvorgehen, das insbesondere das zentrale Projektziel einer möglichst frühen Fehlerfindung völlig außer Acht lässt.

Umgebungen in IT-Standards
Möglicherweise ist es die hohe Komplexität des Themas, die dazu geführt hat, dass auch die bisherige Standardliteratur zum Softwaretest den Umgebungsbegriff tendenziell stiefmütterlich behandelt. In [3] wird die Definition des ISTQB-Glossars (siehe [1]) wiedergegeben, die ihrerseits die Norm IEEE610 referenziert: „Testumgebung: eine Umgebung, die benötigt wird, um Tests auszuführen. Sie umfasst Hardware, Instrumentierung, Simulatoren, Softwarewerkzeuge und andere unterstützende Hilfsmittel“. In [2] ist die Aussage zu finden, dass es im Wesentlichen drei Typen von Testumgebungen gibt: Labor, Systemtestumgebung und „produktionsnahe“ Umgebung.

All dies ist sicherlich korrekt, aber nicht konkret genug für das produktive Geschäft der Definition, der Governance und des Managements von Testumgebungen. (Die betrieblichen Standards wie etwa ITIL treffen übrigens ebenfalls keine operationalisierbaren Aussagen zu Testumgebungen, und selbst TPI als Reifegradmodell für Testprozesse sagt nichts darüber aus, wie „gut“ vorhandene Testumgebungen an die Erfordernisse angepasst sind.)

Produktionsnähe als zentraler Begriff
Den ersten Schritt zur Behebung des Problems bildet die Schaffung einer tragfähigen Definition des Produktionsnähe-Begriffs. Ausgangspunkt hierfür ist eine Beobachtung, die sich an einem Beispiel gut illustrieren lässt: So ist etwa die Umgebungs-Eigenschaft „Bandbreite des Netzwerks, in dem ein Test-PC angesiedelt ist“ für einen Ergonomietest oder einen funktionalen Komponententest völlig irrelevant; für einen Performancetest hingegen ist genau das Gegenteil der Fall!

Einzelne Umgebungsmerkmale können also in einem Fall Teil einer geforderten „Produktionsnähe“ sein, in einem anderen hingegen nicht. Ursache hierfür ist letztlich der *Zweck*, der hinter der jeweiligen Testaktivität steht. Dies führt zu folgender Definition:

> **Produktionsnähe** bezeichnet diejenigen Merkmale einer Testumgebung, die so wie in der Produktionsumgebung ausgestaltet sein müssen, um das in der Testumgebung angestrebte **Testziel** zu erreichen.

„Testziel“ wird hierbei ganz im ISTQB-Sinne verwendet (siehe [1]).

Die Testziele an den Anfang der Betrachtung zu setzen führt nun zu folgendem Vorgehen: Für jede Testaktivität und das damit verbundene Testziel wird die benötigte Produktionsnähe ermittelt. Dies wiederum erlaubt, benötigte Umgebungen zu identifizieren und ggf. zusammenzulegen und ein zweckgetriebenes Mapping zwischen Tests und Umgebungen aufzubauen.

Ein solches Vorgehen erweist sich als tragfähiger als die umgekehrte Richtung, bei der zuerst Umgebungen aufgebaut werden und man sich im Anschluss fragt, welche Aktivitäten man denn dort nun ansiedeln könnte. Letzteres öffnet einem Wildwuchs in der Umgebungsnutzung Tür und Tor - überspitzt gesagt getreu dem Motto: Wir hätten hier eine Lösung, hat jemand vielleicht ein passendes Problem dafür?

Vorgehen: Vom Testziel über die benötigte Produktionsnähe zur Testumgebung
Wie kann man dieses Vorgehen konkret realisieren? Als Erstes versammelt man alle durchzuführenden *Testaktivitäten* und ergänzt diese um das jeweilige *Testziel*. Im Bei-

spiel in der folgenden Tabelle werden dabei bewusst nicht nur Testaktivitäten und Testumgebungen aufgelistet, sondern u.a. auch Produktivbetrieb und Schulung, um das entstehende Bild abzurunden.

(Test-)Aktivität	(Test-)Ziel
Evaluieren	Freies Ausprobieren
Entwickeln	Codeerstellung
Komponententest	frühe Fehlerfindung hinsichtl. Fkt., Robustheit, Performanz
Komponentenintegrationstest	Test der Schnittstellen und komp.übergreifender Fkt.
Fachlicher Systemtest	Test aus Anwendersicht (Fkt., Robustheit, erste Schnittstellen)
Last- und Performanztest	Test des Gesamtsystems auf Perf., Ressourcenverbrauch
Bedienbarkeitstest	Prüfung der Ergonomie, Barrierefreiheit
Fachlicher Abnahmetest	Vertrauensbildung, Nachweis der Erfüllung der Abnahmekriterien
Technischer Abnahmetest	Test gegen betriebliche Anforderungen
Systemintegrationstest	End-to-End-Test wichtiger (Geschäfts-)Prozesse
Betrieb	Produktiver Echtbetrieb
Schulung	Schulung nächster und aktueller Versionen
Fehleranalyse und Re-Test	Nachstellen komplexer Produktionsfehler, Korrekturnachweis

Im zweiten Schritt benötigt man eine Sammlung aller zu betrachtenden *Umgebungsmerkmale*. Diese kann pragmatisch entstehen und mit der Zeit wachsen; es kann sich aber auch lohnen, eine eventuell vorhandene CMDB (Configuration Management Database) heranzuziehen. Neben der Vollständigkeit ist vor allem die *Granularität* der Merkmale sicherlich diskussionswürdig und muss im konkreten Einzelfall festgelegt werden. Entscheidend ist, dass mittelfristig alle relevanten Anforderungen an Umgebungen mit dem entstehenden Schema erfasst werden können. Ein Beispiel für eine Merkmals-Sammlung ist in der nachfolgenden Auflistung zu finden:

- Hardware
  - Server
  - Client
  - Storage
  - Sizing
  - Peripherie
  - Netzwerk
- Software
  - OS Server
  - OS Client
  - Datenbanken
  - Testobjekt-Konfiguration
- Middleware
  - Application Server
  - Enterprise Service Bus
  - Batch Controller
  - Load Balancer
- Daten
  - Basisdaten (z.B. PLZ-Verzeichnisse)
  - Historiendaten (z.B. Kundendatenbestände)
  - Menge
  - Aktualität
- Sicherheit
  - Single Sign On
  - Authentification Server
  - Verschlüsselung
  - Zertifikate
- Administration
  - Deployment
  - Wartungszeiten
  - Nutzerkennungen
- Sonstige
  - interne Partnersysteme
  - externe Partnersysteme
  - Mandantenfähigkeit
  - Barrierehilfen
  - Sprachdienste
  - Laufzeitbibliotheken
  - Datumssimulation.

Setzt man die Informationen aus beiden Schritten - angestrebte Ziele und Umgebungs-Merkmale - nun zusammen, entsteht eine *Matrix*, die es erlaubt, die *benötigte Produktionsnähe pro Aktivität* einfach zu beschreiben. Ein Kreuz in der Matrix bedeutet dabei (eingedenk der Definition von Produktionsnähe): Das jeweilige Merkmal *muss* wie im produktiven Betrieb ausgeprägt sein. Wie eine solche Befüllung beispielsweise aussehen kann, ist in Abbildung 1 zu sehen. Die Zeile „Betrieb“ selbst bezeichnet im Beispiel die Produktionsumgebung und trägt folglich als Referenz überall ein Kreuz.

Identifikation mehrfach nutzbarer Umgebungen und ihrer Eigenschaften

Nach Befüllung der Tabelle durch die jeweiligen Stakeholder lassen sich gleichartige Zeilenbefüllungen zusammenfassen (Clustering) und so die benötigten Umgebungen und ihre Eigenschaften (Obermengen von angekreuzten Umgebungsmerkmalen) ableiten. Als Ergebnis können z.B. Umgebungen entstehen wie:

	HARDWARE						SOFTWARE			MIDDLEWARE			DATEN				SICHERHEIT				ADMINISTR.			SONSTIGE					
AKTIVITÄT	Server	Client	Storage	Sizing (Serv., Storg.)	Peripherie	Netzwerk	OS Server	OS Client	DB	WLS	ESB	Load-Bal.	Basis-daten	Historien-daten	Menge	Aktualität	SSO	Auth.-server	SSL	Serv.-zertifikate	Deployment	Warnungs-zeiten	Kennungen	int. Partner	ext. Partner	Mandant.-fähgkt.	Barriere-hilfen	Runtime-bibliot.	Datums-simu.
Entwickeln	o	o	o	o	o	o	o	x	o	x	o	o	o	o	o	o	?	?	?	?	o	o	o	o	o	o	x	x	o
Unittest	o	o	o	o	o	o	o	x	o	x	o	o	o	o	o	o	?	?	?	?	o	o	o	o	o	o	-	x	o
Integrationstest	o	o	o	o	o	o	o	x	o	x	o	o	o	o	o	o	?	?	?	?	o	o	o	o	o	o	-	x	o
Fachlicher Systemtest	o	o	o	o	o	o	o	x	o	o	o	o	o	o	o	o	?	?	?	?	o	o	o	o	o	?	-	x	o
Last& Perf.-Test	x	o	x	x	o	x	x	x	x	x	x	x	o	o	x	o	x	x	x	x	x	o	x	x	-	?	o	x	o
Usabilitytest	o	o	o	o	x	o	o	x	o	o	o	o	x	o	o	o	?	?	?	?	o	o	x	o	o	o	o	o	o
Test auf Barrierefreiheit	o	x	o	o	x	x	x	x	o	o	o	o	o	o	o	o	?	?	?	?	?	o	x	o	o	o	x	o	o
Fachl. Abnahme	o	o	o	o	o	o	o	x	o	o	o	o	x	o	o	o	?	?	?	?	o	o	o	x	o	?	?	o	o
Techn. Abnahme																													
Systemint.-test	o	o	o	o	o	o	x	x	o	o	o	o	o	o	o	o	?	?	?	?	?	o	x	x	x	?	o	?	x
Betrieb	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Schulung	o	o	o	o	x	o	o	x	o	o	o	o	o	o	o	o	o	o	o	o	o	x	x	x	-	?	x	o	o
Fehleranalyse & Re-Test	x	x	o	o	x	x	x	x	x	x	x	x	x	x	o	x	x	x	x	x	x	o	x	x	x	x	o	x	o

Abbildung 1: Befüllung der Produktionsnähe-Matrix (Beispiel)

- Entwicklungsumgebung
- Integrationsumgebung
- Produktionsumgebung
- Produktionsspiegelung.

Dabei sollte der Aspekt bedacht werden, dass hier über Merkmale physikalischer Umgebungen gesprochen wird. Eine physikalische Umgebung ist Grundlage für i.A. mehrere logische Umgebungen, die parallel oder zeitversetzt in derselben physikalischen Umgebung angesiedelt sind (z.B. eine Integrationsumgebung, die zeitweilig als Umgebung für Abnahmetests und danach für Systemintegrationstests genutzt wird). Auch das Thema Virtualisierung lässt sich in das vorgestellte Schema aufnehmen, denn es verträgt sich mit der Grundfrage: Gefährdet der Einsatz virtueller Maschinen das jeweilige Testziel?

Zuletzt gilt es, den wichtigsten Motor zur Vervielfältigung von Testumgebungen zu bedenken: die Notwendigkeit, parallel verschiedene *Versionsstände* einer Umgebung zur Verfügung stellen zu müssen. Während die meisten Testaktivitäten auf die nächste zu veröffentlichende Version eines Produkts abzielen, entsteht durch die Analyse von Produktionsfehlern oder durch zusätzliche Migrationsaktivitäten (z.B. Datenbank-Upgrades) regelmäßig der Bedarf nach Umgebungen mit den aktuell ausgerollten Versionen. Diese Anforderung muss in der Gesamtkonzeption entsprechend berücksichtigt werden.

Erreichtes
Die vorgestellte Tabelle bildet die Grundlage für Nutzungsregeln, Management, Beplanung und Governance von Testumgebungen. Sie bietet zwei Leserichtungen an: Die eine beantwortet die Frage „WAS wird WO getestet?“, die andere definiert die Menge aller von einer Umgebung zu erfüllenden Anforderungen. (In [2] ist eine im Ansatz ähnliche Tabelle zu finden, die die Themen „Testart - Qualitätsmerkmal - Umgebungstyp“ zueinander in Beziehung setzt, aber noch keine detaillierte Beschreibung der Umgebungsmerkmale vorsieht.)

Sie erlaubt auch die Beantwortung von Detailfragen: Wird etwa der Einsatz von Automatisierungswerkzeugen (Testrobotern) in Integrationsumgebungen mit der Begründung abgelehnt, diese Umgebung solle ja schließlich „produktionsnah“ sein, fällt das Argument nun in sich zusammen, wenn der Einsatz eines Testroboters das konkrete Testziel nicht gefährdet.

Und sie gestattet die zielgenaue Ansiedlung zukünftiger Testaktivitäten auf der dafür geeignetsten Umgebung: Taucht etwa eine Anfrage der Form „Ich benötige eine Umgebung für (...)“ auf, werden ganz analog Testziel und benötigte Produktionsnähe ermittelt. Als Ergebnis ist es ein Leichtes, anhand des entstandenen Profils die geeignete Umgebung zuzuweisen.

Fazit
Mit diesem Schema werden Testumgebungen handhabbar, beschreibbar und vergleichbar. Auch neu auftauchende Anfragen nach zusätzlichen Umgebungen lassen sich fundiert und systematisch klären. Darüber hinaus schafft es ein einheitliches Zielbild, Verständnis und Vokabular bei allen Beteiligten. Nicht zuletzt erfährt es gute Akzeptanz, da es vergleichsweise elementar ist, aber trotzdem eine Chance bietet, unterschiedliche oder diffuse Vorstellungen zu Testumgebungen gemeinsam und verständlich zu dokumentieren.

Literatur

[1] ISTQB-Glossar für Testbegriffe (Version 2.1), www.istqb.org/display/ISTQB/Glossary+of+Terms

[2] Pol, M.; Koomen, T.; Spillner, A.: „Management und Optimierung des Testprozesses“, dpunkt-Verlag, 2. Auflage, 2002 (nur noch als e-Book erhältlich, siehe: www.dpunkt.de/buecher/3081.html)

[3] Spillner, A.; Linz, T.: „Basiswissen Softwaretest“, dpunkt-Verlag, 4. Auflage, 2010

Herzlichen Dank an Prof. A. Spillner für den Austausch zum Thema und die Unterstützung.