Dr. Carsten Weise

(carsten.weise@ivu.de)
arbeitet bei der IVU Traffic Technologies AG als Teamleiter in der Produktentwicklung und ist dort für den Bordrechner zuständig. Er hat umfangreiche akademische wie industrielle Erfahrung mit den Schwerpunkten Spezifikation, Entwicklung, Test, Verifikation und Qualitätssicherung eingebetteter Systeme.
Dr. Weise ist Mitglied im Programmkomitee renommierter Testkonferenzen, wie z. B. ICTSS oder TAICPART.

Testumgebungen für eingebettete Systeme im Griff

Eingebettete Systeme unterscheiden sich von herkömmlichen Softwaresystemen durch den physikalischen Kontext – das sind die zu messenden und zu steuernden Geräte – in den sie eingebettet sind. Beim Systemtest führt dieser physikalische Kontext zu grundlegend anderen Anforderungen an die Testumgebung als im Falle reiner Softwaresysteme. Unterschätzt man die Komplexität der Testumgebung oder fehlt ein strukturierter Prozess für das Management von Testumgebungen, so kann dies zu erhöhten Kosten und Projektverzögerungen führen.

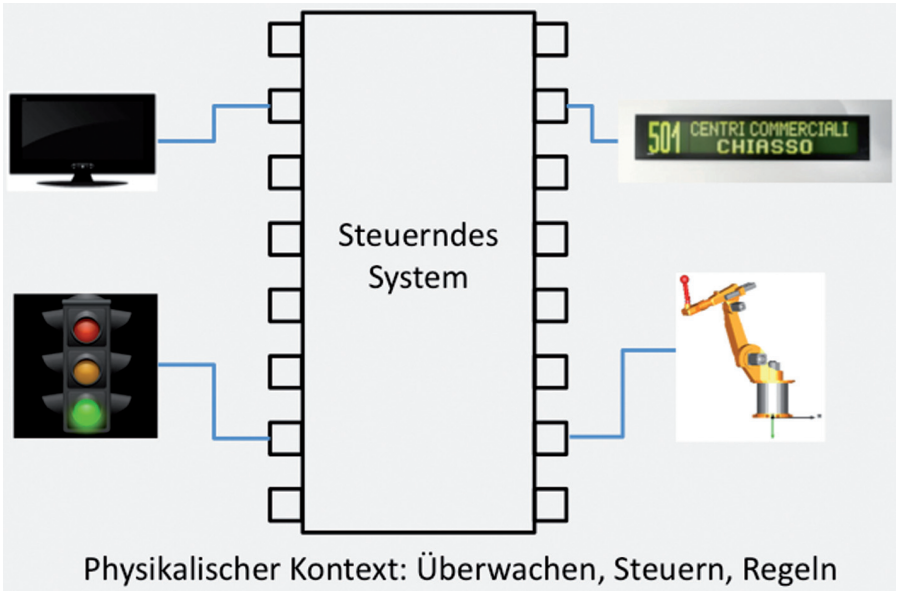


Abb. 1: Ein eingebettetes System existiert in einem Kontext.

Im Folgenden werden die Komplexität und Problematik von Testumgebungen eingebetteter Systeme an mehreren (vereinfachten) Praxisbeispielen illustriert und Maßnahmen für die Definition von Prozessen zum Umgang mit diesen Testumgebungen vorgestellt.

Testinfrastruktur und Testumgebung

Wesentlich für das Gelingen des Systemtests ist die *Testinfrastruktur*. Hierbei handelt es sich um die für die Testausführung benötigten organisatorischen Artefakte, z. B. Testbeschreibungen, Testberichte oder Fehlerberichte.

Zur Testinfrastruktur gehört als wesentlicher Bestandteil die *Testumgebung*. Diese wird zum Ausführen von Tests benötigt. Sie umfasst Hardware, Instrumentierung, Simulatoren, Softwarewerkzeuge und andere unterstützende Hilfsmittel (nach IEEE 610/ISTQB).

Die Unterscheidung zwischen *System under Test* (SUT), Testumgebung und Testinfrastruktur ist dabei nicht immer scharf.

Anwendungsbeispiele für Testumgebungen

Die passende Testumgebung hängt nicht allein vom Testobjekt ab, sondern auch von der Teststrategie und dem Testzweck: welche Funktionalität soll getestet werden, welche Testabdeckung soll erreicht werden? Für den manuellen Test eines einfachen Computerspiels reicht unter Umständen als Testumgebung der Tester selbst aus. Ist das SUT ein Webclient, ist sicherlich der Webserver Teil der Testumgebung. Bei eingebetteten Systemen gehören typischerweise alle Komponenten des physikalischen Kontextes sowie die Schnittstellen des zu steuernden Systems dazu.

Set Top Box

In diesem Beispiel ist eine einfache Set Top Box mit einem digitalen Satellitenreceiver

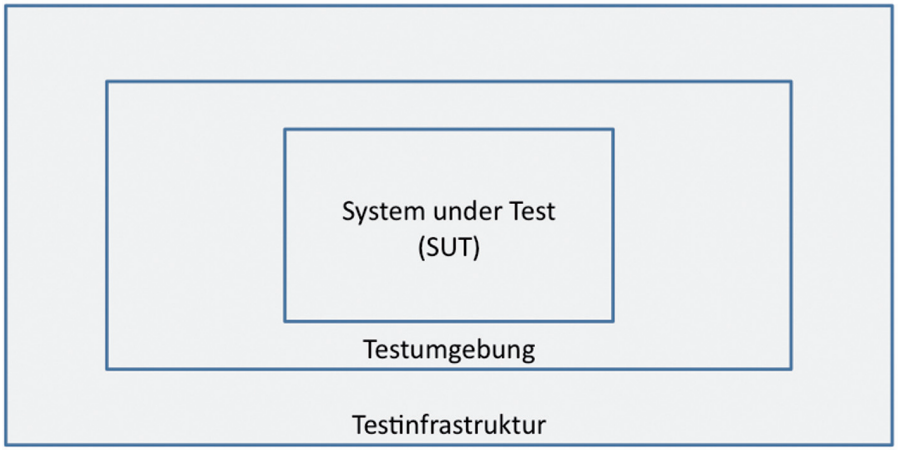


Abb 2: Die Testumgebung gehört zur Testinfrastruktur.

ausgestattet, der das Satellitensignal in geeigneter Form an einen Fernseher weiterleitet. Die dazu benötigten Schnittstellen der Set Top Box sind ein Koaxialeingang und ein SCART-Stecker als Ausgang. Zwischen den einzelnen Sendern kann über eine einfache Infrarot-Fernbedienung umgeschaltet werden.

Das SUT ist die auf der Set Top Box befindliche Software. Das *Device under Test* (DUT) ist die Set Top Box selbst. Zur Testumgebung gehören das Wiedergabegerät (Fernseher), die Fernbedienung und die Signalquelle. Diese einfache Testumgebung kann leicht im Labor nachgebildet werden.

Im Detail besteht jedoch bei der Signalquelle ein Entscheidungsbedarf: verwendet man ein vorhandenes Satellitensignal, das über Parabolantenne eingespeist wird, oder besser ein aufgezeichnetes Signal? Ersteres hat höhere Kosten und einen größeren Platzbedarf, ist aber eventuell einfacher verfügbar. Zweiteres besitzt möglicherweise nicht alle für den Test erforderlichen Eigenschaften, ermöglicht dafür aber die wiederholte Ausführung des Tests. Welche Entscheidung hier die richtige ist, hängt von vielen weiteren Faktoren ab – für den Aufbau einer Testumgebung muss aber eine Wahl getroffen werden.

Für diese einfache Anwendung besteht die Testumgebung somit aus drei Geräten (Wiedergabegerät, Signalquelle, Fernbedienung) und der erforderlichen Verkabelung (Koaxialkabel, SCART-Kabel). Alle Geräte benötigen zudem eine Stromversorgung. Hier sind ein herkömmlicher Hausanschluss und für die Fernbedienung ein Satz Batterien ausreichend. Abbildung 3 zeigt den Aufbau der Testumgebung und eine entsprechende Stückliste.

Vermittlungsstelle

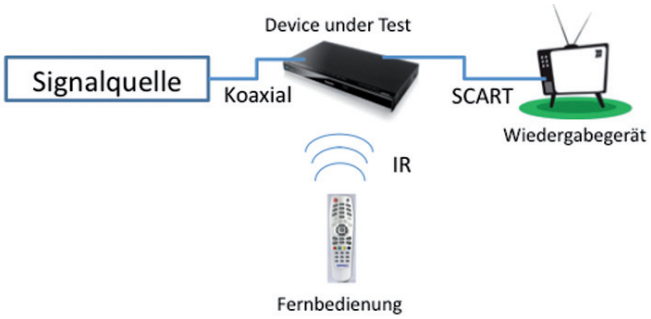
Eine Vermittlungsstelle für den Mobilfunk (MSC = Mobile Switching Center) ist ein weiteres, komplexeres Beispiel für ein eingebettetes System. Die Testumgebung besteht hier aus einem kompletten Mobilfunknetzwerk, den Komponenten für die Übertragung auf dem Funkweg (Basisstationen) sowie den Endgeräten und ihren Benutzern. Für einfache Testfälle wie „A calls B“ reichen zwei Benutzer und entsprechend zwei Basisstationen aus. Für komplexere Szenarien wie dem Handover (Wechsel einer Funkzelle) und Roaming (Wechsel des Netzbetreibers) benötigt man aufwändigere Testumgebungen.

Abbildung 4 zeigt die Komponenten der Testumgebung für die Vermittlungsschnittstelle. Das DUT ist blau eingezeichnet. Die weißen Komponenten gehören zum kabelgebundenen Core Network, das die Verbindung der Vermittlungsstelle zur Außenwelt darstellt. Gelb sind die Komponenten dargestellt, die Funkverbindungen mit den Endgeräten erlauben (Basisstationen). Grün sind erforderliche Endgeräte. Bei den Bezeichnungen in den Komponenten handelt es sich um die Namen der Netzwerkknoten, bei den Bezeichnungen an den Verbindungen um die Schnittstellenprotokolle.

Tests in einem solchen System sind nur aussagekräftig, wenn es unter einer gewissen Last steht. Diese wird in der Testumgebung durch einen Lastgenerator (orange) erzeugt.

Die oben abgebildete Testumgebung ist bereits relativ komplex, kann aber so im Labor aufgebaut werden. Die kabelgebundenen Schnittstellen stellen dabei kein Problem dar, bei den Luftschnittstellen zu den Basisstationen muss jedoch abgewogen werden, ob man diese durch Koaxialkabel ersetzt. Dies kann z. B. sinnvoll sein, wenn Notrufe getestet werden.

Mit dieser Testumgebung soll insbesondere illustriert werden, dass der Aufbau



Lfd. Nr.	Bezeichnung
1	Set Top Box
2	Wiedergabegerät
3	Signalquelle
4	Koaxialkabel
5	SCART-Kabel
6	Fernbedienung
7	Netzkabel Set Top Box
8	Netzkabel Wiedergabegerät
9	Netzkabel Signalquelle
10	Batterien Fernbedienung

Abb. 3: Testumgebung Set Top Box: Aufbau und Stückliste.

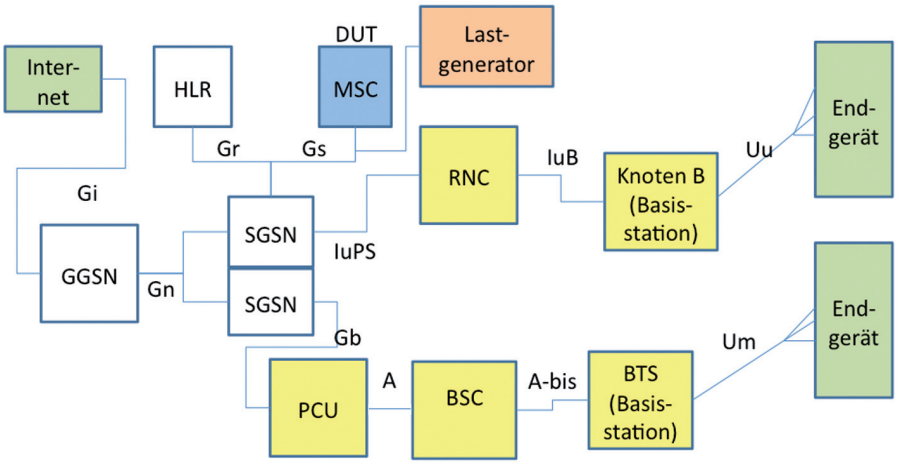


Abb. 4: Testumgebung für eine mobile Vermittlungsschnittstelle

einer solchen Umgebung sowohl Zeit als auch komponentenspezifische Kompetenz erfordert. Auch müssen Entscheidungen getroffen werden, wie naturgetreu der physikalische Kontext nachgebildet wird bzw. inwieweit Teilsysteme abstrahiert werden können. Ebenso ist die Konfigurierbarkeit und Steuerbarkeit der Komponenten einer Testumgebung wesentlich für gezielte Tests.

Bordrechner

Ein weiteres Beispiel zeigt die Nutzung von Simulatoren in Testumgebungen: Im öffentlichen Personenverkehr dienen Bordrechner neben dem Fahrkartenverkauf einerseits dazu, einer Zentrale Auskunft über den momentanen Zustand der Fahrzeugflotte zu geben, andererseits dazu, dem Fahrer die momentane Fahrplanlage mitzuteilen. Hierzu ermitteln moderne Bordrechner die aktuelle Fahrzeugposition entlang des Fahrwegs mittels GPS und leiten diese periodisch an die Zentrale weiter. Da Fahrzeuge mobile Komponenten des Gesamtsystems sind, muss die Kommunikation mit der Zentrale drahtlos geschehen.

Bordrechner steuern weiterhin viele Vorgänge im Bus, wie z. B. die Innen- und Außenbeschilderungen, die Haltestellendurchsagen und die Eingangstüren, oder sie beeinflussen die Ampel auf dem Fahrweg. Eine realistische Testumgebung für einen Bordrechner wäre ein voll ausgerüsteter Bus, der sich bewegt, damit ein sich veränderndes GPS-Signal empfangen werden kann.

Statt eines fahrenden Busses verwendet man im Testlabor spezielle Hardware, die die Busumgebung simuliert, z. B. ein einfacher Schalter für das Öffnen und Schließen der Türen. Die Innenanzeigen hingegen kann man auch im Testlabor direkt an den Bordrechner anschließen. Ein veränderliches GPS-Signal kann per Simulation in das SUT eingespielt werden.

Weiterhin benötigt man eine simulierte Zentrale für die Testumgebung. Die Verbindung kann hierbei naturgetreu, z. B. über GPRS, nachgestellt werden oder über ein IP-Netzwerk simuliert werden.

Wie im Beispiel zuvor handelt es sich um eine komplexe Testumgebung, bei der ein gutes Konzept hilfreich ist. Entscheidungen, wie Komponenten des physikalischen Kontextes simuliert werden, sind notwendig. Nicht zu vernachlässigen ist die Konfiguration der Testumgebung, der beispielsweise ein konsistenter Fahrplan zur Verfügung gestellt werden muss. Ausführliche Informationen zur Komplexität der Software des öffentlichen Nahverkehrs findet man in [Sch11].

Stellwerk

Das letzte Beispiel stammt aus der Großtechnik. Steuerungssoftware ist heutzutage, z. B. in verfahrenstechnischen Anlagen, nicht mehr wegzudenken. Als stellvertretendes Beispiel benutzen wir hier das Steuerungsgerät eines Stellwerks.

Bei einem Bahnübergang sorgt ein Stellwerk dafür, dass die Schranken zum richtigen Zeitpunkt geschlossen sind. In einem einfachen Bahnhof sorgt das Stellwerk dafür, dass Gleisabschnitte nicht von mehreren Zügen gleichzeitig befahren werden können, und stellt entsprechend die Fahrtsignale auf Rot bzw. Grün. Hierzu muss ein Stellwerk die Zugbewegungen detektieren können. Dies geschieht durch so genannte Achszähler.

Die Testumgebung eines Stellwerks ist somit ein Bahnübergang bzw. Bahnhof. Kann man das Innenleben eines Busses im Prinzip noch 1-zu-1 im Labor nachbauen, so ist dies bei einem Bahnhof nicht mehr möglich. Auch ein fahrender Bus könnte unter bestimmten Bedingungen zum Testen genutzt werden; es ist jedoch fast ausgeschlossen, mit einem existierenden Bahnhof zu testen.

Deshalb stellt sich bei einer solchen Testumgebung die Frage nach einer geeigneten Nachstellung des physikalischen Kontextes im Labor. Eine Möglichkeit bieten so genannte Gleisbildstelltische, auf denen in Miniatur das Gleisbild des Bahnhofs, einschließlich der zu steuernden Komponenten, hardwaremäßig nachgestellt wird. Der Vorteil bei einer solchen hardwaremäßigen Nachbildung ist, dass die Schnittstelle und die Software der Peripherie dabei wiederverwendet werden kann.

Noch günstiger ist eine Softwaresimulation des Bahnhofs. Hierbei ist jedoch zu beachten, dass die Protokolle aller Peripheriekomponenten in der Software nachgebildet werden müssen. Der Aufwand für die Entwicklung der Simulation ist also deutlich höher als für eine hardwaremäßige Nachbildung, die späteren Betriebskosten sind dafür jedoch geringer. Weitere Informationen zum Testen von Stellwerken findet man in [Mit09].

Umgang mit den Testumgebungen

In den vorangegangenen Beispielen wurde illustriert, dass Testumgebungen für eingebettete Systeme komplex sein können. Die Erfahrung zeigt, dass Aufwand, Komplexität und Kosten solcher Testumgebungen oft unterschätzt werden und sich erst durch langjährige Erfahrung geeignete Prozesse zum effizienten Aufbau sowie zur effizienten Nutzung und Wartung der Testumgebungen ergeben. Nachfolgend werden die Maßnahmen beschrieben, mit denen man diese Probleme in den Griff bekommen kann.

Konzeption einer Testumgebung

Eine Testumgebung sollte geplant werden, d. h., es sollte ein geeignetes Konzept für sie erstellt werden. Die Testumgebung soll sicherlich so realitätsnah wie möglich sein, aber auch kostengünstig in Aufbau und Bedienung. Die Verwendung von physikalischen Geräten der tatsächlichen Anwen-

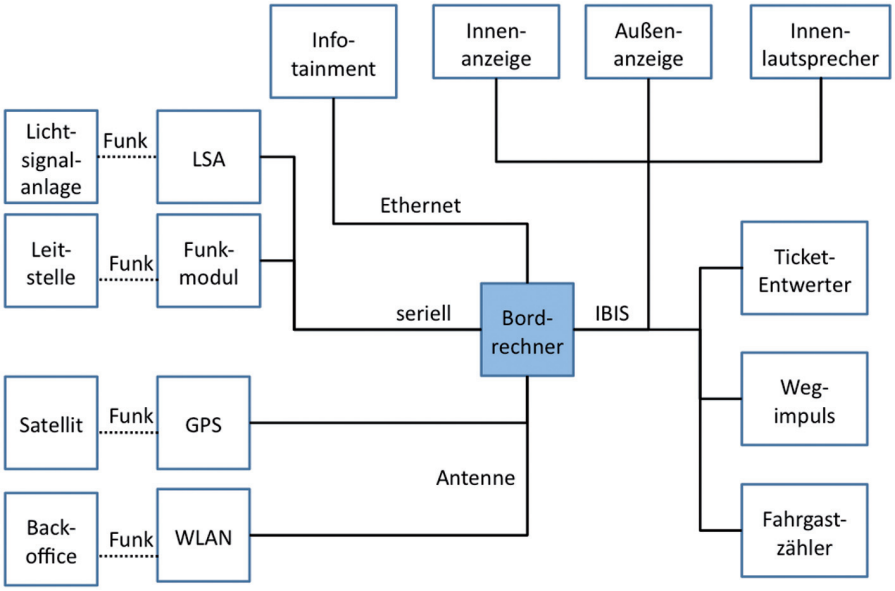


Abb. 5: Kontext eines Bordrechners.

dungsumgebung ist immer dann sinnvoll, wenn dies in Anschaffung, Benutzung, Lagerung und Wartung günstiger ist als eine entsprechende Soft- oder Hardwaresimulation. Ausschlaggebend sind hierbei auch Verfügbarkeit und Platzbedarf.

Simulationen der Komponenten, insbesondere wenn sie zum Großteil softwarebasiert sind, sind in der Regel kostengünstiger als die Verwendung der physikalischen Geräte. Hierbei muss jedoch auch bedacht werden, wie stark dabei vom eigentlichen Gerät abstrahiert wird und ob diese Abstraktion noch dem Testzweck entspricht. Oft ist es ratsam, die Hardwareschnittstellen des DUT tatsächlich zu verwenden, bevor man in die Zwickmühle gerät, ob der Testaufbau das SUT oder etwa die Simulation der Peripherie testet.

Das Ersetzen der Luftschnittstelle durch Koaxialkabel ist für viele Funktionstests sinnvoll und kostengünstig. Störungen der Luftschnittstelle können aber auf diese Weise nicht mehr getestet werden. Die Entscheidungen hängen also auch hier von den Testzwecken ab.

Bei einer Gerätesimulation ist genau darauf zu achten, von welchen Aspekten der Geräte abstrahiert wird. Für Ende-Ende-Tests mag die Abstraktion zu stark sein, sodass Fehler dann erst im Feld und nicht im Systemtest auftreten.

Architektur der Testumgebung

Basierend auf dem Konzept für die Testumgebung wird ihre Architektur festgelegt und dokumentiert. Wichtige Hilfsmittel sind dabei Stücklisten und die grafische Modellierung der Architektur, wie in den Abbildungen zu den Testumgebungen dargestellt.

Bei der Stückliste ist darauf zu achten, dass die Testumgebung nicht nur aus ihren Komponenten besteht, sondern die Schnittstellen zwischen den Komponenten einen wesentlichen Anteil ausmachen. Die Stückliste muss Auskunft über Hard- und Software von Komponenten und Schnittstellen geben sowie gegebenenfalls die korrekte Konfiguration der Teile beschreiben.

Auch bei der Stückliste stellt sich die Frage nach dem Abstraktionsgrad der Testumgebungsbeschreibung: wo hört die Testumgebung auf, und wo beginnt die betriebliche Infrastruktur? Muss z. B. die Stromversorgung der Komponenten gewährleistet oder dokumentiert werden? Diese Fragen können nur in Abhängigkeit von den betrieblichen Vorgängen beantwortet werden.

Stückliste und Architekturbilder sind statisch, d. h., zusätzlich muss auch der Aufbau der Testumgebung dokumentiert werden. Bei allen diesen Dokumenten muss viel Sorgfalt auf Vollständigkeit und Korrektheit verwendet werden.

Aufbau der Testumgebung

Für einen erfolgreichen und termingerechten Aufbau der Testumgebung müssen klare Vorstellungen über den Zeitaufwand und die benötigten Kompetenzen existieren.

Der Aufbau der Testumgebung sollte abgeschlossen sein, bevor der Integrationstest beginnt. In der Praxis erlebt man jedoch oft, dass Tester die Testumgebung als Teil des Systemtests aufbauen. Dies ist wohl darauf zurückzuführen, dass Tester eine klare Zielvorstellung vom Testaufbau haben, ihnen aber häufig die Zeit und Kompetenz für den Testaufbau fehlen. Effizienter ist es, den Testaufbau vor dem Systemtest von entsprechend geschultem Personal durchführen zu lassen. Dafür ist es notwendig, dass die Zielvorstellung (auf Basis der Testzwecke) frühzeitig gefunden wird und bereits in Konzept und Architektur einfließt.

Diese Aufgabe ist meist nicht so einfach im Vorfeld am grünen Tisch zu bewerkstelligen. Deshalb ist es sinnvoll, während des Testaufbaus die entsprechenden Zielvorstellungen sowie das Konzept und die Architektur immer wieder abzugleichen und neu gewonnene Erkenntnisse während des Aufbaus zu dokumentieren. Ein First Time Right sollte man hier nicht erwarten, sondern stattdessen die Zeit während des Aufbaus nutzen, um die Natur der Testumgebung besser verstehen zu lernen.

Für den Aufbau der Testumgebung ist die Verfügbarkeit der Komponenten eine wichtige Voraussetzung. Werden Komponenten der Testumgebung wiederverwendet, stellt ein Belegungsplan der Komponenten ihre Verfügbarkeit sicher.

Der Aufbau der Testumgebung wird durch eine Inbetriebnahme abgeschlossen. Zumindest mit einem einfachen Smoke-Test sollte sichergestellt werden, dass die Testumgebung prinzipiell richtig aufgebaut ist. Selbst dafür ist jedoch eine korrekte Konfiguration der Testumgebung Voraussetzung. Für diese Aufgabe muss entsprechende Kompetenz eingeplant werden. Es gilt: je komplexer die Testumgebung, desto größer ist die Wahrscheinlichkeit, dass es nur wenige entsprechend kompetente Personen gibt.

Nutzung der Testumgebung

Eine gut konzipierte und geplante Testumgebung kann nach erfolgreichem Aufbau und Inbetriebnahme problemlos zur Testausführung verwendet werden. Während der Nutzung beobachtete Mängel sowie Erweiterungs- und Verbesserungsmöglichkeiten sollten dokumentiert werden, um diese Beobachtungen bei der Planung der nächsten Testumgebung berücksichtigen zu können („Lessons learned“).

Konservierung und Wartung der Testumgebung

Nach Abschluss der Testaktivitäten muss die Testumgebung eventuell für spätere Nachtests konserviert werden. Im Falle sicherheitskritischer Anwendungen kann dies tatsächlich bedeuten, dass die gesamte Testumgebung im verwendeten Zustand eingefroren und aufbewahrt werden muss.

Die meisten Unternehmen haben aber nicht diese Notwendigkeit, sondern sind an einer Wiederverwendung der Komponenten der Testumgebung in anderen Testaktivitäten interessiert. In diesem Fall muss der Ist-Zustand der Testumgebung mit dem originären Planungszustand verglichen werden, und Abweichungen müssen dokumentiert werden. Nur dadurch kann gewährleistet werden, dass die durchgeführten Tests auch in der Wiederholung gelingen, insbesondere wenn die Architektur der Testumgebung während der Testausführung verändert wurde.

Essenziell ist auch die Frage nach der Aufbewahrung und Wartung von Komponenten der Testumgebung zwischen den Einsätzen. Ohne entsprechende Vorkehrungen geht man das Risiko ein, dass die Testumgebung beim nächsten Einsatz nicht mehr vollständig ist bzw. wichtige Updates nicht stattgefunden haben. Infolgedessen erhöht sich der Aufwand für die Inbetriebnahme.

Abschließendes

Anhand einiger (vereinfachter) Beispiele aus der Praxis wurde illustriert, wie komplex Testumgebungen für eingebettete Systeme oftmals sind. Ohne entsprechende Prozesse sind erarbeitete Testumgebungen häufig ungeeignet oder unvollständig und ihr Aufbau, ihre Inbetriebnahme und Nutzung führen zu hohen Kosten und Zeitverlust.

Es wurde ein Maßnahmenkatalog aufgeführt, der es erlaubt, Prozesse zum Umgang mit Testumgebungen zu definieren, die zu einer effektiveren und effizienteren Nutzung führen. ■

Referenzen

[Mit09] Mitsching, R., Weise, C., Kolbe, A., Bohnenkamp, H., and Berzen, N., Towards an Industrial Strength Process for Timed Testing, IEEE International Conference on Software Testing, Verification, and Validation, pp. 29-38, 2009.

[Sch11] Scholz, G., IT-Systeme für Verkehrsunternehmen: Informationstechnik im öffentlichen Personenverkehr, dpunkt.verlag GmbH; Auflage: 1. Auflage (28. November 2011).