TESTOVACÍ PROSTŘEDKY A TESTOVACÍ POSTUPY

Branislav LACKO

Fakulta strojního inženýrství VUT v Brně
Technická 2, 616 69 Brno (lacko @ fme.vutbr.cz)

Abstrakt
Příspěvek uvádí různá kritéria pro rozdělení testovacích nástrojů. Popisuje současné testovací nástroje a testovací postupy používané pro zvýšení kvality software. V závěru jsou uvedeny problémy segmentu tohoto typu software na našem trhu.

1 Úvod

Problematika testování software se dostává do centra pozornosti, jak naléhavější je potřeba zajistit kvalitní programové vybavení. Příčiny zvýšeného tlaku na jakost programů byly vysvětleny podrobně autorem [10]. Zde je uveden jen výčet důvodů:

- Software se stalo velmi rozšířeným zbožím a zákazníci by měli být chráněni před nekvalitními produkty
- Do software se investují obrovské finanční částky a z celospolečenského i tržního hlediska je potřeba zajistit jejich efektivní zhodnocení
- Stále více se používá software v aplikacích pro automatické řízení procesů, kde chyby v programovém vybavení mohou mít velké, často katastrofální důsledky (řízení jaderných reaktorů, řízení zdravotnických zařízení, navigace letadel, apod.).

Skutečnost, že kvalita software je dnes klíčovým faktorem spolehlivosti současných složitých výrobků, a že softwarová chyby může způsobit závažné ztráty, potvrdila situace, která vznikla poté, co bylo potřeba zastavit provoz rychlovlaků PENDOLINO (Softwarovou chybu, která v minulých dnech vyřadila z provozu čtyři rychlovlaky Pendolino, by měli technici z Alstomu odstranit do několika dnů. M. Procházka, Právo ze dne 18.1.2006).

Tlak na jakost software je patrný i průběhu přípravy norem ISO [15] podle informací prof. J.Vaníčka, který se za ČR prací na těchto normách účastní

Problematika testování programů je velmi důležitá, protože softwarové inženýrství v současné době nemá jiný prostředek pro zjišťování kvality programů. Tato skutečnost je velmi nepříjemná, protože testováním lze prokázat, že program chybu má, nikoliv, že je program bez chyby!

Kvalita programů je ovlivňována pěti základními faktory (podrobněji viz [12]), které jsou odvozeny od pěti faktorů jakosti prof. Ishikawy:

- Pracovníky, kteří se tvorbě programů podílejí
- Programy.samotnými
- Prostředím, ve kterém se programy vytvářejí a testují
- Postupy, které se při vývoji a testování SW aplikují
- Používanými nástroji pro testování a vývoj programů

V souladu cílem příspěvku se budeme dále zabývat jen nástroji pro testování software.

Nástroje pro testování tvoří jednak různá zkušební technická zařízení, na kterých ověřujeme programy (zejména v případě různých programů pro řízení v reálném čase), a jednak programové testovací prostředky.

Za programový prostředek pro testování software budeme považovat takový programový produkt (resp. část programového produktu), který je navržen pro podporu zajišťování testovacích výpočtů.

2 Rozdělení a klasifikace testovacích SW nástrojů

Testování programů je možno provádět velmi různorodým způsobem. Z této skutečnosti vyplývá i velká různorodost testovacích prostředků, které je možno rozdělit podle různých hledisek.

Následující rozdělení vychází jednak z práce doc.J.Hořejše [3], jednak z příspěvku autora na semináři PROGRAMOVÁNÍ´82. [4]

Rozdělení z hlediska manipulace s programem při testování:

a) Statické testovací prostředky, které nepředpokládají provádění programu při testování. Podrobují analýza zdrojový test resp. přeložený kód a snaží se nalézt nepravděpodobné konstrukce nebo nesprávné posloupnosti příkazů (např. čtení neotevřeného souboru, použití neiniciované proměnné, apod.).
b) Dynamické tetovací prostředky, které předpokládají provádění programu na počítači buď přímo nebo interpretačním programem

Dynamické testovací prostředky dělíme z hlediska způsobu podávání zpráv o průběhu testovacího výpočtu:

a) Sledovací testovací prostředky, které podávají zprávy bezprostředně s průběhem výpočtu
b) Post Mortem prostředky, které podají zprávu o výsledku stavu a uplynulém průběhu programu po jeho přirozeném, nebo mimořádném ukončení.

Rozdělení z hlediska zásahu do programu:

a) Testovací prostředky bez nutnosti zásahu do programu. Těmto prostředkům stačí jen zdrojový text nebo strojový kód.
b) Testovací prostředky s pasivními doplňky v programu (příkazy programovacího jazyka nebo vložené strojové instrukce). Tyto doplňky se nepodílejí na základních funkcích programu, ale provádějí funkce z hlediska potřeb testovacího výpočtu (indikace průchodu částí programu, výpis hodnoty určitých proměnných ve stanoveném okamžiku, apod.). Tyto doplňky se chovají z hlediska kompilátoru jako pouhé komentáře nebo jsou aktivní jen ve zvláštním režimu HW počítače a v běžném pracovním režimu se chovají jako prázdné instrukce, takže je není nutno odstraňovat z programu.
c) Testovací prostředky vyžadující modifikaci zdrojového textu programu nebo provedení zvláštní kompilace, při které se generovaný kód programu doplní o speciální fragmenty programu, zajišťující podporu testování. Tyto prostředky je nutno po testování z programu odstranit.

Sledovací testovací prostředky můžeme rozdělit podle předmětu sledování

a) Sledování průběhu výpočtu. Můžeme sledovat buď adresy strojového kódu nebo symbolická označení ve zdrojovém textu programu, kterými výpočetní proces prochází (jmény návěští resp. čísla příkazů, čísla řádku zdrojového textu, jména vyvolávaných podprogramů či objektů, apod.). Sledování můžeme provádět krok za krokem nebo při skocích v programu. Sledovat lze řízení v celém programu nebo v jeho vybraných částech, přičemž ohraničení může být zadáno jako statické nebo se může dynamicky měnit v průběhu programu.
b) Sledování stavu proměnných. Sledovat můžeme obsahy paměťových buněk, které jsou zadány prostřednictvím adres nebo které jsou zadány symbolickými identifikátory

proměnných. Lze sledovat všechny proměnné v programu nebo vyjmenované vybrané proměnné. Logickým požadavkem je vázání tisku hodnoty na určitou podmínku:

- Proměnná změnila svoji hodnotu ke stavu posledního snímku paměti
- Vyskytl se určitý předepsaný incident (přerušení, přeplnění, stisknuté tlačítko na ovládacím pultě nebo určité klávesnice, apod.)
- Určitá proměnná nabyla specifikované hodnoty (např. čítač)
- Bezpodmínečný tisk hodnoty proměnné při prováděném snímku paměti.

Samozřejmě lze způsoby sledování kombinovat.

Rozdělení podle způsobu režimu provádění testovacího výpočtu:

a) Testování prováděné v dávkovém režimu
b) Testování provádění v interaktivním režimu on-line

Podle přítomnosti prostředků při testovacím výpočtu rozeznáváme:

a) Prostředky rezidentní – přítomné trvale po celou dobu testovacího výpočtu v operační paměti
b) Prostředky přivolávané do operační paměti podle potřeby až na základě předepsaného incidentu

Podle způsobu umístění v programu dělíme testovací prostředky na:

a) Interní, které jsou nedělitelnou součástí programu
b) Externí, které jsou umístěny mimo strojový kód programu

Rozdělení podle závislosti na programovacích prostředku:

a) Prostředky testování, které jsou součástí programovacího jazyka (kompilátoru)
b) Prostředky testování, které jsou nezávislé na použitém programovacím jazyku.

Rozdělení podle režimu, ve kterých pracují testovací prostředky:

a) Testovací režim, kdy je prováděno testování programu pomocí testovacích prostředků na testovacích datech
b) Rutinní výpočetní režim, kdy je prováděn běžný výpočet a testovací prostředky nejsou použity nebo je jejich činnost blokována.
c) Režim latentního testování – kdy testovací prostředky zdánlivě nepůsobí, ale jejich činnost se projeví v okamžiku určitého incidentu (překročení horní meze indexu určitého pole, přeplnění, dělení nulou, apod.)

Samostatně se vyčleňují:

a) Generátory testovacích dat, které vygenerují soubory dat pro testovací výpočty
b) Speciální testovací prostředky, které testují jiné vlastnosti programu (dobu odezvy, čerpání operační paměti, využívání paměťových registrů, reakci na terminálovou zátěž, apod.)
c) Prostředky automatického testování, které automatizují určité činnosti, související s testováním (generování skriptů a scénářů [9], automatické generování požadavků pro zátěžové testování, apod.).

Rozdělení podle druhu podporovaných testů (nejčastěji z hlediska V modelu):

a) Prostředky pro akceptační testování
b) Prostředky pro integrační testy

c) Prostředky pro tetování vlastností jednotlivých programů nebo jejich částí (unit testy)

Rozdělení prostředků podle funkcí, které při testování zajišťují:

a) Prostředky, podporující provádění testovacích výpočtů
b) Prostředky, podporující organizaci procesu testování (termínové plánování testů, přidělování testů testerům, sledování nápravy odhalených chyb, apod.).

Kritéria testů bývají dělena podle způsobu jejich přípravy. Lze je rozdělit na dvě velké skupiny:

I.Skupina: Kritéria závislá na struktuře programu (White Box Testing)

a) Testování cest – testovací data mají zajistit provedení všech realizovatelných cest alespoň jednou
b) Testování větví – testovací data mají zajistit provedení všech realizovatelných větví programu

II.Kritéria závislá na problému bez přihlédnutí ke struktuře programu (Black Box Testing)

a) Testování souborem náhodných hodnot ze zvoleného oboru hodnot.
b) Testování reprezentanty stanovených tříd vstupních hodnot (kladné číslo, záporné číslo, nula, určené číslo)
c) Testování podle tříd výstupních hodnot, kdy soubor testovacích dat musí navodit postupně pro každou třídu výstupních hodnot alespoň jednoho reprezentanta.

Souhlasně s tímto rozdělením lze dělit i prostředky, podporující jednu nebo druhou skupinu přístupu k testům a zajistit tak návaznost testovacích prostředků na testovací postupy.

Poznamenejme, že kritéria I. skupiny vycházejí ze struktury navrženého programu, neumožňují ověřit, zda naprogramovaný algoritmus řeší bezchybně zadaný problém, i když se prověří důkladně všechny části programu. Jejich předností je prověření všech částí programu. Naopak kritéria II. skupiny ověřují správnost naprogramování zadaného algoritmu, ale program může po testech vykázat chybu v důsledku průchodu větví, která nebyla při testech prověřena.

Častým kritériem bývá i zaměření testovacích výpočtů. Proto se často můžeme setkat s nástroji, které jsou speciálně určeny pro testování:

a) Web komponent
b) GUI
c) Síťových komunikačních protokolů
d) Atd.

Účelem uvedených klasifikačních kritérií pro rozdělení testovacích SW prostředků bylo ukázat na jejich mnohotvárnost. Z uvedené klasifikace vyplývá, že jeden a tentýž prostředek můžeme klasifikovat podle různých kritérií. Skutečnost, že se vezmou v potaz různá kritéria rozdělení testovacích prostředků a stanoví se jejich přednosti a zápory, aby se provedl vhodný výběr, má významný vliv na objem a kvalitu testů.

Samozřejmě je možno najít další, zde neuvedená, kritéria pro rozdělení testovacích SW prostředků. Výše uvedená jsou však nejčastěji používaná.

Testovací prostředky jsou obvykle navrženy tak, že v sobě slučují několik různých přístupů k testování podle uvedených klasifikačních hledisek.

Při volbě testovacích prostředků nesmíme brát ohled nejen na volbu ČÍM se testuje, ale také na volbu CO se testuje, PROČ se testuje a JAK se testuje.

3 Stav testovacích prostředků na našem trhu

V současné době programové testovací prostředky v širším rozsahu dodávají na náš trh s podporou zejména tři firmy KOMIX, LBMS a UNICORN. Rozumí se prostředky

podporující platformu testování produktů především v operačních prostředích firmy Microsoft. Pro jiná operační prostředí pracovních stanic firem SUN, HP, IBM, apod., pro operační systém LINUX, UNIX a jiné operační systémy, dodávají specializované programové prostředky jiné firmy (např. IBM, SUN, apod,).

S ohledem na dynamicky se měnící nabídku těchto prostředků je lépe si prohlédnout aktuální informace na stránkách těchto firem, jak uvádějí odkazy v seznamu literatury.

Tato málo početná nabídka je reakcí na velmi malou poptávku po těchto produktech v ČR. Objem prodaných prostředků, vyjádřený jak počtem prodaných kusů, tak objemem finančních prostředků je velmi malý. Některé druhy prostředků nejsou na našem trhu zastoupeny vůbec (např. generátory testovacích dat, prostředky pro testování real-time aplikací, apod.) Vyplývá to z přístupu českých firem k problematice jakosti programů a tím i k testování programů. Situaci napomáhá i skutečnost, že tato problematika se neučí ve větším a potřebném rozsahu na školách.

Odpovídá tomu také stav literatury. V záplavě publikací počítačové literatury je zde pouze jeden titul, který vydalo nakladatelství Computer Press od R.Pattona „Testování softwaru“ . Tato publikace je dobrým úvodem do problematiky testování softwaru a doporučuji ji zakoupit pro potřeby pracovníků softwarových firem, pokud se tak již nestalo. Skriptum ČZU Praha od prof. J. Vaníčka [11] je čestnou výjimkou.

4 Závěr

Jak již bylo uvedeno v souvislosti s nabídkou testovacích softwarových prostředků, u nás se zatím poměrně málo používá těchto specializovaných programových produktů. V drtivé většině případů se využívá jen testovacích prostředků, které jsou součástí dodávaných kompilátorů nebo vývojových prostředí (DELPHI, MS Visual Studio, apod.) a jako nejpoužívanější prostředek pro generování dat slouží program MS EXCEL. To je samozřejmě nedostatečné. Tyto prostředky slouží to spíše k podpoře vlastního vytváření programů, méně k nezávislému samostatnému testování programů. Malá efektivita těchto prostředků se projeví zejména tehdy, když si uvědomíme problematiku a nároky testovacího procesu při testování aplikací počítačové grafiky, real time aplikací, produktů virtuální reality, testování rozhraní HMI (rozhraní člověk-stroj tvoří významnou složku řízení většiny současných složitých strojů a tato komunikace výrazně ovlivňuje výslednou spolehlivost celé soustavy- viz [13]), apod.

Tlak na aplikaci souboru norem ISO 9000:2000 v oblasti SW a tlak na využívání specifických norem jakosti SW bude mít za následek vynucené používání specializovaných testovacích softwarových prostředků. Ty by měly pokrýt celý proces testování včetně plánování a protokolovaní v napojení na řízení verzí (viz normu ISO 10 007 a produkty typu PVCS – Program Version Control System viz nabídku např. firmy AIT Praha).

Zvýšené požadavky na testování, které představují vysoké nároky na čas a objem testovacích činností lze rozumně řešit jen inteligentními testovacími prostředky a vysokým stupněm automatizace. V tomto ohledu není pochyb, že je to cesta, která reálně může pomoci řešit zvýšený objem testovacích prací. Limitujícím momentem na realizaci tohoto způsobu řešené je však kvalitní specifikace softwarového produktu. Konkrétní případ, jak je možno využít automatického generování testovacích scénářů, pokud je specifikace software dobře popsaná jazykem UML ukázal R. Nagy. [9] Bohužel v českých softwarových firmách je problematika specifikace software podceňována ještě více než problematika testování SW.

Ignorovat programové testovací prostředky (např. s argumentem jejich ceny) nebude nadále možné. Je pravda, že tyto prostředky představují určitý náklad, který je lákavé odstranit, obzvláště tehdy, když se zdá, že se nic nestane, když se testování jejich prostřednictvím neprovede a zkrátí se dodací lhůta. Stačí však přistoupit k evidování nákladů na odstraňování chyb, dále k evidování nákladů na ztrátu zakázek v důsledku špatné pověsti,

započítat náklady na úhradu způsobených škod, a porovnávat „úsporu ze zrušení nákupu testovacích prostředků“ s náklady na „nejakost“, aby situace byla zcela odlišně hodnocena!

Pokud se vyvíjel tlak na jakost software jen v tak specializovaných aplikacích, jakou představovaly např. aplikace v jaderné energetice, kdy velmi omezené množství programových produktů, které se použilo bylo možno speciálními postupy za vysoké náklady (které zaplatili zákaznici) řádně otestovat [14], nebyl tlak na jakost software veliký. Pokud se však začne používat mikroprocesorů v tak masivním objemech, jak to předpokládá automobilová, železniční nebo letecká doprava, bude logicky tlak na jakost software daleko vyšší. Jakmile náročné real time aplikace prokáží, že je možno software kvalitně navrhnout a otestovat, bude obtížně udržitelný argument, že běžné programy pro „kancelářské“ a jiné aplikace mohou být plné chyb! V současné době je to právě automobilový průmysl, který začíná klást oprávněné požadavky na kvalitu software v zabudovaných mikroprocesorech automobilových komponent. Pak snad stoupne poptávky i po produktech CASE, které významně jakost software podporuji (viz zkušenosti ze ZČU Praha [16]), podobně jako řízení projektů i v malých a středních SW firmách [17].

Výše uvedené skutečnosti by měly přimět naše softwarové firmy, aby seriózně začaly přemýšlet o výběru a nasazení specializovaných testovacích prostředků.

Literatura:

1 Ron PATTON: Testování softwaru, Computer Press Praha 2002, 313 stran

2 Maggie BIGGSOVÁ: Team Treck udrží obchodní procesy v chodu. Computerworld č.14/2004, str.28

3 Hořejš, J: Ladění programů (Esej o prevenci, diagnose a terapii)
In: Sborník semináře SOFSEM´76. Výzkumné výpočtové středisko OSN Bratislava 1976, str. 7-37.

4 Lacko, B.: Systematický přístup k testování programů
In: Sborník PROGRAMOVÁNÍ´82. Dům techniky ČSVTS Ostrava 1982, str.108-127

5 www.ait.cz

6 www.lbms.cz

7 www.komix.cz

8 Nedomová,L.: Řízení kvality v českých softwarových firmách.
usiness World červen 2000, str.34-35, příloha časopisu Computerworld.

9 Nagy,R.: Automatické generovanie testovacích scénárov na základe špecifikácie v jazyku UML. Automatizace roč.48 (2005) č.3, str. 194-197

10 Lacko,B.:Systémový přístup k jakosti softwaru
In: Sborník z jubilejního 30.ročníku konference Tvorba software 2004.
VŠB-TU Ostrava 2004, str. 129-138

11 Vaníček,J.: Měření a hodnocení jakosti informačních systémů.
ČZU Praha 2000, 212 stran

12 Lacko,B.:Využití teorie kauzality v jakosti software
In: Sborník z mezinárodní konference Evropský týden kvality v České republice.
Česká společnost pro jakost Praha 2003, str. 493-503

13 Havlíková, M: (Kap. X.) Modely spolehlivosti lidského operátora.
Kybernetika a společnost na prahu XXI. Století. VUT v Brně, 2004 Brno, str.53-58

14 Karpeta,Č.-Povalač,O.: Zabezpečování jakosti software bezpečnostních systémů v rámci projektu „Obnova JE Dukovany“. Sborník konference Tvorba software 2005, VŠB-TU Ostrava 2005, str. 78-105

15 Vaníček,J: Stav a perspektivy mezinárodní normalizace v oblasti měření a hodnocení jakost informačních a softwarových produktů. ČZU Praha 2004, 67 stran

16 Merunka, V: První zkušenosti s modelovacím nástrojem Draft Case.
In: Sborník Tvorba softwaru 2006. VŠB-TU Ostrava 2005, str. 137-142

17 Zedek, M.: SW nástroje pro podporu projektového řízení v malých a středních firmách.
In: Sborník Tvorba softwaru 2005. VŠB-TU Ostrava 2005. str. 251-255

Přílohy

Výběr nabídky firmy LBMS:

Relational Tools for Servers
Prostředí pro přípravu testovacích dat a vyhodnocování testů prostřednictvím databáze pro komerčně nejrozšířenější databázové stroje jako jsou DB2, Informix, MS SQL Server, Oracle či Sybase. Relational Tools for Servers umožňuje vytvořit relačně konzistentní výřez produkčních dat, nahrát jej do testovací databáze a případně porovnávat takovéto vzorky dat před a po provedení testů

TeamTrack
TeamTrack je nástroj postavený na webové technologii umožňující evidenci a řízení problémů dle firemních procesů od vrcholové úrovně až po úroveň dílčích problémů. Umožňuje sdílet informace mezi členy řešících týmů a zákazníky

TestDirector
Nástroj TestDirector slouží pro řízení testovacího procesu. Umožňuje vytváření testovacích scénářů, řízení a sledování spouštění testovacích běhů, vyhodnocování výsledků testů a řízení požadavků na odstranění chyb vzniklých v průběhu testování.

QuickTest Professional
Nástroj QuickTest Professional slouží pro automatizované testování funkčnosti webových a Java aplikací. Umožňuje jednoduchým způsobem nahrát scénář testu přímo v testované aplikaci a tento scénář pak opakovaně přehrávat s různými vstupními parametry se současným ověřováním správného fungování aplikace.

LoadRunner
Nástroj LoadRunner slouží pro automatizované zátěžové širokého spektra aplikací a ERP systémů. Umožňuje simulovat zátěž od mnoha souběžně pracujících tzv. virtuálních uživatelů a na základě získaných informací identifikovat úzká místa v technologii a aplikaci, která snižují její potenciální výkonnost.

WinRunner
Nástroj WinRunner slouží pro automatizované testování funkčnosti širokého spektra aplikací a ERP systémů. Umožňuje jednoduchým způsobem nahrát scénář testu přímo v testované aplikaci a tento scénář pak opakovaně přehrávat s různými vstupními parametry se současným ověřováním správného fungování aplikace.

ChangeMan
ChangeMan je určen pro správu rozsáhlých distribuovaných softwarových aplikací. Umožňuje centrálně spravovat softwarový vývoj, nasazení a údržbu aplikací nezávisle na použitých platformách.

Výběr z nabídky firmy KOMIX:

Firma nabízí soubor produktů TestDirector, Quick Test, Win Runner a Load Runner firmy Mercury

Doors

Produkt firmy Telelogic (s plným názvem Doors Enterprise Requirements Suite) je produkt pro sběr, analýzu a řízení správy požadavků s napojením na dostupné přidělené zdroje. Tento produkt vytlačil z našeho trhu produkt Requisite Pro.

Výběr z nabídky firmy AIT:

Řízení testování je nutno propojit s řízením verzí. Zde kromě nabízeného produktu ChangeMan (viz firma LBMS) nabízí firma AIT Praha sadu nástrojů ne řízení verzí a konfigurace od firmy SERENA (firma AIT se věnuje této problematice u nás jednak jako první firma a již poměrně delší dobu – viz dřívější dodávky produktu PVCS):

SERENA Version Manager
Tento produkt je de factó dnes průmyslovým standardem pro organizování, řízení a ochranu vašich firemních softwarových aktivit. Version Manager umožňuje projektovým týmům různé velikosti a případně umístěných i v různých lokalitách koordinovat paralelní vývoj při zajištění bezpečného přístupu a kompletního auditu

SERENA Tracker
Tracker zachycuje a obhospodařuje činnosti nad navrhovanými novinkami, hlášenými chybami a požadavky na změny, které zabírají jinak projektovému týmu mnoho času. Tracker pomáhá týmu komunikovat nad těmito projektovými úlohami a koordinovat práci týmu v životním cyklu.

SERENA Configuration Builder
Produkt automatizuje proces sestavení s možností přenosu id informací o zdrojích do cílové aplikace zákazníka. Při sestavení aplikace dokáže úzce spolupracovat s produktem Version Manager; lze využít libovolný standardní kompilátor a linker.

Výběr z nabídky firma IBM Rational:

BM Rational Functional
Tester provádějící funkční a regresní testování pro Java aplikace a webové aplikace, nově nabízí i funkční testování aplikací vytvořených v novém Microsoft Visual Studiu 2005. Verze 6.1.1. využívá jako skriptovací jazyk Visual Basic .Net a staví na Visual Studio 2005 shellu. To umožňuje testerům výběr vývoj testovacích skriptů, v prostředí Eclipse a nebo .Net.

IBM Rational Performance Tester Extension
Snižuje rizika a chyby při implementaci nových nebo aktualizaci stávajících aplikací SAP. IBM Rational Performance Tester Extension umožňuje výkonnostní testování a provádění testů plánování kapacit. Dále měří a předpovídá výkon řešení SAP před jeho nasazením.

IBM Rational Performance Tester verze 6.1.2
Poskytuje zátěžové a výkonnostní testování a nově integruje Tivoli kombinovanou správu aplikací a monitorovací produkty. Tato integrace umožňuje testovacím a IT operačním týmům snadněji izolovat zdroj od výkonových úzkých míst a zabránit nedodržení SLA.