Rational PurifyPlus

Jan Outrata

14. dubna 2002

1 Rational PurifyPlus: Co je to

3 nástroje pro ladění SW:

Rational Purify - hledá runtime errors a memory leaks

Rational PureCoverage - testuje každou část kódu

Rational Quantify - analýza výkonu, hledá úzká hrdla výkonu

Lze je spustit z VS, s debugerem nebo samostatně.

Purify: chyby práce s pamětí

PureCoverage: prověří všechny části kódu, lze spustit z Purify, při chybě, která přeskočí velkou část programu, použijte testovací nástroj Rational Robot nebo Rational Visual Test pro napsání testů pro novou část kódu

Quantify: při přidání kódu zachytí pokles výkonu a podá informace pro zvýšení výkonu

Následující popis programů se týká jejich verzí pro OS MS Windows. Účel, vzhled a chování programů jsou stejné i u verzí pro unixové operační systémy. Ovládání programů je s nepatrnými obměnami danými jiným stylem práce také stejné.

Další informace o těchto nástrojích lze nalézt na WWW stránkách firmy Rational - `http://www.rational.com` a v nápovědách uvedených programů.

2 Rational Purify (pro Visual C/C++)

2.1 Co to dělá

Runtime chyby paměti jsou nejobtížnější k nalezení a nejdůležitější k opravení. Symptomy jsou nepředvídatelné a typicky se objevují daleko od příčiny chyby. Často jsou skryté, dokud nejsou náhodně vyvolány, program se může zdát bezchybný.

Purify nabízí: detekci runtime chyb, hledání chyb i bez zdrojáku, pokrytí kódu (code coverage) ukáže netestovaný kód

Detekuje chyby paměti: překročení rozsahu pole, přístup přes neplatné ukazatele, čtení neinicializované paměti, chyby alokace paměti, memory leaks

Dnešní SW je komponentový, je potřeba testovat i nevlastní komponenty (můžeme jí dávat špatná data). Prověří každou komponentu (i vícevláknovou, víceprocesovou aplikaci): DLL (i MFC), Visual C/C++ komponenty (VB, IE, NN, MOff), MExcel a MWord pluggins, COM (OLE, ActiveX)

Kontroluje volání API funkcí: GDI, Inet services, registr, COM, OLE, handly a ukazatele do paměti

Umožňuje sdělit pokrytí kódu, identifikovat netestovaný.

2.2 Základní kroky

1. Spustit program pomocí Purify a zjistit chyby, pokrytí kódu
2. Analyzovat chyby a opravit je
3. Analyzovat části, které nabyly prověřeny
4. Spustit program znova pomocí Purify

Spuštění programu pomocí Purify: ve VS Purify z Purify toolbaru nebo Purify s pokrytím kódu, sestavit program s debugovacími informacemi (pro maximální detaily v Purify) a spustit, Purify zkopíruje program i všechny knihovny a přidá instrukce ověření čtení, zápisu, alokace a de-alokace paměti (OCI - Object Code Insertion technology), čítače řádků a funkcí, ukáže průběh každé komponenty, cache kopií, při dalším spuštění mění jen změněné, detekuje a zobrazí runtime chyby a memory leaks, pro client-server a víceprocesové aplikace každý proces spustit zvlášť

Pohled na chyby: chyby a varování, informace o běhu, používaná paměť a handly po ukončení

Když se opakují stejné chyby: ve smyčkách, každá jen jednou (poprvé) a počítadlo, lze změnit

Zaměření na kritické chyby: filtry na skrytí ostatních chyb - zvlášť, podle typu (informační) a zdroje (z jednoho souboru), platí pro všechny další spuštění až do vypnutí filtru

Filtry: Filter Manager pro vytvoření filtru na specifické moduly, globální filtry na všechny moduly, uložení filtru, PowerCheck pro zaměření se na určitý modul

Analýza chyb: kde se objevily, diagnostické informace, proč se objevily, detaily závisí na množství debug informací (i při release sestavení), možnost nastavení formátu (např. počet řádek, ukazatele na instrukce)

Opravení chyb: dvojité kliknutí na řádek s chybou (v Purify Data Browseru)

Pokrytí kódu: kontrola, zda se prověřuje všechen kód, zvlášť přidaný nebo změněný, po řádcích, barevné označení, pokud se testuje manuálně, projít všechna menu, příkazy, hodnoty, pokud automaticky, upravit testy

Porovnání běhů programu: po opětovném spuštění opraveného kódu lze srovnat s předchozím

Uložení dat: jen chyby, chyby s pokrytím, přidání dat o pokrytí do souboru PureCoverage, ASCII soubor (pro skripty)

2.3 Pokročilé možnosti

Nastavení vkládání instrukcí: záleží na velikosti modulu a množství debug informací, precizní - plná detekce runtime chyb, minimal, pro pokrytí po rádcích nebo po funkcích, i pro každý modul zvlášť

Průběžné dubugování: instantní přístup k debugeru při zastavení na chybě (musí se nastavit Break on Error), zastaví program těsně před chybou, možnost testování přímo v debugeru, použití filtrů

Samostatné Purify: bez VS i samostatně s VS, Run Program dialog, pak stejné

Příkazový řádek: v Makefile souborech, skriptech, program.exe na purify program.exe, nebo přidat instrukce (a pak normálně program.exe) purify /Replace=yes /Run=no program.exe (uloží bak), pro data o pokrytí purify /Coverage=yes program.exe, bez grafiky /SaveTextData

Rozšířené hledání chyb pomocí Purify API funkcí: defaultně zobrazí memory leaks až po ukončení programu, po volání PurifyNewLeaks (New leaks button) hned vypisuje nové leaks, volání také z Quick Watch dialogu ve VS

Použití v integrovaném prostředí: spolu s Rational QlearQuest (správa změn) - zaznamenání chyby, Rational Robot - nastavení robota (playback) na shromáždění dat o chybách a pokrytí, Rational Visual Test (testování) - Purify automaticky v testech

3 Rational Purify (pro Javu a .NET kód)

3.1 Co to dělá

V Javě a .NET kódu jsou také memory leaks. JVM a .NET garbage collector automaticky odstraňují z paměti objekty, které již nejsou potřeba. Ale aplikace může spotřebovávat stále více a více paměti: nepotřebné objektové reference a neuvolněné systémové prostředky.

Objektové reference: paměť, na kterou existuje reference nemůže garbage collector uvolnit, např. zapomenuté objekty v poli, neuvolněné reference dočasných objektů, nestačí referenci přiřadit NULL

Systémové prostředky: paměť mimo Java instance nebo .NET kód, např. prostředky oken a bitmap

Purify zjistí metody, třídy a objekty, které alokují velké bloky paměti, kterou garbage collector nemůže uvolnit. Pak můžete eliminovat reference nebo vynutit jejich odstranění pomocí garbage collector.

4 Rational PureCoverage

4.1 Co to dělá

Před vypuštěním produktu je potřeba jej pořádně otestovat, každý řádek, funkci. PureCoverage vyhodnotí úplnost testu a vytipuje části, které nebyly testovány. Monitoruje pokrytí kódu testy a vytvoří report.

Pomocí PureCoverage je možné: vidět procento kódu, který byl testován a který ne, identifikovat netestované funkce nebo řádky, nastavit zaměření na detaily, spojit data o pokrytí z více běhů programu, uložit data o pokrytí, monitorovat pokrytí kódu

Kontrola každé komponenty: Visual C/C++ a Visual Basic exe, dll, OLE/ActiveX, COM, Java applety, třídy, jar, .NET exe, komponenty z IE, Transaction Server, tstcon32, NN, MOff, MExcel a MWord pluginy

Použití při vývoji: začít používat co nejdřív, tím je testován všechen kód od počátku

4.2 Základní kroky

1. Spustit program s PureCoverage

2. Obraz kódu pomocí Coverage Browser a Function List, PureCoverage filtry

3. Identifikovat netestované řádky v Annotated Source okně

4. Modifikovat testy pro pokrytí netestovaných řádek, podmínek, funkcí

5. Znova spustit program, uložit data o pokrytí

Pro data o pokrytí na úrovni řádků musí být program zkompilován s debugovacími informacemi o řádcích.

Spuštění programu: spustit PureCoverage ze Start Menu, Run button, Run Summary okno se stavem pokrytí

Obraz pokrytí: Coverage Browser a Function List okna ukazují celkové pokrytí, funkce buď missed nebo hit, počet řádek missed a hit, Coverage Browser - data o pokrytí hierarchicky podle zdrojáků, Function List - textově, nehierarchicky, pro třídění, možnost nastavení zobrazení

Zaměření pomocí filtrů: data pro každý modul, nezobrazuje všechna - default filter pro systémové komponenty, Filter Manager Tool pro zobrazení a změnu

Identifikace netestovaných řádků: data o pokrytí řádek po řádku v kopii zdrojáku, dvojklik na funkci v Coverage Browser nebo Function List pro zobrazení kódu v Annotated Source okně, rozlišení barvami - netestované červeně, testované modře, řádky nevolané (ze zdrojáku) funkce černě, nevyvolané možnosti podmínky růžově, je možné jej otestovat změnou hodnoty podmínky pomocí QuickWatch

Modifikace testů: provedení netestovaných částí programu při dalším spuštění nebo úprava testů pro větší pokrytí

Znovuspuštění programu: nejen data o pokrytí nového běhu, ale i Auto Merge data - spojení s daty z předchozích běhů (možno nastavit kterých)

Uložení dat o pokrytí: Save Copy As button, Pure Coverage data soubory nebo ASCII text, možnost uložení z příkazové řádky

4.3 Pokročilé možnosti

Integrace s vývojovým prostředím: s VS, VB - PureCoverage menu a toolbar, Rational Visual Test, Rational Robot - testovací skript, Rational ClearQuest - uložení informace o neúplném pokrytí

Vyladění sběru dat o pokrytí: PowerCov options, nastavení úrovně pokrytí (řádky, funkce) pro každý modul, defaultní pro všechny programy nebo nastavení pro konkrétní program

Interaktivní nahrávky: nahrávky dat o pokrytí pro konkrétní rutiny programu při běhu

PureCoverage API funkce: začátek a konec sběru dat o pokrytí, uložení během běhu, sběr dat jen pro určitou část programu, volání API funkcí z programu, QuickWatch dialogu VS, debugeru

Integrace do testovacího prostředí: z Makefile souborů, skriptů - Coverage /SaveTextData program.exe, spojení dat o pokrytí s daty z předchozích běhů - Coverage /SaveMergeData /SaveMergeTextData program.exe uloží do program_AutoMerge.cfy a program_AutoMerge.txt, pro Javu /Java přepínač, pro .NET a VB /Net přepínač, v testovacích skriptech Rational Visual Test a Rational Robot

5 Rational Quantify

5.1 Co to dělá

Zákazníci chtějí co nejrychlejší SW. Musí se najít slabá místa výkonu a vylepšit je. Výkon by se měl začít monitorovat co nejdřív, dokud je ještě kód malý. Také se musí zvážit náklady na zlepšení proti jeho užitku, který od toho čekáme. Rational Quantify provádí výkonnostní průzkum, nasbírá data o výkonu a zobrazí je přehledně tak, že je vidět, kde je program nejpomalejší.

Zjišťuje výkonnost každé komponenty: Visual C/C++ a VB exe, dll, OLE/ActiveX, COM, Java applety, třídy, jar, kód z kontejneru, .NET exe z VS .NET, komponenty z IE, Transaction Server, tstcon32, NN, MOff, MExcel a MWord pluginy

Integrováno do VS a VB.

5.2 Základní kroky

1. Spustit program pod Quantify pro změření výkonu

2. Quantify data okna na analýzu dat o výkonu a nalezení slabých míst

3. Změna kódu pro zlepšení výkonu a odstranění slabých míst

4. Znovuspuštění programu a ověření zlepšení pomocí Compare Runs nástroje

Spuštění programu: Quantify ze Start Menu, Run button pro Run Program dialog, výkon se měří pro funkce a pokud jsou debugovací informace pro řádky, tak i pro řádky (i pro release), zobrazí Run Summary okno během běhu programu, které zobrazuje stav všech vláken programu, pozměněné (instrukce) komponenty v cache, při znovuspuštění se mění jen změněné

Analýza dat o výkonu: Quantify Call Graph okno - po ukončení programu, zaměřeno na hodně používané části (kde by zlepšení výkonu mělo největší dopad), výběr funkcí podle různých kritérií (výkon, vztahy volání, příčiny slabých míst), zaměření se na určitou funkci (zobrazení volání dalších funkcí - Subtree), Function list okno - číselná data o výkonu (čas provádění funkcí), preview zlepšení - smazání funkce z grafu, Quantify přepočítá data bez ní, Function Detail okno - výkon z pohledu funkce (číselné a grafické zobrazení o volajících a volaných funkce), Annotated source okno - zdroják s daty o výkonu pro řádky

Porovnání změněného výkonu programu: Compare Runs nástroj, porovnání mezi dvěma běhy pro ověření užitečnosti změn (změny zeleně v grafu), Diff function list pro číselné porovnání, uložení dat (Quantify data file nebo ASCII)

5.3 Pokročilé možnosti

Integrace s vývojovým prostředím: s VS, VB - menu a toolbar, Rational Visual Test, Rational Robot - v testovacích skriptech, Rational ClearQuest - uložení informace o malém výkonu

Interaktivní nahrávání dat: nahrávání dat jen pro část programu (zastavení, výmaz, pokračování), snímek aktuálních dat, Quantify API funkce

Vytipování funkcí se stejnými klíčovými vlastnostmi: v grafu volání funkcí specifické charakteristiky výkonu nebo vztahy volání, nebo např. funkce se zdrojákem a bez

Zaměření se na určitá data: příkazy filter (vyjmutí funkce nebo modulu z dat) a subtree (detail nebo vyjmutí funkce a volaných), undo, expand a collapse příkazy v Call Graph (nemění data)

Nastavení měření: Quantify PowerTune možnosti, jak jsou specifické moduly měřeny - řádky (čas řádky), funkce, time (čas funkcí dohromady)

Integrace do testovacího prostředí: z Makefile souborů, skriptů - Quantify /SaveData program.exe, pro Javu /Java přepínač, pro .NET kód a VB /Net přepínač