Automated Unit Testing

met SQLDeveloper

Tijdens het softwareontwikkelproces wordt altijd hard gewerkt om de functionaliteit zo goed mogelijk te implementeren, maar tijdens de systeemtestfase worden altijd nog dingen gevonden. En met name bug fixing kan erg tijdrovend zijn waarbij tevens onvoorziene impact op overige functionaliteit van de applicatie kan optreden. In mijn vorig project wilde ik het daarom eens anders aanpakken. Bij aanvang van het project is besloten om de kwaliteit van de functies en procedures tijdens realisatie gestructureerd te (unit)testen.

De applicatie is gerealiseerd in PL/SQL met een Application Express user interface en biedt de functionaliteit om van verschillende systemen data te ontvangen, te verwerken en vervolgens op basis hiervan via onder andere webservices deze data weer te ontsluiten. De user interface dient met name voor monitoring van de processen en het opgeven van sturingsinformatie. Aangezien de functionaliteit van de applicatie in PL/SQL is gerealiseerd is gekozen voor het PL/SQL Unit Test Framewrok van SQLDeveloper (toen nog 2.1) om de unit tests gestructureerd en herbruikbaar op te stellen. In dit artikel wil ik aangeven wat de ervaringen en voordelen zijn van het gebruik van dit framework.

Testmethodes in software ontwikkeling

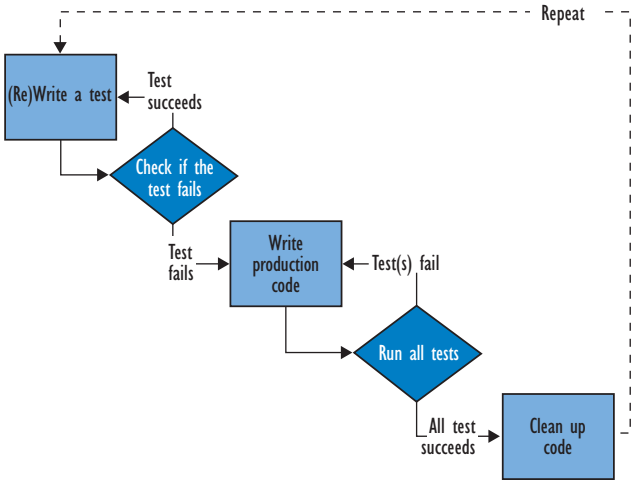
Voor het testen van software zijn verschillende methodes en toepassingen. Zo ondergaat de applicatie een gebruikers- of functionele acceptatietest waarbij de functionaliteit door testers of gebruikers wordt beoordeeld en geaccepteerd voor de inproductiename van de applicatie. Deze fase van het softwareontwikkelproces gaat vooraf aan systeem- en regressietests waarbij de functionaliteit wordt gevalideerd en veelal vaak uitgevoerd in nauwe samenwerking met het ontwikkelteam. Een veel gebruikte testmethodiek hierbij is TMAP. Daarnaast wordt tijdens de realisatie de kwaliteit van de applicatiecomponenten en -units beoordeeld door het ontwikkelteam door middel van code reviews en unit tests. Zonder deze validatie is er een zwak fundament en zal zich vertalen in vele technische bevindingen in de hierop volgende testfases. Met het opstellen van kwalitatieve unit tests is een aantal beperkingen te benoemen. Zo is doorgaans het gedrag van PL/SQL erg afhankelijk van data, zoals inhoud van tabellen, cursor variabelen, parameters. Het uitvoeren van de tests kan daarmee ook erg tijdrovend zijn. Daarnaast is er in de community geen breed gedragen standaardmethode, hetgeen zich vertaalt in weinig ondersteuning vanuit de ontwikkeltooling. En tot slot, sprekend vanuit eigen ervaring, is het implementeren leuker dan testen. Hierdoor kunnen we, ten aanzien van hoe ik het ontwikkelproces aanschouw - waarbij er door de ontwikkelaars wordt getest - de volgende problemen onderscheiden:

- Testen worden niet herhaald, wanneer het eenmaal werkt wordt er niet meer opnieuw naar gekeken.
- Beperkte set aan testcases, onduidelijk welke relevante testscenario's worden doorlopen.
- Resultaat van de unit test wordt veelal handmatige beoordeeld.
- Er wordt pas getest wanneer de code af is waardoor je onbewust naar een positief resultaat wordt getest.

Dit is niet onopgemerkt gebleven bij Oracle's Product Development van SQLDeveloper. Deze afdeling heeft met versie 2.1 een unit test framework voor PL/SQL geïntroduceerd. Het gebruik van een framework is handig, aangezien hiermee een bepaalde structuur wordt afgedwongen waardoor de testcases eenduidig en herhaalbaar kunnen worden opgesteld. Naar mijn inzicht is de toegevoegde waarde van geautomatiseerd unit testen;

- Herhaalbare testcases;
- Uitbreidbaar;
- Automatische resultaatbeoordeling;
- Rapportagemogelijkheden over testuitslagen en code coverage.

Oftewel met een enkele druk op de knop worden er automatisch unit tests uitgevoerd, gevalideerd en gerapporteerd over de uitslagen.



Test Driven Development

Het hebben van een unit test framework is geen garantie voor een verhoogde kwaliteit van de code. Het ontwikkelproces dient goed om te gaan met het opstellen van relevante testcases. Een methodiek hiervoor is Test Driven Development. Volgens deze methodiek begin je met definiëren van de testcases, vervolgens het implementeren van de unit en tot slot valideren of de unit voldoet aan de gedefinieerde testcases. De kracht hiervan zit in het herhalen van de testcases, waarmee wordt geverifieerd of aanpassingen aan de unit niet hebben geleid tot een verschil in het gedrag tijdens reeds succesvolle testcases. In de praktijk blijkt dit niet geheel aan te sluiten aangezien vooraf het gewenste gedrag niet is beschreven of bepaald. Dit leidt dan ook tot een mix van opstellen van de unit en vervolgens opstellen van de testcases, zowel voor de correcte werking als omgaan met uitzonderingsgevallen.

PL/SQL Unit Test Framework

SQLDeveloper biedt nu een unit test framework om de tests te structureren en herbruikbaar op te stellen. Voor het PL/SQL unit test framework van SQLDeveloper moet eenmalig een repository (apart schema) worden aangemaakt op de ontwikkeldatabase. Vervolgens kan deze worden geselecteerd vanuit SQLDeveloper en is de mogelijkheid in het rechtermuisknopmenu aanwezig om voor procedures en functies, ook die in de packages, een unit test te definiëren.
Het definiëren van een unit test verloopt via een wizard waar de verschillende fases van een testcase worden nagelopen.
De testcase begint met een Startup actie. Ik heb al aangegeven dat PL/SQL units veelal afhankelijk zijn van aanwezige data. In een startup action kan met onder andere custom code of een aantal voorgedefinieerde acties de benodigde dataset worden klaargezet. Hiermee is het mogelijk om iedere testcase autonoom te maken. Het definiëren van startup actions kan herbruikbaar beschikbaar worden gesteld in zogenaamde libaries. Dit is handig zodat je voor verschillende testcases niet de start-up actions hoeft te kopiëren.
De volgende stap in het definiëren van de testcase is het opgeven van de input parameters voor de te testen unit. Hierop volgt de eventuele validatie van de output en aanvullende validaties om automatisch te valideren of de functie succesvol de testcase heeft doorstaan. Met het doorlopen van de wizard is al vrij duidelijk wat het framework allemaal biedt. Het PL/SQL unit test framework bestaat uit:

- Tests and suites
- Libraries and Lookups
- Startup and Teardown actions
- Validaties
- Rapportages

In het project hebben we hier veelvuldig gebruik van gemaakt, beginnend met versie 2.x en vervolgens doorgegaan naar 3.0 die sinds begin dit jaar beschikbaar kwam in early adaptors versie. In SQLDeveloper 2.x was er nog niet de mogelijkheid om de testcase te synchroniseren met de betreffende unit, wat er toe leidt dat wanneer de specificatie (in en output parameters van de unit) wijzigen de testcase invalid werd. Gelukkig is in SQLDeveloper 3.0 de feature toegevoegd om dit te synchroniseren. Ook blijkt het framework nog niet alle datatypes te ondersteunen, met name het gebruik van zelf gedefinieerde pl/sql types was tot 3.0 nog niet mogelijk. Wel is er nog steeds de beperking op het gebruik van dynamische types, bijvoorbeeld gedefinieerd met table%rowtype.

Met het uitgebreide unit test framework van SQLDeveloper 3.0 is er goede ondersteuning voor gestructureerd testen van PL/SQL. Met name het gebruik van libaries voor startup acties en het bundelen van verschillende testcases tot een suite geeft de mogelijkheid om de functionaliteit van de applicatie met een druk op de knop te testen. Wel dien je er zelf zorg voor te dragen de testcases zo autonoom mogelijk op te stellen en mogelijk afhankelijkheden met data in de startup te verzorgen. Met de uitbreiding voor ondersteuning van custom PL/SQL types is de inzetbaarheid van dit framework erg vergroot. Ook de mogelijkheid om de testcase te synchroniseren is niet onmisbaar, aangezien gaandeweg de realisatie specificaties kunnen wijzigen waardoor voorgaande testcases na synchronisatie weer herbruikbaar zijn. Om de unit testcases goed op te zetten is initieel wel inspanning noodzakelijk, maar dit komt verdere ontwikkelingen en patches ten goede.
Door een druk op de knop kan een gehele unit test worden uitgevoerd en gevalideerd.

Simon Boorsma is Senior Technologie Specialist Oracle bij Sogeti.