Kwaliteitgedreven softwarearchitectuur

Een aanpak voor betere software

Softwarekwaliteit beschrijft niet zozeer de functionaliteit van het systeem, maar veel meer de manier waarop deze functionaliteit is gerealiseerd. Te denken valt aan de betrouwbaarheid, onderhoudbaarheid en snelheid van het systeem.

Danny Greefhorst en Jan Bosch

Een standaardmodel voor het redeneren over softwarekwaliteit is beschreven in de ISO/IEC 9126-standaard, en het daarop gebaseerde Extended ISO-model (Van Zeist e.a., 1996). In het laatste model worden in totaal 32 kwaliteitseigenschappen gedefinieerd, onderverdeeld in zes hoofdeigenschappen. De hoofdeigenschappen zijn functionaliteit, bruikbaarheid, betrouwbaarheid, onderhoudbaarheid, portabiliteit en efficiëntie. Onder deze laatste noemer vallen bijvoorbeeld de kwaliteitseigenschappen tijdgedrag (performance) en hulpbrongedrag. De kwaliteitseigenschappen kunnen voor een bepaald systeem worden geprioriteerd en omgezet in meetbare kwaliteitseisen. Het meetbaar maken van kwaliteit blijkt in de praktijk niet eenvoudig; de eis 'het systeem moet goed onderhoudbaar zijn' is bijvoorbeeld veel te vaag. Het streven is te komen tot kwantitatieve streefwaarden die zo vroeg mogelijk te meten zijn.

Een architectuur beschrijft de belangrijkste beslissingen omtrent het ontwerp van een systeem en is een belangrijk onderdeel van een softwareontwikkeltraject. Onderdeel van een architectuurbeschrijving is de onderverdeling van het systeem in verschillende soorten onderdelen en hun relaties. De verschillende structuren of gezichtspunten die hierbij worden beschreven hebben een grote invloed op de uiteindelijke kwaliteit van het systeem. Het borgen van kwaliteit zou dan ook in eerste instantie op het niveau van architectuur moeten plaatsvinden. Dit neemt niet weg dat ook in de uiteindelijke realisatie van het systeem deze kwaliteit moet worden gegarandeerd. Het borgen van kwaliteit in software blijkt in de praktijk niet eenvoudig. Niet alleen is het vinden van een architectuur die tegemoet komt aan deze eisen vaak een kwestie van gevoel en ervaring. Ook interfereren kwaliteitseisen en de wijze waarin deze in de architectuur worden bewerkstelligd vaak met elkaar. Er is behoefte aan een standaardaanpak voor het ontwerpen van een architectuur die aan gegeven kwaliteitseisen voldoet.

Een kwaliteitgedreven aanpak voor architectuurontwerp

In dit artikel geven de auteurs een overzicht van een aanpak voor het ontwerpen van architectuur die gestuurd wordt door kwaliteit. Deze aanpak is gebaseerd op het Extended ISO-model voor softwarekwaliteit zoals beschreven door Van Zeist e.a. (1996) en het QUASAR architectuurdefinitieproces zoals beschreven door Bosch (2000).

De eerste stap in de aanpak bestaat uit het opstellen van een kwaliteitsspecificatie. Die beschrijft de voor het systeem belangrijkste kwaliteitseigenschappen en geeft aan hoe deze eigenschappen gemeten dienen te worden. Hiertoe zijn er in het Extended ISO-model zogenoemde indicatoren gedefinieerd die uitspraak doen over de mate waarin een kwaliteitseigenschap aanwezig is. Een typische indicator

Samenvatting

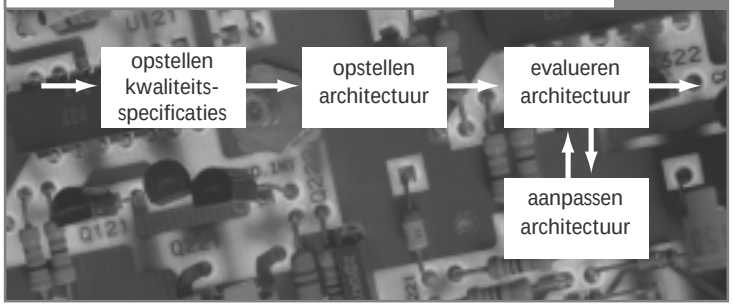
Een architectuur legt de belangrijkste beslissingen omtrent het ontwerp van een systeem vast. Het borgen van kwaliteitseisen die aan een systeem worden gesteld, moet daarom in de eerste plaats in de architectuur plaatsvinden. Dit blijkt in de praktijk niet eenvoudig. Dit artikel beschrijft daarom een praktische aanpak voor het ontwerpen van een kwalitatief hoogwaardige architectuur.

gaat vergezeld van een meetvoorschrift dat aangeeft welke methode en stappen dienen te worden gebruikt om de waarde van de indicator te bepalen. Naast gebruik van de gegeven indicatoren kan er voor een specifiek systeem nagedacht worden over situatiespecifieke indicatoren en meetvoorschriften. Bij het opstellen van een kwaliteitsspecificatie kunnen verschillende middelen worden gebruikt, zoals het interviewen van gebruikers of het bespreken van een bestaand systeem als referentiekader.

De volgende stap bestaat uit het opstellen van een eerste versie van de architectuur, gebaseerd op de functionele eisen die aan het systeem worden gesteld. Hierbij worden de belangrijkste conceptuele componenten in het systeem geïdentificeerd. Deze conceptuele componenten of archetypen geven een goed beeld van de belangrijkste concepten in het probleemdomein, maar komen niet noodzakelijkerwijs overeen met abstracties die ontstaan bij een objectgeoriënteerde analyse van het probleemdomein.

Na een eerste opzet van een architectuur dient te worden bepaald in hoeverre deze tegemoet komt aan de kwaliteitseisen die aan het systeem zijn gesteld. Uitgangspunten voor deze evaluatie zijn de indicatoren die in de kwaliteitsspecificatie zijn gedefinieerd. Er kunnen verschillende methoden van evaluatie worden geïdentificeerd, onderverdeeld in kwalitatieve en kwantitatieve methoden. Voorbeelden van

Aanpak voor architectuurontwerp 1



technieken zijn het gebruik van scenario's, simulaties, wiskundige modellen of reviews van experts. Er zijn specifieke varianten van deze algemene methoden nodig voor specifieke kwaliteitseigenschappen. Ter illustratie beschrijven we in de volgende paragraaf twee specifieke methoden voor het bepalen van performance en onderhoudbaarheid. Het resultaat van de evaluatie is een indicatie van de mate waarin de kwaliteit in de architectuur aanwezig is.

Op basis van de evaluatie kan besloten worden de architectuur aan te passen door gebruik te maken van bekende oplossingsrichtingen. Een steeds groter wordende gemeenschap richt zich hierbij op het beschrijven van standaardpatronen en -stijlen voor architecturen (zie bijvoorbeeld Buschmann e.a., 1996 en Schmidt e.a., 2000). Het probleem van deze initiatieven is dat zij niet gedreven zijn door (kwaliteits)eisen en eigenlijk alleen een lijst van standaardoplossingen beschrijven. Er is dus behoefte aan oplossingen die redeneren vanuit de kwaliteit. Een aanzet daartoe wordt gegeven door Klein en Kazman (1999) die standaardarchitectuurstijlen koppelen aan specifieke kwaliteitseigenschappen en manieren om over de specifieke kwaliteitseigenschap te redeneren. Een overzicht van een aantal mogelijke architecturele maatregelen is weergegeven in figuur 2.

Merk op dat het nemen van maatregelen niet automatisch leidt tot een goede architectuur omdat maatregelen vaak (een mogelijke negatieve) invloed hebben op andere kwaliteitseigenschappen. Zo blijkt bijvoorbeeld in de praktijk aanpasbaarheid een negatieve invloed te hebben op performance (Lundberg e.a., 1999). De echte uitdaging bestaat dan ook uit het maken van de juiste afwegingen om te komen tot een optimale architectuur (Kazman e.a., 1998; Bosch, 2000).

Evalueren van architectuur

Ter illustratie worden hier twee methoden beschreven voor het evalueren van specifieke kwaliteitseigenschappen. Zo is er een aanpak ontwikkeld voor het in een zo vroeg mogelijk stadium modelleren van de performance van een systeem (Hoeben en Sterk, 1998). De

Maatregelen voor het verbeteren van kwaliteitseigenschappen 2

kwaliteitseigenschap	maatregel
betrouwbaarheid	• redundant uitvoeren van componenten • stemmechanisme voor resultaten
portabiliteit	• gebruik van standaarden voor omgeving • laag om omgevingsspecifieke details heen
onderhoudbaarheid	• goede scheiding van verantwoordelijkheden • verbergen van mogelijke variatie achter een component • gemeenschappelijk gebruik van componenten in een productfamilie • ontkoppeling van componenten middels een 'event bus'
efficiëntie	• scheiden tijdkritische componenten van andere componenten • gebruik asynchrone communicatie • statische allocatie van hulpbronnen

noodzaak voor deze aanpak is dat het in de praktijk blijkt dat metingen voor het bepalen van de performance pas (kunnen) worden uitgevoerd als een eerste versie van een systeem bestaat. In dat stadium zullen aanpassingen in de architectuur relatief kostbaar zijn.
De aanpak bestaat uit het maken van een model van het gebruik, de onderdelen en de infrastructuur van een systeem. Voor het gebruik is het van belang te weten wat gebruikers gemiddeld genomen gedurende een drukke periode met de software doen. Bij het modelleren van de onderdelen wordt beschreven hoe deze onderdelen precies samenwerken om gebruikerstaken te vervullen en welke diensten van apparaten ze daarbij nodig hebben. De beschrijving van de infrastructuur bevat deze apparaten en geeft aan hoe snel ze zijn. Hierbij is het dus niet van belang hoe deze apparaten intern werken. Deze modelinformatie wordt tezamen met schattingen over de performance ingevoerd als UML-model in een standaard CASE-tool.
Op basis van het model wordt er dan een interne simulatie van het systeem uitgevoerd waarbij de performancegegevens worden doorgerekend. Hierbij wordt gebruikgemaakt van queuingtheorie om te bepalen wat de gemiddelde wachttijd voor apparaten zal zijn. Deze theorie (zie bijvoorbeeld Smith, 1990) is er op gebaseerd dat de wachttijd per verzoek steeds verder toeneemt naarmate het aantal verzoeken stijgt. Uitkomst van de berekening is een overzicht van de performance van het systeem, onderverdeeld naar de verschillende onderdelen. Gedurende het ontwikkelproces kan er een steeds beter beeld ontstaan van de performance door resultaten van metingen aan het systeem zelf te gebruiken om het model te verfijnen.

Evalueren van onderhoudbaarheid

Een tweede kwaliteitseis die voor vrijwel ieder systeem een rol speelt is onderhoudbaarheid. Verschillende onderzoeksresultaten geven aan dat tot 80% van de totale kosten van een softwareproduct na de eerste release besteed worden. Daarom is er een techniek ontwikkeld waarmee kwantitatieve uitspraken kunnen worden gedaan over de te verwachten onderhoudskosten op basis van een analyse van de softwarearchitectuur (Bosch, 2001).
Een probleem dat hiervoor dient te worden opgelost is het feit dat kwaliteitseisen vaak uiterst onnauwkeurig gedefinieerd zijn. Hiervoor wordt dan ook het concept 'scenarioprofiel' geïntroduceerd. Een scenarioprofiel is een lijst van de meest waarschijnlijke of belangrijkste scenario's voor een softwareproduct. Het soort scenario is afhankelijk van de kwaliteitseigenschap, maar voor het evalueren van onderhoudbaarheid wordt gebruikgemaakt van een onderhoudsprofiel: een lijst met de meest waarschijnlijke onderhoudsscenario's. Op basis van een profiel kan een kwaliteitseis nauwkeurig worden gedefinieerd, bijvoorbeeld: 'Uitgaande van dit onderhoudsprofiel en 15 veranderingsverzoeken per jaar mogen de onderhoudskosten niet meer dan 100.000 gulden per jaar bedragen.' Dit is een expliciet en kwantitatief gedefinieerde eis op basis waarvan een evaluatie kan worden uitgevoerd.
De voorgestelde methode voor het evalueren van de onderhoudbaarheid van een softwarearchitectuur bestaat uit vier stappen (Bengtsson e.a., 2001). Als eerste moet het doel van de evaluatie gekozen worden. Zo kan naast een voorspelling van de onderhoudskosten ook de inflexibiliteit van een architectuur of het vergelijken van twee alternatieven van belang zijn. Daarnaast moet het onderhoudsprofiel gedefinieerd worden. De daarbij behorende scenario's kunnen topdown (bijvoorbeeld op basis van een aantal vooraf gedefinieerde

categorieën) of bottom-up (gebaseerd op gesprekken met belanghebbenden) geïnventariseerd worden. De derde stap is het evalueren van de architectuur op basis van het scenarioprofiel. Hierbij identificeert de architect de componenten die door een scenario veranderd moeten worden, de geraakte operatie(s) van ieder van deze componenten en de eventuele bijeffecten. Als laatste wordt het resultaat van de evaluatie geïnterpreteerd en kan gekomen worden tot een conclusie. In het geval van het voorspellen van onderhoudskosten kan dit een kwantitatieve conclusie zijn in termen van de verwachte kosten per jaar.

Hoewel het evalueren van een kwaliteitseigenschap van groot belang is, leert de ervaring dat softwarearchitecten zich onmiddellijk afvragen in hoeverre de softwarearchitectuur optimaal is. Bosch (2001) presenteert een techniek waarmee op basis van een impactanalyse van het onderhoudsprofiel, uitspraken gedaan kunnen worden over de optimale en worst-case onderhoudskosten. Dit geeft informatie over de grenzen waarbinnen de onderhoudskosten zich kunnen bevinden voor een willekeurige architectuur en daarmee dus in hoeverre de huidige architectuur optimaal is voor onderhoudbaarheid.

Drs. Danny Greefhorst
is werkzaam als senior adviseur bij het Software Engineering Research Centre te Utrecht.

Prof.dr.ir. Jan Bosch
is hoogleraar software engineering aan de Rijksuniversiteit Groningen.

Conclusie

Een systeem is pas succesvol als het voldoet aan de kwaliteitseisen die eraan gesteld worden. Het borgen van deze kwaliteit vindt primair plaats in de architectuur. In dit artikel is daarom een aanpak voor het ontwerpen van architecturen gepresenteerd waarbij het borgen van kwaliteitseisen centraal staat. Hierbij dient vooraf een kwaliteitsspecificatie te worden opgesteld en een eerste versie van een architectuur gebaseerd op functionaliteit te worden ontwikkeld. In meerdere iteraties vinden daarna evaluaties van de architectuur plaats voor specifieke kwaliteitseigenschappen en kan op basis daarvan besloten worden aanpassingen in de architectuur aan te brengen om beter te voldoen aan bepaalde kwaliteitseisen. Hierbij kan gebruikgemaakt worden van standaardoplossingen, maar ligt de werkelijke uitdaging in het maken van de juiste afwegingen tussen kwaliteitseigenschappen.

Literatuur

Bengtsson, P.O., e.a., Analyzing software architectures for modifiability, submitted, 2001.

Bosch, J., *Design and Use of Software Architectures – Adopting and Evolving a Product-Line Approach*, Addison-Wesley, 2000.

Bosch, J., Bengtsson, P.O., Assessing optimal software architecture maintainability, European Conference on Software Maintenance and Reengineering, 2001.

Buschmann, F., e.a., *Pattern-Oriented Software Architecture – A System of Patterns*, Wiley, 1996.

Kazman, R., e.a., The architecture tradeoff analysis method, *Proc. Fourth IEEE Intern. Conf. on Engineering of Complex Computer Systems (ICECCS98)*, Monterey, 1998, p. 68-78.

Klein, M., Kazman, R., Attribute-based architectural styles, Software Engineering Institute, Technical Report, CMU/SEI-99-TR-022, 1999.

Lundberg, L., e.a., Quality attributes in software architecture design, *Proc. IASTED 3rd Intern. Conf. on Software Engineering and Applications*, 1999, p. 353-362.

Hoeben, F., Sterk, M., Performance modellering, white paper, Software Engineering Research Centre, 1998.

Schmidt, D., e.a., *Pattern-Oriented Software Architecture – Patterns for Concurrent and Networked Objects*, Wiley, 2000.

Smith, C.U., *Performance Engineering of Software Systems*, Addison-Wesley, 2000.

Zeist, B. van, e.a., *Kwaliteit van softwareproducten, praktijkervaring met een kwaliteitsmodel*, Kluwer, 1996.