Software-architectuur is een relatief jonge discipline die bij veel bedrijven nog een duidelijke plaats moet krijgen. Een praktisch probleem is het gebrek aan een uniforme standaard voor de precieze invulling van software-architectuur. Dit artikel gaat in vogelvlucht over de definitie en ontwikkelingen van software-architectuur heen en geeft aan hoe het moet worden gepositioneerd ten opzichte van andere architecturen.

Software-architectuur in vogelvlucht

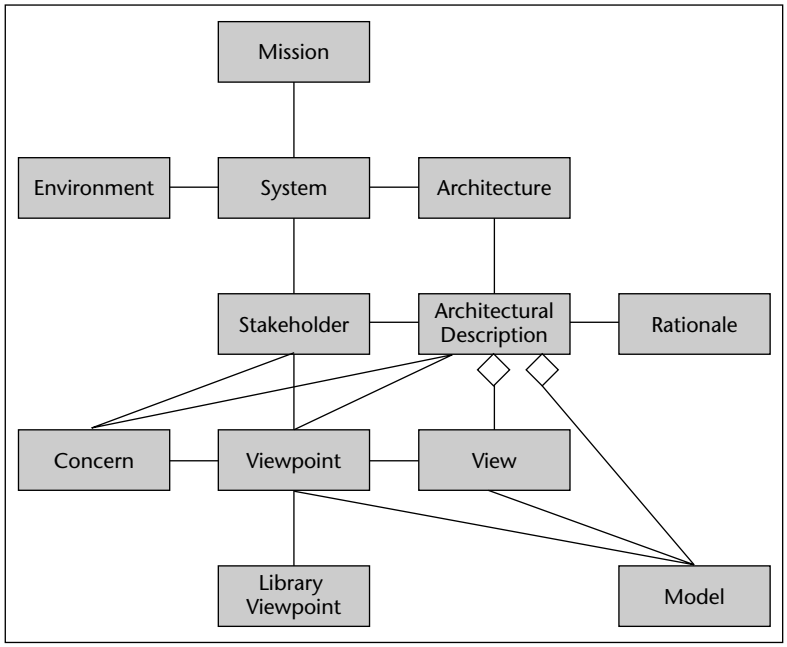
Positionering en ontwikkelingen

Er is de afgelopen jaren veelvuldig aandacht geweest voor software-architectuur; er is een standaard voor architectuurbeschrijving, er is een aantal belangrijke boeken verschenen, er zijn academische artikelen gepubliceerd en ontwikkelmethoden hebben het begrip een plaats gegeven. Toch is er in de praktijk onduidelijkheid over de precieze invulling van software-architectuur. Dit verschijnsel is niet uniek voor software-architectuur; de term "architectuur" in het algemeen lijkt nog onvoldoende afgebakend. Dit artikel geeft een overzicht van de ontwikkelingen op het gebied van software-architectuur en de onduidelijkheden die het toepassen van deze theorie in de praktijk oplevert. In een poging meer duidelijkheid te scheppen wordt software-architectuur gepositioneerd ten opzichte van andere architecturen. Het artikel start met een verkenning van het begrip software-architectuur.

DEFINITIE Wat is nu precies software-architectuur? Als je het eenvoudig zou willen uitleggen zou je kunnen zeggen dat de software-architectuur het hoog-niveau ontwerp van een systeem is. Het beschrijft de belangrijkste eigenschappen van het systeem en is de primaire plaats om kwaliteitseigenschappen te borgen. Een alternatieve definitie is dat een software-architectuur die aspecten van een systeem beschrijft die het moeilijkst kunnen worden veranderd. De eerste echte ideeën over software-architectuur komen uit het begin van de jaren negentig. Kern van deze ideeën is dat componenten en hun relaties de basiselementen van software-architectuur zijn. Componenten moeten hierbij heel breed worden opgevat; het hangt van je gezichtspunt af wat je er precies mee bedoelt. Een aantal invloedrijke boeken zoals die van Shaw en Garlan [Shaw 96] en Bass [Bass 98] dragen bij aan de evolutie van het begrip. De belangrijkste ontwikkeling van de afgelopen jaren is de P1471 standaard voor architectuurbeschrijving [IEEE 2000], zoals die door ANSI en IEEE is gestandaardiseerd. De standaard geeft definities van een aantal belangrijke architectuurbegrippen, zoals de volgende definitie van architectuur: *The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.*

CONCEPTUEEL MODEL Er staan in deze definitie een aantal kernzaken. Niet alleen beschrijft een architectuur componenten en hun samenhang, ook de relatie met de omgeving is van belang. Naast deze meer modelmatige benadering van architectuur geeft de definitie aan dat architectuur ook procesmatige aspecten bevat. In het bijzonder geeft het principes voor het werken (ontwerpen) onder architectuur en zou een architectuur expliciet aandacht moeten besteden aan evolutieaspecten zoals migratie. Merk tenslotte op dat de definitie niet specifiek redeneert over softwaresystemen en dus ook gebruikt kan worden voor andere vormen van architectuur.

Een andere belangrijke bijdrage van de P1471 standaard is een conceptueel model waarin de samenhang tussen de verschillende architectuurbegrippen is beschreven. Figuur 1 geeft een hoog-niveau overzicht van dit conceptueel model. Er zijn een aantal zaken die opvallen aan dit model. Als eerste valt de strikte schei-

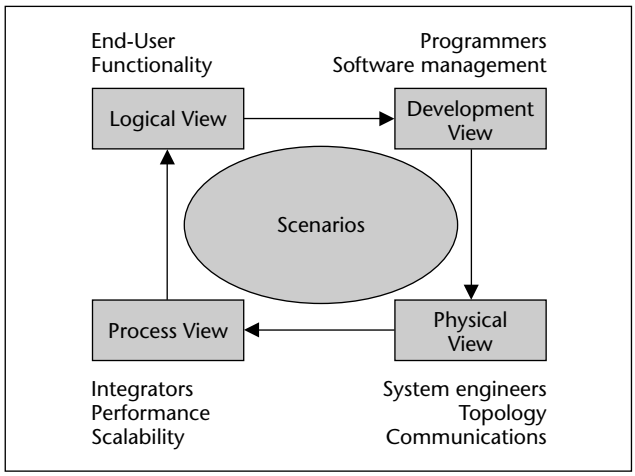


FIGUUR 1. P1471 conceptueel model

ding tussen architectuur en architectuurbeschrijving (de klassen "architecture" en "architectural description") op. Het achterliggende idee is dat een systeem een inherente architectuur heeft die wordt geëxpliciteerd in een architectuurbeschrijving. Een ander belangrijk inzicht is dat er meerdere gezichtspunten op architectuur zijn, die in P1471 viewpoints worden genoemd. Het komt erop neer dat een architectuur complex is en voor het begrip het best vanuit verschillende complementaire gezichtspunten kan worden gepresenteerd. Een architectuurbeschrijving voor een specifiek systeem bevat views met bijbehorende modellen die voldoen aan de viewpoints. Een laatste belangrijk inzicht is dat viewpoints altijd worden gedefinieerd voor concerns (belangen) van bepaalde stakeholders (belanghebbenden). Zo zal een opdrachtgever in geheel andere aspecten van een architectuur geïnteresseerd zijn dan een programmeur. Het is jammer dat de standaard niet aangeeft welke architectuurbeschrijvingen, viewpoints, stakeholders en concerns er in het algemeen zijn; dit wordt overgelaten aan de individuele architect.

OVERZICHT VAN ONTWIKKELINGEN Nu we een beeld hebben van wat software-architectuur is zal ik kort de meest relevante ontwikkelingen van de afgelopen jaren schetsen. Daarbij relateer ik de ontwikkelingen aan de P1471 standaard. In Figuur 3 is daarvoor met nummers weergegeven waar in het P1471 conceptueel model de ontwikkelingen kunnen worden gepositioneerd.

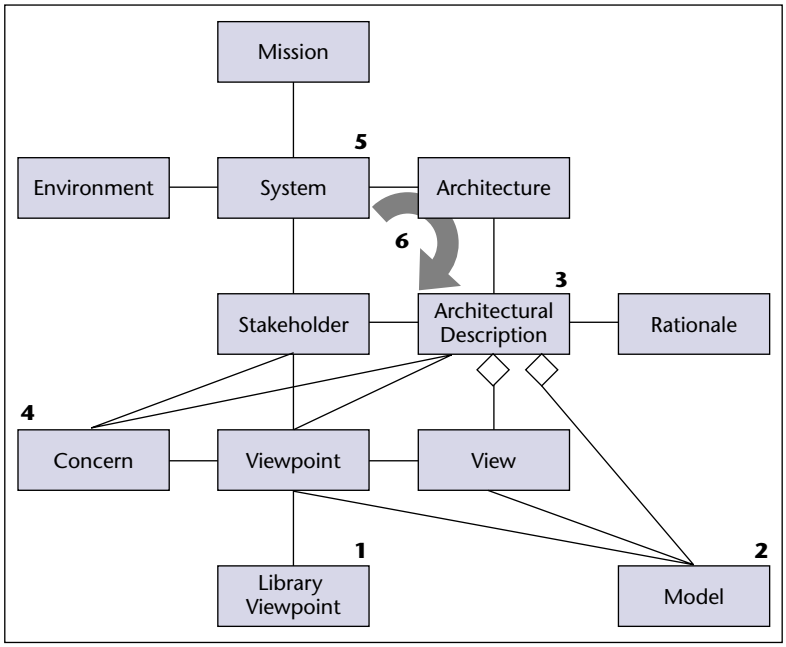
1. *Architectuurraamwerken* bieden standaard viewpoints ("library viewpoints" in P1471 terminologie), die in één of meer dimensies zijn gerangschikt. De grondlegger van de architectuurraamwerk-gedachte is Zachman [Zachman 87] die zijn raamwerk voor informatiesystemen indeelde in de dimensies stakeholders en soort informatie. Mogelijke stakeholders zijn: de planner, de eigenaar, de ontwerper, de bouwer en de onderaannemer van een informatiesysteem. De soorten informatie die Zachman onderscheidt zijn gegevens, functies, lokaties, mensen, tijd en motivatie. Een bekend software-architectuurraamwerk is het 4+1 model [Kruchten 95] (zie Figuur 2) dat vijf viewpoints definieert (logical, development, process, physical, scenario's). Deze viewpoints zijn onderverdeeld naar de betrokken stakeholders en hun concerns. De gezichtspunten hebben een herkenbare relatie naar respectievelijk gebruikers (klassen), ontwikkelaars (packages en bestanden), systeemintegrators (processen, berichten) en infrastructuur ontwerpers (nodes en netwerken). Het vijfde gezichtspunt bevat scenario's die beschrijven hoe de eenheden in de andere gezichtspunten met elkaar samenwerken.
2. *Architectuurpatronen* zijn, analoog aan ontwerppatronen, standaard oplossingen voor veel voorkomende problemen. Je zou kunnen zeggen dat patronen standaard modellen, of delen van modellen bieden. In tegenstelling tot ontwerppatronen beschrijven architectuurpatronen standaard samenwerkingen van componenten op macroniveau. Deze patronen bouwen ook verder op de eerdere ideeën rond architectuurstijlen. Een architectuurstijl beschrijft een standaard samenwerking van componenten, inclusief relaties en beperkingen, maar los van een specifiek probleem. Denk bijvoorbeeld aan een gelaagde architectuurstijl waarbij componenten alleen kunnen communiceren met componenten in een onderliggende laag.
3. In het kader van architectuurbeschrijving is veel onderzoek gedaan naar *architectuurbeschrijvingstalen (ADLs)*. Deze talen bieden typisch een formele architectuurbeschrijving die wordt uitgedrukt in interfaces, relaties en functionaliteit van componenten.



FIGUUR 2. Het 4+1 model

Het uiteindelijke doel is het automatisch kunnen redeneren over- en het genereren van architectuuraspecten van software. Bekende ADLs zijn Wright, Rapide, Acme en UniCon. Er zijn zeer weinig voorbeelden van commerciële ADLs, alhoewel UML ook als ADL kan worden beschouwd.

4. *Architectuurevaluatie* houdt zich bezig met het bepalen van de kwaliteit van een architectuur. Kwaliteitseigenschappen zijn specifieke concerns en je zou kunnen zeggen dat architectuurevaluatie zich bezig houdt met de mate waarin een architectuurbeschrijving voldoet aan de concerns van de verschillende stakeholders. Door een architectuur vroegtijdig te evalueren kan het verdere traject worden bijgestuurd en kunnen alternatieve oplossingsrichtingen worden onderzocht. Er is vooral veel aandacht geweest voor scenario-gebaseerde methoden zoals SAAM en ATAM van het software engineering institute. In dergelijke benaderingen worden meerdere architectuurscenario´s tegen elkaar afgezet op basis van kwaliteitsattributen.
5. *Product-line ontwikkeling* is erop gericht de basis voor gelijksoortige systemen éénmalig te ontwikkelen. In feite zijn product-lines een uitbreiding op het P1471 model, waarbij naast systemen ook systeemfamilies worden onderscheiden. Bij product-line ontwikkeling worden de ontwikkelprocessen gescheiden in domain engineering processen en application engineering processen, waarbij in de eerste de gemeenschappelijke en in de tweede de systeemspecifieke ontwikkeling plaats vindt. Een product-line architectuur is het resultaat van domain engineering en beschrijft de gemeenschappelijke architectuur, inclusief de eigenschappen die per systeem verschillen.
6. *Architectuurreconstructie* gaat over het extraheren van architectuurinformatie uit bestaande systemen. In P1471 terminologie is dat het extraheren van de architectuur uit het systeem en het vertalen daarvan in een architectuurbeschrijving. Architectuurreconstructie is erg belangrijk als je bedenkt dat veel software slecht (of niet) gedocumenteerd is en de beste informatie uit het systeem zelf gehaald kan worden. Het visualiseren van architectuurinformatie levert dan ook veel inzicht op en zorgt ervoor dat aanpassingen sneller kunnen worden doorgevoerd doordat de relevante plaats snel kan worden gevonden.



FIGUUR 3. Ontwikkelingen afgebeeld op het conceptueel model

SOFTWARE-ARCHITECTUUR IN DE PRAKTIJK Zoals aangegeven is er op het gebied van software-architectuur in theorie veel gebeurd. Het is dan ook interessant om te kijken naar de ervaringen met software-architectuur in de praktijk. Het is positief te zien dat software-architectuur op de agenda van organisaties is komen te staan. Geïnspireerd door ontwikkelmethoden die software-architectuur een plaats hebben gegeven beginnen organisaties is de rol van software-architect te onderkennen. Helaas lopen software-architecten in de praktijk aan tegen onduidelijkheden over de precieze rol van de software-architect, inhoud van een software-architectuur, en afbakening van software-architectuur met andere architecturen. In termen van P1471 is er met name onduidelijkheid over de te onderkennen architectuurbeschrijvingen en te hanteren viewpoints. Na kort stil te staan bij het laatste punt zal ik daarom software-architectuurbeschrijvingen positioneren ten opzichte van andere architectuurbeschrijvingen.

De soorten informatie die Zachman onderscheidt zijn gegevens, functies, locaties, mensen, tijd en motivatie

INZICHT IN FUNCTIONALITEIT Het lijkt vreemd dat er onduidelijkheid bestaat over de te hanteren viewpoints gezien de beschikbaarheid van architectuurraamwerken zoals het 4+1 model. Het probleem is dat de viewpoints in veel raamwerken nog niet in detail zijn uitgewerkt waardoor er nog heel veel aan de interpretatie van de individuele architect wordt overgelaten. Een ander probleem is dat veel raamwerken erg gericht zijn op technische (embedded) systemen en dat de bijbehorende viewpoints voor informatiesystemen veel minder relevant zijn. Het klakkeloos toepassen van deze viewpoints leidt dan tot veel redundantie, omdat

Adv.

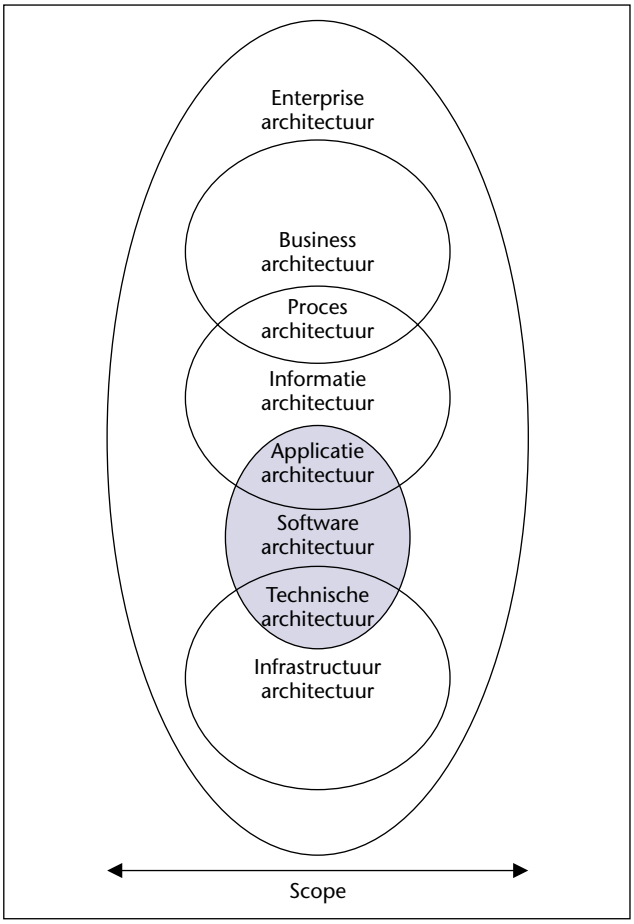
modellen dan erg veel overlap gaan vertonen. Tenslotte is nog niet een wereldwijd uniform standaard architectuurraamwerk en moeten we leven met veel verschillende raamwerken. In de praktijk is het belangrijk je bij het schrijven van een software-architectuurrapport sterk te richten op de belangrijkste stakeholders en de concerns die zij hebben bij het systeem. Het document zal immers alleen gebruikt worden als de stakeholders het als nuttig ervaren. De toegevoegde waarde van de architectuur zal hierdoor sterk worden verhoogd. De belangrijkste stakeholders voor een software-architectuur zijn de ontwerpers, ontwikkelaars en beheerders. Met name de beheerders krijgen nog wel eens minder aandacht, terwijl zij uiteindelijk de meeste tijd aan het systeem zullen besteden. Voor alle stakeholders geldt dat zij van de software-architectuur verwachten dat deze inzicht geeft in de functionaliteit en werking van het systeem. Probeer daarom verschillende overzichtsmodellen op te nemen, in abstractieniveau variërend van conceptueel tot fysiek. Een ander belangrijk doel van een software-architectuur is voldoende informatie te bieden zodat ontwikkeling van componenten parallel kan plaats vinden. Een goede beschrijving van de interface van de componenten is hierbij onontbeerlijk.

Positionering van software-architectuur

Er blijkt vaak veel onduidelijkheid over de relatie van een software-architectuurbeschrijving met andere architectuurbeschrijvingen die in een organisatie worden gemaakt. Een belangrijke oorzaak hiervoor is dat er veel verschillende dimensies zijn waarin architecturen van elkaar kunnen verschillen en dat deze dimensies vrijwel nooit expliciet worden gemaakt. Ook architectuurraamwerken blijken te weinig antwoorden te geven op deze vragen; in plaats van de term “software-architectuur” worden nog al eens woorden als “applicatie-architectuur”, “technische architectuur” en “technologie architectuur” gebruikt. Het gebrek aan standaardisering is een groot probleem. Er moet vaak diep worden doorgevraagd om duidelijk te krijgen wat er precies van een architectuurbeschrijving wordt verwacht.

Alhoewel ik niet de intentie heb een nieuw architectuurraamwerk te verzinnen wil ik toch proberen de plaats van software-architectuur te verduidelijken. In Figuur 4 positioneer ik de verschillende soorten van architectuur ten opzichte van elkaar, inclusief hun overlap. De hoofdarchitecturen die ik onderken zijn:

- Enterprise architectuur: alle mogelijke aspecten organisatiebreed op hoog niveau beschouwd.
- Businessarchitectuur: de doelen, producten, diensten, structuren, middelen en processen van een business van een organisatie.



Figuur 4. Positionering van software-architectuur

- Informatie-architectuur: de informatie en conceptuele componenten benodigd om een business te ondersteunen.
- Software-architectuur: de conceptuele, logische, ontwikkel, deployment en runtime componenten van een specifiek softwaresysteem.
- Infrastructuurarchitectuur: de infrastructurele hardware en software zoals computers, netwerken, besturingssystemen en middleware.

Nuancering Naast deze hoofdarchitecturen onderken ik ook deelarchitecturen die de overlap vormen tussen hoofdarchitecturen, maar ook vaak als zelfstandige architectuur worden genoemd. Zo behelst de procesarchitectuur de processen voor het ondersteunen van een business van een organisatie. De applicatie-architectuur bestaat uit de conceptuele componenten en services

In methoden als SAAM en ATAM worden meerdere architectuurscenario’s tegen elkaar afgezet op basis van kwaliteitsattributen

van een specifiek systeem. De technische architectuur bevat tenslotte de infrastructurele benodigdheden van een specifiek systeem, inclusief infrastructurele delen van de zelf te ontwikkelen software. Ook is in de figuur de scope dimensie weergegeven, die beschrijft hoe groot het aandachtsgebied van een architectuur is. Daar waar andere architecturen betrekking hebben op een specifieke business of een specifiek domein in een organisatie, heeft de software-architectuur vaak betrekking op één specifiek systeem. De scope van de architectuur heeft ook betrekking op het moment in tijd waarop deze wordt ontwikkeld; een software-architectuur zal typisch als laatst in het rijtje van architecturen worden ontwikkeld.

De belangrijkste nuancering die hierin gemaakt moet worden is dat een software-architectuur wel betrekking kan hebben op meerdere systemen. Een goed voorbeeld hiervan is een product-line architectuur, die systemen beschrijft die dezelfde primaire functionaliteit bieden. Een andere mogelijkheid is dat er een software-architectuur wordt gemaakt voor systemen die niet zozeer dezelfde primaire functionaliteit bieden, maar wel anderszins veel op elkaar lijken. Een dergelijke standaard software-architectuur blijkt in de praktijk erg nuttig om gelijkvormigheid over meerdere systemen te creëren en wordt gemaakt voordat de software-architectuur voor een specifiek systeem wordt ontwikkeld.

Conclusies Er is de afgelopen jaren veel gebeurd op het gebied van software-architectuur, zowel in de academische wereld als in het bedrijfsleven. Veel bedrijven weten echter nog steeds niet precies wat er verwacht moet worden van een software-architectuur. In dit artikel is hierin helderheid gebracht door aan te geven waar we staan op het gebied van software-architectuur en hoe software-architectuur zich verhoudt tot andere architecturen.

Referenties

- [Bass 98] L. Bass, P. Clements, R. Kazman: "Software Architecture in Practice", Addison Wesley, 1998.
- [IEEE 2000] IEEE Std 1471-2000: "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems", 2000.
- [Kruchten 95] P. Kruchten, "The 4+1 View Model of Architecture," IEEE Software, November 1995, pp. 42–50.
- [Shaw 96] M. Shaw, D. Garlan: "Software Architecture: Perspectives on an Emerging Discipline", Prentice Hall, 1996.
- [Zachman 87] J. Zachman: "A Framework for Information Systems Architecture", IBM Systems Journal 26, No. 3, 1987.

Danny Greefhorst is werkzaam als IT Architect bij IBM Business Consulting Services en bereikbaar via greefhorst@nl.ibm.com

PATCHES Patches PATCHES Patches PATCHES Patches PATCHES

Nederlandse Java User Group (NL-JUG) van start

Per april 2003 gaat de Nederlandse Java User Group van start, het enige officiële informatie- en overleg platform voor iedereen die actief is met Java.

Nederland kent tot heden een kleine groep Java-gebruikers die zich verenigd hebben in "Dutch Melange". Deze organisatie staat volledig achter de NL-JUG en zal derhalve opgaan in de nieuwe Java User Group.

Het oprichten van de NL-JUG komt voort uit een wereldwijd 'harmonisatieproces' van Java user groups. Jolanda Beerse en Luc Pennings zullen hierin voor Nederland het voortouw nemen. Beiden zijn reeds geruime tijd actief in technische en marketing functies binnen de ICT-branche en zijn naar eigen zeggen 'honderd procent Java-supporter'. Pennings zegt hierover: "Nu Java geen hype meer is maar een common good is de tijd rijp om een onafhankelijke organisatie op te zetten die de belangen van Java en het grote aantal Java-gebruikers in Nederland (40.000+) vertegenwoordigt en de verdere ontwikkeling van Java ondersteunt." Voor meer informatie en aanmelding ga naar http://www.nljug.org of stuur een e-mail naar info@nljug.org.