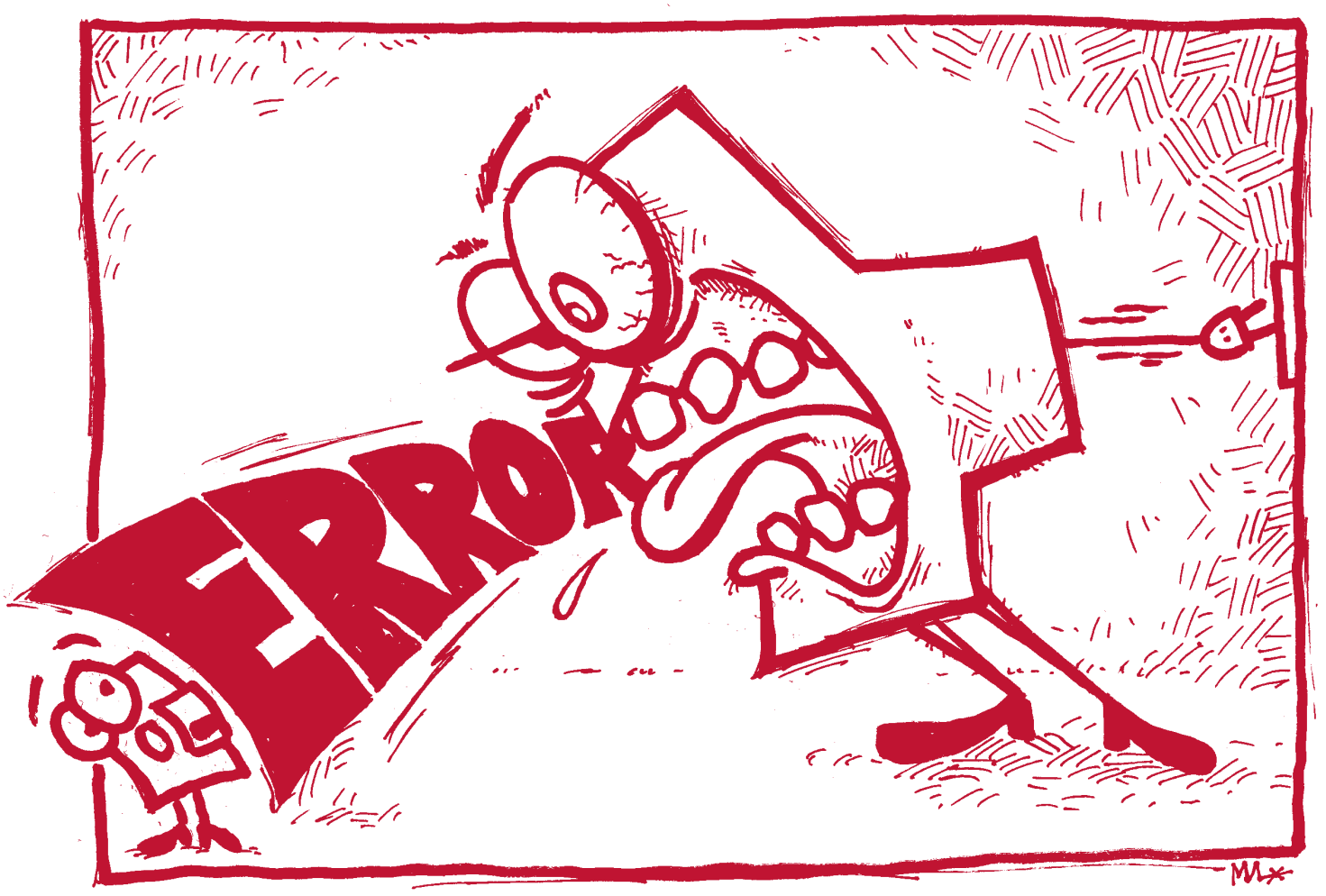
Slinger Jansen

DELIVER-ONDERZOEK

INFORMATIESYSTEMEN EN HET BELANG VAN DE UPDATE KNOP

Informatiesystemen, zoals bijvoorbeeld bibliotheekprogramma's, evolueren. Dat stelt zowel de ontwikkelaars als de gebruikers van zo'n systeem voor problemen. Want aan welke eisen moet zo'n informatiesysteem voldoen om klaar te zijn voor de toekomst? Slinger Jansen beschrijft een aantal voorbeelden uit de bibliotheekpraktijk en gaat in op het Deliver-onderzoeksproject ter verbetering van de levering van software-informatiesystemen.



De aanschafprocedure van een informatiesysteem is een complex proces. Er zijn bijvoorbeeld veel verschillende bibliotheekprogramma's beschikbaar, maar de kans dat een van deze systemen precies de informatiebehoefte van een organisatie dekt is klein. Dit leidt vaak tot het toevoegen van functionaliteiten aan een bestaand informatiesysteem, door het bijbouwen en verder aanpassen van de software. Naast het feit dat informatiesystemen lang niet alle functionaliteiten bevatten die een organisatie nodig heeft, evolueren informatievoorzieningen en -systemen ook. Impliciet betekent dit dat de architectuur van het informatiesysteem evolutionaire veranderingen moet kunnen ondersteunen.

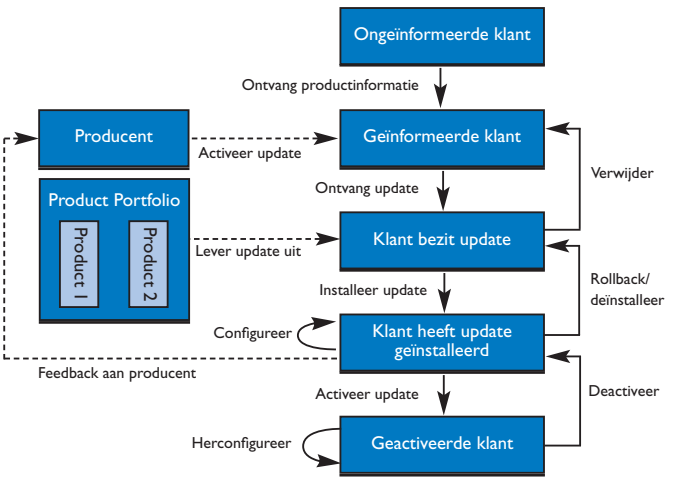
Complexiteiten

Bij een casestudie kwam het voorbeeld naar voren van twee versies van een documentmanagementsysteem, waarin verscheidene werknemers samen aan documenten kunnen werken. In de tweede versie van het systeem kunnen verschillende varianten van een document worden bewaard, terwijl in de eerste versie slechts één variant kan worden bewaard. Daarnaast worden bij klanten door klanten en derden speciale readers voor specifieke documenttypen gebouwd. Al snel bleek de softwareproducent in de problemen te komen.
Als een werknemer met een eerste versie client op de tweede versie van de documentmanagementsysteem-server probeerde in te loggen, kon de werknemer alleen de eerste variant van een document zien. Daarnaast maakte de update een aantal van de readers onbruikbaar omdat de readers op de oude architectuur gebouwd waren. Dit resulteerde vervolgens in een grote hoeveelheid on-site ontwikkelwerkzaamheden.

Uit dit voorbeeld is een aantal complexiteiten te abstraheren die producenten van informatiesystemen moeten managen. Zo zijn er verschillende componenten, in het voorbeeld de client en de server, die afhankelijk van elkaar zijn. Daarnaast kan de software ook afhankelijk zijn van externe componenten, zoals het operating systeem en databasemanagement systemen die op hun beurt ook weer apart evolueren.
Bovendien zijn er de klantspecifieke aanpassingen waarmee moeilijk rekening te houden is, omdat deze niet expliciet door de producent worden gebouwd en onderhouden. De producent heeft waarschijnlijk ook nog verschillende versies van een systeem uitstaan. Dit vanwege het feit dat klanten op verschillende momenten het product hebben aangeschaft en omdat niet elke klant de volledige verzameling componenten van een informatiesysteem aanschaft, maar vaak een subset van de componenten die een producent verkoopt. Voeg hier nog aan toe dat elke klant het product zelf kan inrichten naar de eigen wensen en er ontstaat een complex probleemdomein.

Een voorbeeld

Behalve dat bovengenoemde complexiteiten ontstaan, moet het product ook nog daadwerkelijk bij de klant worden afgeleverd. Dit proces, waarbij een klantconfiguratie wordt gewijzigd, noemen we een *update*. Het updateproces van een klantconfiguratie omvat een aantal aspecten, zoals de aflevering van de update (bijvoorbeeld via internet), het informeren van (potentiële) klanten over updates, het ondersteunen van het installatieproces, en het activeren van een informatiesysteem met de juiste licentie.
Bij een recente evaluatie van software updatetechnieken hebben we het model opgesteld zoals te zien is in *figuur 1*. In deze figuur wordt het updateproces en de interactie tussen klant en softwareproducent weergegeven. De klant doorloopt hierbij vijf stadia, van ongeïnformeerde klant zonder weet van de update, tot een geactiveerde klant, die het product geactiveerd heeft met een licentie en het product daadwerkelijk gebruikt. Het weergegeven proces kan gebruikt worden om te evalueren of een producent alle mogelijkheden benut om een product te evolueren.
Een voorbeeld is de Microsoft Windows Update-dienst, die alle processen dekt. De pop-up in het scherm informeert dat er een update gearriveerd is. Vervolgens kan de update worden gedownload (vaak is dit al eerder door Windows gedaan). Na het downloaden wordt de update geïnstalleerd en geconfigureerd. Tot slot wordt de licentie van het product gecontroleerd.
De feedback naar Microsoft verloopt via de waarschijnlijk wel bekende boodschap 'er is een fout opgetreden in Windows, wilt u dit probleem rapporteren?' Deze feedback geeft de producent van de software de bijzondere mogelijkheid continu te zien hoe de installatie bij een klant verloopt.
Hoewel Windows Update veel oplost, is het maar een versimpelde oplossing voor één product. Het Deliver-onderzoek, dat zich richt op grote informatiesystemen, komt

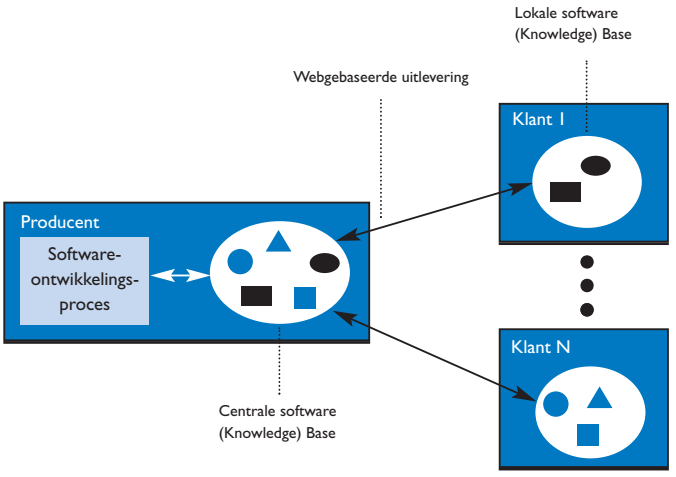


Figuur 1.
Het updateproces en de interactie tussen klant en softwareproducent

veel meer problemen tegen, zoals meer variabiliteiten, aanpassingen van de klant zelf, verschillende operatingsystemen en verschillende klantconfiguraties van een product.

De Deliver-groep

Er zijn doorgaans twee groepen die zich bezighouden met de release, levering en het updaten van software: de ontwikkelaars en de systeembeheerders van informatiesystemen. Het Deliver-onderzoeksproject is opgezet om onderzoek te doen naar de onopgeloste problemen op het gebied van levering van software, maar ook om de Nederlandse software-industrie een extra impuls te geven. Deze impuls is hard nodig, aangezien Nederland in 1999[1] voor 2,1 miljard euro aan software importeerde en voor slechts 0,6 miljard euro exporteerde.
Het Deliver-onderzoek wordt uitgevoerd door twee hoogleraren, een postdoc en twee promovendi. De onderzoekers voeren de casestudies uit bij Nederlandse producenten van informatiesystemen, zoals *Exact*, *Chipsoft*, *Planon* en vele andere. Daarnaast speelt de Deliver-groep een rol in het Platform voor Product Software,[2] een platform waar verschillende bedrijven en wetenschappelijke instellingen informatie uitwisselen over productsoftware en de belangrijke processen daaromheen. Dit forum heeft tot nu toe geleid tot een groot aantal onderzoeksprojecten van studenten en promovendi bij bedrijven.
Ook bestudeert Deliver de resultaten van het *Netherware*-project.[3] Dit project draait om het vak Informatics Business van de studie Informatiekunde aan de Universiteit Utrecht, waar studenten binnen een half jaar een softwareproduct moeten bouwen en lanceren. Intussen werken de studenten tijdens de colleges alsof ze deel uitmaken van een groter bedrijf, en hebben dus ook verschillende nevenfuncties, zoals public relations manager, human resource manager en loonadministrateur, om hen te leren omgaan met een klein tot middelgroot softwarebedrijf. De producten die de studenten ontwikkelen zijn een interessante kweekbodem voor de leverings- en releaseproblematiek die het Deliver-project bestudeert.



Figuur 2.
Gedistribueerde opslag, met de kennis en componenten bij de producent en twee klanten

Eerste conclusies

De kern van het Deliver-onderzoek is het centraal managen van kennis over de uit te leveren software. Als bijvoorbeeld alle relaties tussen componenten vastgelegd zijn, is het installeren van een informatiesysteem in een vreemde omgeving een stuk gemakkelijker, omdat deze informatie gebruikt kan worden om voor de installatie te kijken of het systeem überhaupt wel gaat werken in deze configuratie van componenten.
De belangrijkste kennis concentreert zich rondom de licentie-informatie, de klantspecifieke informatie, de componentrelaties en de diagrammen die weergeven in welke configuraties een product kan voorkomen (feature diagrammen). Het onderzoek poogt deze kennis te bundelen en gedistribueerd op te slaan, zodat – door slim kennismanagement – zo veel mogelijk informatie beschikbaar is bij het ontwikkelen, releasen, afleveren en installeren van een product. Deze gedistribueerde opslag is gemodelleerd in *figuur 2*, waar de kennis en componenten te zien zijn bij de producent en bij twee klanten.

De eerste conclusie van het onderzoeksproject is dat er nog geen ideaal systeem is waarmee een informatiesysteem als product en de componenten ervan beheerd kunnen worden.[4]
Een andere conclusie is dat er nog geen applicatie is die alle processen uit figuur 1 dekt en dat iedere softwareproducent opnieuw het wiel lijkt te willen uitvinden.[5]
De uiteindelijke resultaten van het Deliver-project zullen moeten leiden tot een set van applicaties en processen die deze problemen oplost. Deze applicaties en procesbeschrijvingen zullen gevalideerd en getest worden in het bedrijfsleven om de juistheid van het onderzoek te bewijzen. Vervolgens worden deze applicaties en processen gepubliceerd op wetenschappelijke conferenties en industriële forums om de kennis openbaar te maken aan softwarebedrijven.

Deze conclusies kunnen onder andere gebruikt worden om kwalitatieve eigenschappen van bibliotheeksystemen te identificeren door middel van 'koude' wetenschappelijke analyse.

VubisSmart als beste

Voor het Deliver-onderzoek is gekeken naar VubisSmart,[6] Concerto[7] en Bicat.[8] Deze producten zijn beoordeeld naar de beschikbare informatie. De procesdekking van figuur 1 kon niet mee worden genomen vanwege de gebrekkige informatie van de softwareproducenten.
Een interessante eigenschap van Concerto en Bicat is, dat beide producten alle modules op de server installeren en dat alleen de gekochte modules worden geactiveerd. Dit is

'common practice' in de software-industrie, maar komt ook voor in de auto-industrie, waar regelmatig de airconditioning wel wordt ingebouwd in de auto, maar zonder activeringsknop, omdat dit goedkoper is dan het onderhouden van een variërende productlijn.

VubisSmart komt als beste uit de bus, vooral vanwege de keuze om alle clients zo licht mogelijk te maken. Dat houdt in dat de businesslogica op de server staat en niet op de computer van de clients. De clients kunnen hierdoor afzonderlijk van de server geüpdatet worden, wat de beheerder van het systeem ruimte en flexibiliteit biedt. VubisSmart slaat alle configuratie en aanpassingsinformatie in de centrale database op en dat maakt het updaten van de software alleen maar makkelijker. Geac weet tenslotte zelf wat er in de database staat als het datamodel veranderd moet worden.
Daar tegenover staat wel, waarschijnlijk vanwege druk uit de markt, dat de software toestaat dat via ODBC, een protocol van Microsoft dat het mogelijk maakt de database direct te benaderen, het datamodel door de gebruiker kan worden aangepast. Deze eigenschap maakt de updates van het datamodel onbetrouwbaar.

Verder is gekeken naar Concerto, een pakket met een client server architectuur waar de clients wél versie-afhankelijk zijn en de updates dus synchroon moeten worden uitgevoerd. Hiermee kan Bibliomondo hun claim dat het systeem 24 uur per dag bereikbaar is niet helemaal hard maken, aangezien de server, al is het maar voor een korte periode, uit de lucht moet voor een grote update. Concerto lijkt aanpassingen beter toe te staan vanwege een modulaire opbouw. Dit soort aanpassingen introduceert echter de bekende versie-inconsistentieproblemen.

Tot slot werd gekeken naar Bicat, een product van Nederlandse bodem dat minder volwassen lijkt dan de andere twee producten. Leverancier HKA geeft aan graag aanpassingen te bouwen en lijkt voor een groot deel afhankelijk te zijn van een Linux-server. Bicat is dus een flexibele oplossing met een client server architectuur die dezelfde problemen oplevert als die van Concerto. Bicat verspreidt updates van de client echter via de server en deze eigenschap maakt het updaten tot een minder pijnlijk proces. Als laatste moet er rekening mee gehouden worden dat de aanpassingen en quick fixes van HKA op lange termijn wel meer problemen kunnen opleveren. Deze aanpassingen moeten namelijk slim gemanaged worden om versieproblemen te voorkomen.

Meer studie vereist

De conclusies van het onderzoek kunnen ook gebruikt worden bij het maken van beslissingen over de implementatie van een systeem, zoals bij het opstellen van een Service Level Agreement (SLA). Zo kan worden afgesproken dat updates binnen een maand na het uitkomen ervan getest en geïnstalleerd worden. Deze maand geeft de beheerders van het systeem de tijd om eerst te bezien hoe de updates functioneren in de testomgeving en bij andere gebruikers van het product. Daarnaast is het testen van een update een must, aangezien dat de manier is om te zien of eventuele (klant-specifieke) aanpassingen aan een systeem nog wel bruikbaar zijn na de update.

De problemen die hier worden geschetst zijn van een dusdanige complexiteit dat wetenschappelijke aandacht vereist is. Vooral de onderlinge afhankelijkheden en communicatie tussen softwareproducten en componenten vraagt meer studie.
Het Deliver-onderzoek heeft als primair doel de integratie van softwarecomponenten aan de klant en aan de kant van de ontwikkelaar te vergemakkelijken, om daarmee de software-industrie een sprong voorwaarts te helpen. Dit stelt namelijk de klant en de ontwikkelaar in staat minder tijd te besteden aan de integratie en communicatie tussen softwarecomponenten en -producten.

Selectiecriteria

Het Deliver-onderzoek levert een aantal criteria op waarmee we een informatiesysteem en de producent daarvan kunnen analyseren. Zo kan gekeken worden naar de procesdekking van figuur 1 om te zien of een producent wel alle mogelijkheden aangrijpt om zijn producten up-to-date te houden. De belangrijkste eigenschappen bij dergelijk onderzoek zijn de stabiliteit van een product, de mogelijkheden tot het maken van aanpassingen, de configureerbaarheid van het product, en het medium waarmee updates beschikbaar worden gemaakt, zoals internet, cd-rom en dvd. De stabiliteit kenmerkt zich bijvoorbeeld door een regelmatige en niet overmatige stroom aan updates.
Bij de keuze en de aanschaf van een informatiesysteem, moet dus goed gelet worden op de updatemogelijkheden van het systeem. Een informatiesysteem is ten slotte alleen effectief als het in de toekomst ook nog voldoet aan de wensen van de gebruikers.

Noten

1. OESO – The software sector: Growth, structure and policy issues – 2001.
2. www.productsoftware.nl.
3. www.Netherware.nl.
4. S. Jansen, G. Ballintijn, and S. Brinkkemper, "*Integrated SCM/PDM/CRM and Delivery of Software Products to 160.000 Customers*," in Technical Report CWI, submitted for publication, 2005.
5. S. Jansen, S. Brinkkemper, and G. Ballintijn, "*A Process Framework and Typology for Software Product Updaters*," in Conference on Software Maintenance and Reengineering 2005.
6. www.geac.nl.
7. www.bibliomondo.nl.
8. www.hka.nl.

Slinger Jansen is Assistent in Opleiding bij de Universiteit Utrecht. Daarnaast heeft hij een eigen bedrijf dat webapplicaties bouwt, cgbs.nl.