Nieuws / Development

Minder fouten in features door delta modelling

18-11-2014 13:09 | Door Henk Boot | Er zijn 8 reacties op dit artikel | Dit artikel heeft nog geen cijfer (te weinig beoordelingen) | Permalink

Bij het programmeren van nieuwe features in software worden makkelijk veel fouten gemaakt. De code van meerdere features grijpt vaak in op dezelfde plek, en zo kan de code van de ene feature soms ongemerkt die van een andere overschrijven. Hierdoor ontstaan bugs, die ertoe leiden dat software later op de markt komt en duurder is. Michiel Helvensteijn, promovendus van het Centrum Wiskunde & Informatica (CWI) te Amsterdam, onderzocht manieren om dit soort fouten te voorkomen.

Op 12 november promoveerde Michiel Helvensteijn aan de Universiteit Leiden op zijn proefschrift 'Abstract Delta Modeling - Software Product Lines and Beyond'. De resultaten zijn interessant voor software engineers en de industrie.

Software wordt steeds groter en wordt ook door steeds meer ontwikkelaars gemaakt, waardoor het steeds moeilijker uit te breiden is. Om te voorkomen dat de software te complex wordt, moeten features idealiter elk tot één module worden samengevoegd, zodat code van verschillende features netjes gescheiden is. In realiteit moet code juist over verschillende locaties in de software worden verspreid en gemixt met die van andere features. Helvensteijn ontwikkelde samen met collega's de formele methode van 'abstract delta modelling' (adm) om dit probleem te kunnen lossen.

Demonstratie

De promovendus liet bijvoorbeeld zien hoe een feature met één deltamodule geformuleerd kan worden. Zo'n deltamodule verandert bestaande code van buitenaf en geeft aan welke code waar neergezet moet worden of welke bestaande code aangepast moet worden. Helvensteijn maakte het mogelijk dat ontwikkelaars relaties tussen delta's kunnen aangegeven, zoals welke delta een andere mag overschrijven. Daarnaast ontwierp hij een conflictoplossend model. Helvensteijn: 'Als geen van de delta's 'de baas' is, krijgt de ontwikkelaar nu een duidelijke foutmelding, terwijl in eerdere technieken de ene module stilletjes de code van de ander kon overschrijven. Als ontwikkelaars deze methode voor conflict resolution volgen, scheelt het hun veel bugs. De software is dan modulair en makkelijk te onderhouden.'

De promovendus vervolgt: 'Software product line engineering heeft hier in het bijzonder veel bij te winnen. Hierin kan dezelfde code-base gebruikt worden voor softwaresystemen die elk een ander maar vaak overlappende verzameling features hebben. Een specifiek systeem kan dan gegenereerd worden door een selectie van features te maken: 'automated product derivation'. Abstract delta modelling' kan hier helpen om de code 'componeerbaar' te maken, bijvoorbeeld voor meerdere software edities.' De nieuwe methode is in de praktijk getest. Het promotieonderzoek is mede gefinancierd door het EU HATS-project, waarin onderzoeksorganisaties en de industrie samenwerken aan software-variabiliteit. Software is een van de speerpunten van het Centrum Wiskunde & Informatica (CWI), waar het onderzoek is uitgevoerd.

Meer Nieuws

24-11 ASML verwacht forse groei door EUV-machine
24-11 VVD Amsterdam: Grijp in bij financieel project 1
24-11 'Europa overweegt splitsing Google' 3
24-11 VNSG verhuist en versobert jaarcongres
24-11 'Software defined staat nog in kinderschoenen' 1
21-11 Microsoft wil meer geld van app-ontwikkelaars
21-11 Atos ontvangt BCR-norm beveiliging
21-11 FC Utrecht print voor vijf jaar met Sharp
21-11 CODE_n-wedstrijd staat in teken van Iot
21-11 Detron vernieuwt opslagomgeving Jaarbeurs

Meer Development

24-11 ASML verwacht forse groei door EUV-machine
21-11 Microsoft wil meer geld van app-ontwikkelaars
21-11 Nieuwe versie Tecnomatix van Siemens PLM
21-11 Microsoft Azure plat door performance-update 4
21-11 Profiteer van de kracht van de personal cloud 2
20-11 De 'scrum' van softwarefabrikanten 1
19-11 Nieuwe stichting keurt software op veiligheid 5
18-11 Wetgeving frustreert vernieuwing 3
18-11 ICT Automatisering doet Bulgaarse overname
17-11 Clockwork bouwt Drupal-portal voor Havenbedrijf

Drie sleutels voor meer productiviteit

Servicepunt 71 kiest voor Private Cloud van Imtech ICT — Imtech ICT

Reacties op dit artikel

Henri Koppen, 18-11-2014 13:54

Leuk artikel, maar geen bronvermelding en iets te oppervlakkig.

Hier lees ik wel iets van hem over delta modelling.

https://lirias.kuleuven.be/bitstream/123456789/288822/1/gpce15-clarke.pdf

En het paper is wel erg technisch, logischerwijs vol met wiskunde waardoor een praktische toepassing weer moeilijk te vinden is.

Dat mis ik sowieso in de academische wereld, ook die van machine learning. Heel veel academische papers, maar de mensen die dat schrijven zouden wat meer met ontwikkelaars en commerciële mensen moeten praten om er zodoende ook tastbare producten van te maken. Ik lees nu "The Innovators" van Isaac Walterson en daarin zie je keer op keer dat je moet mixen van mensen en kwaliteiten om echt tot innovaties te komen die groot worden in de werkelijke wereld.

Felix The Cat, 18-11-2014 20:48

Tis een promovendus, dus inderdaad logisch dat het om een abstract model gaat. Bijzonder is dat de nieuwe methode al "in de praktijk getest is" (hmmm alhoewel, misschien net zoals elke Duijvis pinda ok bevonden is)

Codelines groeperen in modules en die aan elkaar relateren, kan een mooie stap zijn op weg naar een N.G. software revision control system, al wordt Henri er niet opgewonden van. Beter geen commerciele mensen erbij, die clicken meer kapot dan je lief is.

Pa Va Ke, 18-11-2014 22:11

Nu ben ik al enige tijd actief in het configuration- en product lifecycle management wereldje, maar dit academische abstractieniveau (van de paper waar Henri naar verwijst) maakt het een stuk moeilijker dan het in werkelijkheid is volgens mij.
(misschien komt het omdat ik geen acadeem ben, maar een man van de praktijk)

Je kunt alles wel in wiskundige modellen willen gieten, maar bij CM en PLM kom je met te GBV methode een heel eind!

Ewout Dekkinga, 19-11-2014 12:01

Mogelijk dat het allemaal een academisch betoog is maar als ik kijk naar de initiatieven (open source) die hier gaande zijn wel interessant, zelfs voor Henri omdat de code ergens moet leven. En dat is steeds vaker de cloud waar de hoop op foutloze software architecturen tot op heden nogal vluchtig is gebleken door met name dus de commerciële aspecten die al dan niet terecht vervloekt worden. Een onberekenbare factor in dit soort modellen is de verwachting, bijvoorbeeld de hoop op foutloze software wat weleens irreëel kan zijn als ik kijk naar de snelheid van veranderingen in niet alleen de code zelf maar ook de omgeving waarin deze leeft. PLC in een software omgeving die uit een compositie van componenten bestaat die door een varia aan belanghebbende geleverd wordt zorgt ervoor dat een bug voor één partij een irritante onderbreking is en voor de andere partij een kansvolle feature. Ik haal dit even aan aangezien in meeste software contracten een clausule staat die leverancier ontslaat van gevolgschade als gevolg van bugs.

En het eventueel oplossen ervan is dus een dienst met een steeds kort wordende levensduur in de closed source industrie. Vraag die na het lezen van dit stuk bij mij naar boven komt is in hoeverre er op Europees niveau samengewerkt wordt aangezien er nog meer universiteiten al enige tijd bezig zijn met dit idee en sommigen zelfs al verder lijken te zijn als ik kijk naar sommige open source projecten. De wens voor modulaire software zonder bugs klinkt in mijn oren namelijk als een oud idee waarbij we telkens tegen dezelfde problemen aan blijven lopen, verwachtingen die als gevolg van marketing niet altijd kloppen. PLC is tenslotte in belangrijke mate een economisch proces, als leverancier wil je producten en diensten niet tot in de eeuwigheid ondersteunen en vanuit dat aspect wordt bij geboorte dus al de dood bepaald. Hierdoor lijken dit soort modellen veel op 'handlezen' waarbij het de vraag is in hoeverre er rekening gehouden wordt met niet alleen levensbedreigende ziekten maar ook menselijke factoren.

Pa Va Ke, 19-11-2014 13:26

@Ewout ... je betoog bestrijkt slechts een deel van het software spectrum. In de maak-industrie is modulaire software gebruikelijk, waarbij lange levensduren (>10 jaar) niet ongebruikelijk zijn.

PLM kom je in deze industrie dan ook veelvuldig tegen binnen de sofware-ontwikkel afdelingen.

Jouw terechte opmerking over keten-afhankelijkheid (een bug in blok A zorgt voor problemen in blok B) los je niet op met PLM of CM systemen, maar met een goede architectuur en change management.

Top 10 reagerende bezoekers

			Aantal reacties	Gemiddelde waardering
	1	Ewout Dekkinga	2037	6.81
	2	Henri Koppen	1525	6.70
	3	Reza Sarshar	1236	6.70
	4	Ruud Mulder	1144	6.65
	5	Pa Va Ke	890	6.56
	6	Felix The Cat	610	6.34
	7	Louis Kossen	433	6.32
	8	NumoQuest	1097	6.11
	9	Jan van van Leeuwen	722	6.08
	10	Willem Oorschot	462	6.06

Felix The Cat, 19-11-2014 17:04

- code opslaan is goed
- versies van sourcecodefiles opslaan is beter
- groeperen in branches/releases is nog beter
- concurrent versioning system met lock/merge mogelijkheden op line niveau is fijn
- codelines groeperen en er betekenis en prio aan toekennen is top.

De modulariteit gaat hier niet om de programma code zelf, maar het management ervan in de repository.
Echte innovaties waardoor Cloud en byod mogelijk is geworden zoals nanotechinologie, komen niet van ontwikkelaars en commercielen. Verwachtingen lossen ook niet echt iets op, betawetenschap wel.

Ewout Dekkinga, 19-11-2014 20:58

Aangaande architectuur bemerk ik dat lifecycles niet vastgelegd worden, vaak ook niet de versioning en de afhankelijkheden tussen technische componenten. Functioneel wordt zelfs niet altijd rekening gehouden met de voortschrijdende technische ontwikkelingen waardoor vervangen van versie 1.0 door 2.0 al problematisch kan zijn. Het zijn namelijk niet alleen maar bugs die voor verandering zorgen, versioning geeft tenslotte vooral de voortschrijding in ontwikkeling aan. En omdat een nieuwe versie niet altijd (volledig) backward compatible is levert dat dus uitdagingen in de vervanging op als de daarop gestapelde oplossingen (van B tot en met Z en nog verder) nog niet aan vervanging toe zijn. Hele leuke uitdaging zie je hier bijvoorbeeld met middleware, vaak een redelijk abstracte laag met toch een behoorlijk aantal technische componenten.

Voordat Felix zich verslikt in een overdosis aan nanotechnologie is het misschien goed als hij even denkt aan de controle zelf. De betawetenschap leverde ons tenslotte ook atoomenergie (en wapens) op waarbij het in de hand houden van de kettingreactie niet onbelangrijk is. Het gaat dus niet zo zeer om het management van de repository maar voorkomen dat je denkt dat de reactorstaven uit de kern zijn terwijl zich ondertussen een metldown aan het ontwikkelen is. Oftewel dient de wetenschap de mens of dient de mens de wetenschap?

Henri Koppen, 20-11-2014 12:16

Wat ik een beetje uit de tekst begrepen heb is dat het ook suggesties geeft voor de plaats van de code op basis van de structuur van de code. en opzich is dat wel een aardige.

Je legt dit dan ook *over* de versioning van de code heen en staat er verders los van. De Delta zal immers een nieuwe versie van de code creëren. Doordat in een branch te doen (zoals gebruikelijk is als je codeert in teams), kun je gewoon met unit testen e.d. ervoor zorgen dat de code ook nog functioneert na de veranderingen.

één van de problemen die delta modelling lijkt op te lossen is deze: Stel een programmeur maakt er wat code bij, maar doet dat op een onlogische plaats. Dus functieachtige zaken in de presentatie laag (front-end) en niet in de libraries. Het algoritme zal dit herkennen en aangeven dat de functie code beter in Lib Y gezet kan worden inclusief de aanpassingen in de code die dat moeten faciliteren.

Een ander probleem die mogelijk wordt opgelost is conventie. Dus wanneer een programmeur afwijkt van de conventie, dan wordt er een suggestie gedaan. Dit zou ik in ieder geval een zeer bruikbaar gegeven vinden. 1 van de lastigste dingen bij het coderen in teams die ook nog wel eens van samenstelling veranderen is het doorgronden en kunnen houden aan conventies en dit logisch en gebruikersvriendelijk te houden. Als je de conventie kan modelleren dan is het mijn inziens niet heel complex om dit dus te checken met delta modelling.

Jouw reactie

Je bent niet ingelogd. Je kunt als gast reageren, maar dan wordt je reactie pas zichtbaar na goedkeuring door de redactie. Om je reactie direct geplaatst te krijgen, moet je eerst rechtsboven inloggen of je registreren.

Naam: E-mailadres: IP-adres: 192.16.191.140

Reactie op dit artikel:

Voorwaarden voor plaatsing van reacties

Reacties van gasten worden niet direct op de site geplaatst. De redactie controleert vooraf of de reactie aan een aantal voorwaarden voldoet. Deze voorwaarden zijn:

- De reactie dient betrekking te hebben op de inhoud van bovenstaand artikel.
- De reactie moet correct, bondig, professioneel en beschaafd zijn.

Ik ga akkoord met de voorwaarden

Plaats reactie