An online corpus of UML Design Models : construction and empirical studies

Karasneh, B.H.A.

Samenvatting

In dit proefschrift richten we ons twee problemen in de software engineering. Het eerste probleem betreft de vraag hoe de ernst ('severity') van software gebreken ('defects') te beoordelen? Het tweede probleem dat we beschouwen betreft de vraag hoe wij het ontwerp ('design') van software kunnen bestuderen. We leggen elk van deze problemen in iets meer detail uit.

Het toewijzen van een ernst-classificatie aan een software defect wordt momenteel gedaan door software-ontwikkelaars op basis van hun ervaring. In de praktijk blijkt dat ontwikkelaars die taak van het registreren van de ernst van de defects niet erg serieus nemen en vaak een default waarde bevestigen die wordt voorgedragen door een defect tracking tool. Om deze situatie te verbeteren bestuderen we of het mogelijk is om geautomatiseerde ondersteuning voor de beoordeling van de ernst van de defects te ontwikkelen. Geautomatiseerde ondersteuning voor de beoordeling van de ernst van software defects kan menselijke ontwikkelaars helpen om deze taak efficiënter en nauwkeuriger uit te voeren. In dit proefschrift presenteren we een nieuwe aanpak (MAPDESO) voor de beoordeling van de ernst van software defects op basis van de IEEE Standard Classification voor Software Anomalies. De innovatie bij deze aanpak ligt in het gebruik van ontologieën en ontologie-gebaseerde redenering die defects koppelt aan systeem-niveau-kwaliteitseigenschappen. Deze aanpak is gevalideerd door te bestuderen hoe zij presteert op een industrieel project. In deze validatie presteert onze geautomatiseerde voorspellingsmethode goed in vergelijking met de handmatige toekenning. De MAPDESO aanpak op basis van ontologieën presteerde beter dan enkele geselecteerde machine learning classifiers. Bij het bedrijf waar we de studie uitgevoerd hebben, waardeeren de engineering de resultaten van MAPDESO.

Ons onderzoek kan gepositioneerd worden in het gebied van de empirische Software Engineering - dit is een tak van de Software Engineering die tot doel heeft kennis te creëren door middel van de analyse van producten en processen die deel uitmaken van software development projecten. Bij een grote meerderheid van studies in dit gebied ligt de nadruk op analyse van programmatekst ('source code') - uitgedrukt in een bepaalde programmeertaal. Hiermee wordt voorbij gegaan aan het feit dat het ontwerp van een software systeem een cruciale rol speelt bij van de meeste software development projecten. Echter de ontwerpen van software systemen worden nauwelijks

bestudeerd door de empirische software engineering gemeenschap.

Een van de belangrijkste redenen waarom het bestuderen van software ontwerpen een uitdaging is, is hun gebrekkige beschikbaarheid. In ons onderzoek hebben we vastgesteld dat software ontwerpen vaak worden gearchiveerd als UML diagrammen in 'image'-formaten en dat software ontwerpen in deze vorm gevonden kunnen worden bij verschillende bronnen op het internet. Om deze reden hebben we besloten om een een poging to doen om een corpus van UML-modellen in image-formats van internet te verzamelen en deze te converteren naar echte modellen. Om dit mogelijk te maken ontwikkelden we de volgende software-instrumenten: 1) UMLCrawler, dit is een software tool die een groot aantal UML diagrammen kan downloaden van het internet, 2) UMLImgClassifier, dit is een automatische classifier die UML class diagrammen kan onderscheiden van andere diagrammen, en 3) Img2UML, dit is een software tool die software-model informatie kan extraheren uit UML-images en ze omzet naar XMI bestanden. De gegenereerde XMI-files zijn compatibel met verschillende UML-tools waarmee de modellen kunnen worden bewerkt en geanalyseerd. Onze validatie toont aan dat UMLImgClassifier en Img2UML een hoge nauwkeurigheid behalen bij het classificeren en omzetten van UML diagrammen naar XMI-model bestanden.

De hiervoor genoemde software tools zijn essentiele ingrediënten voor de bouw van een UML Repository: Tijdens dit onderzoek construeerden we een online repository die UML class diagrammen, XMI-files en enkele bij de ontwerpen behorende metrieken bevat. In dit proefschrift presenteren we deze repository en enkele van haar functies zoals het delen, zoeken, ranken, definiëren van experimenten, uploaden, downloaden en genereren van grafieken.

Deze repository is de eerste corpus in zijn soort en we geloven dat het een nuttige bron zal zijn voor de empirische software engineering gemeenschap. Ter ondersteuning van deze stelling, voerden we een reeks van empirische studies met behulp van deze UML Repository uit. Deze empirische studies zijn slechts een druppel in de oceaan van empirische studies die kunnen worden uitgevoerd met deze repository.

Onze eerste studie toont enkele voorbeelden van interessante relaties tussen klassendiagrammen en software-ontwerp metrieken. Op basis van dergelijke relaties kan het corpus van UML ontwerpen worden gebruikt om patronen in UML-modellen te vinden en er achter komen welke goede en slechte praktijken gangbaar zijn. Dit levert referentie-gegevens die gebruikt kunnen worden bij de kwaliteitsborging van UML ontwerpen.

Onze volgende studie onderzoekt de relatie tussen de kwaliteit van UML ontwerpen en de bijbe-horen¬de broncode. Onze meest prominente bevindingen in deze studie zijn de volgende: Ten eerste, klassen die zowel in het ontwerp en de source code voorkomen ondergaan meer veranderingen en bevatten meer fouten dan klassen die alleen voor komen in de broncode (en dus niet in het ontwerp van de software). Dit is een indicatie dat deze klassen een centrale rol spelen en wellicht van een relatief hoge complexiteit zijn. Ten tweede, we tonen aan dat anti-patterns in een vroeg stadium van software ontwerp geïdentificeerd kunnen worden, en dat anti-patterns die bestaan

in het ontwerp van een software systeem zich vertalen naar anti-patterns bij de corresponderende klassen in de source code. Ten derde, klassen die onderdeel zijn van anti-patterns in het software ontwerp ondergaan meer veranderingen en bevatten meer fouten dan andere klassen die geen onderdeel zijn van anti-patronen in het ontwerp. Onze studie is de eerste studie naar de relatie tussen de kwaliteit van UML-gebaseerde software ontwerpen en de kwaliteit van de source code.

Een andere studie onderzoekt hoe het corpus gebruikt kan worden in een educatieve setting. Onze eerste studie in deze richting onderzoekt hoe studenten en experts verschillen in de manier waarop ze de kwaliteit van UML class diagrammen beoordelen. In deze studie maken we een extra onderscheid tussen een self-assessment en peer-assessment van studenten. We vonden dat er een significant verschil is tussen experts en studenten (zowel hun self-assessment als peer-assessment). We vonden een hoge correlatie tussen experts en peer-assessment van studenten voor de beoordeling van begrijpelijkheid van ontwerpen. We vonden dat enkel het kwaliteitsattribuut begrijpelijkheid is gecorreleerd met de meeste andere kwaliteitskenmerken (zowel voor experts en studenten peer-assessments). Een patroon in de data van de studenten lijkt er op te wijzen dat zij vermijden om onvoldoende beoordelingen te geven aan mede-studenten ('peers'). Een implicatie is dat peer-assessment door student niet zonder meer gebruikt kan worden als beoordelings¬mechanisme in de klas. De studenten werd ook gevraagd om tekstuele feedback te geven op ontwerpen van peers (in samenhang met de kwantitatieve beoordeling). Een analyse van die feedback laat zien dat studenten opmerkingen geven over de kwaliteit van de UML ontwerpen die inhoudelijk gezien vergelijkbaar is met de feedback die experts geven. Uit het voorgaande concluderen we dat peer-feedback van studenten nuttig is, terwijl peer-beoordeling door studenten geen goed idee is.

De laatste studie in educatieve setting onderzoekt het nut van voorbeelden bij het onderwijzen van het ontwerpen van software. Hiertoe boden we studenten de mogelijkheid om de UML repository te gebruiken tijdens het uitvoeren van een aantal opdrachten voor het ontwerpen en verbeteren van een software ontwerp (dat in eerste instantie zonder voorbeelden was gemaakt). We concluderen dat de voorbeelden de studenten helpen bij het creëren en het verbeteren van hun ontwerpen. Een kwantitatieve analyse toonde aan dat experts een hogere beoordeling gaven aan de modellen geproduceerd door studenten die gebruik maken van voorbeelden uit de repository dan aan studenten in de controlegroep (zonder gebruik van voorbeelden uit de repository). Deze hogere beoordeling werd gevonden voor alle kwaliteitsattributen van een software ontwerp. We concluderen dat de model repository een geschikte manier is voor het aanbieden van voorbeelden van software designs – niet alleen in de zin dat de inhoud ervan nuttig is, maar ook dat model -voorbeelden op een efficiente manier te vinden zijn. Bovendien blijkt dat studenten waarderen dat de UML Repository hen in staat stelt om te zoeken naar software ontwerpen op basis van verschillende onderdelen van het ontwerp zoals klasse-namen, attributen en operaties. Deze manier van zoeken is nergens anders beschikbaar, ook niet via generieke zoekmachines op het internet.

Deze serie van empirische onderzoeken illustreert het belang en de mogelijkheden van de corpus van UML modellen van software ontwerpen.