Aanwijzingen voor het schrijven van het Functioneel Ontwerp

Inleiding

Op de vraag 'wat is een goed functioneel ontwerp' is geen eenduidig antwoord te geven. Wat je opneemt in een functioneel ontwerp, tot in welk detail dit beschreven wordt, en in welke vorm is afhankelijk van de context.

Schrijf je het functioneel ontwerp bijvoorbeeld voor een opdrachtgever en ontwikkelaar? En doe je dit voordat er code geschreven gaat worden, zodat zij vooraf kunnen lezen wat er precies ontwikkeld gaat worden? Worden er dan ook op basis van het functioneel ontwerp ook prijsafspraken gemaakt?

Of is het functioneel ontwerp bedoeld voor documentatie achteraf, en voor wie dan en met welk doel?

In dit document geven we aan wat we verwachten van het functioneel ontwerp voor het project dat jullie in deze periode gaan uitvoeren.

De doelgroep voor het functioneel ontwerp

Voor dit project verwachten we een functioneel ontwerp op basis waarvan iemand die niet betrokken is geweest bij het project, zich in korte tijd (maximaal 10 minuten leestijd), een goed beeld kan vormen van de functionaliteit op hoofdlijnen van de applicatie. Verder is het functioneel ontwerp zo geschreven dat de lezer de details er makkelijk op kan nalezen.

Het functioneel moet te begrijpen zijn voor een opdrachtgever, toekomstige eindgebruikers van het systeem, een docent (ook als deze niet betrokken is het geweest bij de ontwikkeling van het systeem) of een stagiaire die in de toekomst verder gaat met de ontwikkeling van het systeem.

Je mag er van uitgaan dat de lezer UML-diagrammen kan interpreteren, je hoeft de tekentechniek dus niet uit te leggen. Wel is het van belang dat je een toelichting geeft op de inhoud van de diagrammen.

Functionaliteit op hoofdlijnen

Voor het globale plaatje is het natuurlijk van belang dat duidelijk wordt wat het doel van de applicatie is. Daarvoor moet je de business case voor het ontwikkelen van de applicatie beschrijven.

Beschrijf de rollen van de gebruikers die met het systeem gaan werken en ondersteun dit met een use case diagram, waarin dan ook staat tot welke functionaliteiten zij toegang hebben.

Naast de functionele kant kun je ook de gegevens kant van je applicatie beschrijven, hiervoor is een klassendiagram waarbij je je beperkt tot de niet-technische klassen heel geschikt. Dit noemen we ook wel een domeinmodel. Uiteraard plaats je dan niet zomaar een los klassendiagram in je functioneel ontwerp, maar geef je ook duidelijke tekstuele uitleg. Neem multipliciteiten, attributen en indien zinvol ook de operaties op in dit klassendiagram. Het kan zijn dat het domeinmodel zo omvangrijk is, dat wellicht meerdere klassendiagrammen nodig zijn om een en ander leesbaar te houden. Maak daarbij gebruik van de beschikbare UML-technieken, zoals het werken met packages of de mogelijkheid om bijvoorbeeld in een klassendiagram op hoofdlijnen, je in eerste instantie te

beperken tot klassennamen en pas in een latere detaillering de attributen en operaties toe te voegen.

Gewenste mate van detaillering

De gewenste mate van detaillering hangt onder andere af van de gerealiseerde applicatie. Zo zou het kunnen voorkomen dat er in de applicatie zelf al help-schermen zijn opgenomen die verduidelijking geven. En misschien heb je een game gemaakt, met een uitgebreide handleiding, al dan niet in de applicatie zelf opgenomen. Je zou dan vanuit je functioneel ontwerp naar deze handleiding kunnen verwijzen om dubbel werk en dubbel onderhoud op je documentatie te voorkomen.

Testgevallen en de gewenste output beschrijven ook de functionaliteit van je applicatie. Ga je de handmatig uit te voeren testen opnemen in je functioneel ontwerp, of leg je deze vast in een apart document? Bespreek dit met je docent.

Ook de unittesten, zeker als je goede naamgeving kiest, beschrijven de gewenste functionaliteit. Je zou dan kunnen overwegen om in je functioneel ontwerp minder op het detail in te gaan en kunnen verwijzen naar deze unittesten. Dit is uiteraard niet voor iedere doelgroep even geschikt. Denk hier dus goed over na, en overleg met je docent.

User story of use case

De gewenste functionaliteit wordt beschreven in het functioneel ontwerp. Nergens ligt formeel vast hoe zo'n beschrijving er uit zou moeten zien. Vaak wordt gekozen voor een beschrijving in de vorm van use cases of user stories. Maar ook deze worden nergens formeel beschreven. Wel kun je in boeken en op het internet veel voorbeelden vinden van hoe dit zou kunnen.

Het is niet zo dat je verplicht bent met use cases te werken als je een use case diagram hebt opgenomen in je ontwerp. Het use case diagram heeft nu eenmaal deze naam, maar je mag dit ook lezen als user story diagram, al is dit geen gangbare term in de UML wereld. Wel dien je alle in het use case diagram genoemde use cases, zowel de primaire als secondaire in je ontwerp toe te lichten. En uiteraard zorg je dan voor consistente naamgeving.

Het kan soms zinvol zijn om een beschrijving van een complexe use case of user story, met een UML activity diagram te ondersteunen.

Ook zijn er soms klassen die zich goed lenen voor een toestandsdiagram, als er zo'n klasse in jullie applicatie voorkomt, verwachten we natuurlijk ook zo'n diagram daarvoor, inclusief de toelichting daar op.

Stem met je docent af hoe jullie de functionaliteit gaan beschrijven. Van belang is dat jullie je aan de definition of done houden, zodat er na de eerste gerealiseerde items een volledig functioneel ontwerp ligt waar de docent feedback op kan geven.

Schermontwerpen

Bij de uitwerking van een use case of user story hoort vaak een scherm. Voordat je een use case of user story gaat programmeren, wil je op zijn minst globaal weten hoe een dergelijk scherm er uit zal komen te zien. Hiervoor kun je, bijvoorbeeld tijdens een sprint planning, schermontwerpen op een whiteboard tekenen en overleggen met je product owner. Overleg met je docent of je een foto

hiervan als ontwerp in je functioneel ontwerp mag opnemen. Een goed alternatief is om hier een tool voor te gebruiken. Een bekend tool hiervoor is Balsamiq. Maar ook schermontwerpen die je maakt met een software framework (bijv. WPF), zouden kunnen voldoen. Overleg ook dit met je docent.

Als het scherm uiteindelijk geprogrammeerd is, kun je hiervan een afbeelding opnemen in je functioneel ontwerp. Overleg met je docent of hij/zij dit een goed idee vind.

Soms is het ook verduidelijkend als er een schermflow diagram wordt opgenomen in het functioneel ontwerp. Helaas is hier in UML geen diagram voor gedefinieerd, dus hiervoor zul je zelf iets moeten bedenken.

Niet functionele requirements

Vermeld in je functioneel ontwerp de eventueel van toepassing zijnde niet functionele requirements. Bekijk om een idee te krijgen de iso-norm 25010.

Consistentie

Uiteraard zorg je voor consistente naamgeving in je ontwerp. Dus bijvoorbeeld een attribuut uit je domeinmodel geef je, als deze ook voorkomt in een scherm, niet ineens een andere naam. En de omschrijving die je in een use case diagram geeft aan een use case, houdt precies deze zelfde omschrijving aan, als je deze nader specificeert. Maak er voor de lezer geen puzzel van!

Hoofdstukindeling

Hierin zijn jullie vrij, zolang jullie maar gehoor geven aan bovenstaande aanwijzingen. Bespreek de indeling met je docent.