Toets Informatica

Modulenaam:	Project Software Engineering 2
Modulecode:	ipsen2
Studiejaar:	2018-2019
Gelegenheid:	1 en 2
Toets opgesteld door:	Michiel Boere
Toetsvorm:	Project
Aantal verwachte deelnemers:	60

Duur van de toets bedraagt 10 weken.
De oplevering is in week 9 van Periode 1, gevolgd door een individueel assessment in week 10.
De herkansing is in week 12 (week 2 van Periode 2), gevolgd door een individueel assessment in week 13.

Hulpmiddelen:
Alles mag gebruikt worden, binnen de kaders van de Nederlandse wet en regelingen van de Hogeschool Leiden zoals, maar het is zeker niet beperkt tot de OER van de opleiding Informatica, zolang de bronvermelding klopt en de groep en de individuele student voldoen aan de opdracht.

Opmerkingen: Geen
Bijzonderheden: Geen

Puntentelling:
De criteria zoals deze beschreven zijn in de volgende opgaven zullen door de examinator worden voorgelegd aan de inhoudelijk begeleider van de groep; zij voorzien de examinator van oordelen op basis van tussenresultaten en de eindoplevering.

De eindoplevering bestaat uit een presentatie en een consistente, transparante, complete set documenten en digitaal materiaal, voorzien van een leeswijzer waarin te lezen is in welk document of deel van de oplevering aangetoond wordt dat de groep of het individu aan de gestelde criteria voldoet.

Per criterium of groep criteria is door de moduleleider vooraf een weging vastgesteld, alsmede een mogelijke ondergrens (cesuur). De beoordelaars worden hiervan op de hoogte gebracht, de studenten echter niet. De oordelen alsmede belangrijke overwegingen van de beoordelaars en de examinator zelf worden vastgelegd in een besloten document. Hieruit volgt een eindscore.

De informatie binnen deze kaders bevat geen criteria voor de toetsing, maar is ter verklaring en als ondersteuning toegevoegd. De vragen die bij de opgaven staan, zijn een indicatie van vragen waarop wij tijdens het beoordelen antwoorden willen vinden in jullie documenten. Je hoeft ze dus niet letterlijk als vraag te beantwoorden, maar we verwachten wel een stelling, conclusie, aanbeveling of berekening terug te vinden in jullie documentatie als 'bewijs' dat je over de vraag hebt nagedacht. Waar dit 'antwoord' volgens jullie te vinden is lezen we terug in de leeswijzer.

Tijdens de inzage geeft de inhoudelijk begeleider, of de examinator zelf, desgevraagd inzage in de totstandkoming van het eindresultaat. Mocht de projectgroep hiertoe gezamenlijk besluiten dan kan deze bij een 2e gelegenheid opnieuw een complete oplevering doen. Het onderdeel presentatie is het enige oordeel dat meegenomen kan worden van gelegenheid 1 naar gelegenheid 2, alle andere onderdelen dienen opnieuw, compleet, consistent en transparant, voorzien van een leeswijzer te worden ingeleverd bij de inhoudelijk begeleider en de examinator.

Toets Informatica

Opgave 1 Samenwerking en projectmanagement

Zoals gebruikelijk wordt direct na het starten van het project een samenwerkingsovereenkomst opgesteld waarin de betrokken projectgroep leden de afspraken vastleggen die nodig zijn voor een succesvolle samenwerking. Deze samenwerkingsovereenkomst is onderwerp van het eerste begeleidingsgesprek met de procesbegeleider.

Vervolgens maken jullie een plan van aanpak. De waterval methode (en specifieker: het model van Royce), welke we gebruiken bij ipsen2, heeft een fasering (Definitiestudie/analyse, Basisontwerp, Technisch ontwerp, Implementatie, Testen, Integratie en Beheer). De fases die doorlopen moeten worden, en de op te leveren producten (mijlpalen) liggen vast. Het is aan jullie om uit te zoeken wat deze methode precies voorschrijft, vervolgens verwerk je deze gegevens in je eigen Plan van Aanpak. Je plan van aanpak bevat ook een beschrijving van de huidige situatie (i.e. het bedrijfsproces) en een beschrijving van de toekomstige situatie.

Het projectmanagement houdt met name in: controleren of iedereen zijn werk op tijd gedaan heeft en of de opgeleverde producten consistent zijn. Als twee personen op basis van een afspraak twee delen maken is de kans klein dat deze twee delen werkelijk passend op elkaar aansluiten. Jullie zullen je allemaal moeten verdiepen in de producten die jullie gezamenlijk opleveren.

De watervalmethode staat niet toe dat je aanpassingen maakt in een mijlpaalproduct na de oplevering ervan. Definitief is definitief. Als een mijlpaal onwerkbaar veel fouten en inconsistenties bevat moet je dus helemaal opnieuw beginnen met deze mijlpaal. Kleinere aanpassingen aan b.v. een UML-schema kun je in een hoofdstuk, "Wijzigingen ten opzichte van ..." opnemen in je reflectieverslag. Hierbij vermeld je concreet de specifieke wijziging en de motivatie voor deze verandering. De wijzigingen mogen niet meer dan 20% van het originele stuk in beslag nemen.

In het hoofdstuk: "Wijzigingen t.o.v. het basisontwerp" van de realisatie zou b.v. kunnen staan:

"We hebben i.p.v. compositie toch voor aggregatie gekozen bij de relatie tussen fiets en wiel, omdat uit nadere bestudering van de wensen en eisen bleek, dat een open fietswiel met spaken vervangen kan worden door een dicht fietswiel bij weinig wind. Het open fietswiel wordt in dat geval niet vernietigd, maar gewoon op het dak van de auto gemonteerd, zodat we na de tijdrit de wielen weer terug kunnen wisselen." (Interview met Francesco Moser, aug. 1984.)

De met zorg samengestelde documenten en producten, waarbij actielijsten en besluitenlijsten van vergaderingen niet mogen ontbreken, worden tijdig aan de inhoudelijk begeleider, procesbegeleider en de examinator van ipsen2 opgeleverd. Bij de eindoplevering vinden we alle documenten terug, tussentijdse vergaderstukken en complete mijlpalen in originele samenstelling. Hou hierbij rekening met de richtlijnen van de opleiding Informatica. We moedigen jullie aan om sjablonen te personaliseren en waar mogelijk te gebruiken.

Op de ELO zijn templates van een aantal documenten terug te vinden. Deze staan in de modules "Projectdocumentatie informatica" en in "IN 1516 IIPSEN"

Op te leveren documenten:

- **GroepX.Samenwerkingscontract.pdf**
- **GroepX.PlanVanAanpak.pdf**

Toets Informatica

Opgave 2 Waterval methode.

Uit Wikipedia:

De **watervalmethode** is een methode voor softwareontwikkeling waarin de ontwikkeling regelmatig vloeiend naar beneden loopt (als een waterval). De ontwikkeling loopt door een aantal fasen, namelijk: definitiestudie/analyse, basisontwerp, technisch ontwerp/detailontwerp, bouw, testen, integratie en beheer en onderhoud.

Voorheen was het ontwikkelen van vooral grote softwareprojecten een groot onoverzichtelijk breiwerk. Met de komst van deze nieuwe methode hoopten de informaticabedrijven meer duidelijkheid te krijgen in hun projecten.

Het watervalmodel is afgeleid van de traditionele manier van werken in grote projecten in de constructiebouw. De bedoeling van deze manier van werken is dat je het project in verschillende fasen opdeelt. Je begint met fase 1 en begint niet eerder met fase 2 dan wanneer je fase 1 hebt afgesloten. En wanneer je in een van de fasen een fout ontdekt moet je helemaal terug om die fase te corrigeren en de daaropvolgende stappen opnieuw uitvoeren. Het watervalmodel bestaat uit de volgende fasen:

1. *Definitiestudie/analyse.*
2. *Basisontwerp.*
3. *Technisch ontwerp/detailontwerp.*
4. *Bouw/implementatie.*
5. *Testen.*
6. *Integratie.*
7. *Beheer en onderhoud.*

Informatie over de waterval methode is ruim beschikbaar op internet. Ook kun je in de bibliotheek op zoek naar literatuur of via google scholar zoeken naar relevante informatie.

Bij de beoordeling van jullie project zal gekeken worden of jullie je aan de watervalmethode gehouden hebben, of de bronnen waaraan jullie refereren betrouwbaar zijn en of de argumenten die jullie aandragen voor keuzes die jullie zelf maken valide zijn. Het kan zijn dat jullie keuzes moeten maken m.b.t. de waterval methode, omdat bronnen elkaar tegenspreken of omdat de voortgang van het project daarom vraagt. We kunnen dit teruglezen in het plan van aanpak.

Een belangrijk begrip zijn de mijlpaalproducten. Het gaat hier om complete consistente sets van documentatie en later ook digitale producten die de oplevering van een fase vormen. Bij dit project is de inhoudelijk begeleider degene die de beoordeling van het mijlpaalproduct voor zijn rekening neemt. Uiteraard heeft deze overleg met de opdrachtgever hierover. Lever de mijlpaalproducten op bij je inhoudelijk begeleider, na een akkoord van je opdrachtgever.

Toets Informatica

Opgave 3 Experimenteren

Jullie gaan een systeem ontwerpen, bouwen en beschikbaar stellen. Een belangrijke eis is dat de gegevens worden opgeslagen in een relationele database. Daar zijn nogal wat varianten van.

Voor sommige betaal je de 'hoofdprijs' zoals Oracle en MS-SQL. Als je project succesvol is dan wordt er veel gebruik van gemaakt. Databasesystemen hebben in geval van succes de neiging om snel te groeien, qua benodigde rekenkracht en opslag. In dat geval is het mogelijk om de werklast te verdelen over meerdere servers. Hoewel dit niet nodig zal zijn voor dit project is het wel een gegeven om rekening mee te houden. Bij betaalde databasesystemen moet je vaak per CPU en/of per user betalen. Wat zou de invloed hiervan zijn op de schaalbaarheid van je systeem? Maak een realistische schatting voor het gebruik van het systeem.

Er zijn ook relationele databasesystemen die een vrije licentie kennen. MySQL is een voorbeeld van een vrij systeem dat toevallig, via overnames nu ook van het bedrijf Oracle is. Hoewel de taal SQL redelijk eenvoudig is qua opzet en leesbaar, zelfs voor mensen die zichzelf "geen programmeur" noemen, is de techniek achter de verwerking van die SQL commando's razend ingewikkeld, complex en in veel gevallen verbluffend snel.

Zoek eens uit wat de ACID eigenschappen eigenlijk precies inhoudt als we het hebben over databasetransacties. Wat betekent "Referentiele integriteit" en waarom is het belangrijk dat het databasesysteem dit bewaakt i.p.v. de Java toepassing die we zelf schrijven?

Aangezien er zeer capabele relationele databasesystemen zijn waar geen licentiekosten voor gelden, die gemaakt zijn om te kunnen worden bestudeerd, die je mag testen en waarvan je de testresultaten mag publiceren zonder het risico te lopen te worden vervolgd, gebruiken we een opensource relationeel databasemanagementsysteem. Bijkomend voordeel is dat de vrije licenties het vaak toestaan om de broncode van het databasesysteem zelf aan te passen en de aangepaste versie te her-distribueren naar een klant of andere betrokkenen. Je mag hier zelfs geld voor vragen aan je klant zonder dat je iets hoeft te betalen aan de copyrighthouder van het originele systeem. Het mag natuurlijk wel, en deze 'donaties' kun je als bedrijf zelfs afschrijven voor de belasting.

Deze lijst: https://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems is een lijst van relationele DBMS en bevat mogelijke kandidaten voor jullie project.

Jullie kiezen een relationeel databasemanagementsysteem. De volgende licentievormen zijn hierbij toegestaan: Apache licence, GPL, BSD, PostgreSQL licence. Bij het maken van de keuze gebruik je:

- Gegevens die openbaar beschikbaar zijn over de betrouwbaarheid, performance en functionaliteit van de verschillende systemen zoals de aangeleverde lijst, regressietesten, testimonials of White papers
- Resultaten van eigen testen met de software. Hiervoor is het nodig om op een vergelijkbaar systeem, onder vergelijkbare omstandigheden de verschillende databasesystemen te installeren, te configureren, vergelijkbare testen uit te voeren, de resultaten te controleren en evt. nogmaals na her-configuratie;
  - Hoe snel wordt een SQL script verwerkt?
    check alle CRUD handelingen met en zonder unique constraint.
  - Wordt het script correct verwerkt?
  - Wordt 'Unique' gehonoreerd als we een insert uitvoeren?
  - Wordt de referentiële integriteit gecontroleerd bij een insert statement?
  - Verwijdert de database records die verwijzen naar een 'parent-record' bij een cascading relatie?
  - Bedenk zelf testen die specifiek voor deze opdracht van belang zijn.

Bekijk de informatie uit de publieke bronnen en de gegevens die uit je eigen experimenten naar voren komen en verklaar waar mogelijk de verschillen en overeenkomsten. Houd in je achterhoofd dat een testset van bijvoorbeeld slecht 5 items misschien niet zo representatief is. Je levert je onderzoek op als :

- **GroepX.DatabaseOnderzoek.pdf**

Toets Informatica

Opgave 4 Basis- en detailontwerp

Bij het maken van de verschillende ontwerpen is het belangrijk dat jullie allemaal een bijdrage leveren. Nu is de opdracht nog te overzien en je zult misschien geneigd zijn om alles zelf te doen. Jullie moeten het werk echter samen maken en iedereen daarbij eigenaar maken van een deel van het werk. Wie waarvoor verantwoordelijk is lezen we terug in de documenten, de reflectie en het blijkt tevens uit de urenverantwoording en de Javadoc.

Uiteindelijk leveren jullie de ontwerpen op per mijlpaal als één geheel. Deze ontwerpen bestaan onder andere uit correcte UML schema's. Het is belangrijk om de juiste schema's te maken in de juiste fase. Soms is het heel duidelijk waar een bepaald schema 'hoort', soms zijn er verschillende varianten van hetzelfde schema die steeds gedetailleerder worden naarmate het project vordert. Denk hierbij, bijvoorbeeld maar niet uitsluitend, aan de verschillende UML-klasse diagrammen die je bij IMUML hebt moeten maken. Activiteiten diagrammen geven goed inzicht in hoe het systeem gaat werken. Voor dit project maken jullie deze voor de niet-triviale usecases. Tevens moet goed bedacht worden hoe de database eruit moet zien. Ook hier maken we een ontwerp voor.

Je gaat niet zomaar een systeem bouwen. Jullie ontvangen aan het begin van het project een brief van een opdrachtgever die een wens heeft. Deze opdrachtgever gaan jullie natuurlijk spreken en door middel van het stellen van vragen, tijdens een interview of schriftelijk, proberen jullie de wensen en eisen op papier te krijgen. Notulen van de gesprekken met de opdrachtgever zijn zeer belangrijk! Sommige wensen en eisen zullen functioneel van aard zijn en andere gaan over de kwaliteit van het product. Het is aan jullie om alle eisen gedetailleerd in kaart te brengen.

Vaak is het verschil tussen functioneel en niet functioneel erg makkelijk te maken. De wens: 'Ik wil een systeem dat facturen maakt' is duidelijk een functionele wens. '12.000 orders moeten binnen 1 minuut gefactureerd zijn' is duidelijk een niet functionele wens.

Er zijn echter ook wensen en eisen die voor sommige mensen functioneel en andere niet functioneel zijn. 'Alle facturen moeten kloppen' kan beteken:
– 'er mogen geen rekenfouten worden gemaakt bij het afronden',
– 'Iedere order resulteert in precies 1 factuur',
– 'Het logo moet precies op de goede plek geprint worden' of

Bij dit type wens en eis is het belangrijk dat je door blijft vragen en terugkoppelt wat je begrijpt tot je er zeker van bent dat er geen 'verborgen boodschap' over is. Vaak moet je meerdere keren spreken met een opdrachtgever en zowel mondeling als schriftelijk terugkoppelen voordat de onderste steen boven is.

Jullie formuleren functionele en niet functionele wensen en eisen op basis van de informatie die je ophaalt bij de opdrachtgever. De lijst van functionele eisen worden opgedeeld volgens het MoSCoW principe. Al in het stadium van opstellen van deze requirements stellen jullie testen op die zullen moeten slagen om, bij de oplevering, van een geslaagd project te kunnen spreken. Ook plannen jullie de testen al tijdens het plannen van het project. Dit testplan en de aanpassingen die erop gemaakt worden en de concrete testcases en testresultaten zijn onderdeel van de oplevering. (zie onderdeel testen)

Op te leveren documenten:

- **GroepX.FunctioneelOntwerp.pdf**
- **GroepX.TechnischOntwerp.pdf**

Toets Informatica

Opgave 5 Bouw

Natuurlijk maken jullie ook software. Bij dit project is het verplicht om in Java te programmeren. Sommige delen maken jullie bij wijze van experiment of als proof of concept ter overtuiging van de opdrachtgever of elkaar. Deze delen moeten natuurlijk controleerbaar zijn, leesbaar voor de groepsgenoten, voorzien van commentaar in de stijl benodigd voor Javadoc als de code onderdeel uitmaakt van het eindproduct.

Het product dat jullie maken dient te voldoen aan de MVC-architectuur. Dit houdt o.a. in dat jullie al bij het eerste klassediagram rekening moeten houden met het MVC-principe. We beoordelen uiteindelijk de consistentie, dus ook of de code en de UML schema's overeenkomen. Zo kan werkende software toch tot een onvoldoende leiden als de schema's niet gevolgd zijn. Denk eraan dat je bij de oplevering van de code een 'wijziging te opzichte van ...' kunt opnemen als dit nodig is om het systeem werkend te krijgen, lees meer hierover bij opgave 1.

De opdrachtgever en/of de inhoudelijk begeleider moet formeel de mijlpalen accepteren. Als het om het programma gaat zal de opdrachtgever kijken naar de functionaliteit. De inhoudelijk begeleider kijkt naar de code en de niet functionele wensen en eisen en controleert of de opdrachtgever akkoord is.

Iedereen programmeert mee aan het systeem dat jullie maken. Daar houden jullie al vanaf het begin rekening mee, zie opgave 5. Het is dus extra belangrijk dat jullie van elkaar de Javadoc kunnen lezen als je een methode van een ander projectlid gebruikt. Dit kan als je gebruik maakt van Javadoc. Wij controleren of een geoefend Java programmeur op basis van jullie Javadoc een correct en compleet beeld krijgt van de functie van classes, methodes en attributen. Het is aan de studenten om, ook in de Javadoc, te verantwoorden wie voor welk deel verantwoordelijkheid draagt zodat wij de individuele verschillen in de beoordeling kunnen verwerken.

De Java code is object-georiënteerd geschreven. Jullie maken naast de MVC-principe gebruik van de technieken die jullie leren bij de module IOPR3; inheritance, encapsulation, polymorphism.
– Wat betekenen de termen inheritance, encapsulation, polymorphism?
– Welke argumenten kunnen jullie vinden voor het gebruik van deze technieken?

De database spreken jullie aan via JDBC en waar mogelijk en van toepassing gebruiken jullie daarbij prepared statements. Als meerdere SQL-commando's geïsoleerd dienen te worden uitgevoerd dan passen jullie transacties toe. Als dit nergens het geval is dan maken jullie een 'log'-tabel waarin logregels dienen te komen van elke geslaagde insert, update en delete statement op de database.

– Welke argumenten kunnen jullie vinden voor het gebruik van prepared statements?
– Welke argumenten kunnen jullie vinden voor het gebruik van transacties?
– Zijn er ook redenen te vinden waarom je geen gebruik wilt maken van deze technieken?
– Bij het gebruik van de MyISAM storage engine[1] van MySQL worden de SQL commando's START TRANSACTION, COMMIT en ROLLBACK geaccepteerd, zonder melding of waarschuwing. Omdat deze storage-engine geen implementatie kent van transacties worden deze commando's genegeerd. Wat vinden jullie hiervan? Je kunt dit bespreken in het database onderzoek.

Op te leveren software:

GroepX.Project.zip (Bevat de broncode)

GroepX.Applicatie.zip (Bevat de executable code voor de klant)

GroepX.JavaDoc.zip (Java documentatie)

1 Zoek maar op met Google "myisam storage engine" wikipedia is in dit geval best een betrouwbare bron.

Toets Informatica

Opgave 6 Testen

De functionele testen die jullie gedurende het project hebben bedacht en afgesproken met de opdrachtgever dienen natuurlijk ook te worden uitgevoerd. Niet een keer door een tester maar verschillende keren door verschillende testers aan de hand van dezelfde testcases. De resultaten publiceren jullie in het testrapport. We concentreren ons op functioneel testen op bases van de use-cases, maar we jullie kunnen ook gebruik maken van hallway testing.

Aangezien we met een waterval methode werken, kan het niet anders dan dat alle testen slagen en de opdrachtgever tevreden kan terugkijken op een geslaagde onderneming die binnen budget is gebleven en die oplevert wat hem bij aanvang is voorgehouden.

Jammer genoeg is de praktijk toch meestal iets minder voorspelbaar. Ook een herkansing zit er vaak niet in. Eens gefaald altijd gefaald denken de meeste opdrachtgevers. Als je al een tweede kans krijgt is dit niet voor hetzelfde aantal "punten". Gelukkig is dit een schoolproject en kunnen jullie nog veel rechtzetten in de reflectie en desnoods in de herkansing maar neem van ons aan, IRL is het een ander balspelletje.

Iedere student voert testen uit en maakt hiervan een eigen testrapport. In een gezamenlijk document schrijven jullie de conclusie van de testen. Deze rapporten zijn onderdeel van de laatste mijlpaal.

Op te leveren documenten:

- **GroepX.Testplan.pdf**
- **GroepX.Testrapport.pdf**

Opgave 7 Invoering

Tijdens de fase na het ontwikkelen van de software dient de projectgroep de migratie en conversie van gegevens uit het oude systeem voor te bereiden. Hierbij worden de technische systeemeisen gespecificeerd voor de apparatuur waarop de nieuwe software zal komen te draaien en de randapparatuur die daarbij aangesloten dient te zijn. Dit alles kunnen we teruglezen in een conversie- en invoeringsplan. De huidige klant heeft hoogstwaarschijnlijk geen database met gegevens welke geconverteerd moeten worden. Ook hoeft er geen waarschijnlijk geen migratie te gebeuren van het ene naar het andere systeem. Dit is wel iets om te bespreken met de klant. Een moment van invoering moet wel bepaald worden en daar schrijven we een invoeringsplan voor. Hierin staat beschreven welke software nodig is en hoe jullie applicatie werkend kan worden neergezet.

Organisatorische veranderingen worden vaak nauwgezet gepland en beschreven. Denk aan, maar niet uitsluitend: sluiting voor migratie, training, administratie, fallback, backups, recovery. Deze organisatorische zaken lezen we terug in het conversie- en invoeringsplan.

Er wordt een gebruikershandleiding voor het systeem gemaakt welke bedoeld is voor de eindgebruikers.

Op te leveren documenten:

- **GroepX.Migratie_Conversie_Invoeringsplan.pdf**
- **GroepX.Gebruikershandleiding.pdf**

Toets Informatica

Opgave 8 Reflectie

Tijdens het uitvoeren van het project letten de beoordelaars en andere betrokkenen ook op de houding van de studenten. We beoordelen of je je professioneel gedraagt. Ben je er als we dat afspreken? Besteed je genoeg tijd aan het werk? Neem je op tijd een pauze en leid je daarmee je projectgenoten en medestudenten niet te veel af? Ben je bereid om andere te helpen met eenvoudige en met complexere vraagstukken? Plan je je werk in? Overzie je de omvang van een opdracht? Werk je mee aan een goede sfeer en ben je niet bedreigend of overmatig aanwezig voor anderen?

Het zijn te veel onderwerpen om allemaal te noemen. Je kunt al deze vragen, en de vragen die jullie als projectgroep erbij bedenken, beantwoorden in je reflectie en een antwoord op deze vragen kunnen onderdeel zijn van de feedback die je aan anderen geeft. Het is verstandig om tijdens de loop van het project in een procesvergadering dit deel van de beoordeling ter sprake te brengen en tussentijdse feedback te vragen aan de betrokken docenten.

Als je extreem goed of extreem slecht scoort op dit punt dan krijg je dat van ons, of je medestudenten te horen op initiatief van de ander. Normale stervelingen moeten over het algemeen vragen om feedback als ze hiervan willen leren.

Naast de reeds genoemde onderwerpen van reflectie en de gangbare onderdelen zoals urenverantwoording en feedback lezen we in de reflectie, tot slot, graag aanbevelingen ten behoeve van de opleiding, de opdracht, de begeleiding en de opdrachtgevers.

Op te leveren documenten:

- **GroepX.ReflectieVerslag.pdf**