SWOP - Hospitaal Iteratie 3

Groep12

Jeroen Van Gool
Ruben Lapauw
Tom De Bie
Jeroen De Coninck

Inhoudsopgave

1 Inleiding

1.1 Overzicht van het verslag

1.2 Veronderstellingen

- "Een patiënt kan aan niet meer dan 10 X-ray scans per jaar onderworpen worden." Hierbij hebben we natuurlijk aangenomen dat het over de tijdspanne van een jaar gaat, en niet over een kalenderjaar. Dit is natuurlijk gemakkelijk aanpasbaar.
- Personeelsleden alsook patiënten hebben een unieke naam. Je kan echter wel een patiënt hebben met dezelfde naam als iemand van het personeel, dit leek ons logisch aangezien een personeelslid ook opgenomen kan worden in het ziekenhuis als hij of zij zelf ziek is.
- De FIFO-queue wordt nu gebruikt zoals gevraagd. Dit was geen probleem met de implementatie.
- Bij het doorspoelen van de tijd wordt als het eten op is geen eten meer gegeven aan de patiënten.
- De Stock van alle items is voldoende groot dat er niet meer items gevraagd worden dan dat het minste aantal items die in een Stock kan ziten zonder dat er bijbesteld moet worden. Met andere woorden, deze is voldoende groot dat 2 dagen na de laatste afspraak altijd voldoende items zullen zijn.

2 Het systeem

2.1 Overzicht

Het hospitaal heeft verschillende subsystemen: De wereld die als oer-Object dient. Deze houdt de tijd, personen, machines en voorraad bij. De personen splitsen zich op in patienten en personeel. Het personeel kan verschillende operaties uitvoeren op het systeem. Terwijl er op patienten operaties worden uitgevoerd: Voeg diagnoses, medische testen en behandelingen toe. Medische testen en behandelingen hebben allebei een afspraak die als alle andere precondities voldaan zijn gemaakt worden. Verder is er nog voorraad. Deze voorziet maaltijden voor de patienten en andere items voor de behandelingen. De bestellingen gebeuren hiervan automatisch. Bestellingen worden niet automatisch verwerkt.

2.2 Gebruikers

2.2.1 Usecases: Login, Logout

De enige verandering is de keuze van een Campus bij het inloggen. Dit gebeurt aan de hand van CampusInfo's die worden gevraagd aan de World. In de CampusInfo zit enkel de naam van de Campus en daardoor kan de Campus niet onnodig gewijzigd worden door de Login. Vervolgens wordt in de login methode van WorldController de Campus weer opgevraagd aan de World via zijn CampusInfo. Ten slotte wordt er een CampusController aangemaakt die de juiste Campus bijhoudt, deze wordt gebruikt door LoginControllers, zoals bv. de DoctorController of de NurseController. Bij logout is er niets veranderd.

2.2.2 Bespreking GRASP, uitbreidbaarheid en nadelen

Dit was een eenvoudige aanpassing. Dezelfde opmerkingen als bij de vorige iteratie zijn van toepassing.

2.3 Input en output

2.3.1 Beschrijving

Input en output is is strikt gereguleerd. Alle output moet ofwel immutable zijn, zoals Strings en andere, ofwel read-only, zoals CampusInfo en LoginInfo, ... Voor Input moet men ook oppassen: deze moet ook immutable zijn, ofwel moet men een perfecte kopie maken van alle Objecten die ingegeven worden. Maar alleen met primitieven en Strings werken is niet uitbreidbaar en onhandig werken. Men kan bijvoorbeeld niet garanderen dat alle behandelingen hetzelfde aantal parameters hebben van hetzelfde type. De oplossing is om de parameters te abstraheren naar een Object Argument. Om de kennis van in het systeem dan ook nog af te schermen van de UI is er een vraag bij gegeven. Hierdoor moet de UI niet de vragen stellen aan de gebruiker en kunnen gemakkelijk nieuwe behandelingen met andere parameters toegevoegd worden. Een andere gevolg is dat men direct kan controleren of de invoer mogelijks correct is.

2.3.2 Publieke Werking

De interface PublicArgument¡E¿ is de basis van de invoer. Deze heeft een vraag die aan de gebruiker moet gevraagd worden. En deze kan beantwoord worden door de setAnswer methode met een String. Deze wordt direct geconverteerd naar het type E. Deze Argumenten zitten in een ArgumentList, samen met objecten van de superklasse Argument¡E¿. Deze objecten zijn voor het interne systeem om in te vullen, op het moment dat de UI de controle teruggeeft aan het systeem.

2.3.3 Interne Werking

Een generische factory verwacht als invoer om een nieuw object te maken een lijst van Arguments. Men kan een nieuwe lege lijst Argumenten opvragen aan deze factory met de methode getEmpty-ArgumentList. Als deze ingevuld is door de UI en door de andere code wordt de make-methode aangeroepen met deze lijst. Deze doet dan de specifieke inputvalidation. Met de validate-methode kan gecontroleerd worden of een lijst correct is voor een bepaalde factory zonder een object te maken.

2.3.4 Bespreking GRASP en uitbreidbaarheid

Deze opbouw laat toe om een grote diversiteit aan invoer uit te lezen op een veilige EN generische manier. Het laat toe om de input statisch te valideren tijdens de invoer en zo duidelijker fouten terug te geven. Wat de cohesie ten goede komt. Het laat ook toe om informatie van het systeem te lezen zonder dat de UI deze zelf moet zoeken. Hierdoor is de koppeling tussen de UI en het systeem lager en duidelijk herkenbaar.

Men kan eenvoudigweg nieuwe argumenten ontwerpen en gebruiken zonder dat er iets moet veranderen aan de UI. Om informatie van andere objecten mee te geven kan men gebruik maken van het visitor-pattern en een filter.

2.4 Magazijn

2.4.1 Beschrijving

Hier volgt een overzicht van de belangrijkste klasse en een korte beschrijving van hun taken in ons ontwerp van het magazijn systeem:

- LIFO-queue: Dit is een FIFO-queue geworden zoals voorzien.
- FIFO-queue: Een wrapper rond Queue¡E¿ en geabstraheerd als ItemQueue.
- ItemReservator: Deze klasse is verwijderd. En is vervangen door de ItemReservationCommand. Deze heeft een andere naam en andere verantwoordelijkheden.
- ReserveItemObserver: Deze klasse is verwijderd, een afspraak wordt direct gepland op het moment dat alle items beschikbaar zijn.
- ItemReservationCommand: Deze klasse reserveert alle nodige items in een stock. Op het moment van de afspraak komen de Items in deze klasse beschikbaar.

- ItemInfo: Dit object is een veilig object om buiten het systeem te gebruiken.
- ItemConstraint: Deze constraint wordt bij het maken van een afspraak in rekening gebracht om een goed moment van afspraak te vinden.

2.4.2 Interne Werking

In tegenstelling tot de vorige iteratie wordt een afspraak nu wel direct gepland. Een geschiedenis van alle afspraken wordt voor ieder item bijgehouden en bij het plannen van de afspraak wordt een moment gezocht waar er geen tekort aan items is, noch waar er ooit in de toekomst problemen zullen komen. Het gebruik van een Item kan zich ver in de toekomst propageren en zo binnen een aantal maanden een probleem veroorzaken.

2.4.3 Bespreking GRASP

De twee subsystemen, het Warehouse en de Scheduler, zijn enkel met een Constraint verbonden. Dit garandeerd een minimale koppeling. De constraint zelf bevat ook weinig logica en wordt door een afgescheiden algoritme afgehandeld die apart getest kan worden.

2.4.4 Nadelen

Bij het plannen wordt geen rekening gehouden met het eventuele vervallen van de items aangezien we dit niet kunnen voorspellen. Het gebruik van een FIFO-queue en een "druk" hospitaal zorgen ervoor dat deze kans klein is.

2.5 Medische testen en behandelingen

2.5.1 Verloop usecases: OrderMedicalTest en EnterTreatment

Bij deze usecases is er een grote aanpassing. Namelijk het gebruik van preemptive-scheduling. Het eerste zal apart in de sectie Scheduling2.6 worden uitgelegd. De prioriteit van deze Appointment wordt gevraagd via een PriorityArgument.

2.5.2 Bespreking GRASP en uitbreidbaarheid

Aan de usecase zelf is niets veranderd. De UI heeft geen enkele verandering in gedrag.

2.6 Tijdsplanning

2.6.1 Beschrijving

Afspraken worden gemaakt voor een Appointable, dit wordt gedaan door AppointmentFactory. Deze zoekt een moment waarop alle constraints voldaan zijn. Ook is de eerste constraint een GetCampusConstraint voor ieder type van afspraak: deze bepaalt de campus waarop deze zich zal voordoen.

Voor een afspraak tussen een docter en een patient moeten bijvoorbeeld aan volgende Constraints voldaan zijn: DoctorBackToBackConstraint (DoctorPatientAppointment), Preference (van Doctor), PriorityConstraint (DoctorPatientAppointment), en WorkingHoursConstraint (Doctor). Elk van deze constraints maken deel uit van alle voorwaarden die opgelegd zijn aan de afspraken. De PriorityConstraint zorgt dat er geen twee appointments op hetzelfde moment vallen en als er op een afspraak op dat moment staat bij een persoon dat die een lagere prioriteit heeft en dus zal verzet worden. De DoctorBackToBackConstraint zorgt ervoor dat de afspraak ofwel op het uur valt ofwel dat de afspraak direct na de vorige valt.

2.6.2 Bespreking GRASP en uitbreidbaarheid

De koppeling tussen de twee belangrijkste systemen is hier laag: De scheduler weet niet eens dat er iets in de warehouse gedaan wordt. En behalve dat het opvragen van alle benodigde informatie is alles afgeschermd. De scheduler weet enkel dat hij constraints opvraagt van een Schedulable, en van een Appointable.

Het algoritme is ook afgeschermd in een eigen deel binnen dit systeem. Dit laat toe om snellere algoritmes te gebruiken, zoals constraintprocessing met backjumping, ...

3 Onbresproken Usecases

4 Onveranderde Usecases

Aan de volgende Usecases is er niets veranderd:

- Undo & Redo: Deze interfaces is volledig hetzelfde.
- Consult Patientfile
- Close Patientfile
- Discharge Patient
- Register Patient
- Enter MedicalTestResult
- Fill Stock
- List Orders
- Enter Diagnosis
- AddEquipment: Voor deze usecase is er niets veranderd, de factories hebben gewoon een CampusInfoArgument gekregen zodat deze automatisch wordt ingevuld in de UI.
- AddStaffMembers: zie AddEquipment

4.1 Enter TreatmentResult

Door het gebruik van een geschiedenis in het Warehouse kunnen resultaten pas ingevuld worden wanneer de items er uit verwijderd zijn. En dit mag enkel op het juiste moment, anders kan zal de planning niet meer juist werken. Het is dus logisch dat resultaten pas uitgevoerd kunnen worden als het na de start van deze afspraak is. Anderzijds moet men tijdens het vooruitspoelen van de tijd direct de resultaten opschrijven van alle afspraken die geweest zijn. Dit zou betekenen dat men als zuster enkel de tijd heeft om deze in te vullen voor het einde van de afspraak. We hebben gekozen om AdvanceTime op een logischere manier te laten werken en toch niet te veel van de opgave af te wijken.

4.2 Advance Time

Door de veranderingen die gebeurd zijn aan het Warehouse was het niet logisch om results in te vullen als meteen wanneer deze verlopen. De aanpassing is gemaakt dat deze moeten ingevuld worden voor het einde van de dag en anders door de HospitalAdministrator. Verder gebeurt AdvanceTime op dezelfde manier als in iteratie 2.

5 Conclusie

De uitbreidbaarheid van ons systeem in hetzelfde gebleven. De gevonden pijnpunten van de vorige iteratie zijn verbeterd. De verscheidenheid aan constraints bij het schedulen is groter geworden zoals verwacht en zonder grote aanpassingen.

6 Appendices

6.1 De user interface

De UserInterface bestaat uit één MainUI, één rolUI per rol die kan inloggen en één usecaseUI per usecase. In de MainUI krijg je de optie om in te loggen en na het inloggen zal de UI van de juiste rol (bvb. Doctor, WareHouseManager, ...) gestart worden. In de rolUI krijg je alle usecaseopties die voor die bepaalde rol gelden en wordt er gefiltert op precondities. Zo zal de DocterUI enkel de optie ClosePatientFile aanbieden, als de doctor van de huidige doctorcontroller een patient file open heeft. In de usecaseUI wordt heel de usecase doorlopen: lijsten worden getoond, er worden opties aangeboden, invoer gevraagd, ...

6.2 Testverslag

6.2.1 Teststrategie

6.2.2 Eclemma-verslag

Eclemma rapporteert op het eerste zicht een slechte testsuite. Wanneer we echter in detail kijken naar het verslag, zien we dat de slechtst gecoverde klassen de UI- en exception klassen zijn. De UI-klassen zijn niet opgenomen in de tests, en de exceptions voorzien vaak alternatieve constructors (met/zonder message) die niet altijd gebruikt worden. Daarbovenop is er in de gewone klassen ook vaak error-handling code voor exceptions die technisch gezien niet voor zouden kunnen komen, of door overerving vereiste methoden die niet gebruikt worden, logischerwijs wordt deze code ook niet covered door de testen waardoor het totale coverage-percentage daalt.

Echter de beste aanduiding van de kwaliteit van onze testsuite is het feit dat doorheen het implementeren van de software we gebruik hebben kunnen maken van onze tests om fouten in het systeem te kunnen opsporen. Deze ervaring van nut uit de tests gehaald te kunnen hebben zonder onzekerheid of het systeem nu wel écht werkt is waarschijnlijk ook de beste indicator voor testsuite-kwaliteit.

6.3 Werkverdeling

De Bie Tom	nog door te geven	¡voeg lijst in¿ Verslag
De Coninck Jeroen	nog door te geven	¡voeg lijst in¿
Lapauw Ruben	~56 uur	Scheduling, AddEquipment & AddStaffMembers Warehouse
Van Gool Jeroen	nog door te geven	¡voeg lijst in¿ Verslag

6.4 Volledig klassendiagram

druk dit apart af gespreid over 2 paginas en voeg in in plaats van dit blad

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
default		0%		n/a	2	2	5	5	2	2	1	1
Hospital		87%		81%	7	24	4	32	2	11	0	1
Hospital.Argument		53%		33%	11	25	26	61	8	22	4	9
Hospital.Controllers		72%		70%	53	177	131	454	23	111	1	18
Hospital.Exception		31%		n/a	33	47	54	75	33	47	22	35
Hospital.Machine		81%		58%	11	38	12	71	2	26	0	6
Hospital.MedicalTest		83%		82%	21	117	49	286	8	75	0	8
Hospital.Patient		64%		55%	51	121	76	236	15	70	1	9
Hospital.People		65%		64%	16	55	24	93	8	41	0	6
Hospital.People.PeopleFactories		53%		42%	7	18	18	39	2	12	0	3
Hospital.Schedules		60%		62%	56	127	124	292	20	68	1	10
Hospital.Schedules.Constraints		85%		62%	10	39	8	69	0	23	0	4
Hospital.Treatments		74%		62%	25	87	48	186	11	62	0	8
Hospital.WareHouse		45%		30%	98	148	163	312	33	74	0	9
Hospital.WareHouse.ItemQueues		62%		62%	5	11	6	16	3	7	0	1
Hospital.WareHouse.Items		50%		40%	24	52	32	76	14	42	0	9
Hospital.WareHouse.OrderPlacers		64%		50%	4	9	3	18	2	7	0	3
Hospital.World		74%		68%	16	57	69	221	9	40	0	4
HospitalUI.AdminUI		0%		0%	23	23	164	164	10	10	4	4
HospitalUI.DoctorUI		0%		0%	53	53	275	275	14	14	7	7
HospitalUI.MainUI		0%		0%	23	23	88	88	7	7	2	2
HospitalUI.NurseUI		0%		0%	31	31	168	168	15	15	4	4
HospitalUI.WareHouseUI		0%		0%	15	15	72	72	6	6	3	3
Total	6,436 of 13,736	53%	515 of 999	48%	595	1,299	1,619	3,309	247	792	50	164

Figuur 1: Eclemma code coverage verslag voor de testsuite