UNIVERSITEIT GENT

Academiejaar 2014–2015

Faculteit Ingenieurswetenschappen en Architectuur
Valentin Vaerwyckweg 1 – 9000 Gent

Combineren van performantie-indicatoren in operationele processen tot key performance indicators

Masterproef voorgedragen tot het behalen van het diploma van
Master in de industriële wetenschappen: informatica

Lando TANGHE

Promotoren: *prof. dr. ir. Jan CNOPS*
dr. ir. Didier COLLE
dr. ir. Sofie VERBRUGGE

Begeleiders: *dr. ir. Koen CASIER*
Jan VAN OOTEGHEM
ir. Gregory CASIER

UNIVERSITEIT GENT

Academiejaar 2014–2015

Faculteit Ingenieurswetenschappen en Architectuur
Valentin Vaerwyckweg 1 – 9000 Gent

Combineren van performantie-indicatoren in operationele processen tot key performance indicators

Masterproef voorgedragen tot het behalen van het diploma van
Master in de industriële wetenschappen: informatica

Lando TANGHE

Promotoren: *prof. dr. ir. Jan CNOPS*
dr. ir. Didier COLLE
dr. ir. Sofie VERBRUGGE

Begeleiders: *dr. ir. Koen CASIER*
Jan VAN OOTEGHEM
ir. Gregory CASIER

Woord vooraf

Graag wil ik iedereen bedanken die heeft bijgedragen aan het tot stand komen van deze scriptie.

Eerst en vooral wil ik mijn begeleiders Koen Casier, Gregory Casier en Jan Van Ootteghem bedanken voor het veelvuldig nalezen van mijn tekst en voor het zoeken naar oplossingen bij problemen met de simulator. Mijn dank gaat tevens uit naar Jonathan Spruytte, die me hielp met de ontwikkeling van het dashboard en het aanleveren van ideeën. Ik wil zeker ook mijn interne promotor, prof. dr. ir. Jan Cnops, bedanken voor zijn onmisbare hulp bij het ontwerp van de analysetool en voor het nalezen van de scriptie. Ook wil ik de externe promotoren, dr. ir. Sofie Verbrugge en dr. ir. Didier Colle, bedanken om de masterproef mogelijk te maken.

Ook een woord van dank aan de mensen van Amaron, die het registratieproces modelleerden dat in deze thesis als voorbeeld werd gebruikt.

Ten slotte wil ik nog mijn familie en vrienden bedanken voor de steun, mijn tante, mijn nicht en mijn zus in het bijzonder voor het nalezen van mijn scriptie.

Toelating tot bruikleen

augustus 2015

Abstract

Combineren van performantie-indicatoren in operationele processen tot key performance indicators

Lando Tanghe

Het doel van deze masterproef is een applicatie te ontwikkelen die toelaat een proces te analyseren met behulp van een *discrete event simulator*. De applicatie moet ertoe bijdragen dat bedrijven hun processen kunnen optimaliseren.

Eerst wordt het procesverloop gemodelleerd, op basis waarvan het proces kan gesimuleerd worden. Uit de simulatierapporten wordt op automatisch een generieke verzameling performantie-indicatoren (PI's) afgeleid, dit zijn metingen op het proces. Uit deze verzameling PI's kan dan een kleinere verzameling van Key Performance Indicators (KPI's), de belangrijkste indicatoren, worden samengesteld. Deze PI's en KPI's worden dan weergegeven op een dashboard waarmee de analyse en beoordeling van het proces mogelijk is.

Als case worden de doorlooptijden en de kosten voor de ontvangst en registratie van leveringen met prothese-onderdelen onderzocht bij een ziekenhuis. Eerst wordt het proces over een maand gesimuleerd, waarna verschillende performantie-indicatoren kunnen bepaald worden. Hieruit wordt afgeleid dat het gemiddeld 8,81 euro kost om een levering te ontvangen en te registreren. Als één medewerker ingezet wordt, dan kan 63,9 procent van de onderdelen binnen het uur worden geregistreerd. Door twee medewerkers in te zetten, neemt dit percentage toe naar 75,8 procent. Dat gaat wel ten koste van een lagere activiteitsgraad van de medewerkers, die daalt van 11,1 procent naar 5,6 procent.

Combining Performance Indicators to Key Performance Indicators

Lando Tanghe

Supervisor(s): Gregory Casier, Koen Casier, Jan Van Ooteghem

Abstract—**The purpose of this thesis is to build an application that allows managers to assess and evaluate a process based upon PIs and KPIs. The process is first modeled using BPMN and additional information is added about the resources, the timing and the choices. The model is then analyzed using a Discrete Event Simulator (DES). The simulator produces a report for every instruction that is processed by the simulated process. The instructions can represent patients to be treated in a health care process, products created in a manufacturing process... From the reports a generic set of Performance Indicators (PIs) can be calculated automatically. Key Performance Indicators (KPIs) are then defined by selecting and combining PIs. The KPIs contain the most important for the organisation and can be displayed on a strategic dashboard.**

Keywords—**Performance Indicator, Key Performance Indicator, Discrete event simulator, BPMN, Dashboard**

I. Introduction

For a company to achieve its goals, its processes have to be as efficient as possible. To make sure the processes are efficient, they have to be continuously monitored, assessed and evaluated. However, measuring the process in real time however can be expensive, which can be avoided by using a simulator.

According to [1] a simulator can be used to provide relatively realistic estimations about the performance of a process, because the simulator can introduce variability in the execution times and in the arrival of the instructions. Because of the variability, queues can be formed, increasing the lead times of the instructions.

The simulation results can then be used to infer a set of PIs. PIs are important, but not always crucial for the success of the organization. KPIs, however, are the subset of PIs that are crucial [2]. The PIs allow a more detailed view of the process and can be displayed in an operational dashboard. The KPIs are represented in a strategic dashboard, allowing the management to assess the process very quickly. The Service Level Agreement (SLA) defines the required values for every KPI and can be used to evaluate them [3].

A PI can contain the progress of a parameter in time or the distribution for all processed instructions of a parameter. For example, one PI shows the usage of a resource in time. Another PI can contain the distribution, summarizing the lead times of the process for all processed instructions at one day. Statistical PIs can be derived from a distribution PI, f.e. the average lead time.

The purpose of this study is to determine how PIs and KPIs can be calculated for a simulated process and how a selection of the PIs and KPIs can be displayed on a dashboard.

Firstly, this paper discusses how PIs and KPIs can be calculated using a simulator II. Then the system is applied to analyze a supply process by building an operational and a strategic dashboard, see III-B.

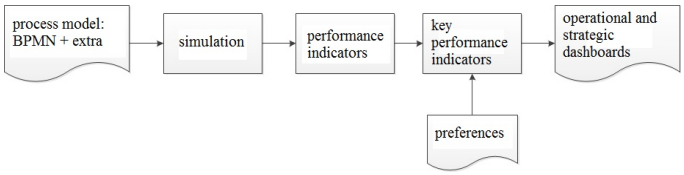


Fig. 1. Steps of the analysis tool

II. The analysis tool

Analysing a process is done in multiple steps, see fig. 1.

A. Simulation of the process

In order to simulate the process, information has to be available about its flow, the required resources and its costs and the timing information. An existing simulator, based on Multi-Agent Simulator Of Neighborhoods (MASON), is used.

Business Process Modelling and Notation 2.0 (BPMN) is used in order to model the process flow. Additional configuration is added about the timing and resources.

After the simulation completes, a report is generated for every processed instruction. Each report contains the following information about the corresponding instruction:

- arrival time at each task
- when each task starts processing the instruction (when required resources are available)
- when the execution of each task is completed

B. Inferring PIs from the simulation reports

The results of the simulation are then analysed and multiple PIs can be calculated, of which the most important are:

- distributions of the lead times and costs for each task and for the entire process
- distributions indicating the choices made
- time graphs indicating resource usage and availability and the input and output of instructions

The distributions and time graphs are added to a tree, making it easy to navigate the many PIs. The tree has a generic structure, allowing PIs from a real environment to be added.

The distributions can be queried for statistical PIs, for example the average lead time or the percentage of instructions with a cost of at least 1,00 EUR. These statistical PIs are only calculated when they are queried.

C. Defining KPIs

From the large set of PIs a subset of single valued KPIs can then be selected for presentation in a strategic dashboard.

Additionally weighted KPIs can be defined, assessing multiple PIs at once. The weighted KPI mechanism was inspired by the UTA method [4]. The value of each PI is first translated to a score between 0 and 100, representing worst and best case respectively. The evaluation of the individual PIs and the weights can be manipulated completely by managers, allowing the definition of any KPIs they want. A single KPI can then replace multiple PIs in a dashboard, making the dashboard easier to read.

III. Results

A. Scalability

The time needed for both the simulation and the calculation of the PIs is linear in relation to the number of tasks and the number of instructions. Simulating 100 instructions for a process containing 1000 tasks takes 5,6 s to complete and the calculation of all PIs for the same process requires 14,0 s to complete.

Calculating the PIs can take a while for larger processes. Therefore it is recommended to run the PI calculation on multiple servers simultaneously.

B. Case study: supplying the operating room

The purpose of the HIPS project [5] is to streamline healthcare processes. In a first phase all processes concerning a hip surgery are analysed.

A hip surgery requires multiple components of a hip replacement. In order to operate a patient, hip components of the right sizes have to be available. Therefore supplying the operating room (OR) with hip components is of vital importance. The reception and registration of hip components at the hospital will be analysed in this case study.

Every week day 0 to 5 deliveries are made to the hospital each containing 1 up to 5 hip components. The OR worker receives each delivery and checks it for errors, f.e. wrong quantities. If errors occur, they have to be solved first, increasing the lead time of the delivery. The OR worker also checks every hip component for damage and reorders the component if it is damaged. If the hip component is not damaged, it can be registered by a pharmacist and put away in the OR.

The delivery process was analysed for a month, comprising 22 working days. One OR worker and one pharmacist are available from 8 am to 5 pm every day except on Sundays.

An operational dashboard was constructed allowing a detailed inspection of the process.

A distribution is shown for the lead times of both deliveries and individual hip components. Also a distribution is displayed indicating the costs of each delivery.

Sometimes the lead times for deliveries take from 24 up to 28 hours, see figure 2. These long lead times occur due to a broken component that has to be reordered, taking 24 hours on average. Nine deliveries take between 2 and 4 hours to process, because they arrive shortly after another delivery and the OR worker has to finish the other one first. Most deliveries can be processed within 2 hours, however.

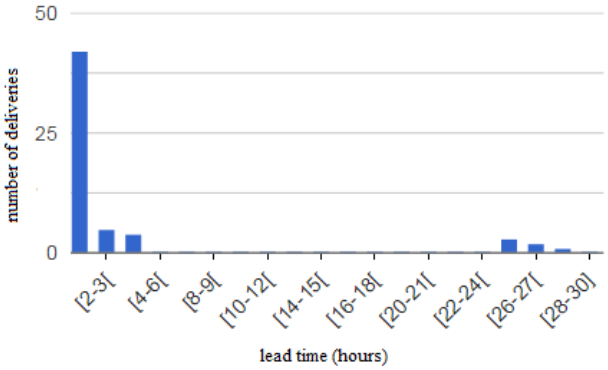


Fig. 2. Lead times of deliveries

The five most expensive and the five most time consuming tasks are shown. These tasks most influence the costs and the lead time of the entire process respectively. Thus, improving these tasks will have more influence than improving other tasks.

Time lines are shown for:

- the number of deliveries and hip components received and processed every day
- the number of deliveries and hip components in progress at each moment in time
- activity of the OR worker and the pharmacist at each moment in time

These time lines allow a detailed inspection of when each resource type is best deployed. Deliveries arrive between 8 and 12 am, causing peaks in the number of deliveries and the number of components in the system. The OR worker immediately starts processing the deliveries and components. The number of deliveries and hip components in the system rapidly decreases after every peak. Therefore the OR worker usually has no work after 12 am, see figure 3. Thus, deploying additional OR workers is only useful between 8 and 12 am.

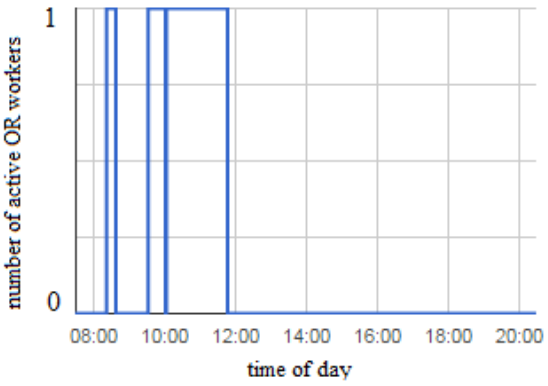


Fig. 3. Activity of OR workers

A strategic dashboard was also constructed and is used to evaluate the process and compare it with different alternatives. It displays all single valued KPIs, these are always numbers because they have to be evaluated. The SLA is used for the evaluation of the KPIs. F.e. the SLA defines that at least 50% of the deliveries has to be processed within 24 hours. Therefore the corresponding KPI is displayed as a gauge, see figure 4. Of all

deliveries, 89,5% are registered within 24 hours. The remaining 10,5% were delayed because at least one hip component had to be reordered. The meter is in the green zone indicating that more than 50% are processed within 24 hours, the KPI thus complies with the SLA. Failure is indicated with a red zone.

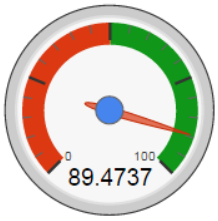


Fig. 4. Percentage of deliveries processed within 24 hours

The values of multiple PIs can be used to construct a weighted KPI, a KPI *efficiency* is defined to showcase how this works. Two PIs were used to calculate *efficiency*: the average cost of a delivery and the percentage of deliveries processed within 24 hours. The score of the lead times f.e. is 0 if the percentage is below 50% and 100 if it is 100%, percentages in between are scored linearly. Then each PI score is multiplied with a weight, in this case 0,2 is chosen for the cost and 0,8 for the lead time. The weighted sum of both PI scores results in the score of the KPI. This KPI is again displayed as a gauge with colours indicating failure or success. The weights and evaluation rules are chosen randomly in this case, but managers can construct KPIs to their liking, by selecting the PIs they want to include, defining their own set of evaluation rules and adding weights to each PI.

A gauge is also displayed for the percentage of components processed within one hour. Only 63,9% of the components were registered within an hour. There were three causes for components to be delayed:

- errors with the delivery
- the component is broken, thus requiring a reorder
- the OR worker is not available, because he is busy

Also a statistical overview for cost and lead times of deliveries and hip components is shown. The average delivery costs 8,81 euro and has a lead time of 3,59 s.

A pie chart displays the percentage of deliveries with at least one error. A similar pie chart indicates the percentage of broken components.

The activity rate of both the OR worker and the pharmacist is shown on a bar chart. The activity rate of the OR worker is only 11,1%. The cause of this can be determined by analysing the operational dashboard. Most of the days the OR worker is only active between 8 and 12 am and, some days there is even no work at all.

Every element on the strategic dashboard has a more detailed counterpart on the operational dashboard. The percentage of deliveries processed within 24 hours for example, is displayed as a gauge on the strategic dashboard, see fig. 4. The same information can be retrieved visually from the distribution of lead times on the operational dashboard, see fig. 2.

After analysing the case, some parameters were changed in order to see their effect on the lead time of deliveries and hip components. This led to the following conclusions:

If the number of deliveries or the amount of hip components is increased, the lead times of deliveries and hip components will increase. If on average 5 instead of 3 deliveries arrive, only 33,7% instead of 63,9% are registered within an hour. This is caused by the high occupation of the OR worker in the morning. The activity rate of worker increases from 11,1% to 20,6%.

To increase the number of components registered within an hour, two OR workers can be deployed. This way 75,8% of the components are registered within one hour, however the activity rate for the OR workers drops from 11,1% to 5,6%.

IV. Conclusion

A tool was constructed that allows managers to analyse and evaluate a process. The tool calculates PIs based on a simulator. A manager can indicate which PIs are of vital importance, thus marking the set of KPIs. A selection of PIs and KPIs can then be shown on an operational or a strategic dashboard. These dashboards are then used to analyse and evaluate the process.

A. Future work

The case has been configured manually. In order for the tool to be practical, a graphical environment has yet to be developed for configuring the process.

Only one process was considered in this case. In health care many processes interfere with each other, for example the supply of hip components and the hip surgery. The simulator has to be expanded in order to simulate multiple interacting processes.

Information about individual instructions can be retrieved from the simulation results. That information could be further exploited to determine the path and costs for a specific instruction.

References

[1] Benneyan, J.C. (1997). An introduction to using computer simulation in healthcare: patient wait case study
[2] Parmenter, D. (2015). Key Performance Indicators: Developing, Implementing, and Using Winning KPIs. https://books.google.be/books?id=bKkxBwAAQBAJ&oi=fnd
[3] Service level agreements and key performance indicators. https://www.springtideprocurement.com/?wpfb_dl=479
[4] Siskos, J., Jacquet-Lagreze, E.(1981). Assessing a set of additive utility functions for multicriteria decision-making, the UTA-method.
[5] HIPS. (2014). http://www.iminds.be/hips

Inhoudsopgave

Hoofdstuk 1

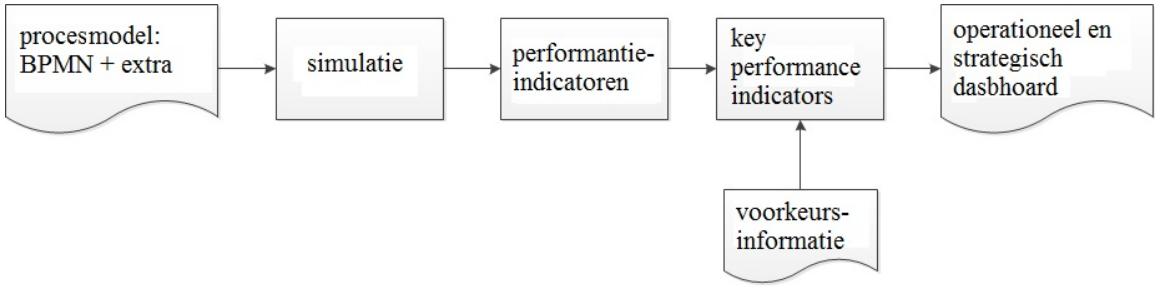
Doelstelling en terminologie

1.1 Doelstelling

Bedrijven streven er steeds naar hun productieprocessen zo efficiënt mogelijk te maken. Het doel hiervan kan bijvoorbeeld zijn zoveel mogelijk winst te maken of zoveel mogelijk eenheden te produceren. Om dat doel te bereiken is het noodzakelijk de bedrijfsprocessen op te volgen door data te verzamelen, zodat er kan bijgestuurd worden wanneer er problemen dreigen op te treden. Meestal zijn er zeer veel meetwaarden beschikbaar, deze kunnen worden geëvalueerd met performantie-indicatoren (PI). Opdat het management snel een beslissing zou kunnen nemen, mogen ze niet geconfronteerd worden met een overvloed aan data. Daarom is het beter verschillende PI's te combineren tot een beperkte verzameling Key Performance Indicators (KPI's).

Het doel van deze thesis bestaat eruit om met behulp van een bestaande simulator op een generieke en automatische manier PI's en KPI's af te leiden voor een proces. Deze PI's en KPI's moeten weer te geven zijn op een dashboard. Op basis hiervan kan het proces beoordeeld en geanalyseerd worden.

In hoofdstuk 2 wordt een systeem beschreven dat toelaat een verzameling PI's en KPI's te bepalen en weer te geven in een dashboard voor een gesimuleerd proces. Figuur 1.1 toont hierbij de verschillende stappen.



Figuur 1.1: Van proces naar meetwaarden en informatie

Het proces wordt beschreven met behulp van *Bussiness Process Modeling Notation* (BPMN). Daarnaast is voor de simulator informatie nodig over de uitvoeringstijden van de taken. Ook is er informatie vereist met betrekking tot de kosten en de beschikbaarheid van de bronnen zoals materieel en personeel. Hiervoor moet eerst statistische informatie verzameld worden uit reële processen. Al deze informatie wordt gebruikt door een *Discrete Event Simulator* (DES) om hieruit een verzameling PI's uit af te leiden. Doordat een DES sprongen maakt in de tijd is de simulatie efficiënt [1]. Hierbij wordt steeds een sprong gemaakt naar de volgende gebeurtenis. Bij de simulatie van processen zijn gebeurtenissen bijvoorbeeld het begin en het einde van een taak. Dat een DES succesvol kan gebruikt worden om een proces te analyseren en te verbeteren wordt aangetoond in [2]. De gebruikte simulator werkt op basis van MASON (Multi-Agent Simulator Of Neighborhoods) [3]. MASON is een Java-framework voor de ontwikkeling van *discrete event simulators* en is sneller is dan vergelijkbare alternatieven zoals Repast [4].

Daarna wordt aangetoond hoe KPI's worden afgeleid door middel van informatie die het management invoert. Hierbij kan een KPI eventueel samengesteld worden uit meerdere PI's.

Er wordt gepoogd om PI's en KPI's zo generisch mogelijk te definiëren zodat het systeem uitbreidbaar en zo algemeen mogelijk toepasbaar is. Het systeem moet dus bruikbaar zijn voor verschillende soorten productieprocessen. Bovendien kan er ook vertrokken worden van PI's die niet bepaald zijn met de simulator. De PI's en KPI's worden logisch gegroepeerd in een boomstructuur zodat ze snel terug te vinden zijn en gemakkelijk kunnen geïntegreerd worden in een dashboard. De bedoeling is dat de verzameling KPI's veel kleiner wordt dan het aantal PI's zodat de beoordeling vlotter kan gebeuren door het management.

Vervolgens wordt aangegeven hoe een overkoepelende boom kan gebruikt worden om de PI's en KPI's van meerdere dagen bij te houden. Deze boom kan dan gebruikt worden om een dashboard aan te maken. Er wordt een strategisch dashboard gebouwd dat toelaat in een oogopslag na te gaan of de doelstellingen van het bedrijf zijn bereikt. Daarnaast wordt een

operationeel dashboard gebouwd dat toelaat de data gedetailleerder te bestuderen.

Tevens wordt de efficiëntie van de simulator en van de berekening van de PI's nagegaan.

Ten slotte zal het systeem gebruikt worden om een ziekenhuisproces te analyseren waarbij heupprothese-onderdelen geleverd worden. Hierbij wordt een operationeel en een strategisch dashboard gebouwd met de berekende PI's. Op basis hiervan kunnen de kosten, de doorlooptijden en de personeelsbezetting bestudeerd worden.

1.2 Procesbeschrijving

Opdat het proces kan gesimuleerd worden, moet het voldoende beschreven zijn. Zowel het procesverloop als de verschillende vereiste bronnen moeten hierbij worden aangegeven. Daarnaast is informatie vereist over de duur van de taken en de invoersnelheid van het proces.

1.2.1 Business Process Model and Notation 2.0

Om het procesverloop te modelleren zal Business Process Model and Notation (BPMN) versie 2.0 gebruikt worden. BPMN is een standaard om bedrijfsprocessen mee te modelleren [5].

Volgens Reale is BPMN 2.0 eenvoudig genoeg zodat niet-technische gebruikers er een proces mee kunnen modelleren met behoud van voldoende informatie voor technische gebruikers, zoals informatici [6]. Dit betekent dat procesanalisten gemakkelijk processen kunnen beschrijven en dat deze beschrijving kan gebruikt worden om simulaties uit te voeren.

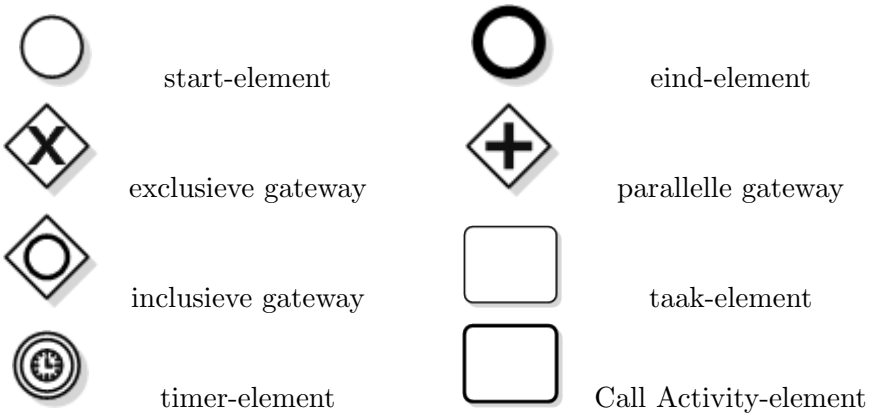
1.2.2 Belangrijkste BPMN-elementen

Tabel 1.1 geeft een overzicht van de belangrijkste BPMN-elementen die de simulator gebruikt. Elementen worden verbonden met pijlen die de volgorde aanduiden waarin ze worden doorlopen. Op die manier kunnen taken die na elkaar worden uitgevoerd, worden gemodelleerd door taakelementen in serie te plaatsen en met pijlen te verbinden.

Het startelement vormt het begin van het proces. Bij een proces dat de behandeling van patiënten beschrijft, komen hier de patiënten binnen. Het eind-element duidt aan wanneer de patiënten het proces verlaten.

Voor het beschrijven van parallelle uitvoering of voorwaardelijke opsplitsing kunnen *gateways* gebruikt worden. De verschillende paden tussen twee parallelle gateways kunnen tegelijk en onafhankelijk van elkaar afgewerkt worden. Om taken uit te voeren als aan een bepaalde voorwaarde is voldaan, worden exclusieve of inclusieve gateways gebruikt. Bij de opsplitsing

Tabel 1.1: De belangrijkste BPMN-elementen die gebruikt worden door de simulator

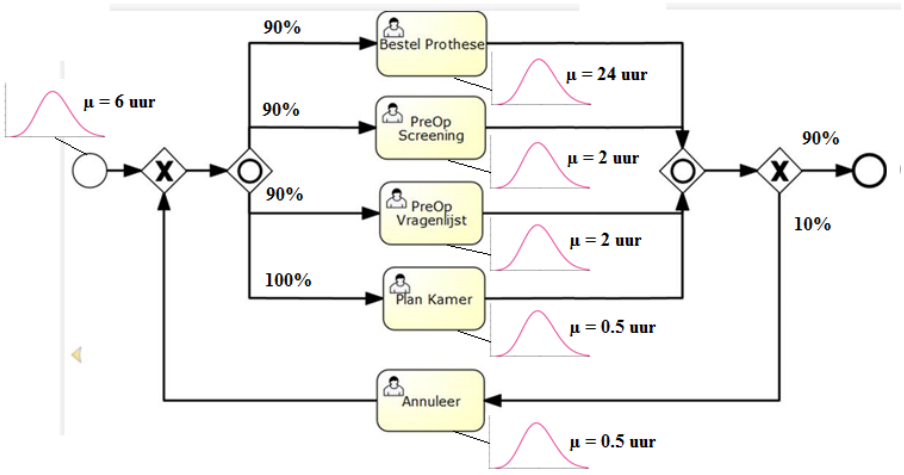


vanuit de gateway wordt aan elke tak een voorwaarde gekoppeld. Als de voorwaarde voldaan is dan wordt die tak uitgevoerd. Het verschil tussen exclusieve en inclusieve poorten is dat bij de exclusieve steeds exact één voorwaarde voldaan is en bij de inclusieve kunnen er meerdere voorwaarden voldaan zijn. Hierdoor kunnen bij een inclusieve gateway meerdere paden worden uitgevoerd.

Een timer duidt aan dat er bepaalde tijd moet gewacht worden. In deze masterproef wordt een timer na een taak geplaatst om de doorlooptijd van die taak te modelleren.

Call Activity-elementen worden gebruikt om binnen een proces een ander proces op te roepen. Op die manier kan een groot proces opgesplitst worden in verschillende deelprocessen die apart beschreven worden.

Figuur 1.2 toont een preoperatieproces waarin de meest rechtse exclusieve gateway gebruikt wordt om het proces te herstarten indien er een fout gebeurde. De meest linkse inclusieve gateway gaat na welke taken allemaal moeten uitgevoerd worden. De twee andere gateways brengen de gescheiden paden terug samen.



Figuur 1.2: Verloop van de voorbereiding voor een heupoperatie uitgetekend in BPMN met aanvullende informatie voor de simulator

Om de analysetool zo algemeen mogelijk uit te leggen, wordt in deze scriptie vaak de term instructie gebruikt. Deze instructie is hetgeen behandeld of verwerkt wordt in het proces. Zo worden de patiënten uit het preoperatieproces beschouwd als instructies. De term *start event* wordt gebruikt voor het binnenkomen van een instructie in het proces. Telkens wanneer een patiënt toekomt aan het preoperatieproces, treedt er dus een start event op.

1.2.3 Aanvullende proces-informatie

BPMN kan wel aanduiden hoe het procesverloop eruit ziet, maar er is geen standaardmanier om aan te geven welke bronnen de taken vereisen en wat de uitvoeringstijden zijn van die taken. Er moet dus nog extra informatie toegevoegd worden aan het BPMN-model.

Bij elke taak worden een of meerdere bronnen gebruikt, zoals bijvoorbeeld personeelsleden, operatiezalen, ER-scanners... Van elk type bron moet opgegeven worden hoeveel en wanneer ze beschikbaar zijn. Daarnaast moet per taak het aantal benodigde bronnen van elk type worden ingesteld.

Tevens moet opgegeven worden hoe lang elke taak duurt en hoe vaak er start events optreden, zie figuur 1.2. Aan elke taak wordt een distributie gekoppeld die de uitvoeringstijd aanduidt. Telkens een taak wordt uitgevoerd, gebruikt de simulator de distributie om de uitvoeringstijd te genereren. Ook het interval tussen twee opeenvolgende start events wordt opgegeven aan de hand van distributies. Door distributies te gebruiken wordt de variabiliteit die in de realiteit

optreedt, meegerekend in de simulator.

De keuzes bij de inclusieve en exclusieve gateways kunnen niet gesimuleerd worden zonder extra informatie. In de realiteit komt elke pijl die vertrekt uit deze gateways overeen met een voorwaarde. Deze voorwaarden hebben elk een bepaalde kans om voldaan te zijn. Deze probabiliteiten kunnen gebruikt worden om de gemaakte keuzes te simuleren. Zo zal de voorbereiding voor een operatie opnieuw moeten gebeuren indien er een fout optrad, de kans hierop is 10%. Merk op dat voor de exclusieve gateways de som van de kansen gelijk moet zijn aan één, aangezien er slechts één voorwaarde kan voldaan zijn.

Het procesverloop uitgetekend in BPMN 2.0 gecombineerd met aanvullende gegevens over de invoer, de taken en de gateways levert voldoende informatie voor de simulator. Na de simulatie van het proces kunnen dan verschillende PI's afgeleid worden.

1.3 Performantie-indicatoren

In deze thesis wordt een verzameling meetwaarden als performantie-indicatoren (PI) beschouwd. Ook de statistische waarden die uit die verzameling kunnen worden afgeleid, worden als PI's aanzien.

Zo wordt de verzameling meetwaarden voor de uitvoeringstijd van een knie-operatie voor alle patiënten in een distributie opgeslagen. Deze distributie wordt als één enkele PI beschouwd, waaruit statistische PI's, zoals de gemiddelde tijd en de variantie afgeleid kunnen worden.

De waarden van de PI's zijn afhankelijk van hoe het proces is georganiseerd, welke bronnen elke taak vereist en welke bronnen er beschikbaar zijn.

1.3.1 Bezettingsgraad van de gebruikte bronnen

Personeel en materiaal zijn schaarse bronnen, in die zin dat er steeds een beperking op het aantal staat. Voldoende bronnen inzetten heeft als voordeel dat ze steeds beschikbaar zijn wanneer nodig, zodat de instructies minder moeten wachten. Helaas stijgt de kost met het aantal ingezette bronnen, zodat dit niet mag overdreven worden. Door na te gaan wat de bezettingsgraad van elk type bron is, kan nagegaan worden of er minder dan wel meer van dit type bron nodig is. Het aantal bronnen dat beschikbaar is varieert in de tijd, zo worden er vaak minder personeelsleden ingezet in het weekend.

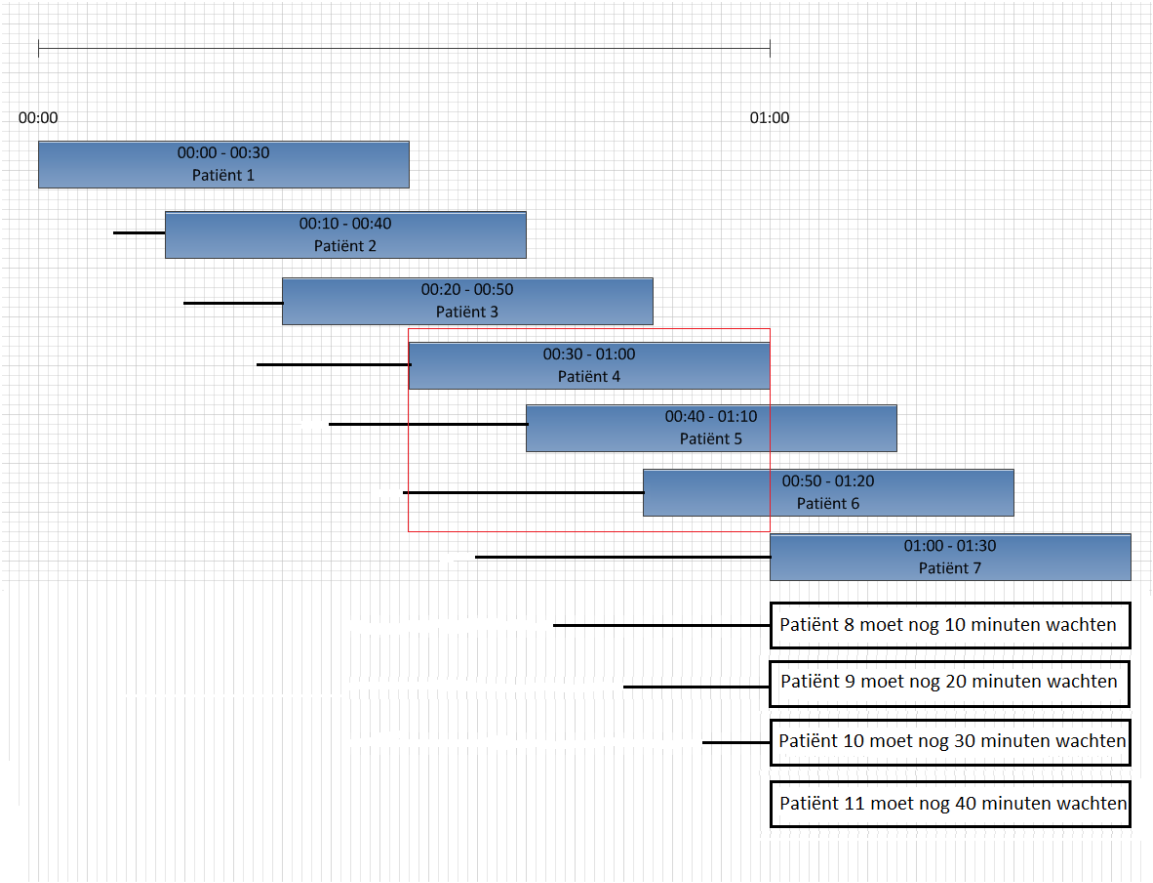
1.3.2 Doorlooptijden en wachttijden van elke taak en elk proces

Als de instructies zo snel binnenkomen dat dat het aanwezige personeel en materieel de vraag niet kan opvangen, zullen sommige instructies moeten wachten. Dit is uiteraard niet wenselijk. Beschouw bijvoorbeeld een proces waarbij patiënten geopereerd moeten worden. Ten eerste zullen patiënten het wachten vervelend vinden. Indien er wachttijden optreden bij cruciale taken zoals de operatie kan dit zelfs de gezondheid van de patiënten schaden. Daarom is het belangrijk de groei van de wachttijden in de gaten te houden.

Stel bijvoorbeeld dat er zich gemiddeld 10 patiënten per uur zich aanmelden aan het ziekenhuis, dit wordt de invoersnelheid genoemd. Dit betekent ook dat er ongeveer 6 minuten tussen twee opeenvolgende patiënten zit. Stel ook dat de behandeling van een patiënt gemiddeld een half uur duurt, dit is de uitvoeringstijd van het proces. Om te vermijden dat het aantal wachtende patiënten continu toeneemt moeten er ook 10 patiënten per uur geholpen worden, dit is de doorvoersnelheid van het proces. Als de gemiddelde doorvoersnelheid en de gemiddelde invoersnelheid gelijk zijn, zullen de wachttijden klein zijn, maar niet nul aangezien patiënten soms net iets sneller na elkaar binnenkomen. De doorlooptijd van elke patiënt is dan gelijk aan de som van de uitvoeringstijd en de wachttijd en is in dit geval gemiddeld net iets hoger dan een half uur.

Maar als er gemiddeld slechts zes mensen per uur kunnen geholpen worden, dan zal de wachtrij gemiddeld groeien met vier patiënten per uur, zoals op figuur 1.3. De aankomst van elke patiënt wordt hier voorgesteld als het begin van een horizontale lijn. Het einde van de zwarte lijn duidt het tijdstip aan waarop de behandeling aanvat voor een patiënt. De horizontale lijnen stellen dus voor hoe lang elke patiënt al gewacht heeft na een uur. De gekleurde rechthoeken duiden aan wanneer een patiënt behandeld wordt, de zwart omlijnde rechthoeken stellen de patiënten in de wachtrij voor net na een uur.

Er kunnen zes mensen per uur geholpen worden, dit betekent dat het ziekenhuis gemiddeld om de tien minuten iemand anders kan helpen. De elfde patiënt, die na een uur toekomt, zal moeten wachten op vier anderen en dit duurt gemiddeld 40 minuten. De doorlooptijd is nu 70 minuten, dit is ongeveer het dubbele van een proces waar voldoende bronnen aanwezig zijn.



Figuur 1.3: Tijdsverloop bij een proces met onvoldoende bronnen tegenover het aantal patiënten

Als er lange tijd te weinig bronnen worden ingezet, dan blijven de wachttijden en doorlooptijden stijgen. Daarom is het van belang om ervoor te zorgen de taken kort te houden en voor voldoende bronnen te zorgen om de invoerstroom van patiënten aan te kunnen. Door de uitvoerings-, wacht- en doorlooptijden van de verschillende patiënten op te volgen kunnen probleempunten in het proces opgespoord worden.

1.3.3 Start events

Om de resultaten nauwkeurig te kunnen analyseren is het handig om te weten wanneer de start events optraden. Zo kan bijvoorbeeld verklaard worden waarom de wachttijden bij de taken plots de hoogte ingaan, meestal is dit gecorreleerd aan de snelheid waarmee de instructies binnenkomen.

1.3.4 Keuzes van gateways

Hoe vaak een bepaald pad wordt genomen bij een exclusieve of inclusieve gateway kan invloed hebben op de doorlooptijden en bezetting van de bronnen. Zo kan een exclusieve gateway bijvoorbeeld testen of een checklist correct is ingevuld zoals in figuur 1.2. Als de lijst verkeerd is ingevuld, dan moet deze opnieuw ingevuld worden. Dit zorgt voor een langere doorlooptijd van de behandelde patiënt en zorgt voor een verspilling van de bronnen van de taak die moet herhaald worden. Het is in dit geval belangrijk de gemaakte keuzes bij de gateways op te volgen.

1.4 Key Performance Indicators

Key Performance Indicator (KPI's) zijn indicatoren die aspecten van de organisatorische performantie meten die kritisch zijn voor het succes van een organisatie [7]. PI's omschrijven verschillende aspecten van het proces, maar er zijn te veel indicatoren om te analyseren. Daarom moet een verzameling KPI's geselecteerd worden uit de PI's, zodat enkel de KPI's moeten worden bestudeerd.

Het management moet kunnen aangeven hoe deze KPI's samengesteld worden uit de PI's. Uiteindelijk krijgen ze een beperkte en overzichtelijke verzameling van KPI's. Deze KPI's kunnen dan worden weergegeven op een strategisch dashboard.

Een afzonderlijke PI kan worden aangeduid als KPI, hierbij moet dan opgegeven worden wat de aanvaardbare waarden zijn voor die KPI. De Service Level Agreement (SLA) definieert welke waarden voor een KPI aanvaardbaar zijn [8].

Er wordt ook toegelaten dat meerdere PI's worden samengenomen om hieruit één KPI te berekenen. Door meerdere PI's om te vormen naar één enkele KPI kan het aantal KPI's verder gereduceerd worden.

De KPI's bevatten een score die aanduidt hoe gewenst de waarden van één of meerdere PI's zijn. De score van een KPI wordt bepaald als het gewogen gemiddelde van de scores van zijn onderliggende PI's. Hierbij wordt de waarde van elke PI eerst omgezet in een score van 0 tot en met 100. Soms heeft een PI een totaal onaanvaardbare waarde, zodat de KPI een score van 0 krijgt ook al scoren de andere PI's wel goed.

De keuze van de juiste KPI's en vastleggen hoe ze gemeten worden is noodzakelijk om ze bruikbaar te maken voor analyse. De gekozen KPI's moeten de missie en visie van het bedrijf weerspiegelen.

1.5 Belang van een juiste selectie van KPI's

PI's en KPI's dienen om de huidige toestand van het bedrijfsproces in kaart te brengen en de evolutie van het proces op te volgen in de tijd. Door slechts enkele PI's en KPI's weer te geven, kunnen verschillende alternatieven snel vergeleken worden en kan de meest optimale oplossing gekozen worden.

Het is daarom van belang dat de gebruikte PI's en KPI's kunnen aantonen of de missie bereikt wordt. Deze missie kan verschillende zaken inhouden, zoals de doorlooptijd van het proces verkleinen, medische fouten vermijden, patiëntvriendelijkheid verhogen, technische vooruitgang boeken, verspilling van bronnen vermijden...

In de gezondheidszorg is het bijvoorbeeld vooral belangrijk het aantal heropnames zo laag mogelijk te houden. Daarnaast moet ook vooral gelet worden op problemen die bij veel patiënten voorkomen, zoals besmetting met de ziekenhuisbacterie MRSA. Meetwaarden die met deze zaken verband houden zijn sterke kandidaten om als KPI te kiezen [9].

De volledige organisatie moet de PI's en KPI's begrijpen en het moet duidelijk gedefinieerd zijn hoe ze moeten gemeten worden. De indicatoren moeten steeds op dezelfde manier gemeten worden om ze te kunnen vergelijken over de tijd heen. Eventueel kan het management voor gestandaardiseerde KPI's kiezen wat als voordeel heeft dat het ziekenhuis zichzelf met gelijkaardige bedrijven kan vergelijken, waardoor beter kan worden nagegaan of het goede kwaliteit aflevert [9]. Zo bestaat er in Vlaanderen het Vlaams Indicatorenproject (VIP^2) dat verschillende indicatoren onderverdeelt in zes domeinen. Er bestaat bijvoorbeeld een indicator die het percentage van de patiënten aangeeft dat een polsbandje met een correcte identificatie draagt [10].

De indicatoren die bepaald worden met behulp van de simulator zullen steeds op dezelfde manier bepaald worden voor de verschillende processen, taken en bronnen. Hierdoor kunnen ze dus ook onderling vergeleken worden net als de gestandaardiseerde indicatoren.

Hoofdstuk 2

De analysetool

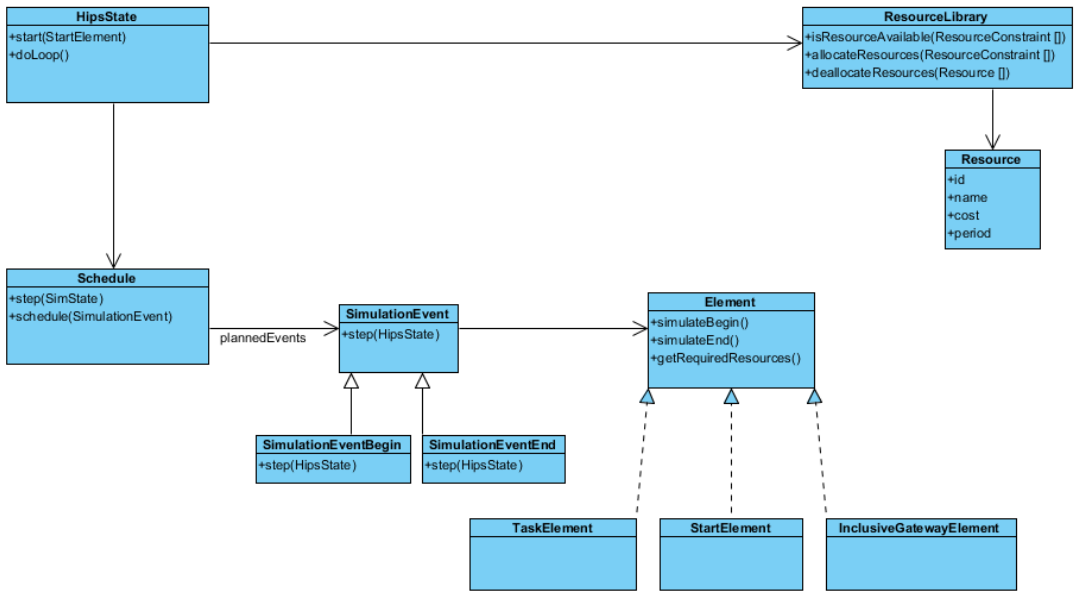
Om door middel van een simulator een proces te analyseren, worden verschillende stappen doorlopen, zie figuur 1.1 op pagina 2.

2.1 Simulator

Om de werking van een proces te evalueren, moeten er eerst voldoende resultaten (meetwaarden van de PI's) beschikbaar zijn. Het is onmogelijk elk proces eerst uit te testen in de realiteit indien meerdere alternatieven moeten vergeleken worden. Dit zou namelijk veel te lang duren en het voortdurend meten van de PI's voor alle alternatieven doet de kosten stijgen. In de gezondheidssector zou bovendien de gezondheid van de patiënten hierdoor in het gedrang komen. Daarom is een simulator aangewezen om snel veel resultaten te bekomen. Natuurlijk is het nadeel van een simulatie dat het fictieve resultaten aflevert. Daarom is het belangrijk dat de simulator zich zoveel mogelijk baseert op reële informatie. Zo wordt bijvoorbeeld voor elke taak aangeduid hoe lang ze gemiddeld duurt.

Om de bedrijfsprocessen te simuleren, wordt een discrete event simulator (DES) gebruikt. De gesimuleerde gebeurtenissen zijn discreet, er is dus steeds een duidelijk verschil tussen de toestand voor en na de gebeurtenis. Bij de simulatie van een fabricageproces waarbij bakvormen moeten gevuld worden, zal er bijvoorbeeld een gebeurtenis zijn die aanduidt dat de vorm vol is. Hoe de vorm van leeg naar vol gegaan is, wordt hierbij niet beschreven, wel wordt aangeduid hoe lang dit duurde. Voor de aanvang werd reeds een DES beschikbaar gesteld die in staat is om BPMN-processen te simuleren. Een kort overzicht van de werking ervan staat in figuur 2.1. Deze simulator is gebaseerd op de MASON-bibliotheek omdat hiermee snellere simulatoren

kunnen gebouwd worden dan met vergelijkbare bibliotheken. Daarboven ondersteund MASON de Activiti-engine zodat BPMN-processen kunnen gesimuleerd worden.



Figuur 2.1: Klassendiagram van de belangrijkste klassen van de gebruikte simulator

2.1.1 Scheduler

Om steeds te kunnen springen naar de volgende gebeurtenis worden de toekomstige gebeurtenissen en het bijhorende tijdstip bijgehouden in een wachtrij. Telkens een gebeurtenis wordt afgewerkt, wordt de eerstvolgende geplande activiteit uit de wachtrij gehaald en afgehandeld.

Elke DES bevat steeds een centrale *scheduler*, deze entiteit houdt de nog uit te voeren gebeurtenissen bij in de volgorde waarop ze zullen optreden. Daarnaast is er een simulatie-*engine* die één gebeurtenis tegelijk kan verwerken. Tijdens de verwerking van een gebeurtenis kunnen nieuwe gebeurtenissen gegenereerd worden. Deze nieuwe gebeurtenissen worden naar de scheduler gestuurd en worden daar toegevoegd aan de gesorteerde wachtrij. Als een simulatie-engine een gebeurtenis heeft afgewerkt, dan zal de scheduler de vroegst geplande gebeurtenis uit zijn wachtrij halen en aan de engine geven [1].

Het HIPSState-object is de specifieke engine voor de processimulator. Om de simulatie te starten moet een eerste gebeurtenis gepland worden met behulp van de start-methode. Daarna wordt de simulatie effectief gestart met de doLoop-methode. Deze methode voert een lus uit die bij elke iteratie de eerst geplande gebeurtenis opvraagt aan de scheduler en die vervolgens

uitvoert. Tijdens de uitvoering van elke gebeurtenis worden nieuwe gebeurtenissen aangemaakt en toegevoegd aan de scheduler. De lus zal uiteindelijk moeten stoppen, wanneer dit moet gebeuren wordt aangeduidt met behulp van de stopvoorwaarde.

2.1.2 Stopvoorwaarde

De gebruikte simulator laat toe om twee stopvoorwaarden toe te voegen voordat de simulatie gestart wordt. Indien beide stopvoorwaarden worden ingesteld, moeten ze beide voldaan zijn om de simulatie te beëindigen. De eerste mogelijke stopvoorwaarde duidt het aantal iteraties aan. De andere stopvoorwaarde stelt het maximaal gesimuleerde tijdstip in.

Daarnaast wordt in de achtergrond een convergentievoorwaarde getest, als deze voldaan is zorgt ze ervoor dat de simulatie stopt, ook al zijn de andere stopvoorwaarden niet voldaan. Deze convergentietest gaat na of de gesimuleerde uitvoeringstijden van de taken en van het volledige proces niet te veel verschillen van de geconfigureerde tijden. Als de gemiddelde uitvoeringstijd van elke taak ongeveer gelijk is aan de tijd die in de configuratie staat, dan wordt de simulatie vroegtijdig afgebroken.

2.1.3 Gebeurtenissen

De Schedule houdt de geplande gebeurtenissen (*SimulationEvents*) bij, zie figuur 2.1. Er worden slechts twee soorten gebeurtenissen beschouwd: de start en het einde van een proceselement (*Element*). Deze gebeurtenissen worden respectievelijk voorgesteld door de klassen *SimulationEventBegin* en *SimulationEventEnd*. Voorbeelden van gebeurtenissen in de gezondheidszorg zijn:

- een patiënt komt het ziekenhuis binnen of vertrekt
- een CT-scan die begint of eindigt
- een operatie begint of eindigt
- een bestelling van prothesen wordt geleverd

Een gebeurtenis bevat een verwijzing naar het proceselement dat moet worden uitgevoerd. Daarnaast bevat de gebeurtenis een aanduiding voor welke instructie het proceselement wordt uitgevoerd. Telkens een instructie door een proceselement moet passeren, wordt er steeds een begin- en een eind-gebeurtenis gesimuleerd. Voor taken en timers treden deze gebeurtenissen

op verschillende tijdstippen op. Het verschil tussen deze tijdstippen is dan de uitvoeringstijd van dat proceselement. Voor het startelement en de gateways vinden de beide gebeurtenissen op hetzelfde tijdstip plaats.

Een gebeurtenis wordt uitgevoerd door de step-methode op te roepen. Afhankelijk van het type gebeurtenis, wordt hierbij de methode simulateBegin() of simulateEnd() uitgevoerd van het bijhorende proceselement.

2.1.4 Rapporteren per instructie

Voor elke instructie die door het proces loopt, wordt een *Report*-object aangemaakt. Dit object wordt meegegeven aan de verschillende gebeurtenissen die optreden voor een instructie. Op deze manier kan voor elke gebeurtenis informatie worden opgeslagen in het *Report*-object. Elk rapport bevat bovendien een unieke identificatie, zodat de bijhorende instructies ook identificeerbaar zijn. De volgende informatie wordt voor elke gebeurtenis opgeslagen:

- het gesimuleerde tijdstip waarop de gebeurtenis optreedt
- de variabele kost per seconde en de vaste kost van het gesimuleerde element
- de bronnen die toegekend of vrijgegeven werden en hun bijhorende kosten.
- de unieke identificatie van het proceselement waarvoor de gebeurtenis voorkwam
- het type van het proceselement waarvoor de gebeurtenis voorkwam

Na afloop van de simulatie worden de rapporten opgeslagen, waarna ze gebruikt kunnen worden om PI's uit af te leiden.

2.1.5 Elementen plannen de volgende gebeurtenis

Tijdens de simulateBegin-methode van een proceselement wordt de bijhorende eindgebeurtenis van dat element toegevoegd aan de Schedule. Voor een taak zal de eindgebeurtenis op een later tijdstip gepland worden dan de begingebeurtenis, zodat het tijdsverschil tussen beide gebeurtenissen de uitvoeringstijd van de taak voorstelt.

Bij een startelement zal de simulateBegin-methode er bovendien voor zorgen dat de aankomst van de volgende instructie gepland wordt. Dit wordt gedaan door een nieuwe begingebeurtenis voor het startelement toe te voegen aan de Schedule. Hierdoor bevat de Schedule op elk moment minstens een gebeurtenis.

Tijdens de simulateEnd-methode van een element zal worden nagegaan naar welke elementen de instructie moet. Voor elk van deze elementen wordt een begingebeurtenis aangemaakt en toegevoegd aan de Schedule. Voor elke taak is er slechts één opvolgend element, maar voor gateways kunnen er meerdere zijn. Enkel bij het EndElement van het hoofdproces wordt geen nieuwe gebeurtenis aangemaakt, aangezien de instructie daar wordt afgewerkt.

2.1.6 Benodigde bronnen nagaan

Voordat de uitvoering van een proceselement kan beginnen, moeten alle benodigde bronnen beschikbaar zijn. De SimulationEventBegin gaat dit na en plaatst zichzelf terug in de Schedule indien niet alle bronnen beschikbaar zijn.

Voor elke taak kan een lijst van de benodigde bronnen worden opgevraagd. In de ResourceLibrary wordt dan nagegaan of al deze bronnen beschikbaar zijn. De bronnen worden toegekend bij begin van de taak, zodat ze niet kunnen gebruikt worden door andere taken zolang de taak bezig is. Na afloop van de taak worden deze bronnen terug beschikbaar gesteld voor andere taken.

Voor sommige taken moet de instructie ook beschikbaar zijn. Zo is een patiënt bijvoorbeeld steeds aanwezig bij zijn eigen operatie. Bij het klaarleggen van het materiaal moet de patiënt echter niet aanwezig zijn. Om ervoor te zorgen dat de instructie vereist is bij een taak, wordt voor die taak een extra bron vereist. Deze extra bron wordt aangemaakt tijdens de start event van de instructie, aangezien dat het moment aanduidt waarop de instructie het proces betreedt en dus beschikbaar wordt. Zo wordt een patiënt beschikbaar vanaf dat hij het ziekenhuis binnenkomt en wordt hij vereist voor de operatie. Na de operatie wordt de patiënt opnieuw beschikbaar gesteld voor andere taken.

2.1.7 Onzekerheid simuleren met behulp van een distributie

Enerzijds is een simulator aan te raden omdat snel veel meetwaarden kunnen verzameld worden en omdat het veiliger, sneller en goedkoper is dan het proces rechtstreeks te testen in een werkomgeving. Anderzijds moet er rekening mee gehouden worden dat de resultaten fictief zijn. Daarom is het belangrijk de simulatie zo op te bouwen dat ze zo goed mogelijk het echte proces benadert. Daarnaast moet bij de interpretatie van de resultaten aandacht zijn voor eventuele inschattingsfouten.

In een proces zijn volgende zaken onderhevig aan onzekerheid:

- het tijdstip waarop de instructies het proces binnenkomen
- de duur van de verschillende taken
- de gemaakte keuzes

Een knie-operatie bijvoorbeeld duurt niet altijd even lang, er zit een zekere spreiding op, daarnaast kunnen er complicaties optreden. Al deze onzekerheid zorgt ervoor dat instructies in een wachtrij kunnen terechtkomen als er tijdelijk onvoldoende bronnen beschikbaar zijn. Stel dat er bijvoorbeeld gemiddeld om de 10 minuten een patiënt binnenkomt en dat zijn behandeling ook gemiddeld 10 minuten duurt. Dan lijkt het alsof er geen wachtrijen zullen optreden, maar dit is enkel mogelijk als de intervallen steeds exact 10 minuten zijn. In de realiteit zal de tijd nodig voor de behandeling soms 12 minuten duren en zal de volgende patiënt zich al aanbieden na 9 minuten. Hierdoor zal deze patiënt 3 minuten moeten wachten. De wachttijden zijn dus niet alleen afhankelijk van de gemiddelde waarde voor de invoersnelheid, de uitvoeringstijden en de beschikbaarheid van bronnen, maar ook van de variabiliteit op deze parameters [11].

Als er voldoende statistieken verzameld zijn over de uitvoeringstijd van een taak zoals het klaarleggen van het operatiemateriaal kan een distributie worden opgesteld. De distributie duidt dan aan wat de kans is dat een bepaalde tijdsduur zal gesimuleerd worden tussen de gebeurtenissen voor het begin en het einde van een taak. Ook om de intervallen tussen twee opeenvolgende start events te simuleren wordt een verdeling gebruikt. Daarnaast wordt de kans van elke keuzemogelijkheid in het proces bepaald.

2.1.8 Simulatie per week

De PI's zullen bepaald worden uit de simulatie van het proces over een week. Er wordt een stopvoorwaarde gebruikt zodat de simulatie nooit meer dan één week simuleert. Oorspronkelijk garandeerde de simulator echter niet dat er een volledige week werd gesimuleerd doordat de convergentievoorwaarde vroeger kon voldaan zijn en de simulatie dus kan afbreken. Daarom werd de convergentievoorwaarde uitgezet, zodat er steeds exact een week wordt gesimuleerd.

2.1.9 Invoerbestanden

De simulator verwacht drie XML-bestanden als invoer: een bestand met de procesbeschrijving in BPMN, een bestand waarin de beschikbare bronnen worden gedefinieerd en een bestand met extra informatie over de taken zoals uitvoeringstijd en gebruikte bronnen. Voor eenzelfde proces,

moeten deze bestanden dezelfde naam hebben, enkel de suffix is anders, deze is respectievelijk *.bpmn20.xml*, *.bronnen.xml* en *.attributes.xml* . Om de simulatie te beginnen moet het bestand met de BPMN-beschrijving van het proces geselecteerd worden.

2.1.10 Procesbeschrijving in BPMN

Het bestand met de procesbeschrijving kan manueel aangemaakt worden, maar het is eenvoudiger om eerst het proces in een grafische BPMN-tekentool te tekenen en dan de beschrijving naar een xml-bestand te exporteren. Om dit bestand te kunnen gebruiken in de simulator moet de extensie *.bpmn20.xml* zijn. Voor de configuratie van de volgende twee bestanden is het handig als in de procesbeschrijving de automatisch gegenereerde id's van taken, gebeurtenissen en gateways vervangen worden door een leesbaardere vorm.

Momenteel gebeurt de toekenning van leesbare id's aan de verschillende proceselementen volledig manueel. Ook de koppeling van taken aan verschillende bronnen gebeurt manueel door middel van xml-bestanden zoals beschreven in hoofdstuk 2.1.11. Dit is echter tijdrovend, zodat in de toekomst best een grafische user interface ontwikkeld wordt hiervoor. In deze tool zou de gebruiker dan zelf namen aan de verschillende taken kunnen geven. Daarnaast kan de tool ook toelaten dat verschillende bronnen worden gedefinieerd.

2.1.11 Definitie bronnen

De simulator vereist ook een bestand met extensie *.resources.xml* waarin de bronnen gedefinieerd worden. Enkele voorbeelden van bronnen zijn verpleegkundigen, hoofdverpleegkundigen, nierspecialisten, operatiezalen...

Het basiselement *resources* bevat zowel kindelementen met de definities van bronnen, als elementen die types bronnen groeperen. Elk type bron wordt gedefinieerd met een *resource-tag.* De verschillende kindelementen van de resource-tag beschrijven de verschillende eigenschappen van het type bron. Deze kindelementen zijn:

id Een unieke naam die gebruikt wordt door de simulator om de verschillende brontypes te beheren.

cost Dit element duidt aan hoeveel een bron van dit type kost. De kost bestaat uit een vast en een variabel deel, die respectievelijk worden voorgesteld de kindelementen *fixed* en *variable* van het *cost*-element. Als de kost wordt weggelaten zijn beide waarden nul. De vaste kost is onafhankelijk van de uitvoeringstijd en wordt uitgedrukt in een munteenheid, bijvoorbeeld

de euro. Het variabel deel wordt uitgedrukt in een munteenheid per tijdseenheid, dit moet nog vermenigvuldigd worden met de uitvoeringstijd van een taak om de variabele kost te bekomen. Voor een instructie is de totale kost van de bron bij een taak waarvan uitvoeringstijd t is, gelijk aan de som van de vaste en de variabele kost:

$$kost\ bron = fixed + variable * t$$

periods De periodes duiden aan hoeveel bronnen van dit type er beschikbaar zijn gedurende een bepaald interval. De simulator doorloopt deze periodes cyclisch, dat wil zeggen als het laatste tijdstip dat opgegeven werd bereikt is, dan wordt voor dit type bron de tijd herstart vanaf nul. De eenheid van de tijd wordt niet opgelegd door de simulator, maar in deze masterproef wordt ervoor gekozen om de tijd uit te drukken in seconden.

Naast de bronnen zelf, kunnen groepen gedefinieerd worden met het element met als tagnaam *groups*, het kindelement *id* bevat een unieke identificatie van de groep. Daarnaast bevat *groups* de tak *elements* waarin dan een opsomming van *id*-tags staat die de *bron*-types voorstellen die tot die groep behoren.

Groepen zijn handig als er verschillende bronnen zijn die op dezelfde manier kunnen ingezet worden. Zo kunnen zowel een verpleegkundige als een hoofdverpleegkundige ingezet worden om een patiënt voor te bereiden op een operatie. Toch is er een onderscheid voor andere taken tussen beide types, zodat in dat geval het id van de brontypes aan een taak moeten gekoppeld worden. Dus zowel groepen als types van bronnen kunnen aan een taak gekoppeld worden in het attributenbestand (*.attributes.xml*) door de ids van de groepen of van de types bronnen op te geven.

2.1.12 Extra informatie over taken, gateways en het startelement

Eens de bronnen en het proces gedefinieerd zijn, moeten ze nog aan elkaar gekoppeld worden. Daarnaast moeten nog verschillende eigenschappen van taken, gebeurtenissen en gateways ingesteld worden, zoals uitvoeringstijd, kans van elk pad... Deze instellingen komen in een xml-bestand met achtervoegsel *.attributes.xml*.

De instellingen van een processtap zijn kindelementen van een *element*-tag dat verwijst naar het bijhorende proceselement in de procesdefinitie met behulp van het id-attribuut. De tags die de verschillende processtappen aanduiden, zijn op hun beurt kindelementen van het wortelelement *attributes*. De volgende tags beschrijven eigenschappen van processtappen:

rescheduleDistribution Dit element beschrijft het interval tussen twee opeenvolgende start events. Het attribuut *type* duidt aan welke verdeling gebruikt wordt om dit interval te bepalen. Er kan gekozen worden tussen een Poissonverdeling, een binomiale verdeling... Als bijvoorbeeld voor *poisson* gekozen wordt, dan duidt het kindelement *mean* de gemiddelde waarde van deze verdeling aan. *rescheduleDistribution* beschrijft hoe vaak instructies het proces binnenkomen en wordt daarom gekoppeld aan het startelement.

package Als dit element aan het startelement gekoppeld wordt, dan worden er pakketten gesimuleerd in plaats van individuele instructies. Het kindelement *size* beschrijft de verdeling die gebruikt wordt om de grootte van elk pakket te bepalen.

distribution Elk taakelement vereist exact een distributie die de uitvoeringstijd van de taak beschrijft. Opnieuw kan gekozen worden tussen een Poissonverdeling, een binomiale verdeling...

needsPatient Duidt aan of de instructie aanwezig moet zijn om de taak uit te voeren of niet. Bij administratieve taken betrokken bij een operatie bijvoorbeeld moet de patiënt niet aanwezig zijn, maar bij een operatie wel. Aangeven dat de instructie niet vereist is, gebeurt door *false* in te voeren, dit is de standaardwaarde. Als de instructie daarentegen wel vereist is, wordt *true* opgegeven.

needsPackage Duidt aan of voor de uitvoering van de taak het volledig pakket aanwezig moet zijn. De werking is volledig gelijkaardig aan *needsPatient*.

resources Duidt aan hoeveel bronnen van de verschillende groepen en types nodig zijn voor de uitvoering van elke taak. Voor elk type bron dat de taak vereist, is er een *resource*-element waarin het id van de groep of het type bron wordt opgegeven en het vereiste aantal.

cost Kosten kunnen ook gekoppeld worden aan een processtap in plaats van aan bronnen. Opnieuw bestaat deze kost uit een vast en een variabel deel, waaruit de totale kost van enkel de taak kan bepaald worden. Deze kosten worden dan opgeteld bij de kosten van de gebruikte bronnen om de totale kost van de taak te bepalen voor een instructie.

probability Elke exclusieve en inclusieve gateway moet evenveel *probability*-elementen bevatten als er pijlen uit vertrekken. De probabiliteiten moeten in dezelfde volgorde staan als de *flow*-elementen in BPMN die de pijlen voorstellen. Bij exclusieve gateways moet de som van de probabiliteiten gelijk zijn aan 1, aangezien exact een pad genomen wordt.

2.2 Performantie-indicatoren berekenen uit de resultaten van een simulatie

Het doel van dit onderdeel is om uit de rapporten bekomen van de simulator verschillende performantie-indicatoren te berekenen. Deze indicatoren worden dan opgeslagen in een generiek formaat dat zowel uitbreidbaar, als vervangbaar is door reële data.

2.2.1 Boomstructuur van PI's

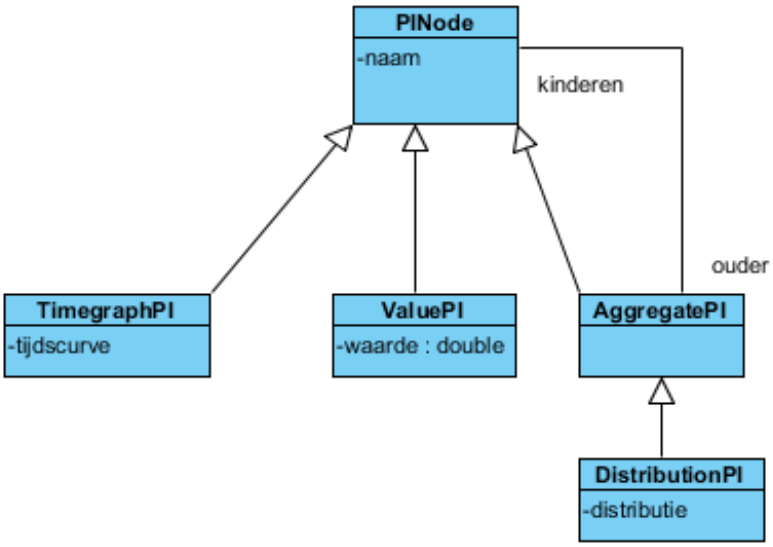
Omdat er zodanig veel PI's kunnen zijn, worden ze logisch geordend in een boomstructuur, zodat alle PI's makkelijk terug te vinden zijn. Er zijn vier soorten knopen beschikbaar om de boomstructuur van PI's mee op te bouwen zoals in figuur 2.2 te zien is:

ValuePI Een knoop van dit type bevat één enkele meetwaarde.

AggregatePI Een knoop van dit type bevat zelf geen waarde, maar heeft wel een aantal kindknopen die van een willekeurig type mogen zijn. AggregatePI-knopen zijn dus vergelijkbaar met de mappen van een bestandssysteem.

DistributionPI Een knoop van dit type bevat een distributie van een meetwaarde voor de verschillende instructies die doorheen het proces liepen. Er kunnen statistische waarden worden aangevraagd, zoals het gemiddelde, percentielen... Als zo een statistische waarde wordt opgevraagd, dan wordt een kindknoop van het type ValuePI aangemaakt die de statistische waarde bevat. De volgende keer dat dezelfde statistische waarde opgevraagd wordt, bestaat de knoop nog en moet de waarde dus niet meer worden herberekend.

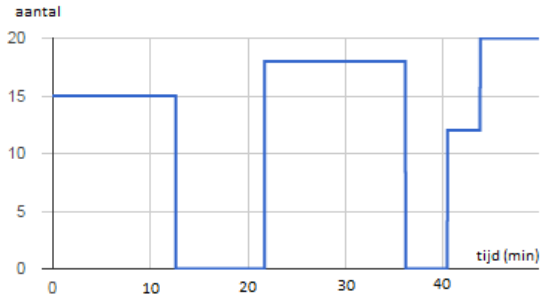
TimegraphPI Een TimegraphPI bevat een tijdscurve die het verloop van een meetwaarde in de tijd aanduidt. Zo is de PI *aantal instructies in het proces* een aanduiding van het aantal instructies die zich tegelijkertijd in het proces bevinden.



Figuur 2.2: De vier types PI-knopen waaruit elke PI-boom is opgebouwd

2.2.2 Tijdscurve

Verschillende analyses construeren een tijdscurve voor één of meerdere parameters. Zo wordt voor het aantal bronnen dat ingezet is op een gegeven moment een tijdscurve opgesteld. Deze curve kan worden opgeslagen als een tabel van punten, waarbij telkens het verband tussen tijd en waarde wordt aangetoond. Op de tijdstippen aangegeven in de tabel zal de waarde dan een sprong nemen naar de opgegeven waarde, dit principe wordt geïllustreerd in figuur 2.3.



tijd(min)	aantal
0	15
12,1	0
21	18
36,2	0
40,3	12
43,8	20

Figuur 2.3: Het tijdsverloop van het aantal leveringen in verwerking in een proces (links) en de interne voorstelling door middel van een tabel (rechts)

2.2.3 Distributie

Naast de tijdscurve, is een distributie een handige voorstelling van informatie. Vele analyses berekenen een distributie, zoals bijvoorbeeld de analyse van wachttijden van alle instructies bij

een taak. De distributie kan voorgesteld worden als een verzameling sleutel-waarde-paren. De sleutel in het voorbeeld van de wachttijden stelt dan een wachttijd voor en de waarde duidt aan hoeveel instructies zo lang moesten wachten.

Een voordeel van een distributie is dat ze eenvoudig te interpreteren is als ze grafisch getoond wordt. Bij de weergave moeten individuele waarden wel worden samengenomen tot intervallen. Daarnaast is het eenvoudig om de distributies van verschillende dagen op te tellen om zo een totaaldistributie te bekomen per maand of per jaar. Bovendien kunnen statistische waarden eenvoudiger berekend worden met deze voorstellingswijze dan bij een tijdscurve. Het nadeel van deze voorstellingswijze is dat informatie over het tijdsverloop verloren gaat.

De distributie ondersteunt de berekening van volgende statistische waarden:

- centrummaten: gemiddelde en mediaan
- spreidingsmaten: percentielen, minimum, maximum, variantie
- percentage van de waarden dat groter of kleiner is dan een opgegeven waarde

2.2.4 Welke performantie-indicatoren er worden berekend

Voor elk proces dat gesimuleerd wordt kan een volledige PI-boom berekend worden die informatie bevat over de afzonderlijke proceselementen, de subprocessen en de verschillende bronnen. De structuur van de boom is gelijkaardig voor elk proces. In figuur 2.4 is een voorbeeld van zo een PI-boom te zien voor een proces met de naam Bevoorrading.

Figuur 2.4: PI-boom berekend uit de resultaten van een simulatie van een bevoorradingsproces

De wortel van een boom draagt steeds de naam van het proces. Op het eerste niveau volgt een onderverdeling van de types instructies, bij de bevoorrading zijn er twee soorten instructies die worden opgevolgd: leveringen en onderdelen. De deelboom voor elk instructie-type bevat enkel PI's voor dat type. Zo wordt in de deelboom *onderdelen* enkel de doorlooptijden van de onderdelen opgeslagen en niet van de leveringen.

Op het tweede niveau is er een knoop *elements* die alle PI's bundelt over de verschillende proceselementen die de instructies doorliepen. Er wordt informatie bijgehouden over de verschillende gateways, taken en CallActivity- elementen. Voor elke inclusieve en exclusieve gateway wordt voor ieder mogelijk pad een distributie bijgehouden die aanduidt hoeveel instructies dat pad namen. Voor de taken worden distributies bijgehouden voor de doorloop-, uitvoerings- en wachttijden en voor de vaste, variabele en totale kosten.

Daarnaast bevat het tweede niveau ook een knoop *process*. Deze knoop bevat PI's voor de doorlooptijden, de variabele, vaste en totale kosten voor het volledige proces. Daarnaast bevat de *process*-knoop PI's die de invoer en uitvoer van instructies in de tijd beschrijven.

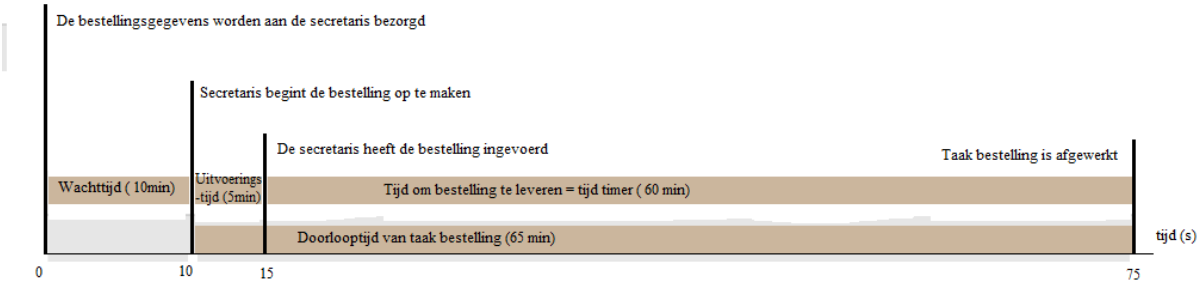
Ten slotte is er op het tweede niveau een knoop *resources* die voor elk brontype een kind

bevat. Voor elk brontype wordt bijgehouden hoeveel bronnen er beschikbaar en ingezet zijn op elk moment.

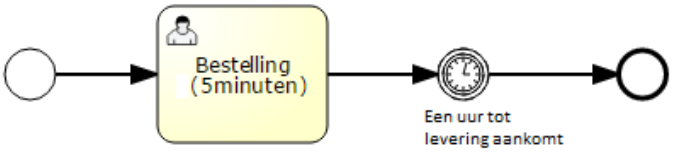
Tijdsanalyses van de diverse taken en het proces

Voor elke taak worden de distributies opgesteld voor de uitvoeringstijden, wachttijden, doorlooptijden en totale tijden uit de rapporten van alle instructies. De uitvoeringstijd is de tijd nodig om de taak uit te voeren, gedurende deze periode zijn de verschillende bronnen actief betrokken bij de uitvoering van de taak. De wachttijd van een instructie bij een taak is de tijd tussen het beschikbaar worden van die instructie voor die taak en het begin van uitvoering van die taak. Wachttijden ontstaan doordat er niet voldoende bronnen zijn om de taak uit te voeren op het moment dat de instructie bij de taak aankomt. Na de uitvoering van de taak moet soms nog gewacht worden totdat het resultaat van de taak beschikbaar is.

Beschouw bijvoorbeeld een taak *bestelling* waarbij enkele onderdelen moeten besteld worden. In figuur 2.5 worden de verschillende gebeurtenissen en de intervallen die worden berekend weergegeven voor een instructie. De bestellingsgegevens worden op het bureau gelegd van de secretaris, die op dat moment nog met iets anders bezig is. Een kwartier later geeft de secretaris de bestelling in en stuurt ze door naar de leverancier, dit duurt 5 minuten. De wachttijd is dus 15 minuten en de uitvoeringstijd 5 minuten. Nadat de bestelling is verstuurd, duurt het nog een uur voordat de bestelling toekomt, maar gedurende die tijd kan de secretaris andere taken uitvoeren. De doorlooptijd van de bestelling is dus een uur en vijf minuten, dit kan in BPMN gemodelleerd worden door middel van een timer, zie figuur 2.6.



Figuur 2.5: Berekening van de uitvoerings-, wacht- en doorlooptijd voor een taak



Figuur 2.6: Doorlooptijd in BPMN modelleren voor een besteltaak

Daarnaast wordt voor elke taak het tijdsverloop van de grootte van de wachtrij in de tijd opgeslagen. In het voorbeeld is dit het aantal bestellingsgegevens dat op het bureau van de secretaris ligt.

Ook voor het volledige proces en voor de subprocessen worden distributies aangemaakt voor zowel de uitvoeringstijden als de doorlooptijden. Hierbij wordt voor elke instructie de uitvoeringstijd van het proces berekend als de som van de uitvoeringstijden van alle doorgelopen taken. De doorlooptijd wordt bepaald als het verschil in tijd tussen het begin en het einde van het proces, bij het voorbeeld is dit dus 75 minuten.

Kostenberekening van taken en van het proces

De kost van elke bron omvat een vast en een variabel deel. De vaste kost wordt hierbij uitgedrukt in een munteenheid zoals de euro. Het variabele deel wordt uitgedrukt in een munteenheid per seconde. Voor een instructie zijn de vaste en variabele kost van de taak afhankelijk van welke bronnen er ingezet werden. De vaste kost van de taak wordt dan berekend als de som van de vaste kosten van alle bronnen. Voor de variabele kost van de taak worden eerst de variabele kostendelen van alle gebruikte bronnen opgeteld, waarna dit product vermenigvuldigd wordt met de uitvoeringstijd. De totale kost is dan gelijk aan de som van de vaste en de variabele kost.

Op deze manier worden de vaste, variabele en totale kost berekend van de taak voor elke instructie. Voor elk kostentype wordt dan een distributie gemaakt die de kosten van alle instructies samenvat. Deze distributies worden voor elke taak aangemaakt.

Ook voor het proces worden drie distributies aangemaakt net als bij elke afzonderlijke taak. De kost van het proces voor een instructie wordt berekend door de kosten van de verschillende taken op te tellen, net zoals bij de uitvoeringstijden gedaan werd.

Instroom en uitstroom van instructies

Er wordt een distributie aangemaakt die aanduidt hoeveel keer er een bepaalde tijd tussen twee opeenvolgende instructies ligt. Dit dient om bij onverwachte waarden van bepaalde PI's na te gaan of gesimuleerde tussentijden groter of kleiner uitvallen dan verwacht.

Daarnaast wordt een tijdscurve aangemaakt voor het proces en alle subprocessen die weergeven hoeveel instructies zich in dat proces bevinden. Het aantal instructies dat zich in een proces bevindt is op elk moment gelijk aan de totale invoer min de totale uitvoer.

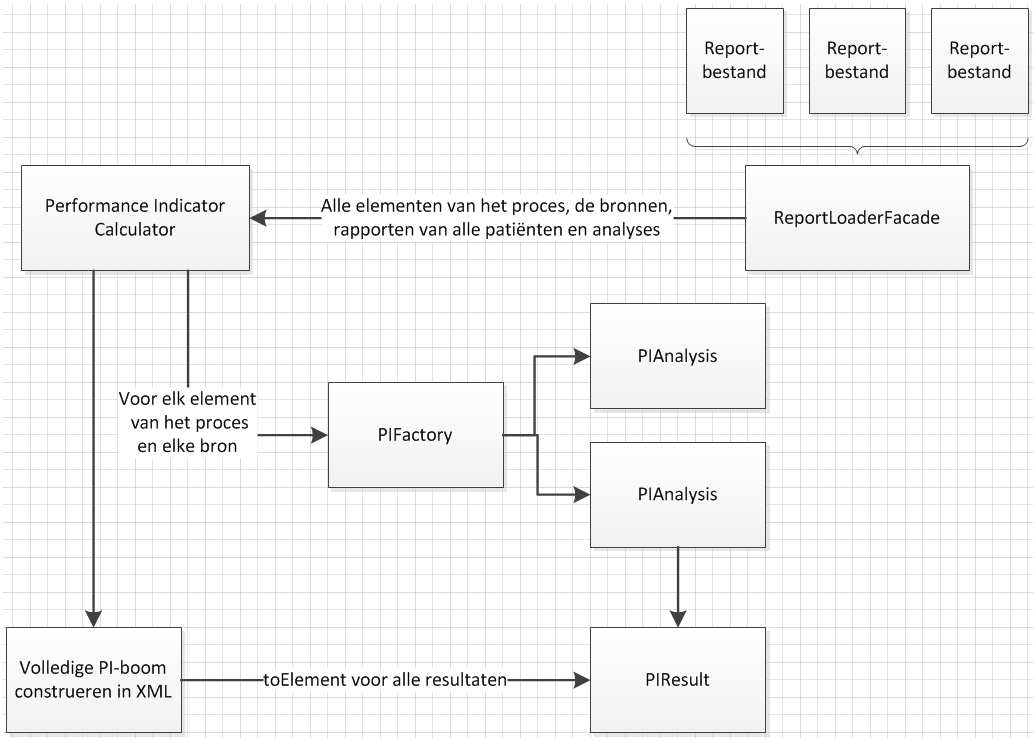
Inzet en beschikbaarheid van elk brontype

Het aantal beschikbare bronnen van eenzelfde type wordt in een tijdscurve opgeslagen. Daarnaast wordt een distributie opgesteld die de tijdsduur aanduidt dat een aantal bronnen tegelijk beschikbaar zijn. Voor het aantal ingezette bronnen van eenzelfde type worden een gelijkaardige tijdscurve en distributie aangemaakt.

De bezettingsgraad op een bepaald moment wordt gedefinieerd als het aantal bronnen van een type dat ingezet is, gedeeld door het aantal beschikbare bronnen van dat type. Ook voor de bezettingsgraad wordt zowel een distributie als een tijdscurve opgeslagen.

2.2.5 Hoe de performantie-indicatoren worden berekend

Nadat de simulatie van het proces is uitgevoerd, zijn de rapporten van alle instructies beschikbaar in xml-bestanden. De Performance Indicator Calculator kan met behulp van deze rapporten de hierboven beschreven PI's berekenen. Vaak moeten gelijkaardige PI's berekend worden op verschillende delen van het proces, waardoor dezelfde berekenmethode meermaals kan worden hergebruikt. Figuur 2.7 duidt aan hoe de berekening van de PI's verloopt.



Figuur 2.7: Performantie indicatoren worden berekend uit de rapporten van een simulatie

De ReportLoaderFacade zorgt ervoor dat er geen rekening moet gehouden worden met verschillende xml-bestanden en laat een objectgeoriënteerde toegang tot de rapporten van de simulator toe. De ReportLoaderFacade kan ook de namen van alle proceselementen en bronnen achterhalen.

De Performance Indicator Calculator maakt de volledige PI-boom aan voor het proces door voor elk proceselement en elk brontype een afzonderlijk *PIFactory*-object aan te maken en uit te voeren. De PIFactory-objecten maken de juiste analyses aan en die analyses maken op hun beurt *PIResult*-objecten aan. Deze resultaten bevatten één of meerdere distributies en tijdscurves. De Performance Indicator Calculator gebruikt deze resultaten om de PI-boom te construeren.

Voor elke bron en elk element van het proces wordt een tak in de PI-boom gecreëerd. Voor de verschillende types elementen is telkens andere informatie gewenst. Daarom wordt afhankelijk van het type van het proceselement een bijhorend PIFactory-object aangemaakt. Een PIFactory verzamelt de verschillende analyses die nuttig zijn voor een bepaald type element. Zo zal de TaskPIFactory analyses uitvoeren voor wachttijden en uitvoeringstijden, terwijl een GatewayPIFactory de wachttijden en de gemaakte keuzes analyseert. Voor bronnen is er een ResourcePIFactory-object nodig. Bij de aanmaak van een PIFactory wordt ingesteld welk pro-

ceselement of welke bron er geanalyseerd moet worden.

Voor verschillende proceselementen moeten gelijkaardige analyses uitgevoerd worden. Zowel voor gateways als voor taken bijvoorbeeld worden de wachttijden geanalyseerd. Door de verschillende analyses onder te brengen in aparte PIAnalysis-klassen kunnen ze hergebruikt worden door meerdere PIFactory-klassen. Zo wordt de WaitingTimePIAnalysis door zowel GatewayPIFactory als de TaskPIFactory. De PIAnalysis voert de analyse uit op de verzameling rapporten van alle instructies.

De resultaten van de analyses moeten nog verpakt worden in PI's. Deze resultaten worden niet rechtstreeks in een PI-boom omgezet, maar eerst in een xml-bestand opgeslagen. Een parser kan dan dit bestand inlezen en daaruit de PI-boom opstellen. Door deze aanpak is het ook mogelijk om PI-bomen die niet berekend werden met behulp van de simulator in te voeren. Hierdoor kan ook een PI-boom met waarden uit reële metingen van het proces ingevoegd worden.

2.2.6 Bottlenecks zoeken met behulp van PI's

Bottlenecks (Engels voor flessenhals) zijn onderdelen van het proces die ervoor zorgen dat het proces vertraging oploopt. In een operationeel proces zijn niet alle productiestappen even snel. De taak met de laagste doorvoersnelheid bepaalt de doorvoersnelheid van het volledige proces. De doorvoersnelheid is het aantal instructies dat het proces kan verwerken per uur. Stel bijvoorbeeld dat een patiënt steeds twee stappen na elkaar moet ondergaan: scannen en opereren. De scan duurt een kwartier, zodat er vier patiënten per uur kunnen gescand worden. De doorlooptijd van de operatie is echter een half uur, zodat er slechts twee patiënten per uur kunnen geopereerd worden. Hierdoor is het niet voordelig om meer dan twee patiënten per uur te scannen, aangezien ze toch zullen moeten wachten op een operatie. In dit geval is de operatie dus een bottleneck voor het proces.

Om de doorvoersnelheid toch te kunnen verhogen, bestaan er twee oplossingen. Eventueel kan de doorlooptijd van de operatie verlaagd worden. Als dit niet mogelijk is dan kan ervoor gezorgd worden dat er twee operaties tegelijk kunnen doorgaan door de benodigde bronnen te verdubbelen. De operatie was namelijk een bottleneck doordat er onvoldoende bronnen beschikbaar zijn, in dit geval chirurgen. Daarom kunnen deze bronnen ook als een bottleneck beschouwd worden. Door twee chirurgen in te zetten, wordt de bottleneck weggewerkt, aangezien de wachttijd na de scan vervalt.

Bronnen waarvan er tekorten zijn, kunnen aan de hand van de bezettingsgraad worden

opgespoord. Stel bijvoorbeeld dat bij de operatie een chirurg en twee verpleegkundigen nodig zijn. Als de chirurg constant bezet is, maar de verpleegkundigen zijn slechts 20% van de tijd bezig dan is het duidelijk dat er enkel meer chirurgen nodig zijn om meer patiënten te kunnen opereren. Door het aantal chirurgen te verdubbelen, kunnen er twee operaties tegelijk uitgevoerd worden. Hierdoor kunnen er wel vier patiënten per uur geopereerd worden, zodat de doorvoersnelheid ook vier patiënten per uur is en de bottleneck weggewerkt is. De bron chirurgen wordt hier ook als bottleneck beschouwd, aangezien hun aantal niet voldoet om de taken te voltooien. Zodat zowel taken als bronnen moeten gezocht worden die als bottleneck optreden.

Een goede indicator dat er bottlenecks bestaan in het proces is de wachtrij. Als het proces overbelast wordt, zullen de wachtrijen bij de meeste taken groeien. Dit is steeds zo, onafhankelijk of er bottlenecks zijn of niet. Maar bij aanwezigheid van bottlenecks zullen de wachttijden echter ongelijkmatig groeien. Taken waarbij de wachtrijen veel sterker groeien zijn bottlenecks. Door na te gaan waar de wachttijden het grootst zijn, kan de taak gevonden worden die bottleneck is voor het proces.

Aangezien een taak pas kan uitgevoerd worden als de benodigde bronnen beschikbaar zijn, is het ook van belang te monitoren welke bronnen de hoogste bezettingsgraad hebben. Meestal wordt een bron ingezet voor meerdere taken. Als het proces zo zwaar wordt belast dat die bron continu in gebruik is, dan zullen alle taken waarvoor de bron wordt ingezet hieronder lijden. Elk van die taken is dus een bottleneck, maar de schaarste van de bron is hiervan de oorzaak.

2.3 KPI's samenstellen uit PI's

De grote verzameling van PI's moet gereduceerd worden tot een beperkte collectie van KPI's. De verzameling KPI's is de deelverzameling van de PI's, waarvan het management vindt dat zij de doelstellingen weerspiegelen. Voor elke KPI kan worden aangeduid wat de gewenste waarden zijn met behulp van de SLA. Door de KPI-waarde en de gewenste waarde van elke KPI te vergelijken, kan worden afgeleid of de doelstelling al dan niet bereikt is.

Aan de hand van de KPI's kunnen processen beoordeeld worden en verschillende alternatieven vergeleken worden. Bij het vergelijken van processen kunnen veel verschillende factoren van belang zijn, zodat de verzameling KPI's nog atlijd groot kan uitvallen. Bovendien is er meestal niet een alternatief dat uitblinkt in al zijn aspecten. Zo kan het ene proces een betere doorlooptijd hebben, maar kan een ander proces dan weer goedkoper uitvallen. Welk proces er verkozen wordt, hangt af van welke KPI het belangrijkst is en wat de ideale waarde is van elke

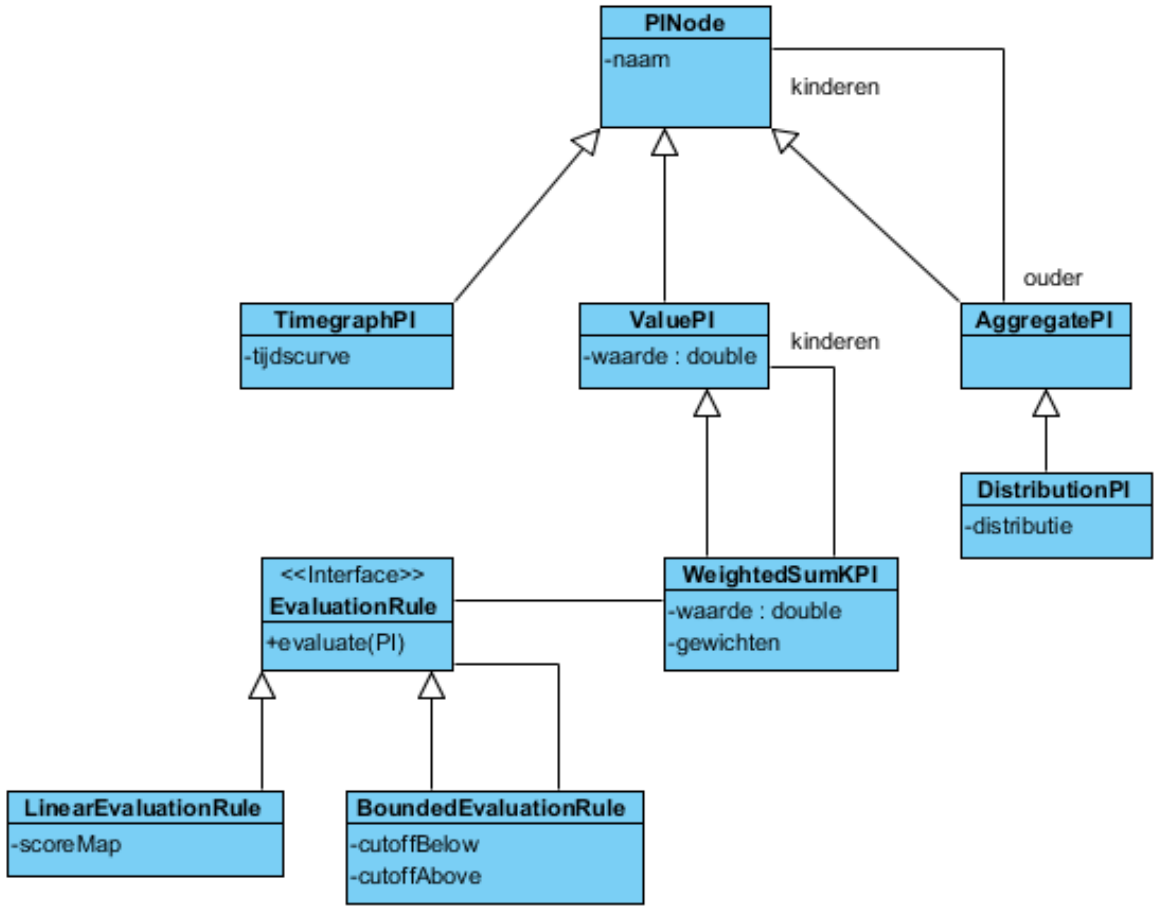
KPI. Daarom wordt een gewogen KPI voorgesteld die de waarden van meerdere PI's samenvat, om zo het aantal KPI's verder te reduceren.

Aan de hand van de voorkeursinformatie van de manager kunnen gewogen KPI's worden samengesteld uit meerdere PI's. Deze KPI's zijn eenvoudiger te vergelijken zodat de verschillende alternatieven sneller vergeleken kunnen worden.

Er zijn verschillende manieren om de beoordeling van een verzameling PI's uit te drukken met één waarde zodat dit overeenkomt met de voorkeur van de gebruiker. Een functie die een PI afbeeldt op een score heet een nuttigheidsfunctie (*utility function* in het Engels). De veelgebruikte en meest praktische benaderingen maken hierbij gebruik van een gewogen som [14]. Eerst wordt aan elke PI een score toegekend, waarbij bijvoorbeeld 100 het ideale geval voorstelt en 0 het slechtste geval. Daarna wordt aan elke PI een gewicht gekoppeld, waarbij de som van de gewichten 1 is. De voorkeur van de gebruiker wordt dus uitgedrukt in de gewichten en scoringsmechanismen van elke PI.

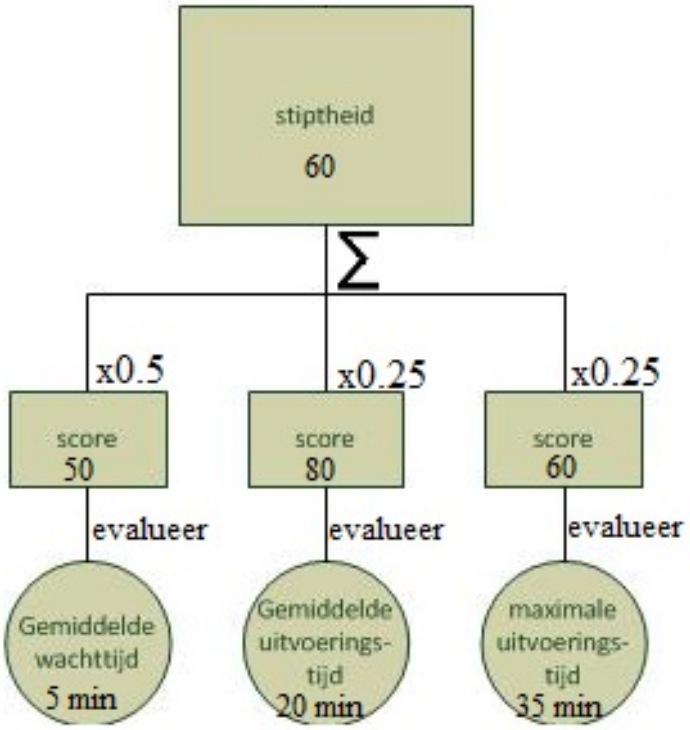
2.3.1 KPI's als uitbreiding op de boomstructuur

Er is reeds een boom van PI's aanwezig waarin de verschillende PI's snel gevonden kunnen worden. Deze boom zal door toevoeging van de gewogen KPI's worden uitgebreid, zie figuur 2.8. Het klassendiagram is een uitbreiding op dat van figuur 2.2 op pagina 21.



Figuur 2.8: klassendiagram PI's en KPI's

Een gewogen KPI bevat enkel ValuePI's als kinderen en is zelf ook een ValuePI. Elk kind van de KPI wordt gekoppeld aan een evaluatieregel gelijkaardig aan de UTADIS-methode [12]. Deze evaluatieregel vertaalt de waarde van de PI naar een score tusssen 0 een 100. Hierbij is een score van 0 zeer slecht en duidt een score van 100 op een ideale PI-waarde. De KPI-waarde wordt dan berekend als de gewogen som van de scores van de kinderen zoals ook beschreven in [13]. Hierbij duiden de gewichten de relatieve belangrijkheid van de PI's aan. Een voorbeeld hiervan is te zien in figuur 2.9.



Figuur 2.9: Samenstelling van KPI's uit één of meerdere PI's of uit andere KPI's

Doordat KPI's zelf ValuePI's zijn kunnen meerdere KPI's opnieuw worden samengenomen om een overkoepelende KPI te maken. Als er veel PI's van belang zijn kan dit gebruikt worden om de evaluatie van het proces in meerdere stappen op te bouwen. Eerst kan bijvoorbeeld per taak een gelijkaardige PI aangemaakt worden en deze kunnen dan opnieuw samengenomen worden tot een KPI op procesniveau.

2.3.2 Opvragen van de voorkeursinformatie

De gewichten en de evaluatieregels kunnen pas worden opgesteld als de voorkeursinformatie gekend is. Deze voorkeur kan op verschillende manieren achterhaald worden.

De eenvoudigste methode voor een gebruiker is om een gegenereerde lijst van opties met bijhorende PI's te ordenen, waaruit dan de parameters worden berekend. Het voordeel is dat de gebruiker zich niet moet bekommeren om de uitdrukking in gewichten en dat hij gevoelsmatig zijn voorkeur kan aanduiden. Het nadeel is dat de uiteindelijk verkregen KPI's die de gebruiker krijgt afhankelijk zijn van de implementatie van het systeem.

Een gelijkaardig mechanisme is het Analytic Hierarchy Process (AHP) waarbij men de relatieve belangrijkheid van paren PI's moet aanduiden [14]. Deze techniek werd ontwikkeld omdat

mensen het makkelijker vinden om twee zaken te vergelijken dan een hele reeks waarden in een keer. Een nadeel is hier dat de gewichten kunnen afhangen van de volgorde waarin de paren PI's beoordeeld worden. Met AHP kunnen ook de evaluatieregels afgeleid worden door paren waarden van de PI's te vergelijken. Er zal wel nog moeten opgegeven worden in welk bereik mogelijke waarden van de PI liggen. Als het bijvoorbeeld over een kost gaat dan is de waarde steeds groter dan nul, terwijl de winst ook negatief kan zijn. Bovendien is het moeilijk een bijpassende curve te bepalen en in te schatten hoeveel punten hiervoor nodig zijn. Is het voldoende om te stellen dat 2500 euro winst het ideale geval is, 500 euro verlies het slechtste geval en dat daartussen een lineair verband is? Of zijn er meerdere paren bedragen die moeten beoordeeld worden en is de uiteindelijke grafiek misschien een parabool of zelfs een willekeurige veelterm?

Bij beide benaderingen moeten de juiste parameters berekend worden zodat de nuttigheidsfunctie zo goed mogelijk aansluit bij de voorkeur van de gebruiker. UTilité Additive (UTA) maakt gebruik van lineair programmeren om de best mogelijke nuttigheidsfunctie te vinden, dit wordt beschreven in [15]. Het kan zijn dat er geen functie kan gevonden worden die aan alle opgegeven voorwaarden voldoet. Dit is bijvoorbeeld het geval als dezelfde optie tweemaal gegenereerd wordt en dat de gebruiker ze de ene keer bovenaan en de andere keer onderaan de voorkeurslijst plaatst.

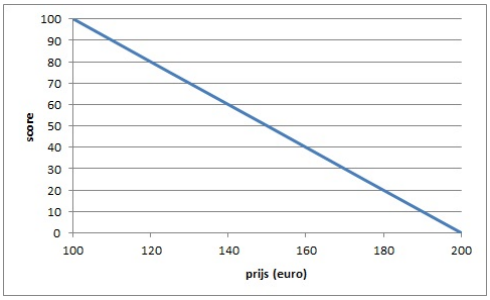
Uiteindelijk werd ervoor gekozen om de gebruiker expliciet alle parameters zelf te laten invoeren. Op die manier kan de gebruiker alles configureren zoals hij het wil. Hierbij is het belangrijk dat de gekozen instellingen goed overwogen zijn, aangezien een klein verschil in gewichten reeds kan resulteren in een totaal andere waarde voor de KPI. De beste optie voor het opstellen van de beoordelingsregels en de gewichten is hiervoor beroep te doen op een expert.

2.3.3 Evaluatieregels met behulp van lineaire intervallen

Om de waarden van verschillende PI's te reduceren naar een totaalscore, moeten ze eerst vergelijkbaar gemaakt worden. Onderstel bijvoorbeeld een PI die de wachttijd in minuten aanduidt en een PI die de gemiddelde kost per patiënt aanduidt. Het aantal minuten kan niet rechtstreeks vergeleken worden met een bedrag door hun verschil in maateenheid. Door de waarden te reduceren naar een score die aanduidt hoe geschikt de PI is, wordt dit probleem opgelost.

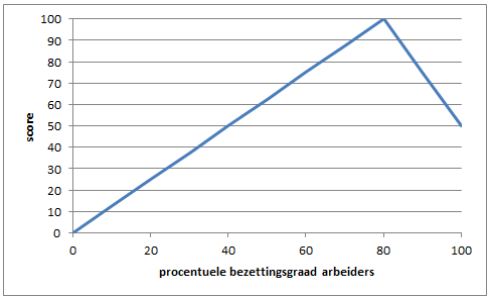
De beoordeling van een PI kan uitgedrukt worden op een schaal van 0 tot 100. Als bij de aankoop van een apparaat vier alternatieven beschikbaar zijn die respectievelijk 100, 125, 150 en 200 euro kosten. Dan kan het goedkoopste alternatief als referentiepunt dienen en kunnen

de andere ermee vergeleken worden. Deze benadering stelt dat het duurste alternatief van 200 euro twee keer zo duur is als het referentie-apparaat van 100 euro. Deze schaal zou echter een verkeerd beeld geven. Als er een apparaat moet aangeschaft worden, dan zal er sowieso 100 euro uitgegeven worden. Er kan dan enkel gekozen worden of er nog 0, 25, 50 of 100 euro extra besteed wordt aan het toestel. Hierbij wordt de goedkoopste optie als het ideale geval beschouwd, hieraan wordt een score van 100% gekoppeld. Aan het duurste toestel wordt de slechtste score van 0% gegeven. Intuïtief wordt aan het toestel van 125 euro een score van 75% gekoppeld. De score wordt dus volgens een lineair model bepaald, zoals geschetst in figuur 2.10 Elke euro die extra besteedt wordt, verlaagt de score met 1%.



Figuur 2.10: Score voor de aanschaf van een machine in functie van de prijs

De beoordeling van een PI is echter niet steeds lineair. Beschouw bijvoorbeeld de PI bezettingsgraad van de arbeiders. Deze PI wordt uitgedrukt als het percentage van de tijd dat de arbeiders effectief werken. Als de bezettingsgraad laag is dan zijn er te veel arbeiders, waardoor er meer kosten zijn dan nodig. Een bezettingsgraad van 100% wijst er dan weer op dat er niet voldoende arbeiders zijn voor de hoeveelheid werk, waardoor de productie wordt belemmerd. Het ideale geval ligt dus ergens tussenin, zoals bijvoorbeeld in figuur 2.11.



Figuur 2.11: Score voor de bezettingsgraad van de arbeiders

Door de gebruiker scores te laten aanduiden voor verschillende waarden van de PI kan een

evaluatieregel afgeleid worden die de score bepaalt volgens een stuksgewijze lineaire functie. De evaluatieregel in figuur 2.11 wordt bekomen door aan de PI-waarden 0, 80 en 100 respectievelijk de scores 0, 100 en 50 te koppelen.

Soms is het onaanvaardbaar dat een PI boven of onder een bepaalde drempel zit. Een mogelijke regel zou zijn dat de gemiddelde tijd van een harttransplantatie zeker niet langer mag duren dan een uur, omdat anders de kans op sterfte te groot wordt. Stel dat een lineaire score zou gebruikt worden, waarbij een operatie die te lang duurt, wordt bestraft met de score 0. Als de KPI nog afhangt van andere PI's dan zal de score van de KPI gewoon wat lager liggen, terwijl de patiënten een groot risico ondergaan en dit absoluut moet vermeden worden.

Er moet dus een manier zijn om voor PI's aan te duiden dat vanaf een bepaalde waarde, de volledige KPI onaanvaardbaar is. Wiskundig kan dit voorgesteld worden door er een score van min oneindig aan toe te kennen. In de gewogen som om de KPI te berekenen komt dan enkel de negatief oneindige score tot uiting, zodat duidelijk is dat de bijhorende optie moet verworpen worden. Onaanvaardbare waarden van een PI worden aangegeven door middel van een BoundedEvaluationRule, zie figuur 2.8.

2.3.4 Definiëren van KPI's met behulp van paden

De gebruiker moet op een eenvoudige manier KPI's kunnen definiëren. Hij wil meestal ook gelijkaardige KPI's berekenen op verschillende plaatsen. Stel bijvoorbeeld dat voor elke taak een KPI moet worden gedefinieerd. Het zou dan zeer omslachtig zijn als de gebruiker voor elke taak dezelfde KPI moet aanmaken en configureren. Daarom wordt een beperkte taal gedefinieerd waarmee de gebruiker meerdere KPI's tegelijk kan aanmaken.

Om naar de verschillende PI's te verwijzen in de boom worden paden gebruikt, gelijkaardig aan bestandspaden bij klassieke bestandssystemen. Een pad wordt gevormd uit de namen van opeenvolgende PI's. Zo duidt het pad */registratie/taken/taak1* de PI met naam *taak1* aan dat kind is van de PI *taken*.

Een KPI wordt gedefinieerd door eerst een pad op te geven naar een PI p waaraan de KPI moet worden toegevoegd. Daarnaast moet nog een naam opgegeven worden voor de KPI. Ten slotte moet nog opgegeven worden welke PI's deel uitmaken van de KPI en welke evaluatieregels en gewichten bij elke PI horen. De PI's die worden toegevoegd aan de KPI worden aangeduid met een relatief pad ten opzichte van de ouder van de PI p. Hierdoor kan een KPI geen PI's bevatten die hoger in de boom zitten dan de KPI zelf.

Vaak moeten gelijkaardige KPI's op verschillende takken van de boom gedefinieerd worden. Zo kan bijvoorbeeld voor de verschillende taken van een proces eenzelfde KPI berekend worden. Om met een definitie tegelijk meerdere KPI's aan te maken, moet het pad meerdere knopen aanduiden. Ook voor de kindknopen kunnen paden gebruikt worden die meerdere PI's tegelijk aanduiden, bijvoorbeeld om een KPI te definiëren die de KPI's van verschillende taken omvat. In dat geval wordt aan elk van die PI's hetzelfde gewicht en dezelfde evaluatieregel gekoppeld.

Om met een pad meerdere knopen aan te spreken, worden twee speciale operaties gedefinieerd. De *wildcard (*)* vervangt de naam van een PI zodat alle kinderen van eenzelfde PI worden toegevoegd aan het resultaat in plaats van een enkele PI. Met de of-operator (|) kunnen ook meerdere kinderen toegelaten worden, maar dan moet de naam van de PI een van de opgegeven namen zijn.

Met */taken/taak1|taak2* bijvoorbeeld worden de knopen taak1 en taak2 aangeduidt. Het pad */taken/** duidt alle taken aan.

In bestandssystemen kan ook naar ouderknopen verwezen worden, maar dit wordt niet toegelaten bij de KPI-boom. De aanpak met ouderverwijzingen werd uitgeprobeerd, maar de ouderverwijzingen maakten het mogelijk dat een KPI aan zichzelf kan worden toegevoegd. Daardoor was de berekening van de KPI afhankelijk van zijn eigen waarde en bleef ze hangen in een oneindige lus.

2.4 Dashboard op basis van een samenhangende KPI-boom over meerdere dagen

Op een dashboard is in een oogopslag te zien of een proces efficiënt is en waar de eventuele problemen zich bevinden. Voor een proces kunnen veel PI's beschikbaar zijn, deze worden gestructureerd in een PI-boom. Voor elke dag wordt een PI-boom beschikbaar gesteld en deze bomen kunnen eenvoudig samengebracht worden tot een enkele boom. Het is de bedoeling dat de gebruiker gemakkelijk de waarden van verschillende PI's kan terugvinden in deze boom zodat ze kunnen worden weergegeven in een dashboard.

2.4.1 Structuur van de samenhangende boom

Uit de resultaten van de simulatie van een proces kan een PI-boom gegenereerd worden, waarbij de specifieke structuur afhangt van het aantal proceselementen en het aantal brontypes. Maar

de algemene structuur is steeds gelijkaardig voor een PI-boom die bepaald wordt aan de hand van de simulator, zie hoofdstuk 2.2.4. Indien de PI-boom bepaald wordt uit de meetwaarden van een reële omgeving zal de boom een zelfgekozen, vaste structuur hebben voor de verschillende dagen. Doordat de structuur vastligt, kan de gebruiker eenzelfde PI opvragen via hetzelfde pad voor een boom van een willekeurig gekozen dag.

Om ervoor te zorgen dat navigeren tussen de PI's van verschillende dagen eenvoudig is, worden extra niveau's aangemaakt die de datum weerspiegelen. Aan de wortel worden kindknopen toegevoegd voor de verschillende jaartallen waarvoor data beschikbaar is. Onder deze jaarknopen worden opnieuw knopen aangemaakt voor de verschillende maanden waarvoor data beschikbaar is. Ook aan de maandknopen worden enkel kindknopen toegevoegd voor dagen waarvoor data beschikbaar is. Onder elke knoop die een dag voorstelt, komen dan bijvoorbeeld de verschillende deelbomen die elk een proces voorstellen indien het om gesimuleerde resultaten gaat. De eerste drie delen van het pad vormen dan de datum en het vierde deel duidt het proces aan. Zo zal het pad */2015/7/15/harttransplantatie/taken/pre-operatie/uitvoeringstijden/gem* duiden op de PI */taken/pre-operatie/uitvoeringstijden/gem* van het proces *harttransplantatie* op 15 juli 2015.

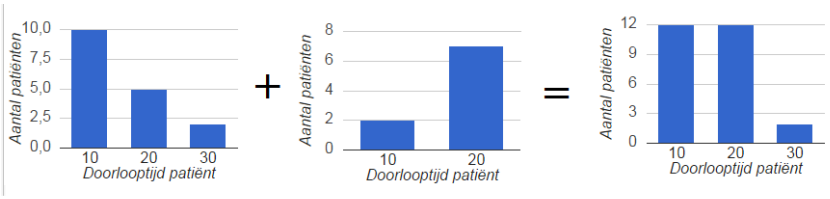
Op de server wordt de volledige KPI-boom bijgehouden, de verzameling evaluatieregels en alle KPI-definities. Met behulp van REST kan de toestand van de boom opgevraagd en gewijzigd worden.

2.4.2 Maandelijkse en jaarlijkse totalen

Aangezien de PI-bomen voor de verschillende dagen steeds dezelfde structuur hebben, is het eenvoudig om dezelfde PI op te zoeken voor verschillende dagen. Hierdoor is het ook eenvoudig om de totale waarde van een PI te berekenen over een maand of over een jaar. Om de totale waarde van een PI over een maand te bepalen worden de waarden van de individuele dagen opgeteld. Vervolgens wordt een tak aangemaakt onder de maandknoop met hetzelfde pad als het pad van de PI binnen de dagknoop. Stel bijvoorbeeld dat het totaal moet berekend worden voor de PI *taken/taak1/uitvoeringstijden* voor de maand juli 2015. Eerst worden alle knopen met het pad *2015/7/*/taken/taak1/uitvoeringstijden* gezocht en opgeteld bij elkaar, hierbij wordt * ingevuld met alle getallen van 1 tot en met 31. Daarna wordt een nieuwe knoop aangemaakt met het pad *2015/7/maand-totaal/taken/taak1/uitvoeringstijden*, eventueel moeten de intermediare knopen, *taken* en *taak*, aangemaakt worden.

Op een analoge manier kan de totale waarde bepaald worden voor een jaar door de waarden van de verschillende maanden op te tellen. Als het jaartotaal moet berekend worden voor de PI uit het vorig voorbeeld dan zullen alle PI's met een pad van de vorm *2015/*/maandtotaal/taken/taak1/uitvoeringstijden* opgeteld worden, * wordt hier vervangen door de waarden 1 tot en met 12. Als het totaal nog niet is berekend voor een bepaalde maand, dan zal eerst gepoogd worden om dit te berekenen en de knoop aan te maken zoals hierboven beschreven.

Er moet wel opgelet worden hoe deze totale waarden bepaald worden. Voor ValuePI's die eenvoudige getalwaarden bevatten, zoals bijvoorbeeld het aantal patiënten dat per dag een bepaalde operatie onderging, kunnen de getallen gewoon opgeteld worden. Maar bij statistische waarden zoals het gemiddelde kunnen de waarden niet zomaar opgeteld worden, aangezien dit een vertekend beeld zou geven. PI's die statistische waarden aanduiden, worden in het systeem steeds bepaald uit een distributie. Als de distributies eerst worden opgeteld, dan kan wel de statistische waarde correct bepaald worden door middel van de totale distributie. Het jaargemiddelde van de doorlooptijd per patiënt wordt dus niet berekend als de som van de maandgemiddeldes, maar wel als het gemiddeld van de jaardistributie, zie figuur 2.12.



Figuur 2.12: Optellen van twee distributies voor de doorlooptijd van de patiënten.

Statistische waarden zijn steeds een direct kind van een distributie, dus moet gewoon nagegaan worden of de ouder van de PI een distributie is. Als dat het geval is dan wordt eerst een totale distributie berekend. Daarna moet nog het juiste kind opgevraagd worden aan de distributie-PI, deze zal er dan voor zorgen dat de statistische knoop wordt aangemaakt. Als er later nog een statistische waarde van dezelfde distributie wordt opgevraagd, dan moeten de distributies niet meer opgeteld worden. De distributie-PI zorgt dan voor de automatische aanmaak van het gevraagde kind, zoals al in hoofdstuk 2.2.1 werd aangehaald.

2.4.3 Dezelfde PI opvragen voor alle taken of voor alle bronnen

Het kan belangrijk zijn om een bepaalde PI voor de verschillende taken of bronnen te vergelijken. Zo kan bepaald worden welke taak de grootste gemiddelde kost per instructie heeft of bij welke taak zich de langste wachttijden voordoen. Hierop kan het management zich dan baseren om te besluiten waar verbeteringen het eerst moeten worden doorgevoerd. Alhoewel de server een methode aanbiedt om de waarde van een PI op te vragen, is het niet handig om deze meermaals te moeten oproepen.

Daarom is er een extra methode beschikbaar die toelaat de waarde op te vragen van meerdere PI's waarvan het pad slechts in een pad-element verschilt, zoals bijvoorbeeld de taaknaam verschilt voor alle taken. Er worden twee paden als argument vereist, waarbij het nut best aan de hand van een voorbeeld wordt uitgelegd. Stel dat managers de gemiddelde wachttijd van de verschillende taken willen vergelijken. Het eerste pad */taken* duidt dan aan waar de verschillende taken zich bevinden. Het tweede argument is dan een relatief pad ten opzichte van een taak, in dit geval */wachttijden/gemiddeld*. In principe worden dus alle PI-knopen aangesproken met het pad */taken/*/wachttijden/gemiddeld* Het resultaat is een tabel die de koppeling aanduidt tussen de PI-waarden en de naam van de taak.

2.4.4 Huidige staat van een PI-boom opvragen

Aangezien de boom zeer veel PI's kan bevatten, kan een gebruiker moeilijk bepalen welke PI's hij kan en wil opvragen. Daarom is er een methode voorzien op de server die toelaat een tak van de boom op te vragen. Het enige argument is een pad dat aanduidt welke tak wordt opgevraagd, waarbij het laatste pad-element overeenkomt met de wortel van de gevraagde deelboom.

Sommige PI's bevatten een distributie of een tijdscurve. Deze distributies en tijdscurves kunnen veel plaats innemen, daarom worden ze niet meegestuurd. Om de gebruiker toch in staat te stellen de distributie of tijdscurve op te vragen, worden hiervoor aparte methoden voorzien. Een gebruiker moet dan echter wel eerst weten van welk type de knopen zijn, daarom wordt met elke knoop een type-aanduiding meegestuurd. Als de PI een distributie bevat dan weet de gebruiker bovendien dat hij er statistische waarden kan van opvragen.

Hoofdstuk 3

Schaalbaarheid van de simulator en van de PI-berekeningen

In dit hoofdstuk wordt nagegaan hoe schaalbaar de simulaties en de berekeningen van de PI's zijn. Hoe meer gebeurtenissen er zijn, hoe langer de simulatie zal duren. Een gebeurtenis treedt telkens op wanneer een instructie aan het begin of het einde van een proceselement passeert.

Als het aantal taken in een proces toeneemt, zal het aantal gebeurtenissen stijgen en zal de simulatietijd dus ook stijgen. Als er meer start events worden gesimuleerd, moeten er meer instructies verwerkt worden. Daardoor zullen er dus meer gebeurtenissen optreden waardoor de simulatie langer duurt. Voor elke taak worden er PI's berekend, zodat ook de berekening afhankelijk is van het aantal taken. Daarnaast zijn voor de berekening van elke PI de rapporten nodig van alle instructies.

Er worden twee testen uitgevoerd. De eerste test gaat het effect van het aantal taken na op de simulatie- en berekentijd voor een vast aantal start events. De tweede test gaat het effect van het aantal start events na op de simulatie- en berekentijd bij een vast aantal taken.

3.1 Opbouw testen

Beide testen zijn opgebouwd als een lus waarin telkens de configuratie van een proces in het geheugen wordt aangemaakt, waarna het proces wordt gesimuleerd en de PI-boom wordt berekend uit de resultaten van de simulatie.

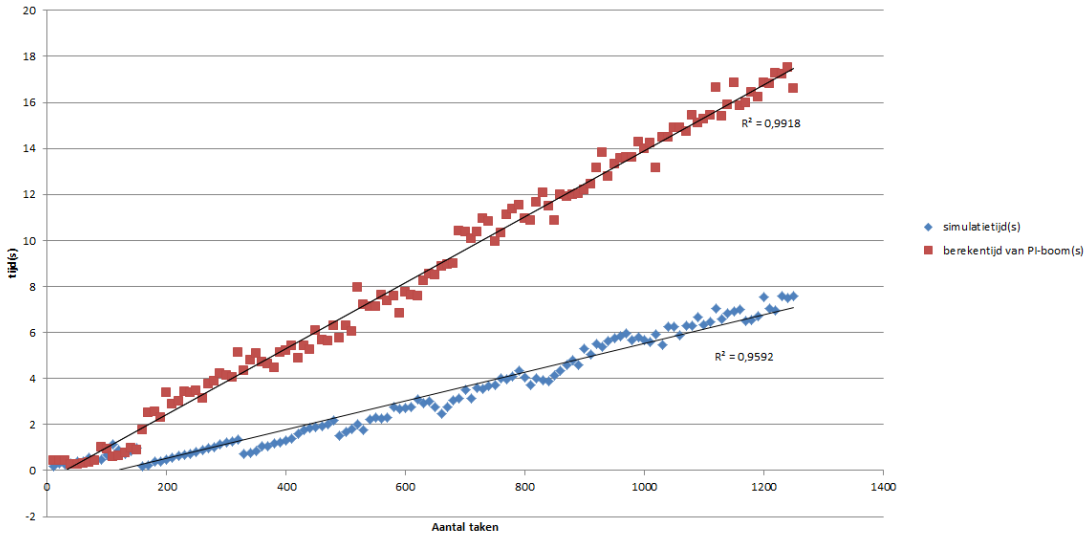
Het proces bestaat steeds uit een aantal taken in serie, waardoor het opbouwen van de configuratie vereenvoudigd wordt. De configuratie wordt in het geheugen opgebouwd en wordt niet

uit XML-bestanden gehaald omdat anders de benodigde tijd voor het inlezen van die bestanden meegeteld wordt voor de simulatietijd.

De eerste simulatie en PI-berekening duren langer dan de daaropvolgende simulaties en berekeningen. Dit komt doordat de Java Class Loader eerst alle klassen moet inladen voordat er simulaties en PI-berekeningen kunnen worden uitgevoerd. Een Java virtuele machine zorgt ervoor dat elke klasse slechts eenmaal ingelezen wordt [17]. Dit betekent dat enkel bij de eerste simulatie alle klassen moeten worden ingeladen. Daarom wordt voor het uitvoeren van beide testen een simulatie en een PI-berekening uitgevoerd zodat de klassen worden aangemaakt voor de testen.

3.2 Invloed van het aantal taken

Om het effect van het aantal taken na te gaan op de simulatietijd en de berekentijd van de PI's, wordt het aantal taken met stappen van tien verhoogd. Hierbij worden steeds honderd start events gesimuleerd. In de grafiek van figuur 3.1 is te zien dat zowel de simulatie- als de rekentijd linear afhankelijk zijn van het aantal taken.



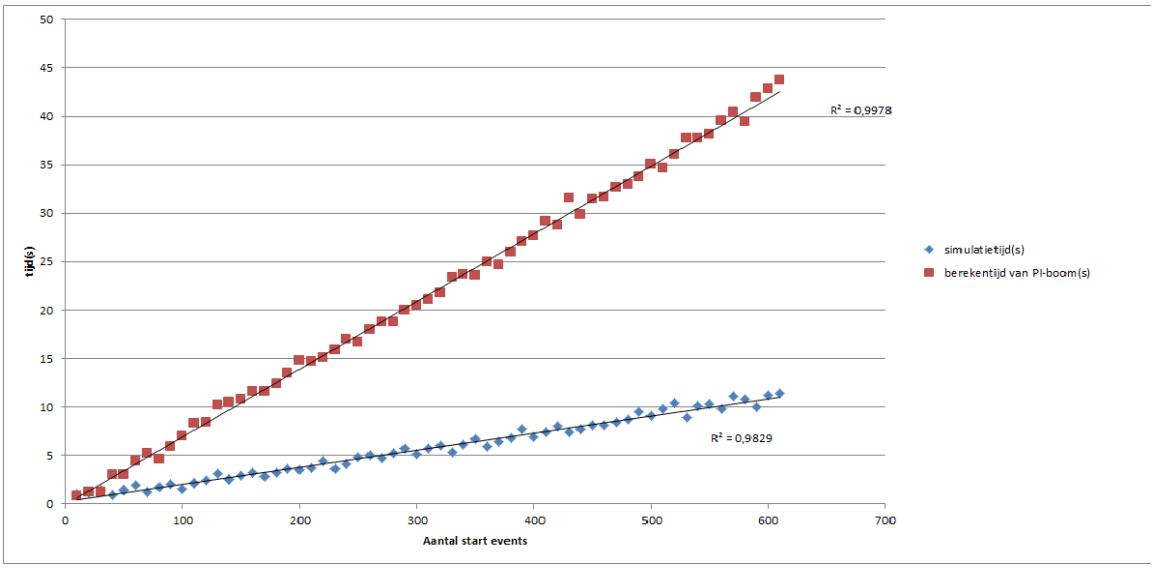
Figuur 3.1: Effect van het aantal taken op de simulatietijd en de berekentijd van de PI-boom

De simulatie van een proces met duizend taken in serie duurt 5,6 seconden. De berekening van de PI-boom voor hetzelfde aantal taken vereist echter 14,0 seconden. De berekening van de boom duurt dus relatief lang tegenover de simulatietijd. Gemiddeld is de berekentijd 3.0 keer langer dan de simulatietijd. Bij de berekening van de boom wordt er voor elke taak een

analyse uitgevoerd op alle rapporten. Deze analyses worden allemaal na elkaar uitgevoerd, maar het is mogelijk om deze op verschillende servers tegelijk uit te voeren. Als er bijvoorbeeld twee servers zijn, kunnen ze elk de PI's voor 500 taken berekenen. Hierdoor zou de berekening van de PI-boom sneller zijn. Dit is in deze masterproef niet nodig, aangezien het aantal taken steeds lager is dan 40 en de berekeningstijd van de PI-boom volgens de test slechts 0,27 seconden is.

3.3 Invloed van het aantal start events

De invloed van het aantal start events wordt gemeten door telkens hetzelfde proces te simuleren, maar het aantal start events met stappen van tien toe te laten nemen. In figuur 3.2 is de invloed op de simulatie- en berekentijd te zien voor een proces met 500 taken in serie. Opnieuw stijgt de simulatietijd lineair doordat er meer gebeurtenissen zijn.



Figuur 3.2: Effect van het aantal start events op de simulatietijd en de berekentijd van de PI-boom

Aangezien elk proces evenveel taken bevat, worden er telkens evenveel knopen aangemaakt in de PI-boom. Doordat de berekening van elke PI alle rapporten moet overlopen, duurt de berekening van een PI langer als er meer start events zijn. Het berekenen van de PI-boom duurt gemiddeld 3.6 keer zo lang als de simulatie.

3.4 Veralgemening voor realistische processen

De tijd om een proces te simuleren en de constructietijd van de bijhorende boom zijn recht evenredig met het aantal gebeurtenissen. Het aantal gebeurtenissen neemt toe als er meer instructies zijn of als het aantal elementen dat doorlopen wordt per instructie toeneemt.

In de twee testen werden enkel processen met taken in serie gesimuleerd. Een proces kan echter ook nog gateways en timers bevatten. Voor elk gesimuleerd proceselement, worden er twee gebeurtenissen gegenereerd. Zolang het proces geen lussen bevat, kan elk proceselement maximaal eenmaal uitgevoerd worden per instructie. Hierdoor kan de simulatietijd uit figuur 3.1 beschouwd worden als de bovengrens voor de simulatie van honderd instructies in een willekeurig proces met een bepaald aantal proceselementen.

De berekentijd van de PI's voor de de proceselementen verschilt sterk voor de verschillende types proceselementen. Toch kan er worden gesteld dat de berekentijd van de PI-boom recht evenredig is met het aantal proceselementen. De berekening van de verschillende PI's gebeurt steeds op dezelfde manier. Enkel het aantal PI's dat wordt berekend verschilt voor de verschillende types proceselementen.

3.5 Grootte van de PI-boom

Het aantal knopen van de PI-boom is afhankelijk van het aantal proceselementen en van het aantal brontypes. Er wordt namelijk per proceselement en per brontype een knoop aangemaakt.

De knopen voor alle brontypes bevatten steeds dezelfde deelstructuur. Zo is er steeds een kindknoop die bijhoudt hoeveel bronnen er van dat type op elk tijdstip worden ingezet. Het aantal knopen in de boom neemt dus lineair toe met het aantal brontypes.

Ook alle knopen voor eenzelfde type proceselement zijn steeds even groot. De knopen voor de verschillende taken bijvoorbeeld bevatten steeds kindknopen voor de uitvoeringstijden, de wachttijden, de doorlooptijden...

Alle knopen die een proceselement of een brontype voorstellen bevatten enkel kindknopen die een distributie of een tijdscurve bijhouden. De grootte van een distributie neemt enkel toe als een waarde wordt toegevoegd die er nog niet inzit, als de waarde er reeds inzat dan wordt gewoon de teller verhoogd voor die waarde. Dit doet zich bijvoorbeeld voor als twee instructies dezelfde doorlooptijd hebben. Voor elke instructie wordt slechts een waarde toegevoegd aan een distributie, zodat het aantal bytes van elke distributie sublineair toeneemt met het aantal start

events. De grootte van de PI-boom in bytes stijgt dus sublineair in functie van het aantal start events.

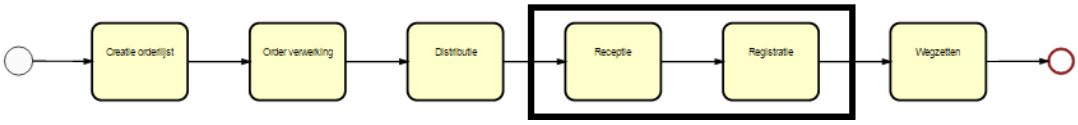
Hoofdstuk 4

Een voorbeeldcase: bevoorrading van een operatiekwartier

In het kader van het HIPS-project zal de bevoorrading van een operatiekamer met heupprothesen bestudeerd worden. Het doel van HIPS-project is om een set van tools te ontwikkelen waarmee de verschillende processen binnen een ziekenhuis geoptimaliseerd kunnen worden [19]. In een eerste fase worden de verschillende processen betrokken bij een heupoperatie besproken om de bruikbaarheid van de tools te testen. In deze case wordt enkel de bevoorrading van het operatiekwartier behandeld.

4.1 Procesbeschrijving van bevoorrading operatiekwartier

Voor een heupoperatie zijn verschillende componenten van een heupprothese nodig. De grootte van deze onderdelen verschilt van patiënt tot patiënt. In de operatiekamer worden er een beperkt aantal onderdelen van de verschillende groottes bijgehouden. Na het uitvoeren van een operatie kan het dus zijn dat er geen onderdelen meer zijn van een bepaalde grootte. Daarom wordt het operatiekwartier elke werkdag bevoorraad. Het bevoorradingsproces bestaat uit zes verschillende stappen zoals te zien is in figuur 4.1. Voor elke stap wordt een afzonderlijk proces aangemaakt, deze worden opgeroepen in het hoofdproces door middel van Call Activity-elementen.

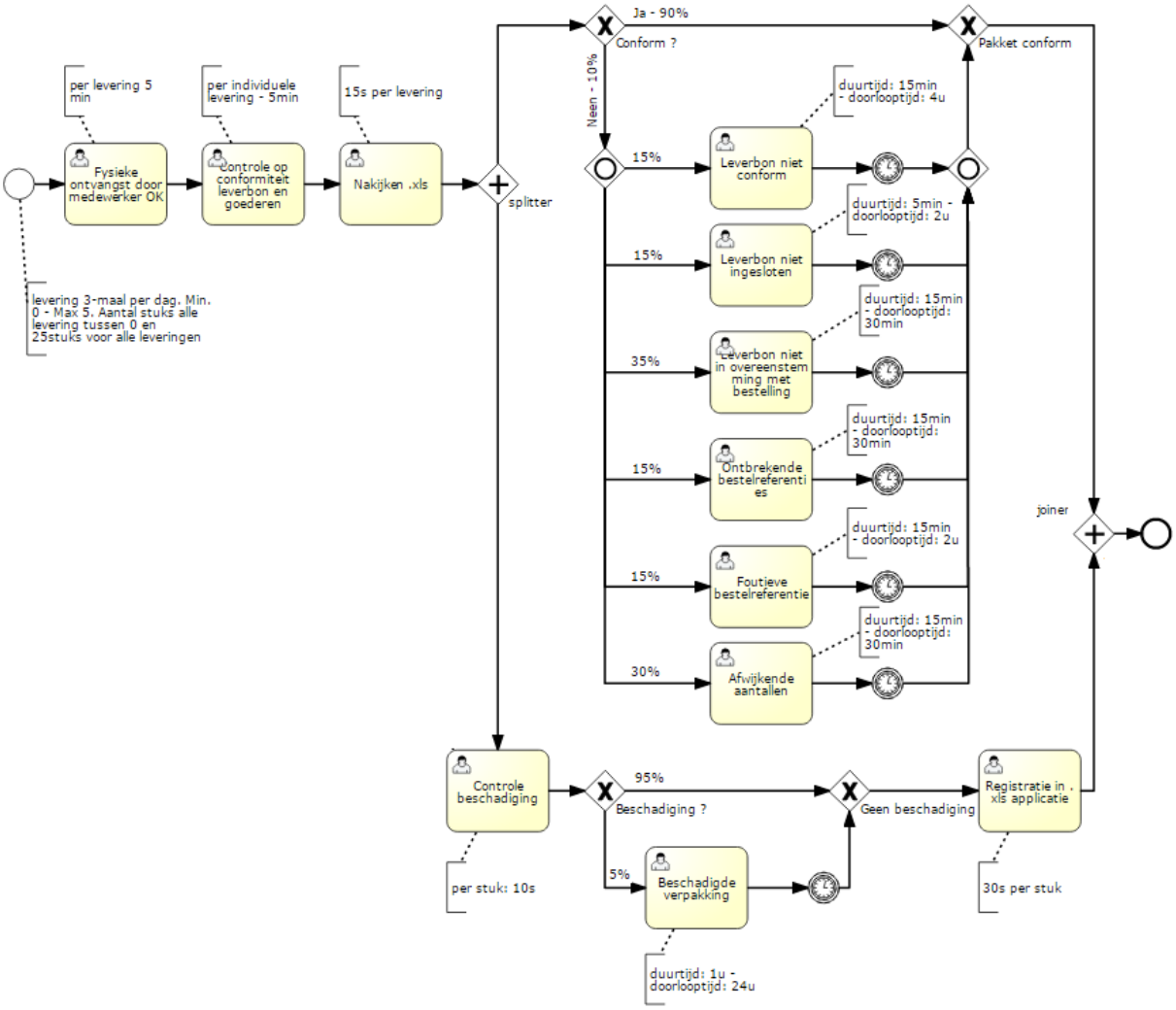


Figuur 4.1: Procesbeschrijving van de bevoorrading van het operatiekwartier

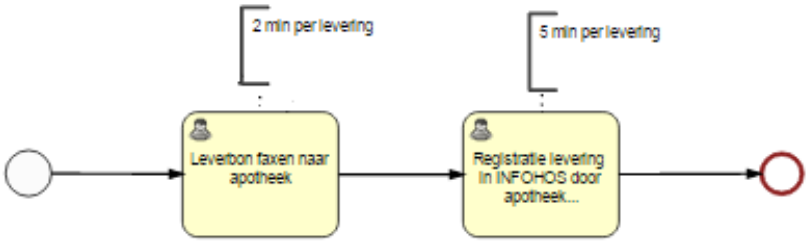
Elke dag wordt de voorraad gecontroleerd en een bestelling geplaatst voor de onderdelen waarvan er geen of weinig beschikbaar zijn. De bestelling moet dan nog bevestigd en gedistribueerd worden.

Daarna moet een medewerker van het operatiekwartier (OK) de bestelling ontvangen, dit gebeurt in het receptieproces. Bij de receptie wordt ook nagegaan of de bestelling volledig is en worden de onderdelen nagegaan op beschadiging. Het verloop van dit proces wordt getoond in figuur 4.2.

Daarna moet een apotheker de bestelling registreren, dit proces is uitgetekend in figuur 4.3. Ten slotte worden ze weggezet in het operatiekwartier.



Figuur 4.2: Procesbeschrijving van de ontvangst van leveringen



Figuur 4.3: Procesbeschrijving van de registratie van leveringen

Er worden slechts twee processen geanalyseerd: de receptie en de registratie van leveringen. Dit deel is aangeduid met een zwart kader in figuur 4.1. De start events stellen dus de aankomst van de leverancier aan het ziekenhuis voor.

De receptie en registratie worden geanalyseerd omdat ze in het ziekenhuis zelf plaatsvinden in tegenstelling tot de processen ervoor. Bovendien vormen de twee processen een samenhangend geheel. De analyse moet meer inzicht verschaffen in de doorlooptijden en de kost om elke levering te ontvangen en te registreren.

4.1.1 Verloop van het receptie- en registratieproces

Elke werkdag komen er 0 tot 5 leveringen aan, gemiddeld 3, tussen 8 en 12 uur. De leveringen bevatten elk minstens 1, gemiddeld 3 en maximaal 5 onderdelen. Iedere werkdag worden er dus 0 tot 25 onderdelen ontvangen. Er wordt verondersteld dat het aantal leveringen per dag en het aantal onderdelen per levering uniform verdeeld zijn.

Als de leverancier aan het ziekenhuis aankomt, zal een OK-medewerker de bestelling eerst ontvangen. Daarna controleert hij de conformiteit van de bestelling en kijkt hij de xls-applicatie na.

In de exclusieve gateway wordt nagegaan of er fouten zijn met de conformiteit van de bestelling. De kans dat er minstens een fout is met de bestelling is 10%. Indien er fouten zijn dan zal de OK-medewerker ze proberen op te lossen. De medewerker zal iemand anders contacteren om de fout op te lossen en kan zelf andere taken uitvoeren terwijl de fout wordt opgelost.

De werktijd van de medewerker en de totale tijd nodig om die fout op te lossen verschilt naargelang de aard van de fout. Het kan bovendien gebeuren dat er meerdere fouten tegelijk optreden, daarom wordt voor het bepalen van de fout een inclusieve gateway gebruikt. Elke taak na de inclusieve gateway stelt de afhandeling van een specifieke fout voor. De kans dat elke fout optreedt staat bij de pijlen naar de taken in figuur 4.2.

Daarnaast moeten de verschillende onderdelen gecontroleerd worden op beschadiging. Indien er een onderdeel defect is dan moet dit worden herbesteld. Een timer duidt aan dat er 24 uur moet worden gewacht op de volgende levering. In principe kan dit onderdeel opnieuw een defect hebben, zodat tweemaal moet herbesteld worden. De kans dat dit zich voordoet is 0.25% en is dus verwaarloosbaar en wordt daarom niet meegerekend in de case.

Als het onderdeel geen defecten vertoont dan registreert de OK-medewerker dit onderdeel in de xls-applicatie. Pas nadat alle onderdelen geregistreerd zijn in de xls-applicatie en de leverbon terug conform is, kan het registratieproces aanvangen.

In het registratieproces van figuur 4.3 zal de OK-medewerker de bestelbon naar de apotheker faxen. Ten slotte zal de apotheker de levering registreren. Nadat de levering is geregistreerd, wordt hij als afgehandeld beschouwd in deze case.

4.1.2 Beschikbaarheid en kosten van de bronnen

Er zijn twee personeelsleden bij de receptie en registratie betrokken: een OK-medewerker en een apotheker.

De OK-medewerker voert alle taken in het receptieproces uit. Daarnaast is de medewerker verantwoordelijk voor het faxen van de leverbon naar de apotheker in het registratieproces. De apotheker heeft als enige taak de levering te registreren in de INFOHOS-applicatie.

In de case wordt aangenomen dat de OK-medewerker 50 000 euro per jaar kost en de apotheker 75 000 per jaar. De kost per jaar is de kost van een personeelslid als hij een jaar lang elke dag 8 uur zou werken. De kosten voor de OK-medewerker en de apotheker zijn dus respectievelijk 17,12 en 25,68 per uur.

Er wordt tevens verondersteld dat het personeel continu beschikbaar is tussen 8 en 17 uur van maandag tot en met zaterdag. In de realiteit zal een medewerker nooit zo lang aan een stuk doorwerken. Er wordt echter verwacht dat er bij de simulatie voldoende momenten zullen zijn waarop er geen werk is voor het personeel, zodat deze momenten als pauze kunnen beschouwd worden.

Het personeel wordt vanaf 8 uur ingeschakeld omdat vanaf dan leveringen kunnen arriveren. Daarnaast zijn ze tot 17 uur beschikbaar, zodat er voldoende tijd is om bestellingen met fouten af te werken. Ook op zaterdag is het personeel nodig, om de onderdelen te kunnen verwerken die op vrijdag worden herbesteld omwille van beschadiging.

4.2 Case-specifieke aanpassingen

4.2.1 Aanpassingen aan de simulator om leveringen te kunnen simuleren

De originele simulator kan perfect gebruikt worden om patiënten of productie-eenheden te simuleren, maar er is steeds slechts één soort instructie. Hierdoor is het niet mogelijk om leveringen van onderdelen te simuleren waarbij zowel de levering in zijn geheel, als de aparte onderdelen opgevolgd kunnen worden. Bij het voorbeeldproces worden echter sommige taken eenmaal uitgevoerd per levering, zoals bijvoorbeeld de ontvangst. Terwijl andere taken inwerken op de individuele onderdelen, zoals de controle op beschadiging. De simulator moet dus worden uitgebreid voordat het bevoorradingsproces kan gesimuleerd worden.

Verschillende types instructies voor verpakkingen en onderdelen

De oorspronkelijke simulator laat slechts een soort instructies toe. Deze waren niet afhankelijk van elkaar en konden dus los van elkaar verwerkt worden in het proces. Er zijn nu echter twee soorten instructies nodig: verpakkingen en onderdelen. De verpakkingen en onderdelen kunnen niet zomaar los van elkaar behandeld worden, aangezien de onderdelen in een verpakking zitten. Om ervoor te zorgen dat alle onderdelen en de bijhorende verpakking apart kunnen behandeld worden, moeten ze eerst worden uitgepakt. Dit uitpakken wordt in figuur 4.2 op pagina 46 voorgesteld door de gateway *splitter*.

Voordat het pakket wordt uitgepakt, worden eerst drie taken uitgevoerd: de fysieke ontvangst, de controle op conformiteit van de bestelling en het nakijken in xls-applicatie. Bij de originele simulator kan elke instructie zich onafhankelijk van de andere instructies voortbewegen in het proces. Hierdoor zouden de verschillende prothese-onderdelen zich afzonderlijk kunnen voortbewegen, terwijl ze in realiteit nog samen in de verpakking zitten. Er moet dus gezorgd worden dat de verpakking en alle onderdelen van een levering steeds wachten op elkaar.

De onderdelen kunnen niet apart worden beschouwd voordat de levering wordt uitgepakt. De eerste drie taken worden daarom slechts eenmaal uitgevoerd per verpakking en niet voor de onderdelen afzonderlijk. Toch moeten de onderdelen door deze taak-elementen passeren. Daarom wordt voor de onderdelen de uitvoeringstijd ingesteld op 0 seconden en worden er geen bronnen vereist bij de uitvoering van deze taken. Hierdoor zullen de onderdelen veel vroeger toekomen bij de volgende taak dan de verpakking. Het is dus telkens bij het volgende proceselement dat moet aangeduid worden dat de onderdelen moeten wachten totdat de verpakking ook bij dat

proceselement is. Zo wachten alle onderdelen in de gateway *splitter* totdat de verpakking er ook is.

Na de *splitter* zijn de onderdelen uitgepakt en worden alle prothese-onderdelen in het procesdeel onderaan de figuur afgehandeld. In het bovenste deel worden fouten met de leverbon behandeld. De leverbon kan hier als de verpakking worden beschouwd, zodat de volledige stroom is opgesplitst: de verpakking gaat naar boven, de onderdelen naar beneden.

De opsplitsing wordt met een parallelle gateway gemodelleerd omdat in de realiteit de verwerking van de verpakking en van de onderdelen tegelijk kunnen worden uitgevoerd. De simulator zorgt er echter voor dat elke instructie alle paden doorloopt die vertrekken uit de parallelle gateway. Dit betekent dat een onderdeel dus ook het pad neemt dat bestemd is voor verpakkingen en omgekeerd. Daarom wordt voor elke taak tussen de openende en sluitende parallelle gateways aangeduid voor welk type instructie de taak is bestemd. Als de taak bestemd is voor een verpakking, dan worden de onderdelen direct doorgelaten. Hiertoe worden geen bronnen vereist en wordt de uitvoeringstijd ingesteld op 0 seconden. Hetzelfde mechanisme wordt gebruikt om verpakkingen door te laten bij taken bestemd voor onderdelen.

Een alternatieve oplossing zou erin bestaan om met een exclusieve gateway te werken in plaats van een inclusieve. De gateway bepaalt dan het pad van een instructie volgens het type van die instructie. Hierdoor zouden de instructies slechts een pad nemen, waardoor de simulatie versneld wordt. De verwerking van verpakkingen en onderdelen kan hierbij nog steeds parallel gebeuren, omdat ze verschillende instructies zijn die zich onafhankelijk van elkaar in een taak kunnen bevinden. Het is echter logischer om de twee parallelle takken effectief te modelleren met een parallelle gateway, zodat de voorkeur wordt gegeven aan de parallelle gateway.

Het receptieproces kan voor een levering pas beëindigd worden als alle onderdelen geregistreerd zijn in de xls-applicatie en als de levering conform is met de bestelbon. Hiertoe wordt bij het eindelement van het receptieproces ingesteld dat de verpakking moet wachten tot alle onderdelen gepasseerd zijn. De onderdelen van eenzelfde levering kunnen verschillende doorlooptijden hebben voor het receptieproces. Daarom wordt toegelaten dat de onderdelen het proces direct verlaten na de registratie in de xls-applicatie, zodat het rapport van een onderdeel de doorlooptijd van dat onderdeel bevat en niet de doorlooptijd van de levering.

Eens alle onderdelen en ook de verpakking zijn afgewerkt, kan het registratieproces aanvatten. In dit proces zijn er enkel taken die inwerken op de levering. Toch moeten de onderdelen doorgelaten worden tot op het einde van het globale proces, omdat de simulator het rapport

van een instructie enkel opslaat als de instructie het eindelement van het globale proces bereikt. Bovendien kan het zijn dat eventuele processen die erna komen wel taken hebben die inwerken op de aparte onderdelen. De onderdelen worden opnieuw direct doorgelaten door de uitvoeringstijd in te stellen op nul en geen bronnen te vereisen.

Configureren wanneer leveringen toekomen

Een levering komt toe als een verpakking met daarin de bijhorende onderdelen. De start events van de verpakking en de bijhorende onderdelen moeten dus op hetzelfde moment gegenereerd worden. Aangezien een instructie zowel een verpakking als een onderdeel kan voorstellen, kan de invoer niet meer louter beschreven worden met een interval tussen de instructies. De intervallen duiden nu de tijd tussen opeenvolgende verpakkingen aan.

Telkens een start event optreedt voor een verpakking, bepaalt de simulator het aantal onderdelen en maakt evenveel start events aan. Het aantal onderdelen wordt bepaald aan de hand van een distributie die in het attributenbestand wordt opgegeven. Voor de case wordt bijvoorbeeld ingesteld dat een uniforme distributie wordt gebruikt die minimaal 1 en maximaal 5 onderdelen aanduidt.

Er komen elke dag tussen 8 en 12 uur 0 tot 5 verpakkingen toe. Dit kan niet aan de hand van een distributie beschreven worden die het interval tussen twee verpakkingen aanduidt, zodat de configuratiemogelijkheden moeten worden uitgebreid. Er kunnen verschillende perioden gedefinieerd worden. Elke periode duidt dan aan hoeveel verpakkingen er in die periode moeten gegenereerd worden aan de hand van een distributie. Binnen een periode is het tijdstip waarop een verpakking toekomt steeds uniform verdeeld. Voor het bevoorradingsproces wordt ingesteld dat het aantal leveringen uniform is verdeeld met een minimaal 0 en maximaal 5 leveringen.

Inclusieve gateway selecteert steeds minstens een pad

De inclusieve gateway kan de instructies door een of meerdere paden doorsturen. De kans dat de instructie door het ene pad gestuurd wordt is hierbij onafhankelijk van de kans voor een ander pad. Hierdoor is het mogelijk dat de instructie door geen enkel pad wordt gestuurd. Dit zou echter betekenen dat de instructie verloren gaat. Daarom kiest de originele simulator in dat geval steeds het eerste pad.

In de voorbeeldcase is de kans dat geen enkel pad genomen wordt 23,75%. Dit wordt berekend als het product van de kansen dat elk pad niet genomen wordt, zie de eerste drie kolommen van

tabel 4.1. Hierdoor stuurt de gateway 39% van de niet conforme leveringen in plaats van 15% door naar de taak *Leverbon niet conform*. Dit grote verschil is ongewenst, daarom wordt de simulator aangepast.

Tabel 4.1: Kansberekening voor de verschillende paden van een inclusieve gateway

naam van de taak op het pad	kans pad gekozen (eerste poging)	kans pad niet gekozen (eerste poging)	kans pad gekozen (tweede poging)	extra kans voor pad door tweede poging
Leverbon niet conform	15%	85%	12%	2,85%
Leverbon niet ingesloten	15%	85%	12%	2,85%
Leverbon niet in oveenstemming	35%	65%	28%	6,65%
Ontbrekenede bestelreferenties	15%	85%	12%	2,85%
Foutieve bestelreferentie	15%	85%	12%	2,85%
Afwijkende aantallen	30%	70%	24%	5,70%
som van de kansen	**125%**		**100%**	
product van de kansen		**23,75%**		

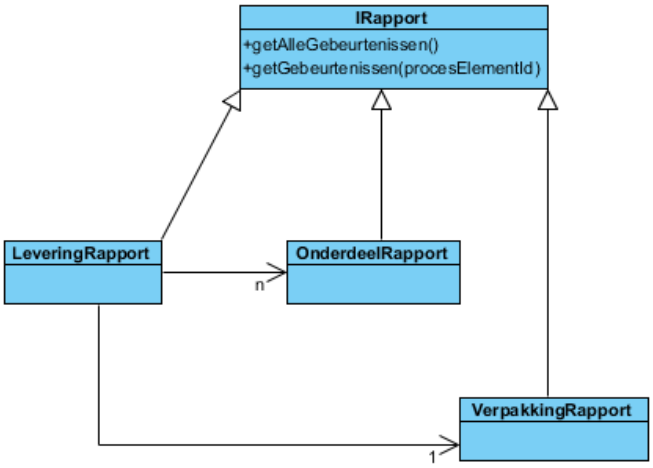
Indien de inclusieve gateway oorspronkelijk geen enkel pad kiest, zal de aangepaste simulator een evenwichtigere keuze maken. Eerst wordt zoals in de originele simulator voor elk pad afzonderlijk bepaald of het pad moet genomen worden. Indien er geen enkel pad wordt gevonden, dan wordt er willekeurig één pad gekozen. Hierbij worden de kansen voor elk pad herschaald, zodat de som van de kansen voor alle paden één wordt, zoals in de vierde kolom van de tabel te zien is. De kans voor *Leverbon niet conform* bijvoorbeeld is 12%. De kans dat er eerst geen pad gevonden wordt is 24%, zodat er 2,88% kans extra bestaat dat dit pad wordt gekozen. De totale kans op *Leverbon niet conform* is dus 17,88%. Er wordt dus een veel kleinere fout gemaakt dan bij de originele simulator.

4.2.2 Aanpassingen aan de berekening van de PI-boom

Aangezien er twee types instructies zijn, kunnen er ook twee aparte PI-bomen worden opgesteld. Zo wordt een boom opgesteld met PI's die informatie bevat over alle onderdelen waarbij alle onderdelen apart worden beschouwd. De andere boom bevat PI's met informatie over alle leveringen, waarbij elke levering inclusief de onderdelen wordt beschouwd. Aan de berekening van de PI's zelf moet hiervoor niets meer veranderen. De module die de PI-boom berekent moet wel tweemaal geactiveerd worden, eenmaal met alle rapporten voor de onderdelen en eenmaal met de rapporten van alle leveringen.

Uit een rapport voor een instructie kan afgeleid worden wanneer elke taak begint en eindigt en welk bronnen hierbij gebruikt werden, zie 2.1.4. Voor de analyse van de onderdelen kunnen de rapporten ongewijzigd gebruikt worden.

Om de leveringen als een geheel te beschouwen, moet er nog voor elke levering een rapport samengesteld worden uit de rapporten van de bijhorende onderdelen en het rapport van de verpakking, zoals in figuur 4.4. Bij het opvragen van de gebeurtenissen aan een leveringsrapport, wordt de navraag doorgestuurd naar de rapporten van de verpakking en van alle onderdelen. Het leveringsrapport bundelt de teruggekregen gebeurtenissen in éé lijst.



Figuur 4.4: Leveringsrapport als aggregatie van de verpakkings- en onderdeelrapporten

De PI-analyses vragen aan alle rapporten van een instructie telkens de gebeurtenissen op die relevant zijn voor die analyse. Voor de analyse van taken en gateways worden enkel de gebeurtenissen met betrekking tot die taak of gateway opgevraagd door het id van dat proceselement op te geven. Bij de analyse van de bronnen worden alle gebeurtenissen voor alle taken opgevraagd.

Voor taken die uitgevoerd worden voor instructies van het verkeerde type zijn de uitvoeringstijden nul en worden er geen bronnen toegekend. Hierdoor moeten de analyses uit 2.2.5 niet aangepast worden. Als een gebeurtenis niet relevant is voor de analyse van een taak of een bron, dan is de uitvoeringstijd steeds 0 en draagt dan gewoon niets bij aan het resultaat.

De uitvoeringstijd van een taak voor een pakket wordt bijvoorbeeld beschouwd als de som van de uitvoeringstijden van de verpakking en de onderdelen. Taken die eenmaal worden uitgevoerd per levering hebben een uitvoeringstijd gelijk aan nul voor onderdelen, zodat de uitvoeringstijd gelijk is aan de uitvoeringstijd voor de verpakking. Omgekeerd is de uitvoeringstijd voor de verpakking nul als de taak inwerkt op de onderdelen. De uitvoeringstijd van het pakket is dan gelijk aan de som voor alle onderdelen.

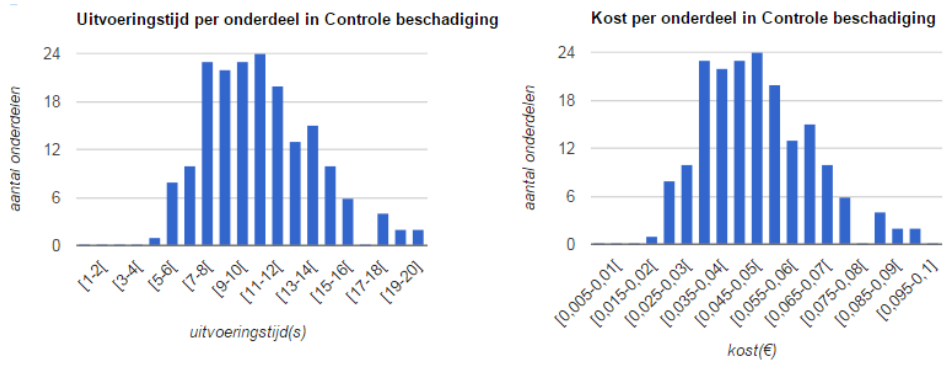
Bij het bepalen van de kost voor van een taak voor een pakket wordt ook de som van kosten

voor de onderdelen en de verpakking genomen. De kost wordt bepaald aan de hand van de ingezette bronnen. Doordat de taak geen bronnen vereist voor instructies van een verkeerd type, is ook de kost voor instructies van het verkeerde type nul.

De simulator werd gedeeltelijk aangepast om leveringen te simuleren waarbij een levering meerdere onderdelen bevat. Hierbij wordt een levering voorgesteld door meerdere instructies: een verpakking en een aantal onderdelen. De berekening van de PI-boom moest niet aangepast worden, doordat de rapporten van een verpakking en de bijhorende onderdelen geaggregeerd worden in een rapport.

4.3 Controle van de kosten en uitvoeringstijden

Alle uitvoeringstijden en doorlooptijden worden bepaald met een poissonverdeling. Dit komt ook tot uiting in de berekende PI's. Figuur 4.5 toont de distributies voor de uitvoeringstijden en kosten voor de controle op beschadiging van alle onderdelen. De gemiddelde uitvoeringstijd is 10.005 s en ligt dus dicht bij de geconfigureerde waarde. De kost van de controle op beschadiging is enkel afhankelijk van de OK-medewerker, zodat de distributie van de kosten dezelfde vorm heeft als die voor de uitvoeringstijden. De OK-medewerker kost 17,12 euro per uur of 0,476 eurocent per seconde, daardoor bedraagt de gemiddelde kost per onderdeel 4,76 eurocent.

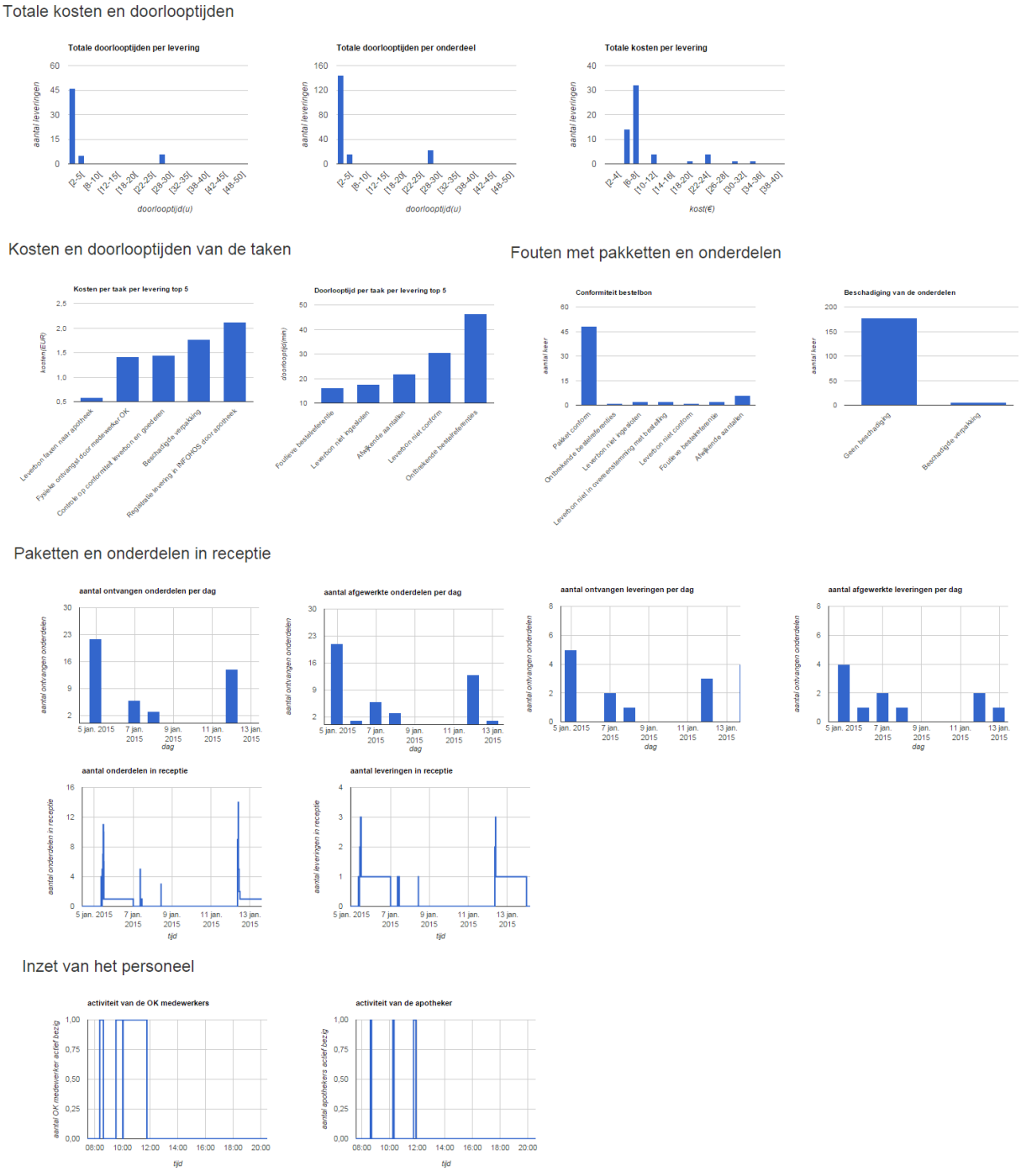


Figuur 4.5: Uitvoerinstijden en kosten van de controle op beschadiging

4.4 Een operationeel dashboard voor het bevoorradingsproces

Het operationeel dashboard toont zoveel mogelijk informatie over het proces op een overzichtelijke wijze weer. Dit dashboard laat een gedetailleerde analyse van het proces toe aan de hand van distributies en tijdslijnen. De tijdslijnen kunnen volledig worden in- en uitgezoomd zodat

afhankelijk van de situatie zowel op het niveau van dagen of uren een analyse mogelijk is. Het operationeel dashboard van het bevoorradingsproces is te zien in figuur 4.6. De case wordt geanalyseerd voor de maand januari van 2015, waarin 22 dagen zijn waarop leveringen kunnen binnenkomen. De simulatie wordt echter enkele dagen ervoor gestart zodat er op 1 januari al onderdelen herbesteld kunnen zijn.



Figuur 4.6: Het operationeel dashboard voor januari 2015 voor het bevoorradingsproces

Eerst is er een overzicht van de doorlooptijden per levering en per onderdeel. De leveringen komen niet allemaal ineens toe, waardoor de OK-medewerker voldoende tijd heeft om de levering af te werken voordat de volgende toekomt. Dit zorgt ervoor dat de doorlooptijd van de meeste onderdelen minder dan 2 uur is. Er zijn zes onderdelen met beschadiging, door de herbestelling hebben ze een doorlooptijd tussen 25 en 28 uur.

Daaronder volgt een overzicht van de vijf duurste en de vijf meest tijdrovende taken. Hierbij wordt de gemiddelde kost en doorlooptijd over alle leveringen van de maand gebruikt. De taken met de hoogste kost zijn de taken die men als eerste kan optimaliseren als de kost een belangrijke PI is. Ook taken met een hoge doorlooptijd kan men als eerste proberen te optimaliseren om de globale doorlooptijd te verhogen.

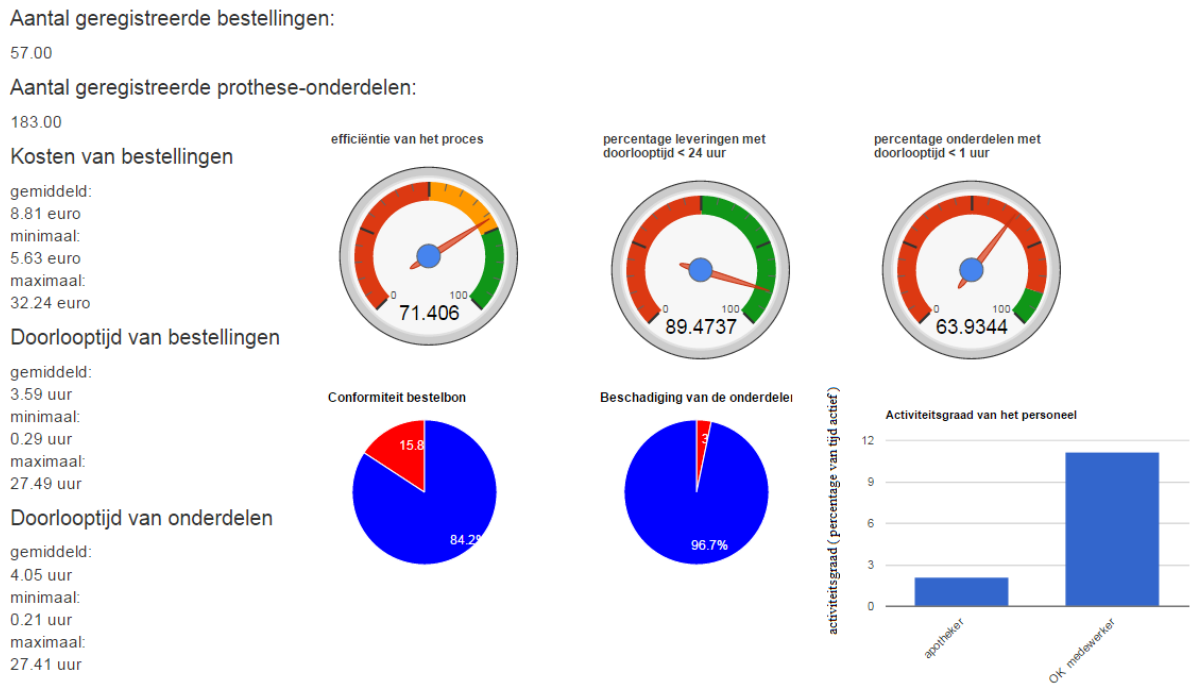
De doorlooptijd van leveringen en onderdelen is minimaal als er op elk moment voldoende personeel beschikbaar is zodat ze nooit moeten wachten. De doorlooptijd kan dus ook verlaagd worden door personeel op de juiste momenten in te schakelen. Door de inzet van het personeel en het aantal leveringen en onderdelen in de tijd te analyseren kan nagegaan worden wanneer het personeel het best wordt ingezet. Op het operationeel dashboard is te zien hoeveel medewerkers en apothekers actief bezig zijn op elk moment. Op figuur 4.6 is hiervoor ingezoomd op donderdag 1 januari. Voor deze case blijkt dat de OK-medewerker na 13 uur bijna nooit werk heeft. Dit komt doordat de leveringen ten laatste om 12 uur toekomen en een beperkt aantal onderdelen bevatten, zodat ze ook snel afgehandeld zijn.

Daarnaast kan opgevolgd worden hoeveel onderdelen en leveringen zich in het proces bevinden. Dit wil zeggen dat ze geleverd, maar nog niet afgewerkt zijn. Elke werkdag kunnen er pieken optreden tussen 8 en 12 uur, aangezien alle leveringen dan toekomen. Na die piek begint het aantal onderdelen terug te dalen, doordat de OK-medewerkers ze afwerken. Hoe meer personeel er beschikbaar is, hoe meer onderdelen er tegelijk kunnen verwerkt worden en hoe sneller het aantal onderdelen dus afneemt. Ideaal is er net na een piek veel personeel beschikbaar zodat de doorlooptijden klein zijn. Bij het bevoorradingsproces hebben de OK-medewerkers het meeste nut tussen 8 en 13 uur.

Ten slotte worden de fouten die optreden met leveringen en onderdelen getoond. Links staat hoeveel keer elke fout voor een bestelling optreedt. Rechts staat hoeveel onderdelen er beschadigd zijn. Er zijn negen leveringen met minstens een fout, toch zijn er in totaal veertien fouten opgetreden. Dit betekent dat er leveringen zijn met meerdere fouten.

4.5 Een strategisch dashboard voor het bevoorradingsproces

Een strategisch dashboard wordt door het management gebruikt en geeft op één scherm de belangrijkste performantie-indicatoren overzichtelijk weer [18]. Het toont aan of de organisatie al dan niet haar doelstelling behaalt. Figuur 4.7 toont het dashboard voor de case.



Figuur 4.7: Het strategisch dashboard voor januari 2015 voor het bevoorradingsproces

Links op het dashboard staat een beknopt overzicht met de belangrijkste statistieken. Hierop staat het aantal bestellingen en onderdelen dat die dag werd geregistreerd. Ook wordt informatie over de kosten en doorlooptijden getoond. Deze statistieken zijn een beknopte weergave van de distributies uit het operationeel dashboard.

Er is 86% kans dat er geen enkel onderdeel in een bestelling van 3 onderdelen defect is. Dit betekent dat ongeveer 86% van de bestellingen een doorlooptijd van minder dan 24 uur moet hebben. Leveringen met minstens een defect onderdeel hebben een doorlooptijd van meer dan 24 uur, aangezien de bestelling pas is afgewerkt als alle onderdelen zijn afgewerkt.

Ter illustratie wordt een SLA opgesteld die eist dat 50% van de bestellingen binnen de 24 uur moeten geregistreerd zijn. Op het dashboard wordt de SLA aangeduid met een rode en een groene strook op een metertje. De groene strook duidt hierbij aan dat de SLA is bereikt, zoals bij de doorlooptijd van de bestellingen. De rode strook duidt aan dat de SLA niet is bereikt, dit is het geval bij de doorlooptijd van de onderdelen, waarbij de SLA eist dat 90% van de

onderdelen in minder dan een uur worden verwerkt.

Slechts 64% van de onderdelen kan binnen het uur verwerkt worden. Er zijn zes leveringen met een fout, gemiddeld zijn er dus 18 onderdelen die moeten wachten op een leveringsfout, waardoor hun doorlooptijd minstens 1 uur is. Daarnaast zijn er zes onderdelen defect, zodat ze een doorlooptijd hebben van meer dan 24 uur. Een onderdeel opnieuw bestellen vraagt 1 uur werk van de OK-medewerker. Hierdoor wordt ook de doorlooptijd van onderdelen die erna verwerkt worden groter, dit kunnen onderdelen van dezelfde of een volgende levering zijn.

Het operationeel dashboard toont de doorlooptijden van de bestellingen en onderdelen meer in detail door de distributie weer te geven. Hierdoor kan het management een PI meer in detail bestuderen indien de SLA niet werd bereikt.

Daarnaast wordt de bezettingsgraad van het personeel getoond. De bezettingsgraad wordt berekend per personeelstype en niet per apart personeelslid. De bezettingsgraad is het quotiënt van het aantal uren dat de personeelsleden actief zijn gedeeld door het aantal uren dat ze beschikbaar zijn. De OK-medewerker is 9 uur beschikbaar per dag, maar wordt gemiddeld slechts een uur per dag ingezet, zodat zijn bezettingsgraad 11,1% bedraagt.

De apotheker werkt slechts 2,1% van de tijd, deze lage waarde is logisch gezien hij slechts een taak uitvoert in deze case. De bezettingsgraad van de OK-medewerker is hier belangrijker, aangezien hij het meeste werk uitvoeren.

De OK-medewerker heeft ook een lage bezettingsgraad, doordat er slechts sporadisch leveringen toekomen. Toch treden er problemen op met de doorlooptijden, doordat er plotse pieken zijn in het werk dat hij moet presteren. De grootste oorzaak van de lage bezettingsgraad is dat er geen leveringen toekomen na 12 uur, maar dat de medewerker tot 17 uur beschikbaar is.

Ten slotte wordt rechts in een taartdiagram weergegeven hoeveel percent van de bestellingen conform zijn. Dit is hier 84,2%, dit ligt dicht bij de 90% die ingesteld werd. Doordat een simulator gebruikt wordt zal de waarde steeds dicht bij 90% liggen als er voldoende bestellingen zijn. Deze PI bevat meer variatie indien ze voor een dag wordt opgevraagd in plaats van per maand.

Het taartdiagram toont niet aan welke fouten er precies opgetreden zijn. Deze informatie staat opnieuw in het operationeel dashboard, onder het dashboardelement met dezelfde naam.

Ter illustratie wordt er een gewogen KPI *efficiëntie* gedefinieerd. Deze wordt samengesteld uit de gemiddelde kost per levering en het percentage leveringen dat binnen een dag afgewerkt is. De score voor de kost daalt lineair als de kost stijgt en is 100 als de kost 0 euro bedraagt

en 0 als de kost 15 euro of meer bedraagt. De score voor de doorlooptijden neemt lineair toe als het percentage afgewerkte leveringen toeneemt. De relatieve belangrijkheid van de kost en doorlooptijd wordt aangegeven met gewichten, in dit voorbeeld zijn ze respectievelijk 0,2 en 0,8. De gewichten en de evaluatieregels kunnen volledig gemanipuleerd worden, zodat het management zelf gewogen KPI's kan definiëren.

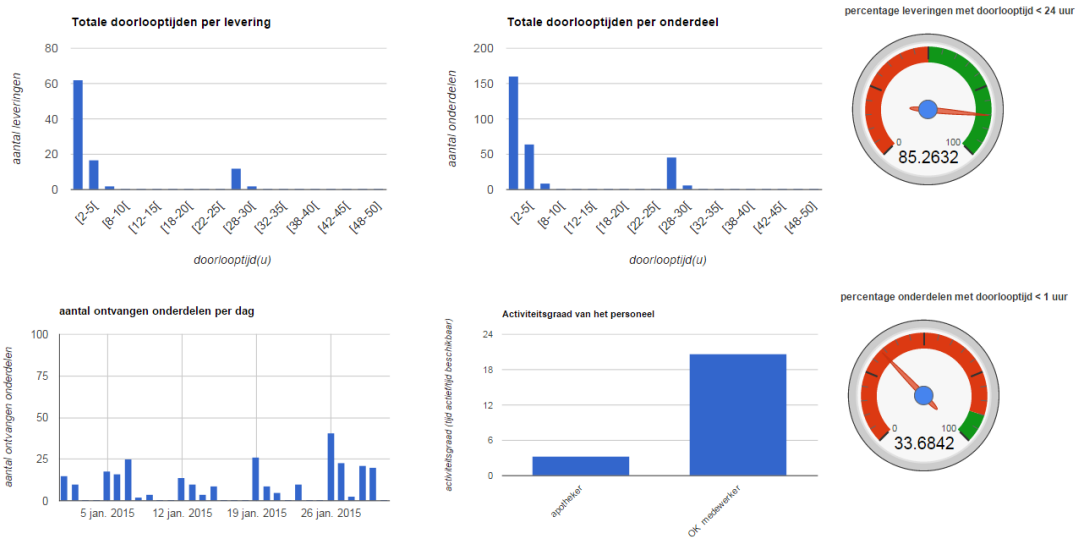
4.6 Sensitiviteitsanalyse

In de praktijk worden er niet alleen heupprotheses geleverd, maar bijvoorbeeld ook beenprotheses. Daarom wordt in dit hoofdstuk nagegaan wat er gebeurt als het aantal leveringen en het aantal onderdelen wordt verhoogd. Daarnaast wordt nagegaan wat er gebeurt als er twee OK-medewerkers worden ingezet. Tabel 4.2 toont de configuratie van de verschillende scenario's.

scenario	gemiddeld aantal leveringen per dag	gemiddeld aantal onderdelen per levering	aantal OK-medewerkers
basiscase	3	3	1
scenario 1	5	3	1
scenario 2	3	5	1
scenario 3	5	5	1
scenario 4	3	3	2
scenario 5	5	3	2

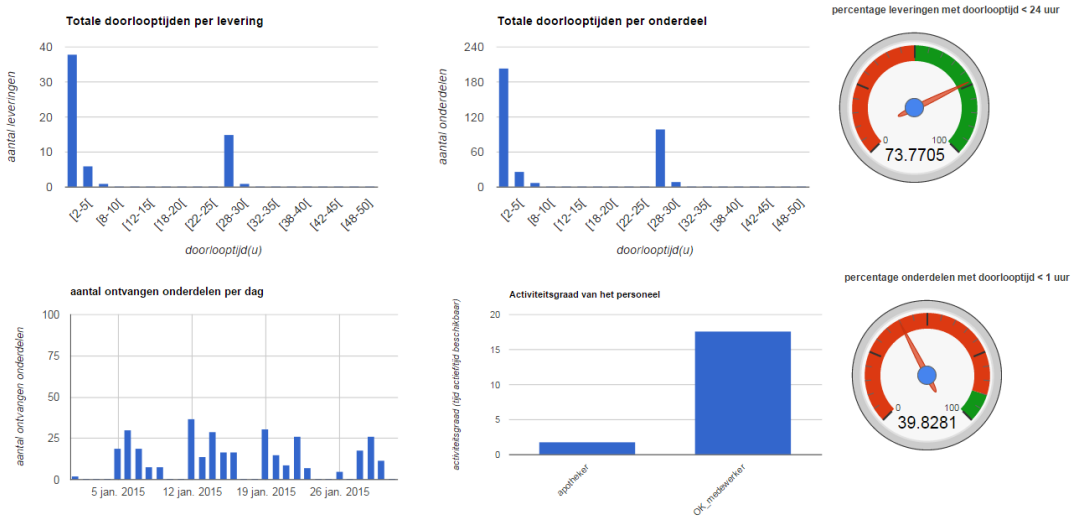
Tabel 4.2: Configuratie van de verschillende scenario's

In het eerste scenario wordt het gemiddeld aantal leveringen per dag verhoogd van 3 naar 5, zie figuur 4.8. Doordat er meer leveringen per dag zijn, moeten er meer leveringen en onderdelen verwerkt worden. Hierdoor is de bezettingsgraad van het personeel toegenomen. De leveringen komen ook sneller na elkaar toe, zodat leveringen en onderdelen vaker moeten wachten en de doorlooptijden toenemen. Het aantal onderdelen dat binnen een uur verwerkt kan worden daalt hierdoor drastisch. Het aantal leveringen dat binnen 24 uur wordt afgewerkt is 85%. De OK-medewerker slaagt er dus nog steeds in om alle leveringen, waarbij geen onderdelen moeten worden herbesteld, af te werken.



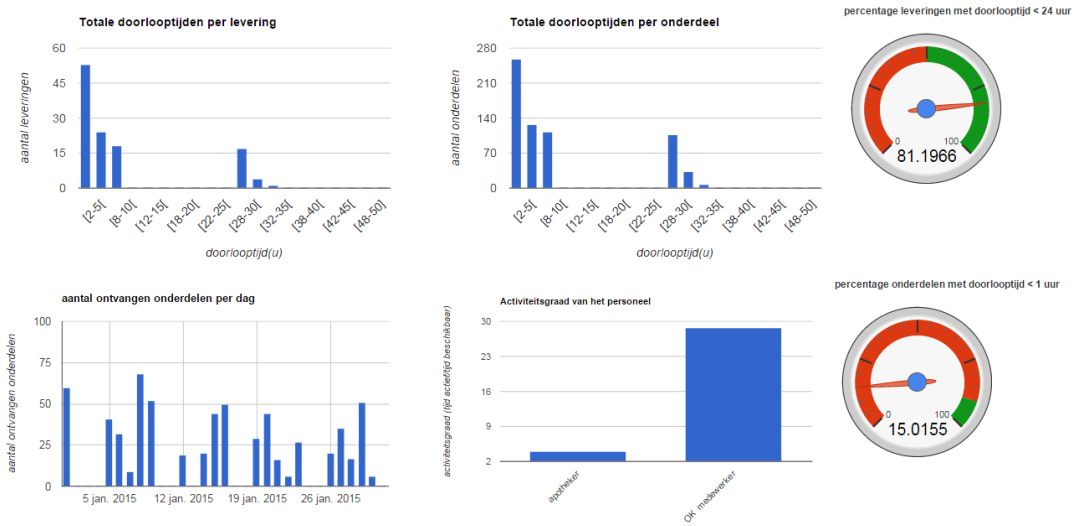
Figuur 4.8: Performantie van scenario 1

Bij het tweede scenario moeten er gemiddeld evenveel onderdelen(9) per dag verwerkt worden als bij het eerste scenario. Daardoor zijn de doorlooptijden van de onderdelen en de bezettingsgraad van het personeel ongeveer hetzelfde als bij het eerste scenario. De hoeveelheid werk is echter niet alleen afhankelijk van het aantal onderdelen, maar ook van het aantal leveringen. De OK-medewerker moet nu minder leveringen verwerken, waardoor zijn bezettingsgraad daalt. Hierdoor zullen de onderdelen ook minder vaak moeten wachten en neemt de doorlooptijd af. Tegenover de basiscase en het eerste scenario is het percentage leveringen dat binnen 24 uur verwerkt wordt afgenomen. Dit komt doordat de kans nu groter is dat een levering minstens één defect onderdeel bevat, zodat leveringen vaker moeten wachten. De kans dat een levering geen defecte onderdelen bevat is 86% als het drie onderdelen bevat en 77% voor vijf onderdelen.



Figuur 4.9: Performantie van scenario 2

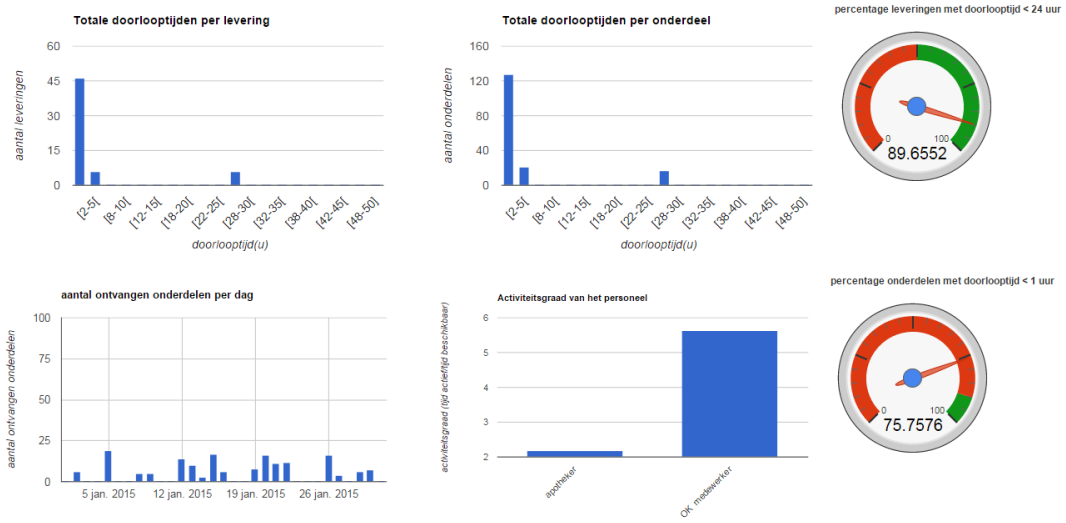
In het derde scenario is het gemiddeld aantal onderdelen dat per dag verwerkt moet worden 25. Hierdoor heeft de OK-medewerker nog meer werk dan in de vorige scenario's, zijn bezettingsgraad is 28%. Gedurende de voormiddag moeten vaak zeer veel onderdelen na elkaar verwerkt worden, waardoor de meeste onderdelen moeten wachten. Hierdoor kan slechts 15% van de onderdelen binnen een uur verwerkt worden.



Figuur 4.10: Performantie van scenario 3

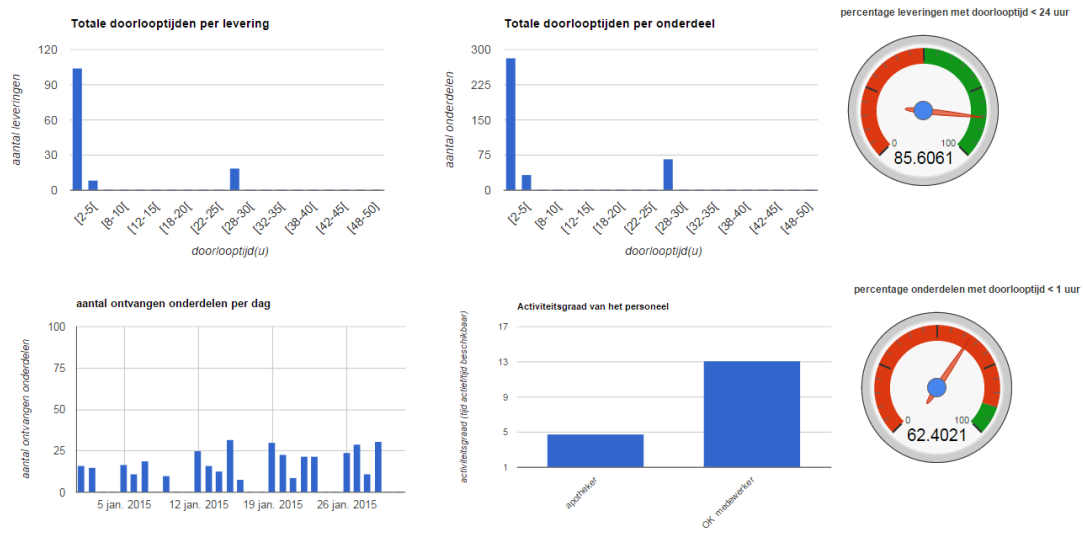
In het vierde scenario wordt een extra medewerker ingezet ten opzichte van de basiscase. Door de extra OK-medewerker kunnen de onderdelen continu verwerkt worden. Als de eerste OK-medewerker bijvoorbeeld een herbestelling uitvoert vanwege een defect onderdeel, dan kan de

tweede medewerker ondertussen andere onderdelen verwerken. Hierdoor liggen de doorlooptijden voor onderdelen lager dan bij de basiscase. Nu is 76% van de onderdelen binnen 1 uur verwerkt tegenover 64% bij de basiscase. Doordat er nu twee medewerkers zijn voor ongeveer dezelfde hoeveelheid werk, is de bezettingsgraad gedaald naar 5,6%.



Figuur 4.11: Performantie van scenario 4

Ten slotte wordt bij het vijfde scenario het aantal leveringen verhoogd ten op zichte van het vierde scenario. Aangezien er twee keer zoveel werk is, verdubbelt de activiteitsgraad van het personeel, voor de OK-medewerker bedraagt dit 13,2%.



Figuur 4.12: Performantie van scenario 5

Als het aantal leveringen en onderdelen dat per dag moet verwerkt worden stijgt, dan neemt

de activiteitsgraad van het personeel toe. Er ontstaan dan wachtrijen voor de onderdelen en leveringen, zodat ook de doorlooptijden toenemen. Als er meer OK-medewerkers beschikbaar zijn, dan zal de doorlooptijd terug dalen.

4.7 Bruikbaarheid van de analysetool voor de case

Hoewel met succes PI's voor het bevoorradingsproces konden worden afgeleid, moet er rekening mee gehouden worden dat het geanalyseerde proces een vereenvoudigd model is. In de realiteit maakt de bevoorrading deel uit van een groter geheel, zodat andere processen de bevoorrading kunnen verstoren. Bovendien onderstelt de simulator dat instructies steeds direct verwerkt worden. In de realiteit wordt soms echter gewacht tot er voldoende werk is, zodat de taak efficiënter kan worden uitgevoerd.

4.7.1 Invloed van de operatie

Indien bij een heupoperatie een prothese-onderdeel vereist is dat niet in voorraad is, dan moet een spoedbestelling worden gedaan. Deze spoedbestelling brengt een extra kost met zich mee.

In deze case wordt verondersteld dat als er een onderdeel defect is, enkel de doorlooptijd toeneemt doordat het onderdeel herbesteld wordt. Als dit onderdeel vereist is bij een operatie, dan zal in realiteit een spoedbestelling nodig zijn. Met de extra kost van deze spoedbestelling wordt in deze case geen rekening gehouden.

Om deze kosten in rekening te brengen, moet het systeem verder worden uitgebreid. Het volledige bevoorradingsproces en het operatieproces moeten tegelijk gesimuleerd worden, zodanig dat er interactie is tussen beide processen. De instructies stellen dan bij de operatie patiënten voor en bij de bevoorrading stellen ze leveringen en onderdelen voor. Aan de instructies voor de leveringen moet dan een eigenschap toegevoegd worden, waaruit kan bepaald worden of het om een spoedbestelling gaat. De gateways moeten de volgende actie bepalen aan de hand van deze eigenschap en niet met behulp van probabiliteiten.

Na afloop van de operatie wordt een bestelling geplaatst voor de verbruikte onderdelen door een start event aan te maken in het bevoorradingsproces.

Als een onderdeel het einde van het bevoorradingsproces bereikt, wordt deze toegevoegd aan de voorraad. Bij het begin van de operatie worden de benodigde onderdelen uit de voorraad gehaald. Eventueel moet de patiënt wachten totdat de juiste onderdelen beschikbaar zijn in de voorraad. Op het moment dat een onderdeel uit de voorraad wordt gehaald, wordt het rapport

voor dat onderdeel toegevoegd aan het rapport voor de patiënt. Hierdoor kan de bestelkost van het onderdeel toegekend worden aan een patiënt.

Er is dus een grote uitbreiding aan de simulator vereist opdat de operatie en de bevoorrading kunnen interageren. Daarnaast zal aanvullende informatie nodig zijn over de voorraad, aangezien dit niet voorzien is in BPMN. Er moet worden aangeduid wat de voorraad kan bevatten, bijvoorbeeld prothese-onderdelen van verschillende groottes. Bovendien moet worden opgegeven wanneer er materialen toegevoegd worden aan de voorraad en terug verdwijnen. Daarnaast moet opgegeven worden wat er gebeurt als iets niet in voorraad is.

4.7.2 Buffers van instructies bij taken

De simulator onderstelt dat de instructies direct worden doorgegeven aan de volgende taak, wat niet altijd zo is in de realiteit. Nadat de OK-medewerker een levering heeft geregistreerd, plaatst hij die op een kar. Pas wanneer er voldoende leveringen op de kar staan, zal hij de kar naar de operatiekamer brengen. De doorlooptijd van de meeste leveringen zal hierdoor groter zijn dan bepaald met de analysetool.

De simulator kan worden uitgebreid zodat er na de taken buffers mogelijk zijn. BPMN voorziet geen standaardmanier om dit op te geven, waardoor opnieuw aanvullende configuratie zal vereist zijn.

Er zijn dus nog veel aanpassingen nodig aan de simulator voordat een echt realistische analyse mogelijk is van de case. Toch kan de analysetool reeds gebruikt worden om verschillende alternatieven voor eenzelfde proces te vergelijken.

Hoofdstuk 5

Conclusie

5.1 Bereikte resultaten

Het doel van deze thesis bestond eruit om met behulp van een bestaande simulator op een generieke en automatische manier PI's en KPI's af te leiden voor een proces. Deze PI's en KPI's moesten weer te geven zijn op een dashboard, zodat op basis hiervan het proces kan beoordeeld en geanalyseerd worden.

Het procesverloop werd gemodelleerd met BPMN. Dit model werd verder aangevuld met informatie over de uitvoeringstijden van de taken, de gemaakte keuzes, beschikbaarheid van het personeel... Op basis van deze informatie kon het proces gesimuleerd worden.

De rapporten van de simulator werden daarna gebruikt om verschillende PI's te berekenen. Deze PI's werden gestructureerd in een boom met een generieke structuur. Er werden PI's berekend voor de doorlooptijden, kosten, personeelsbezetting... De berekende PI's werden opgeslagen als een distributie of als een tijdscurve. Uit de distributies konden dan statistische waarden berekend worden zoals bijvoorbeeld de gemiddelde doorlooptijd.

Daarna werden slechts enkele KPI's gekozen uit de verzameling door het pad in de boom op te geven. Er konden ook gewogen KPI's gedefinieerd worden die de waarden van meerdere PI's samenvatten, zodat het aantal KPI's verder gereduceerd kon worden.

De simulatie van honderd productie-eenheden in een proces met duizend taken in serie duurde 5,6 seconden. De berekening van de bijhorende PI-boom duurde echter 14,0 seconden zodat het aangeraden is om de werklast te verspreiden over meerdere servers indien grotere processen moeten geanalyseerd worden.

Als case werd de bevoorrading van een ziekenhuis met prothese-onderdelen onderzocht. Er

werd een operationeel en een strategisch dashboard gebouwd, op basis waarvan het proces werd geanalyseerd. De bezettingsgraad van de OK-medewerker bedroeg slechts 11,1%. Toch moesten er veel onderdelen wachten, doordat het werk niet gelijk verdeeld was over de dag. Als gevolg van deze wachttijden en de fouten die optreden met leveringen en onderdelen wordt slechts 63,3% van de onderdelen in minder dan een uur afgewerkt.

Door twee OK-medewerkers in te zetten konden de doorlooptijden verbeterd worden, zodat 76% van de onderdelen binnen een uur werden afgewerkt.

5.2 Mogelijke uitbreidingen

Momenteel gebeurt de configuratie van het proces grotendeels manueel. Het procesverloop kan al uitgetekend worden in een grafische omgeving gebaseerd op de Activiti-engine. De bronnen moeten echter manueel gedefinieerd en gekoppeld worden aan de verschillende taken. Ook de eigenschappen van de verschillende proceselementen zoals de uitvoeringstijden van taken en de kansen van de verschillende paden voor gateways... moeten manueel worden ingevoerd.

Om de eigenschappen aan de verschillende proceselementen te koppelen, moet steeds het id van het proceselement opgegeven worden bij de juiste eigenschap. Het opzoeken van de id's van alle proceselementen is zeer omslachtig. Bovendien is de manuele configuratie gevoelig voor typefouten. Om de analysetool in de praktijk bruikbaar te maken is het dus aangewezen om de grafische omgeving waarmee de processen worden getekend uit te breiden, zodat hierin ook de proceselementen aan de bronnen en eigenschappen kunnen gekoppeld worden.

Ook het strategische dashboard kan nog gebruiksvriendelijker worden gemaakt door details van elk dashboardelement opvraagbaar te maken. De elementen van het operationeel deel zijn een gedetailleerdere versie van de elementen uit het strategisch deel. In het operationeel deel wordt bijvoorbeeld het aantal ingezette bronnen per type in aparte tijdscurves weergegeven. Op het strategische deel wordt enkel de bezettingsgraad van elk brontype weergegeven als een element. Alhoewel managers een beknopt dashboard verkiezen, willen ze toch soms het proces meer in detail analyseren. Daarom zou het handig zijn als er voor elk element op het strategisch een link bestaat naar een gedetailleerdere versie op het operationele dashboard.

Uit de PI-boom kan statistische informatie over de doorlooptijden worden afgeleid. De PI-boom laat echter niet toe het pad van een specifieke instructie op te vragen. Het pad bestaat uit de verschillende proceselementen die de instructie doorliep. Bij dit pad kunnen de doorlooptijd en de gebruikte bronnen voor elke taak aangegeven worden. Dit zou een nog gedetailleerdere

analyse toelaten. Deze functionaliteit kan toegevoegd worden door het rapport voor een instructie te analyseren.

Bij de analyse van het receptieproces wordt geen rekening gehouden dat er een spoedbestelling kan optreden als er niet voldoende onderdelen beschikbaar zijn voor een operatie. Deze spoedbestellingen brengen een extra kost met zich mee, waardoor het wenselijk is ze wel mee te rekenen.

De spoedbestellingen treden op afhankelijk van welke onderdelen de operatie vereist, er is dus een interactie tussen de operatie en de bevoorrading. De simulator moet dus worden uitgebreid zodat verschillende processen kunnen interageren met elkaar.

Lijst van figuren

Lijst van tabellen

Literatuurlijst

[1] Hoare, R., Ahn, J., and Graves, J. (2002). Discrete event simulator. US Patent App. 10/043,847.

[2] Benneyan, J.C. (1997). An introduction to using computer simulation in healthcare: patient wait case study

[3] Luke S. (2015). Multiagent Simulation And the MASON Library. Geraadpleegd op 13 augustus via https://cs.gmu.edu/ eclab/projects/mason/manual.pdf

[4] Railsback, S.F., Lytinen, S.L., Jackson, S.K. (2006). Agent-based Simulation Platforms: Review and Development Recommendations. Geraadpleegd op 16 augustus 2015 via http://sim.sagepub.com/content/82/9/609.abstract

[5] Owen, M., Raj, J. (2003). BPMN and Business Process Management. Geraadpleegd op 19 augustus 2015 via http://www.omg.org/bpmn/Documents/6AD5D16960.BPMN_and_BPM.pdf

[6] Earls, A. (2012). BPMN 2.0: A key modeling standard continues to evolve.

[7] Parmenter, D. (2015). Key Performance Indicators: Developing, Implementing, and Using Winning KPIs. Geraadpleegd op 19 augustus 2015 via https://books.google.be/books?id=bKkxBwAAQBAJ&oi=fnd

[8] Service level agreements and key performance indicators. Geraadpleegd op 8 augustus 2015 via https://www.springtideprocurement.com/?wpfb_dl=479

[9] Guidance on Developing Key Performance Indicators and Minimum Data Sets to Monitor Healthcare Quality. (2013). Dublin: Health Information and Qualtiy Authority.

[10] Ziekenhuisbreed domein. Geraadpleegd op 14 augustus 2015 via http://www.zorg-en-gezondheid.be/Beleid/Kwaliteit/Ziekenhuisbreed-domein

[11] Little, J.D.C., Graves., S. C. (2008). Little's Law. *Building Intuition: Insights From Basic Operations Management Models and Principles, pp 81-100* via http://web.mit.edu/sgraves/www/papers/Little's%20Law-Published.pdf

[12] Özpeynirci, S.B., Köksalan, M. (2008). An interactive sorting method for additive utility functions.

[13] Van Der Plas, C.(2012). Evolutionary Multi-Objective Optimization and Preference Modeling in Green Logistics.

[14] Multi-criteria analysis: a manual. (2009). Geraadpleegd via http://eprints.lse.ac.uk/12f761/1/Multi-criteria_Analysis.pdf

[15] Siskos, J., Jacquet-Lagreze, E. (1981). Assessing a set of additive utility functions for multicriteria decision-making, the UTA-method.

[16] Pauwels, K., Ambler, T., Clarck, B.H., LaPointe, P., Reibstein, D., Skiera, B., Wierenga, B., Wiesel, T. (2009). Dashboards as a Service: Why, What, How, and What Research Is Needed?

[17] Binildas, C. (2005). Internals of Java Class Loading. Geraadpleegd op 14 augustus 20015 via http://www.onjava.com/pub/a/onjava/2005/01/26/classloading.html

[18] Kruger, M. (2007). Management dashboards: stuurinstrument of vrijblijvend speeltje? geraadpleegd op 5 augustus 2015 via http://www.invoorzorg.nl/docs/ivz/interviews/Management_dashboards_Stuurinstrument_of_vrijblijvend_speeltje.pdf

[19] HIPS. (2014). Geraadpleegd op 24 augustus 2015 via http://www.iminds.be/hips