2014•2015
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
master in de industriële wetenschappen: elektronica-ICT

Masterproef
ICT-platform voor geïntegreerde gezondheidszorg

Promotor :
dr. Kris AERTS

Copromotor :
ing. Leo RUTTEN

Promotor :
dr. ing. PAUL VALCKENAERS

Daan Mouha

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

2014•2015

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterproef

ICT-platform voor geïntegreerde gezondheidszorg

Promotor :
dr. Kris AERTS

Copromotor :
ing. Leo RUTTEN

Promotor :
dr. ing. PAUL VALCKENAERS

Daan Mouha

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

universiteit hasselt | KU LEUVEN

Woord vooraf

De keuze omtrent het onderwerp *eHealth*, binnen deze thesis, is ontstaan uit mijn interesse in de zorgsector. Enerzijds wil ik mensen helpen tijdens hulpbehoevende momenten en dit door een ondersteunend platform te ontwikkelen. Anderzijds wens ik mij ook te richten op de mensen rondom een patiënt, dit om hun werk te verlichten en efficiënter te maken.

De zorgsector, beroepsmatig maar zeker ook emotioneel, vooruit helpen was de grote drijfveer tijdens de keuze aangaande deze thesis. Een bijkomende, geheel andere reden, was het gegeven dat er binnen dit 'schrijven' gewerkt moest worden met een massief schaalbare programmeertaal. Voorgaand aspect in samenhang met een sector (*healthcare*) waar ik amper kennis van had, vormde een grote uitdaging die ik met veel genoegen wou aangaan.

De scriptie is bedoeld voor informatici, onderzoeksgroepen en andere actoren die geïnteresseerd zijn in het *NEU*-protocol en hoe dit in de zorgsector valt in te passen.

Graag zou ik mijn externe promotor dr. ir. Paul Valckenaers willen bedanken voor de ondersteuning omtrent de programmeertaal Erlang en het *NEU*-protocol. Alsook voor de snelle *feedback* en het snel beantwoorden van mijn vragen. Daarnaast wil ik ook ing. Leo Rutten bedanken voor het geven van de nodige uitleg over de Cowboy webserver en het beschikbaar stellen en opzetten van een server.

Daarnaast zou ik graag mijn interne promotor dr. Kris Aerts willen bedanken om mijn thesis na te lezen en te verbeteren. Tot slot wens ik mijn vriendin, familie, schoonfamilie en alle mensen die betrokken waren willen te bedanken voor de steun tijdens de schakel-, masterjaren en thesis.

Inhoudsopgave

Lijst van figuren

Verklarende woordenlijst

BEAM	Bogdan's Erlang Abstract Machine
DETS	Disk Erlang Term Storage
Erlang	Ericsson LANGuage
ERTS	Erlang Run-Time System
EShell	Erlang Shell
ETS	Erlang Term Storage
GNU	GNU is Not Unix
GPS	Global Positioning System
GZIP	GNU ZIP
HRL/hrl	HeadeR fiLe
ICT	Information and Communication Technology / Informatie- en CommunnicatieTechnology
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology / InformatieTechnologie
KIC	Kennis & Innovatie Centra
KHLim	Katholieke Hogeschool Limburg
KHL	Katholieke Hogeschool Leuven
MinGW	Minimalist GNU for Windows
MPI	Message Passing Interface
MSYS	Minimal SYStem
NEU	Next, Execute & Update
NOK/nok	Not ok / niet ok
OOP	Object-Oriented Programming / object-georiënteerd programmeren
OS	Operating System
OTP	Open Telecom Platform
PALANTE	PAtients Leading and mANaging their healThcare through Ehealth
PDA	Personal Digital Assistent
PDF	Portable Document Format
PID	Process Identifier
SASL	System Architecture Support Libraries
SNMP	Simple Network Management Protocol
TAR	Tape Archive
TCP	Transmission Control Protocol
UCLL	University College Leuven Limburg
UID	Unique Identifier
VM	Virtual Machine
ZIT	Zorgzame IT

Abstract

Heden ten dagen is vergrijzing in onze regio's een niet weg te denken problematiek. Mensen gaan steeds langer leven waardoor er nieuwe, agressieve en chronische ziektes ontwikkelen. Onderzoek naar die ziektes gebeurt vaak met beperkt beschikbare gegevens en er is een lange tijd nodig eer een conclusie omtrent een aandoening getrokken kan worden. Het gevolg is dat de zorg in het algemeen, zowel voor de patiënt, de instellingen en de overheid duurder wordt. Bijkomstig is er de nalatigheid van de patiënt welke tot ernstige complicaties kan leiden.

Bovenstaande problemen kunnen voor een groot deel opgelost worden via *patient empowerment*, samen met een digitaal opvolgsysteem. Deze zal de eindgebruiker bijstaan binnen zijn therapie en informatie op een gerichte wijze doorsturen naar het medisch team.

Via modulair programmeren werd er met de programmeertaal Erlang een applicatie ontwikkeld die later eenvoudig kan worden uitgebreid. Deze heeft twee abstracte *resources*, nl. patiënt en verpleegkundige, ter beschikking voor het controleren en uitvoeren van mogelijke stappen in een behandeling. De implementatie hiervan is bovendien eenvoudig te volgen door een *tutorial*. Realistische situaties worden zo gevormd tot efficiënte processen via het *NEU*-protocol, wat de kern is van dit *ICT*-platform. De verkenning tot de vorming van het meest optimale pad naar een oplossing zal parallel verwerkt worden naast de realistische uitvoering ervan. Tenslotte resulteert dit in een praktische digitale medische assistent.

Engels abstract (Summary)

Ageing of the population is a present-day problem in our regions. The life expectancy of humanity progressively increases which will develop new, aggressive and chronic diseases. Medical research often uses limited available data and requires a long period of time before one can reach a definitive conclusion concerning a disease. The consequence is that the healthcare for a patient, institutions and the government in general will become more expensive. Additionally, the negligence of the patient himself can lead to serious complications.

The above-mentioned problems can to a great extent be resolved through patient empowerment, together with a digital follow-up system. This system will guide the patient through his therapy and will send information in an efficient way towards the medical team.

An application is developed through modular programming, in the language Erlang, which can simply be extended afterwards. It has two abstract resources, i.e. patient and nurse, at hand for controlling and executing the possible steps of a treatment. Moreover, this implementation can be followed easily by a tutorial. Realistic situations are hereby formed to efficient processes through the NEU-protocol, which is the core of this ICT-platform. Surveying until the formation of the most optimal path to a solution will be processed in parallel next to the realistic execution hereof. Ultimately this results in a practical digital medical assistant.

1 Inleiding

1.1 Situering

Dit masterproefvoorstel is opgesteld door UCLL, meer bepaald door de onderzoeksgroep Zorgzame InformatieTechnologie.

UCLL, *University College* Leuven-Limburg, is een fusie van drie hogescholen in Vlaanderen, namelijk: Katholieke Hogeschool Limburg (KHLim), Katholieke Hogeschool Leuven (KHLeuven) en Groep T met als motto '*Moving minds*' [1].

De onderzoeksgroep Zorgzame InformatieTechnologie (ZIT) is een onderdeel van de Kennis & Innovatie Centra (KIC) van KHLeuven. Deze onderzoeksgroep behoort tot het departement Technologie. KIC verzamelt kennis over zorgtopics die enkel kunnen bestaan dankzij het gebruik van technologie. Deze kennis wordt gebruikt om te delen met de maatschappij.

InformatieTechnologie (IT) is een belangrijke onderzoekscategorie betreffende de vakkennis van de opleiding Toegepaste Informatica. Hierbij wordt er meegewerkt aan nationale en internationale projecten en wordt er *consultancy* aangeboden voor projecten van organisaties of bedrijven die beperkte kennis hebben over dit vakgebied [2].

1.2 Probleemstelling

De problematiek van de vergrijzing speelt een belangrijke rol in onze huidige samenleving. De mens gaat steeds langer leven waardoor er nieuwe, agressievere, chronische ziektes zullen ontwikkelen. De beperkt beschikbare gegevens binnen een onderzoek en een lange onderzoekstijd hebben als gevolg dat de zorg in het algemeen voor de patiënt, instellingen en overheid duurder wordt.

Niet alleen de onderzoeken maar ook omslachtige procedures, weinig innovatie en verschillende Informatie- en CommunicatieTechnologie (ICT) platformen die niet met elkaar kunnen communiceren zorgen ervoor dat er niet efficiënt gewerkt kan worden en veroorzaken in de toekomst enkel een duurdere zorg.

Tot slot is er het gevaar op nalatigheid van de patiënt in verband met de therapie, innemen van medicatie, etc., wat in sommige gevallen tot ernstige complicaties kan leiden.

1.3 Probleemanalyse

Onderzoek en innovatie zijn duur en zullen voor chronische patiënten enkel nog duurder worden. Dit komt enerzijds door omslachtige procedures, dure materialen en machines, vele consultaties, etc. Anderzijds komt dit omdat de patiënt te weinig betrokken wordt bij zijn aandoening en hij er maar mee moet leven. Een gepaste oplossing zou zijn dat de patiënten een actievere rol speelt in de gezondheidssector, dit noemt men *patient empowerment* [3], waardoor de kosten kunnen dalen (zie Figuur 1).

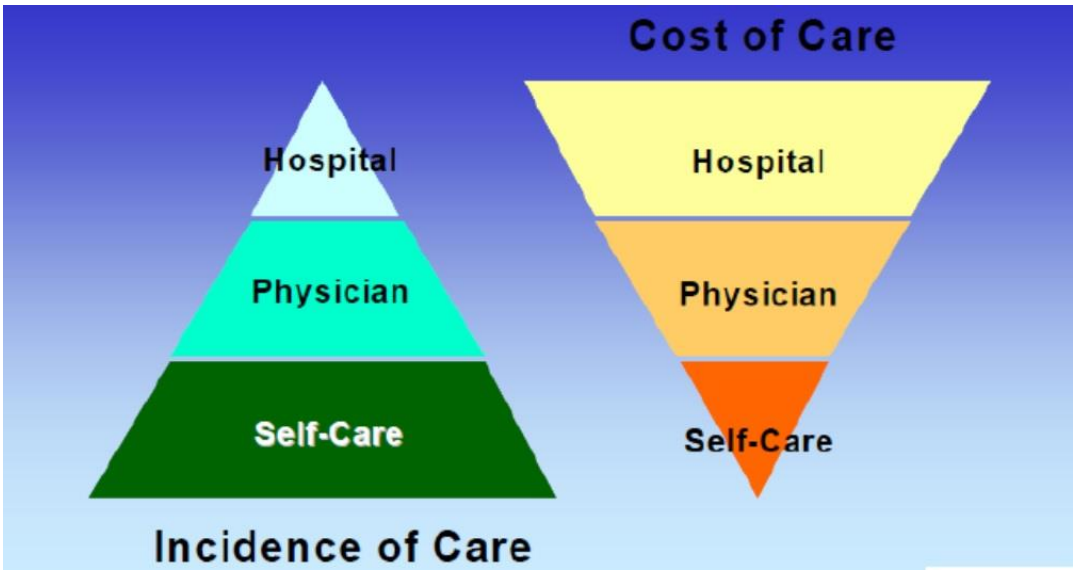
Patient empowerment wordt ten sterkste aangeraden door het Europees *PAtients Leading and mANaging their healThcare through Ehealth* (*PALANTE*) project [4]. Voor deze thesis ben ik naar de *PALANTE final conference* te Brussel geweest op 25 juni 2015. De inleiding is gebaseerd op bevindingen die op deze conferentie zijn aangehaald.

Via het integreren van andere systemen zoals *Global Positioning System* (*GPS*) *tracking*, alarmen en sensoren kunnen zelfs gezonde personen, in mindere mate en mits het recht op privacy in acht genomen, opgevolgd worden zodat aandoeningen sneller opgemerkt kunnen worden.

Door de verschillende platformen beter met elkaar te laten communiceren binnen dezelfde of verschillende diensten en instellingen zorgt dit voor een verbeterde informatiestroom. Op deze

manier worden ziektebeeld(en), situatie en klinische toestand gedeeld waardoor men grondiger en efficiënter de zorg kan toepassen.

Naast de communicatie tussen de verschillende platformen moet men tijdens het programmeren rekening houden met de noden en eisen van de eindgebruiker, terwijl de expertise in de achtergrond gehouden wordt. Bijvoorbeeld in een opvolgsysteem voor patiënten moet men geen termen gebruiken die men niet kan verstaan, maar eenvoudig te begrijpen tekst zodat er geen extra *tools* gebruikt moeten worden voor opzoekwerk. Op deze manier zullen het medisch team en de patiënt sneller overweg kunnen met de beschikbare technologie.



Figuur 1: Kosten op verschillende niveaus [5].

1.4 Doelstellingen

Verschillende platformen integreren is de algemene vereiste van het onderzoek waar binnen deze thesis deel van uitmaakt. Hierbij worden gegevens, mits toestemming van de patiënt, beschikbaar gemaakt en kan die beter worden gedistribueerd naar de betrokken personen. Er zal voor ieder lid van het medisch team en de patiënten een persoonlijk digitale assistent ter beschikking staan, zodat de noden en eisen van de gebruiker met kennis van experten gelost worden. Deze assistent, niet te verwarren met de vroegere *organiser* of kleine computer (PDA), zal voor de gebruiker (patiënt, medisch team) bepaalde taken verlichten of automatisch uitvoeren, zoals het bestellen van medicatie, het regelen of verplaatsen van afspraken. Op deze manier zullen alle, in deze thesis, voorgestelde problemen min of meer opgelost worden in de toekomst. Via de schaal- en uitbreidbaarheid zal het programma up-to-date kunnen blijven.

De doelstellingen van deze thesis zijn:

- Het stap voor stap uitwerken van een ICT-platform voor de zorgsector met het *NEU*-protocol, in de programmeertaal Erlang, tot een basisapplicatie. Deze moet toelaten om in latere fases experimenten uit te voeren en uitbreidingen te voorzien om de toekomstige visie van een digitale assistent te bekomen. Met de bijhorende teksten in deze masterthesis is de werking van de broncode eenvoudiger te begrijpen en te volgen.
- De vernieuwing van deze thesis is een virtuele verkenning. Deze heeft als doel complicaties of aanpassingen tijdig te ontdekken en een aangepast alternatief voor te stellen en te regelen. Bij medische taken moeten deze alternatieven, in een latere fase, altijd door de betrokken arts nagekeken en goedgekeurd worden.

2 Erlang

De Deense ingenieur Agner Krarup Erlang (Figuur 2) concipieerde het begrip Erlang. Hij introduceerde de Erlang-verdeling in de kansberekening voor het modelleren van de tijdsduur van de oproepen in een telefooncentrale (de wachtrijtheorie). Er wordt in deze thesis niet dieper ingegaan op bovenstaande theorie, voor meer informatie kan men terecht op volgende referentie [6].

Figuur 2: Agner Krarup Erlang [7]

Figuur 3: Erlang logo [8]

In 1987 ontwikkelde *Ericsson* een programmeertaal genaamd Erlang, deze term stond tevens voor de afkorting *Ericsson LANGuage* [9].

Erlang is een functionele, declaratieve programmeertaal. Deze werd ontwikkeld voor het bouwen van massief schaalbare *soft real-time* systemen met hoge inzetbaarheid. Dit houdt in dat wanneer een module, op een bepaald ogenblik, veel *input* moet verwerken de *load balancing* zeer snel en efficiënt kan worden gedaan over meerdere processen en/of verschillende computers. Tevens kan er terug worden afgebouwd wanneer dit niet meer nodig is. Omwille van de ingebouwde ondersteuning van gelijktijdigheid, de gemakkelijke distributie en fouttolerantie, wordt Erlang gebruikt in de telecommunicatie, het bankwezen, *e-commerce* en *messaging* [10].

Deze programmeertaal maakt gebruik van *Message Passing Interface* (*MPI*). Dat houdt in dat er berichten verstuurd worden tussen de verschillende modules of nodes, zo moet er de facto geen synchronisatie gebeuren en kan elke module op zijn eigen snelheid blijven werken zonder onderbrekingen.

De voordelen van Erlang zijn (referenties: [9], [11], [12], [13] en [14]):

- Er is ondersteuning voor parallellisme. Tientallen tot duizenden *threads* kunnen zonder of met weinig moeite tegelijkertijd gecreëerd en beheerd worden.
  Een voorbeeld is Klarna, een Zweeds *e-commerce* bedrijf. Het gebruikt Erlang om 9 miljoen gebruikers en 50 miljoen transacties af te handelen sinds 2005.
- Erlang heeft een geïntegreerde *MPI* in tegenstelling tot C waar men gebruik maakt van een aparte bibliotheek welke standaard wordt meegeleverd.
- De schaalbaarheid kan worden bewezen door de systemen, geschreven in Erlang, met enorm veel regels zoals:
  - Cowboy, *Yet Another Web Server* (Yaws) en Zotonic
  - Github
  - WhatsApp

    - Facebook chat
    - Call of Duty *server core*
    - League of Legends chatsysteem
    - Etc. … zie referentie [11] voor meer programma's

  Er wordt een onderscheid gemaakt tussen horizontale en verticale schaalbaarheid:
    - Horizontale schaalbaarheid is het communiceren met processen op verschillende machines.
    - Verticale schaalbaarheid is het communiceren met processen op dezelfde machine.
- Het werkt op elk *Operating System* (*OS*) dankzij de Erlang *Shell* (*EShell*). Deze *EShell* creëert als het ware een eigen *virtual machine* (*VM*) voor Erlang. De *EShell* neemt niet veel plaats in in het geheugen zowel na installatie, als tijdens de werking ervan.
- Het blijven lichtgewicht processen waardoor er geen zware eisen gesteld moeten worden aan de servers.
- Er is een distributie(*OTP*), zie Bijlage 0, en faaldetectie ingebouwd.
- Het is een functionele taal met degelijke, *dynamic typing* en vergelijkconstructie (*matching constructs*).
- *Het bevat hot code swapping* wat wil zeggen het *updaten* en *upgraden* van een applicatie mogelijk is zonder dat deze opnieuw opgestart of stil gelegd moet worden.
- Bij de meeste programmeertalen worden de processen van de applicatie tijdelijk bevroren om aan *garbage collection* te doen. Bij Erlang gebeurt dit terwijl de processen blijven werken.
- Er is een mogelijkheid om te communiceren met C code, dit wordt gedaan door *Input/Output* (*I/O*) poorten te kunnen aanspreken of om in Erlang verder te kunnen bouwen bovenop de bestaande C code.

De nadelen van Erlang zijn (referenties: [9], [11] en [14]):

- Er is een gebrek aan *static typing*. Dit wil zeggen dat de meeste fouten pas worden opgemerkt tijdens de werking en niet tijdens het compileren.
- Het leren programmeren met deze taal voor programmeurs, die gewoon zijn te werken met imperatieve talen, vergt een grote omschakeling in redeneren.
- Er zijn weinig hulpmiddelen, boeken, bibliotheken, ... Dit komt omdat Erlang nog geen populaire programmeertaal is. Desalniettemin ontstaat stilaan een evolutie, steeds meer bedrijven en programmeurs gebruiken Erlang, en dit omwille van de vele voordelen die de programmeertaal te bieden heeft. Enkele voorbeelden hiervan zijn:
    - Amazon.com
    - Yahoo!
    - Facebook
    - WhatsApp
    - Huffington Post
    - Etc. … zie referentie [11] voor meer bedrijven en waarvoor ze Erlang gebruiken.

Op volgende websites bevindt zich informatie voor het programmeren met Erlang:

- Learn You Some Erlang [15], ideaal voor beginners.
- Op StackOverflow [16] hier kan men zoeken naar een oplossing of zelf een vraag stellen.

- http://www.erlang.org/doc/man/ hier staan alle functie beschrijvingen en hoe ze gebruikt dient te worden.

Dankzij de voordelen, met name vooral de hoge schaalbaarheid, is het een logische keuze te opteren voor Erlang. Niet enkel de aangehaalde voordelen zijn van belang maar ook de persoonlijke keuze om een keer met een lichtgewicht, functionele programmeertaal te werken spelen een rol.

Hieronder zal een simpel Erlang teller programma met *messaging*, als voorbeeld, worden toegelicht. De teller zal per seconde de huidige waarde met één verhogen. De gebruiker kan een beginwaarde invoering alsook starten met de *default* waarde '0'. Eerst zullen de nodige *headers* (*hrl*) worden toegelicht alvorens de eigenlijke code. In Bijlage A wordt het maken van een *release* verder aangehaald.

Vooraleer er gestart kan worden met de eigenlijke code en functies moeten de juiste *headers* gebruikt worden. Het eerste is de modulenaam welke hetzelfde moet zijn als de bestandsnaam.

```
-module(voorbeeld).
```

Daarna volgt de opsomming van alle functies, met het aantal parameters, die zich in deze module bevinden.

```
-export([create/0, start/1, setWaarde/2, getWaarde/1, stop/1]).
```

Op identieke wijze worden tevens de private functies opgesomd. Zo vormt er zich een globaal overzicht van de functies die zich in de module bevinden.

```
-export([loop/1]). % For internal use only.
```

In dit voorbeeld zijn alle *headers* afgehandeld en kan er begonnen worden aan de eigenlijke code. Via een functieoproep zal een instantie van de module aangemaakt worden. Deze functie zal een *Process IDentifier* (*PID*) teruggeven, met deze *PID* kan men de juiste instantie van de module oproepen.

De `spawn` functie start zo'n instantie op. Het roept via de `?MODULE`, een macro die voor de huidige modulenaam instaat, de `loop` functie op met als parameterwaarde `voorbeeld`.

```
create() ->
  spawn(?MODULE, loop, [voorbeeld]).
```

Betreft het een module die overal bereikbaar moet zijn en waarvan er maar één instantie nodig is, zoals een logging module, wordt er een *event manager* aangemaakt. In de code hieronder zal er eerst nagegaan worden of er al zo'n *event manager* bestaat, zo ja wordt deze eerst verwijderd vooraleer er een nieuwe wordt aangemaakt. Dit gebeurt als volgt:

```
create() ->
  case whereis('logger manager') =:= undefined of
    true ->
      register('logger manager', spawn(?MODULE, start, [])),
      {ok, 'logger manager'};
    false ->
      unregister('logger manager'),
```

```
    register('logger manager', spawn(?MODULE, start, [])),
    {ok, 'logger manager'}
  end.
```

Na de aanmaak zal de teller gestart worden door middel van de startfunctie. Als parameter wordt hier de *PID* meegegeven zodat de juiste instantie wordt aangesproken, deze wordt tegelijkertijd op geldigheid gecontroleerd.

```
start(Pid) when is_pid(Pid) ->
```

Om aan *messaging* te doen zal, zoals eerder aangehaald, de *PID* gebruikt worden om de juiste instantie aan te spreken. De module zal een bericht versturen naar de bevoegde instantie met als parameters:

- start voor de functienaam.
- Een referentie zodat een tegenbericht steeds bij de juiste functie terug komt.
- Een eigen *PID* van de aangesproken module.

```
  Pid!{start, R = make_ref(), self()},
```

Deze functie zal een bericht terug krijgen en zal opgevangen worden door de `receive`, dit wordt in de `loop` functie verder besproken.

```
  receive
```

Het uiteindelijke bericht.

```
    {R, D} -> D
  end;
```

Onderstaande functie zal een foutieve *PID* opvangen. Hier zal aan de gebruiker worden gemeld dat er iets fout is gelopen.

```
start(_Pid) ->
  'Ongeldige ID.'.
```

De volgende functie zal de waarde van de teller veranderen naar een nieuwe, door de gebruiker, ingevoerde waarde. Ook hier zullen de parameters worden gecontroleerd.

```
setWaarde(Pid, NieuweWaarde) when is_pid(Pid),
is_integer(NieuweWaarde) ->
  Pid!{setWaarde, NieuweWaarde};
setWaarde(_Pid, _NieuweWaarde) ->
  'Ongeldige ID en/of waarde.'.
```

Met de volgende functie kan de teller waarde worden opgevraagd.

```
getWaarde(Pid) when is_pid(Pid) ->
  Pid!{getWaarde, R = make_ref(), self()},
  receive
```

```
    {R, D} -> D
  end;
getWaarde(_Pid) ->
  'Ongeldige ID.'.
```

Het stoppen van de instantie kan via deze functieoproep.

```
stop(Pid) when is_pid(Pid) ->
  Pid!{stop, R = make_ref(), self()},
  receive
    {R, D} -> D
  end;
stop(_Pid) ->
  'Ongeldige ID.'.
```

De private *loop* functie is een recursieve functie en kan worden gezien als een *while loop* in imperatieve talen.

```
loop(Waarde) ->
```

De *loop* zal telkens, door de `receive`, blijven wachten op een binnenkomend bericht en zal na afhandeling verder gaan.

```
  receive
```

Elk bericht is een *tuple* waarin volgende gegevens staan: de naam van de uit te voeren logica, een referentie voor een tegenbericht, een *PID* voor een tegenbericht en/of variabelen staan. Deze manier van werken wordt ook *type matching* genoemd.

```
    {start, R, From} ->
      From!{R, gestart},
```

Omdat *loop* een recursieve functie is moet deze elke keer opnieuw worden opgeroepen.

```
      ?MODULE:loop(1);
    {setWaarde, NieuweWaarde} ->
      ?MODULE:loop(NieuweWaarde);
    {getWaarde, R, From} ->
      From!{R, Waarde},
      ?MODULE:loop(Waarde);
    {stop, R, From} ->
```

Om de module af te sluiten dient men ervoor zorgen dat de *loop* functie niet opnieuw herhaald zal worden.

```
      From!{R, gestopt}
```

`after` zorgt dat de functie maar een bepaalde tijd wacht op binnenkomende berichten. In dit voorbeeld zal er na één seconde (1000 milliseconden) de teller worden verhoogd met één.

```
  after
    1000 ->
      loop(Waarde + 1)
  end.
```

Wanneer men in de *EShell* navigeert naar de locatie van het programma kan men de applicatie testen. Men typt volgend commando in:

```
cd('PATH').
```

Waarbij `PATH` de locatie omvat bv.: `c:/Users/Gebruiker/Desktop`.

Vervolgens zal de module eerst gecompileerd moeten worden, daarvoor wordt volgend commando ingetikt in de *EShell*:

```
c(voorbeeld).
```

Bij fouten, tijdens het compileren, zal de broncode nagekeken en aangepast moeten worden vooraleer er opnieuw wordt gecompileerd. Wanneer de code is gecompileerd kan men beginnen met het testen van de applicatie. Om een instantie aan te maken en te starten worden volgende regels in de *EShell* ingevoerd:

```
Pid = voorbeeld:create().
```

```
voorbeeld:start(Pid).
```

Bij de `create` functie moet de *PID* worden opgevangen zodat men de andere functies kan aanspreken van de instanties in kwestie. Nu deze is aangemaakt en is gestart kan men een nieuwe waarde geven aan de teller of na enkele seconden de waarde opvragen. Om deze functies te gebruiken typt men in de *EShell*:

```
voorbeeld:setWaarde(Pid).
```

```
voorbeeld:getWaarde(Pid).
```

3 Next, Execute and Update protocol (NEU)

De afkorting *NEU* staat voor het *Next, Execute & Update* protocol, zie referentie [17]. Een module kent alle mogelijke stappen van een proces, dit wordt ook *eStep* genoemd. *eExe*, een tweede module, gaat binnen dit proces aan de *eStep module* vragen wat hij moet volbrengen. Vervolgens geeft de *eStep module* een reeks stappen door aan de *eExe module*, zoals deze dienen te gebeuren op dit actuele moment en in de desbetreffende omgeving bijvoorbeeld een fabriek, een ziekenhuis, etc. Dit wordt aangetoond in Figuur 4.

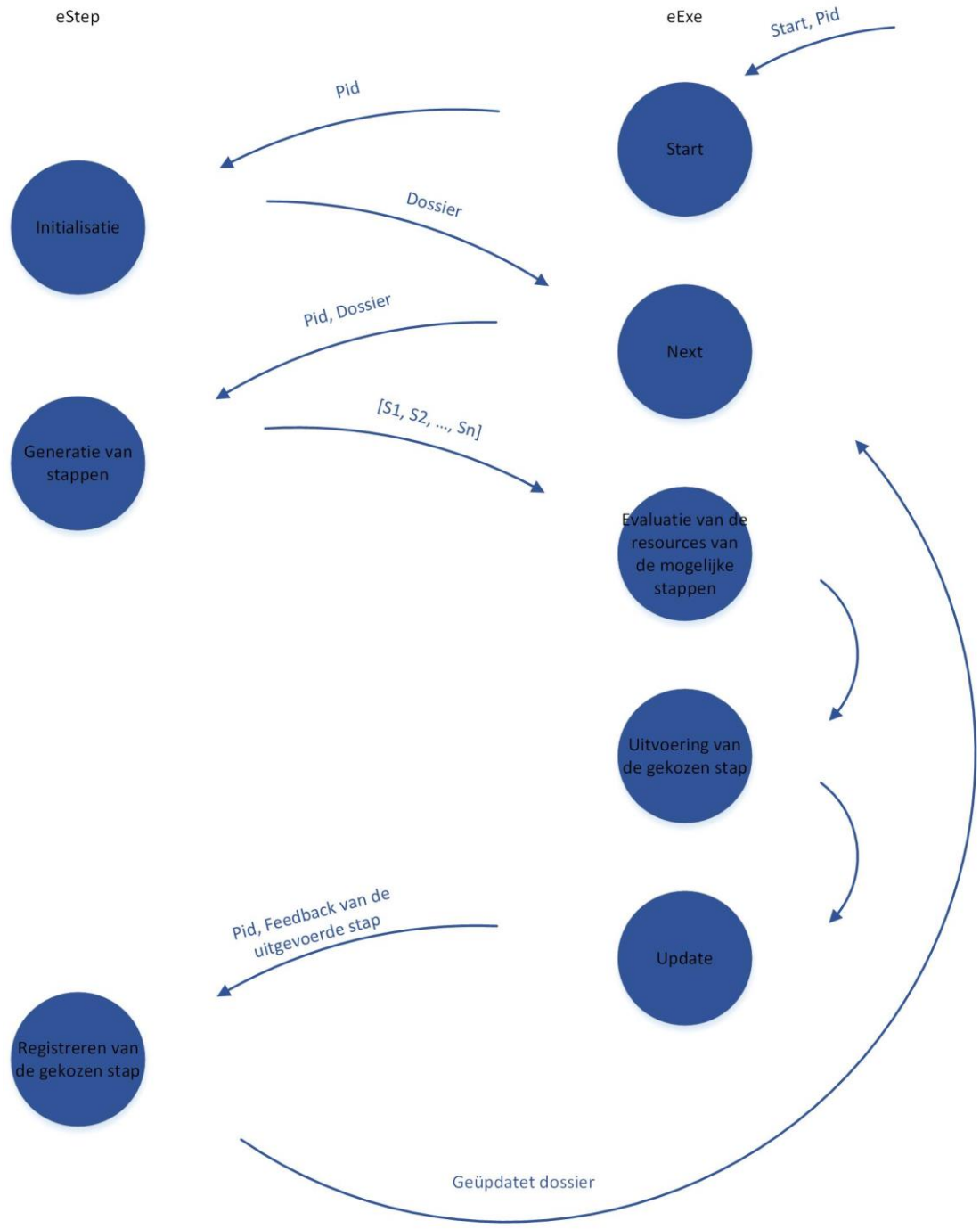
De *eExe module* zal voor elke stap de *resources* bekijken en evalueren. Een *resource* op zijn beurt is een middel zoals een auto, grondstoffen, medicatie, personen of personeel afhankelijk van de toepassing van de applicatie. De *resource* dient geëvalueerd te worden aangezien deze kan stuiten tegen belemmeringen als een uitgeputte voorraad of een foute type *resource*.

Als alle middelen van elke stap zijn doorlopen, wordt er een keuze gemaakt met welke stap er verder gegaan zal worden. De keuze wordt voltrokken aan de hand van het onderdeel dat de meeste beschikbare *resources* heeft. Ook zijn aspecten als ecologie en economisch voordeel belangrijk bij deze keuze.

Daaropvolgend dient de stap werkelijk uitgevoerd te worden. De *eExe module* zal op dat ogenblik de *eStep module* updaten over datgene dat uitgevoerd is, waarbij er eventueel *feedback* wordt meegestuurd.

Met de kennis over het te volgen pad en de eventuele *feedback* kan de *eStep module* nieuwe vervolgstappen genereren die opnieuw geëvalueerd en uitgevoerd moeten worden door de *eExe module* en die op zijn beurt de *eStep module* update met de eventuele *feedback*. Dit blijft zo doorgaan tot het proces ten einde is gelopen, daarna zal alles van vooraf aan opnieuw beginnen met, afhankelijk van de behoefte, dezelfde of nieuwe parameters.

In de volgende secties worden er voorbeelden gegeven om het *NEU*-protocol te verduidelijken. De voorbeelden zijn divers en betreffen verschillende activiteiten in onze maatschappij, zoals het stilvallen van een transportband, een gestrande koerier en de behandeling van een patiënt met diabetes.



Figuur 4: Schema NEU-protocol.

3.1 Voorbeeld 1: Defecte transportband

Een fabriek heeft op een zekere plaats twee transportbanden geïnstalleerd waarvan er maar één in gebruik is.

Deze transportband valt om een bepaalde reden in panne. Er zijn verschillende opties mogelijk:

- Alles omleiden zodat de tweede transportband in gebruik kan genomen worden en dan de eerste transportband repareren.
  Dit proces kan met volgende stappen opgesteld worden:
  - S1: Leg alles stil.
  - S2: Herleid de machines.
  - S3: Test gehele werking.
  - S4: Herneem productie.
  - S5: Repareer de eerste transportband.
  - S6: Zet eerste transportband op reserve.
- De eerste transportband repareren en de tweede transportband links laten liggen.
  Hieruit volgen onderstaande stappen:
  - S1: Leg alles stil.
  - S2: Repareer eerste transportband.
  - S3: Test gehele werking.
  - S4: Herneem productie.

De *eExe module* krijgt van de *eStep module* uit alle mogelijkheden de eerste stap S1 door. In dit geval is het bij beide opties hetzelfde en zal S1 uitgevoerd worden. Na uitvoering zal de *eExe module* het gedane proces terug rapporteren aan de *eStep module* en eventueel bemerkingen meesturen zoals de starttijd van het proces, optreden van fouten, etc.

Vervolgens zal de *eExe module* informeren naar de volgende stap. Deze wordt beantwoord door de *eStep module* die voor beide mogelijkheden de tweede stap S2 doorgeeft. Vanaf dit moment worden de *resources* in beraad genomen. Tijd is een belangrijke factor vermits de productie niet te lang kan stoppen want productie die stil ligt kost geld.

Een bijkomende *resource* die kan aangesproken worden is de hoeveelheid onderhoudspersoneel die aanwezig is en hoeveel ervaring deze mensen hebben. Meer personeel en meer ervaring zorgen ervoor dat een probleem sneller zal opgelost worden. De hersteltijd is bovendien ook afhankelijk van het probleem en de beschikbare reserve onderdelen.

Het protocol zal de stap kiezen met de minste opstarttijd én hersteltijd. Want, zoals reeds eerder aangehaald, productie die stil ligt kost geld en is dus economisch nefast.

Om deze panne te vermijden zal er *feedback* meegegeven worden over de keuze die gemaakt werd en hoelang deze optie heeft geduurd. Het systeem zal hiervan leren en de informatie opslaan om alzo in de toekomst dadelijk een gepaste keuze te maken tussen de mogelijkheden of een beter alternatief te geven.

De gegevens van de verschillende *resources* zoals de werktijd van de transportband na een onderhoud en de slijtage van onderdelen zullen het protocol ondersteunen om in dit geval een nieuwe onderhoudsbeurt te plannen en eventueel de nodige onderdelen te bestellen indien ze niet voorradig zijn. Deze verbetering zal in dit voorbeeld er voor zorgen dat de tweede transportband

klaar is om bij een eventuele panne of onderhoudsbeurt de eerste transportband over te nemen. Zo blijft er enkel nog de stop- en starttijd over als vertraging bij het wisselen.

3.2 Voorbeeld 2: De gestrande koerier

Een koerierdienst zendt al zijn koeriers uit via een efficiënte route om pakjes te bezorgen. Tijdens zijn dienst valt een koerier in panne wegens een defect aan zijn voertuig. De koerier belt zijn hoofdkwartier op om zijn probleem te melden. Hier beschikt men over reservewagens en een eigen reparatiedienst. Het hoofdkwartier heeft nu verschillende mogelijkheden om dit op te lossen:

- Laat de koerier het zelf oplossen.
- Laat een andere koerier zijn route wijzigen door een nieuwe efficiënte route zodat deze onderweg zijn pakjes en de gestrande koerier kan oppikken en de wagen laten slepen.
  - S1: Route aanpassen van de dichtstbijzijnde koerier.
  - S2: Sleepdienst contacteren om wagen te laten slepen.
  - S3: Pakjes overladen en koerier meevoeren.
  - S4: Aangepaste route vervolgen.
- Laat de koerier slepen en laat een andere koerier zijn route wijzigen door een nieuwe efficiënte route zodat deze koerier onderweg de pakjes en koerier kan oppikken bij de sleepdienst.
  - S1: Sleepdienst contacteren.
  - S2: Route aanpassen van de dichtstbijzijnde koerier.
  - S3: Pakjes overladen en koerier meevoeren.
  - S4: Aangepaste route vervolgen.
- Laat de wagen eerst slepen naar het hoofdkwartier
  - Er is een reservewagen beschikbaar, zodat de koerier zijn route kan verder zetten.
    - S1: Sleepdienst contacteren of sleepwagen van bedrijf op pad sturen.
    - S2: Pakjes overladen van wagen naar reservewagen.
    - S3: Route vervolgen.
  - Er is geen reservewagen beschikbaar
    - Wachten tot een andere koerier klaar is met zijn ronde om daarna de ronde over te nemen van de gestrande koerier en stuur hem naar huis.
      - S1: Sleepdienst contacteren of sleepwagen van bedrijf op pad sturen.
      - S2: Stuur gestrande koerier naar huis.
      - S3: Wachten tot een koerier klaar is met zijn ronde.
      - S4: Pakjes overladen.
      - S5: Route vervolgen door de andere koerier.
    - Wachten tot een andere koerier klaar is met zijn ronde om daarna de gestrande koerier zijn ronde te laten afmaken en de collega naar huis te sturen.
      - S1: Sleepdienst contacteren of sleepwagen van bedrijf op pad sturen.
      - S2: Wachten tot een koerier klaar is met zijn ronde.
      - S3: Stuur aangekomen koerier naar huis.
      - S4: Pakjes overladen.
      - S5: Route vervolgen door de gestrande koerier.
  - Laat de pakjes liggen tot de volgende werkdag.
    - S1: Sleepdienst contacteren of sleepwagen van bedrijf op pad sturen.

- S2: Pakjes uitladen.

Nu de mogelijkheden gekend zijn gaan de middelen geëvalueerd worden. Door deze middelen te evalueren zal er een optie gekozen worden en het stappenplan gevolgd worden dat bij de desbetreffende mogelijkheid hoort.

Het wikken en wegen van deze middelen hangt af van verschillende factoren die door het bedrijf kan worden ingegeven, zoals de voorkeuren van het bedrijf, welke kunnen gemeten worden bv.: financiële toestand, de tijden van het slepen, tijden van hoelang het duurt vooraleer er een koerier tot bij de gestrande koerier kan zijn of klaar is met zijn ronde, etc.

- De koerier zijn plan laten trekken is geen optie, aangezien deze niet zal weten wat te doen.
- Andere koeriers zijn te ver weg om een alternatieve route voor te stellen.
- Het hoofdkwartier heeft een reservewagen beschikbaar
    - Het bedrijf heeft zelf geen sleepwagen dus moet er een extern bedrijf gebeld worden
        - Sleepkosten
        - Sleeptijd
        - Afhankelijk vanwaar de sleepdienst komt kan er eventueel de reservewagen gesleept worden tot bij de gestrande koerier om van daar uit zijn route verder te zetten. De auto in panne wordt dan naar het hoofdkwartier gesleept.
        - Afhankelijk vanwaar de sleepdienst komt sleept deze de gestrande wagen naar het hoofdkwartier.
    - Er is nog tijd over om alle pakjes te bezorgen.
    - Personeelskosten blijven op deze manier zo laag mogelijk.

Zoals men zelf al kan voorspellen aan de hand van deze gegevens zal er gekozen worden om de gestrande koerier te laten slepen naar het hoofdkwartier zodat hij met de reservewagen zijn ronde kan afmaken.

Het systeem kan dan zelf de dichtstbijzijnde sleepdienst opbellen voor het wegslepen. Indien blijkt dat het zeer druk is bij de sleepdienst en ze pas over een uur iemand kunnen sturen zal deze *feedback* worden meegegeven aan de *eStep module*.

De *eStep module* van het systeem zal nieuwe stappen moeten genereren om met deze situatie om te gaan:

- Toch laten slepen.
- Een andere koerier zijn route laten wijzigen.

Omdat alle koeriers meer dan een uur onderweg zijn om tot bij de gestrande koerier te geraken, zal er voor gekozen worden om toch te wachten op de sleepdienst. Hierdoor kan er maar een deel van de route worden gedaan en de rest van de pakjes zal tot de volgende werkdag moeten wachten. Met deze *feedback* zal de *eExe module* de *eStep module* updaten en de nodige stappen genereren.

Als het systeem de mogelijkheid heeft om advies te geven zou het eventueel de volgende voorstellen om de economische rendabiliteit te verhogen kunnen bieden aan de *CEO* van het bedrijf:

- Een eigen sleepwagen aanschaffen.
- Als er geen reservewagen is of er zijn er te weinig, een nieuwe of extra wagen aankopen.

- Een afspraak maken met een sleepdienst voor een bepaalde service aan te kopen zodat ze voorrang krijgen op de andere klanten.

3.3 Voorbeeld 3: Patiënt met diabetes mellitus type 1

Een patiënt met diabetes mellitus type 1 of gewoonweg suikerziekte type 1, moet regelmatig (minimaal 4 keer per dag) zijn bloedglucosewaarde meten en de nodige insuline inspuiten. De stappen kunnen als volgt beschreven worden. Hierbij wordt enkel de eerste stap volledig uitgewerkt omdat de andere hoofdstappen gelijkwaardig zijn:

- S1: Meet de bloedglucosewaarde na het ontbijt.
    - S1.1: Hypoglycemie.
        - S1.1.1: Aanvaardbare laagte.
            - Suikerhoudende producten eten of drinken.
        - S1.1.2: Niet aanvaardbare laagte
            - S1.1.2.1: Zelf Glucagen Hypokit toedienen. Arts en thuisverpleging verwittigen.
            - S1.1.2.2: Bewustzijn verloren
                - Hulpdiensten verwittigen.
    - S1.2: Neem de juiste dosis insuline.
    - S1.3: Hyperglycemie.
        - S1.3.1: Aanvaardbare hoogte.
            - S1.3.1.1: Gaan sporten, nadien terug testen.
            - S1.3.1.1: Niets meer eten, na een bepaalde tijd terug testen.
        - S1.3.2: Niet aanvaardbare hoogte.
            - S1.3.2.1: Arts en/of thuisverpleging verwittigen voor controle.
            - S1.3.2.2: Naar ziekenhuis gaan.
            - S1.3.2.3: Hulpdiensten verwittigen.
- S2: Meet de bloedglucosewaarde na het middagmaal.
    - S2.1: Hypoglycemie.
    - S2.2: Neem de juiste dosis insuline.
    - S2.3: Hyperglycemie.
- S3: Meet de bloedglucosewaarde na het avondmaal.
    - S3.1: Hypoglycemie.
    - S3.2: Neem de juiste dosis insuline.
    - S3.3: Hyperglycemie.
- S4: Meet de bloedglucosewaarde voor het slapen gaan.
    - S4.1: Hypoglycemie.
    - S4.2: Neem de juiste dosis insuline.
    - S4.3: Hyperglycemie.

Het systeem zal nagaan wanneer het tijd is om de bloedglucosewaarde te meten ende patiënt op dit moment verwittigen. De patiënt zal zijn metingen terug rapporteren aan het systeem. Dit zal dan bepalen welke volgende stap men zal moeten ondernemen.

Bij normale waarden zal de juiste dosis insuline worden bepaald en zal het systeem via een melding de patiënt op de hoogte brengen. De patiënt zal op zijn beurt het systeem laten weten of dit gebeurd is.

Bij een te lage bloedglucosewaarde, ook wel hypoglycemie genoemd, zijn verschillende opties mogelijk. Als de waarde aanvaardbaar laag is, kan de patiënt suikerhoudende producten eten of drinken en zich opnieuw testen. Wanneer de waarde onder de laagste grenswaarde komt zal de patiënt de Glucagen Hypokit (een hormoon dat de bloedsuikerspiegel snel doet verhogen) moeten toedienen en ook de arts zal verwittigd worden voor het uitvoeren van een controle. Indien de patiënt, door deze lage waarde, niet meer bij bewustzijn is om zelf deze kit toe te dienen, zal het systeem de hulpdiensten verwittigen.

Bij een te hoge bloedglucosewaarde, hyperglycemie, kunnen er op langere termijn ernstige complicaties ontstaan. Ook hier zijn verschillende stappen mogelijk. Als de waarde aanvaardbaar hoog is kan het systeem melden dat sporten de waarde doet verlagen of dat men beter niets suikerhoudend eet en drinkt tot de waarde terug normaal is. Men dient dus wel na te gaan wanneer de bloedglucosewaarde opnieuw normaal is. Bij een niet aanvaardbare hoge waarde kan de arts en/of thuisverpleging worden verwittigd voor een controle. Er kan ook geopteerd worden om naar het ziekenhuis te gaan voor controle en observatie. Als de patiënt zulke verplaatsing niet op eigen krachten kan, worden de hulpdiensten gecontacteerd.

De informatie omtrent diabetes mellitus komt uit referentie [18].

4 Activity types en bijhorende activity instances

Gedurende het jaar is er gezocht naar een implementatie en optimalisatie voor het integreren van het NEU-protocol, zie hoofdstuk 3, in een ICT-platform in de zorgsector.

Eén van de voordeel van de huidige implementatie is dat de broncode eenvoudig uitbreid- en schaalbaar is.

Een ander pluspunt binnen dit programma is de toepasbaarheid in andere sectoren, dit door enkele kleine aanpassingen bij de *eStep module* binnen het *NEU*-protocol door te voeren.

Het huidige programma is modulair geprogrammeerd. Deze aanpak is doorgetrokken naar de wijze waarop de uitleg is geschreven. Het programma zal stap voor stap uitgelegd worden om zo eenvoudiger en sneller de uiteindelijke opbouw te begrijpen. Om de verschillende fases te kunnen testen wordt er verwezen naar de instructies in bijlage B.

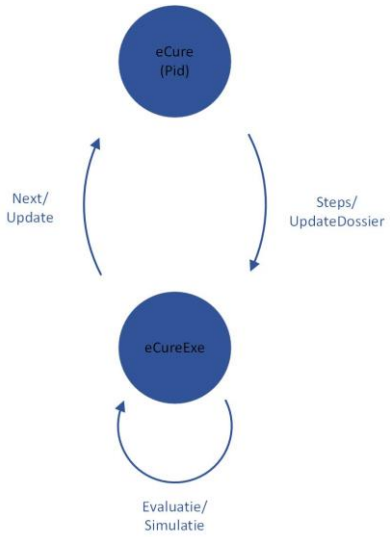
Er zal telkens maar één ziekte gebruikt worden, namelijk bronchitis. De broncode zelf is getest met een bijkomende ziekte, namelijk Diabetes Type 1, dit om de uitbreidbaarheid te demonstreren naar meerdere ziektes toe.

Bij het basisprincipe zal een zeer simpele versie van het *NEU*-protocol aangehaald worden. Op deze manier kan men beter kennis nemen van de implementatie van het protocol zelf. Tenslotte zullen er stukken code bijkomen en mogelijk aangepast worden naarmate men vordert in de fases.

4.1 Eén activity type of één therapie

Deze fase zal de basisimplementatie bespreken waarin één ziekte, namelijk bronchitis (*eBronchitisTN module*), gebruikt zal worden. De therapie voor de ziekte bestaat uit een dosis antibiotica van 1mg welke tweemaal daags moeten ingenomen worden. Voor elke ziekte wordt er een stappenplan opgesteld en een uitvoering aangemaakt.

Figuur 5 stelt de implementatie van het *NEU*-protocol op een eenvoudige manier voor binnen de zorgsector. De modules waartussen de communicatie verloopt zijn *eCure* (*activity type*) en *eCureExe* (*activity instance*), respectievelijk *eStep* en *eExe* in hoofdstuk 3.



Figuur 5: Fase1 implementatieschema van het NEU-protocol.

Figuur 6 toont de werking zoals Figuur 4, met één enkel verschil dat er nu met *headers* (*hrl*) gewerkt wordt. Oorspronkelijk gebeurde de functieoproepen door de module, dit wordt nu overgenomen door de bijhorende *headers* binnen deze module. De reden hieromtrent is dat de modules en *headers* eenvoudiger kunnen vervangen en/of aangepast worden zonder andere, niet bijhorende, modules te wijzigen. Men dient wel rekening te houden met de functienamen in *hrl*'s, deze moeten hetzelfde blijven, zo niet zullen de andere modules wel aangepast moeten worden. Hieronder bevindt zich een broncodevoorbeeld om de modulariteit aan te tonen. Deze code komt voort uit de *header* van de interactie van het *NEU*-protocol, dit is analoog voor de *resources*.

eCureExe module:

```
Steps = eCureNext(DiseasePid, Dossier).
```

eCure header:

```
eCureNext(Pid, Dossier) ->
  Pid!{next, Ref = make_ref(), self(), Dossier},
  receive
    {Ref, Steps} -> Steps
  end.
```

eBronchitisTN:

```
{next, Ref, From, {_, Days, not_started}} ->
      From!{Ref, [{{'Antibiotica', '1mg'}, Days, am},
{{'Antibiotica', '1mg'}, Days, pm}]},
      ?MODULE:loop();
    {next, Ref, From, {_, Days, am}} ->
      From!{Ref, [{{'Antibiotica', '1mg'}, Days + 1, pm}]},
      ?MODULE:loop();
```

eDiabetesT1:

```
{next, Ref, From, {_Specs, Days, not_started}} ->
      From!{Ref, [{{'Measure device'}, Days, measure}]},
      ?MODULE:loop();
    {next, Ref, From, {{_, _, _, {_, High}}, Days, {Value,
measure}}} when Value > High ->
      From!{Ref, [{{'Insuline', '2ml'}, Days, inject}]},
      ?MODULE:loop();
```

Figuur 7 toont de meer gedetailleerde communicatie. Hierin staat wat, hoe en wanneer welke data verstuurd wordt. Aan de hand van deze figuur zal de volledige werking worden uitgelegd. Op deze afbeelding is een testmodule op te merken welke de verschillende modules compileert, aanmaakt en start. Omdat deze module in de toekomst geen nut meer zal hebben zal het in het verdere verloop van deze thesis niet meer aangehaald worden.

Naast de testmodule is er nog de logmodule. Deze staat niet getekend op de figuren en wordt niet besproken in deze tekst aangezien deze van constante aard is. Het doel is om te loggen welke stappen gezet zijn zodat de werking nadien gemakkelijker nagegaan kan worden.

De log module zal evenzeer in alle fases gebruikt worden en zal het hele logging-gebeuren opslaan in een *ETS* tabel welke kan bekeken worden, zie Bijlage B voor de instructies. De volgende tabellen worden gebruikt om de logging gegevens in op te slaan:

- logD is de *debug* tabel waar alle logging-gegevens worden geplaatst die tijdens *debug* modi gebruikt worden.
- logR is de *reality* of realiteitstabel en slaat alle keuzes, die door de patiënt of simulator, gemaakt worden op.
- logV is de *virtual* of virtuele tabel en houdt de keuzes bij die de virtuele verkenning maakt.

Figuur 8 geeft de *output* en tussentijdse evaluatie weer van de applicatie. De opmaak van de *debug* logging is voor alle fases hetzelfde en ziet er als volgt uit:

- Eerste kolom bevat een referentie of *primary key*. Deze bestaat uit een *tuple* met als waarden een referentie, gemaakt door de `make_ref()` functie, en een *timestamp* in microseconden (welke geplaatst is in een *tuple*). Deze kan men aanmaken door de `now()` functie op te roepen.
- De tweede kolom bevat een datum en een tijdstip. Dit kan met volgende functie opgesteld worden: `calendar:now_to_local_time(now()).`
- De derde kolom is de voornaamste omdat hier de uiteindelijke bruikbare gegevens komen te staan. De *tuple* wordt als volgt opgesteld:
    - Modulenaam waarin de logging gebeurd.
    - Functienaam waar de actie plaats vind.
    - Wat de gegevens moeten voorstellen bv.: Dossier, mogelijke stappen, etc.
    - De gelogde gegevens.

In deze fase van één therapie zijn er geen *resources* en zullen de stappen niet geëvalueerd worden. Vandaar zal de *eCureExe* alle stappen goedkeuren en uitvoeren. In de latere fases komen er echter de patiënt en verpleegkundige *resource* bij, welke in de toekomst verder uitgebreid kunnen worden.

Er is ook geen gebruikers*interface* alsook geen simulatie om de gebruiker te vervangen tijdens het testen. Bijgevolg zal de *eCureExe module* een willekeurige keuze maken tussen het innemen van medicatie of niet. In het verloop van deze thesis zal de *eCure module* gepast reageren op de *feedback* afkomstig van de gebruiker.

Zoals reeds aangehaald is maakt de implementatie gebruik van een *header* (*hrl*) bestand, zie Figuur 6. Dit bestand wordt geïntegreerd in de *eCureExe module* en zal het hele *NEU*-protocol op zich nemen. Zodoende zal de *eCure module* enkel de specifieke data doorgeven aan het *hrl*-bestand. In de test module zullen de nodige stappenplannen en *resources* aangemaakt worden, deze *Process IDentifier* (*PID*) wordt meegestuurd naar de *eCureExe module*.

Bij creatie wordt de *eCureExe module* aangemaakt en krijgt deze als parameter de *PID* van een bepaalde therapie mee. Deze module zal op zijn beurt het *hrl*-bestand aanspreken en de *PID* meegeven. Met deze therapie zal de *header* de juiste *eCure module* initialiseren en het begindossier terug geven aan de *eCureExe module*. Dit dossier ziet er als volgt uit:

```
{{Disease, Meds, Dose, N}, Days, not_started}
```

- `Disease`, de eigenlijke ziektenaam, die in dit voorbeeld 'Bronchitis' is.
- `Meds` staat voor medication en bevat de naam van de te nemen medicatie, bv. 'Antibiotica'.
- `Dose` bevat de dosis van de medicatie, bv. 1 mg. Later zal dit een variabele worden zodat de dosis aansluit bij de gemeten waarden.
- `N` is het globale totaal van de in te nemen medicatie. Deze parameter dient voor het nagaan van het tijdstip wanneer men met de medicatie mag stoppen, bv. 28 innames voor een volledige kuur.
- `Days` houdt op welke datum de inname moet gebeuren. Het formaat is een Gregoriaanse telling en zal later handig blijken bij het loggen van de complexere implementaties. De dagtelling wordt niet enkel bijgehouden in het dossier maar ook bij het updaten na uitvoering en later bij voorspelling en voortijdige inplanning. Zodoende kan het dossier aangepast worden met de dagtelling waarin de stap zich bevindt.
- `not_started` is de beginstatus, welke wil zeggen dat de kuur nog niet begonnen is. Deze dient om de nodige mogelijke stappen te kunnen genereren en zal gedurende het proces veranderen.

Met dit dossier zal *eCureExe* aan de *header*, via de *eCureNext* functie, de volgende mogelijke stappen opvragen. Het *hrl*-bestand zal deze specifieke data opvragen aan *eCure* door middel van de meegestuurde *PID*. Met deze `not_started` status zullen dit maar twee mogelijkheden zijn, namelijk `am` en `pm`. De weergave voor deze mogelijkheden worden weergegeven als volgt:

```
[{{'Antibiotica', '1mg'}, Days, am}, {{'Antibiotica', '1mg'}, Days, pm}]
```

Zoals eerder vermeld zal in *eCureExe* een willekeurige generatie plaats vinden om de gebruikersinteractie tijdelijk te vervangen. Deze generatie zal een stap kiezen samen met een willekeurige mogelijkheid om medicatie in te nemen, respectievelijk `takenDose` en `notTakenDose`. Ook zal in een later stadium *resources* of (hulp)middelen samen met een gebruikerssimulatie aangewend worden om een goede keuze te maken tussen de verschillende mogelijkheden. Deze verschillende factoren kunnen zijn; tijd, omgevingsfactoren, vaardigheden van de patiënt (mentaal of fysiek zwak), mantel- en/of thuiszorg, etc.

Nadat *eCureExe* een stap heeft gekozen en de nodige *feedback* heeft gegenereerd, zal deze module het dossier met de gegevens over de uitgevoerde stap doorgeven aan de *header* via de *eCureUpdate* functie. De *header* zal het dossier door het juiste stappenplan laten updaten. Bij deze update zal de status vervangen worden door de gekozen stap en de totale hoeveelheid worden verminderd met één. Stel dat *eCureExe* de `pm` keuze heeft gemaakt en de patiënt de medicatie heeft ingenomen, zal *eCureExe* onderstaande code naar het *hrl*-bestand doorgeven:

```
eCureUpdate(Pid, Dossier, {takenDose, Days, pm})
```

- via het commando `eCureUpdate` zal de juiste functie van de *header* worden aangesproken.
- Het `Dossier` is nodig om de status te kunnen aanpassen.

- `{takenDose, Days, pm}` is de *feedback* met de uitgevoerde stap, dit is nodig om een nieuwe status te kunnen doorgeven.

Het dossier ziet er dan als volgt uit:

```
{{Disease, Meds, Dose, N - 1}, Days, pm}
```

Zoals op te merken is, is de `not_started` status vervangen door de `pm` status. Bij de inname van de medicatie zal de totale hoeveelheid met één verminderen. Zo wordt er bijgehouden hoeveel medicatie nog ingenomen dient te worden.

eCureExe zal opnieuw aan de *header* vragen wat de volgende mogelijkheden zijn, deze vraagt dit op zijn beurt aan de *eCure module*.

```
eCureNext(Pid, Dossier)
```

In deze fase zal dit slechts één stap zijn, namelijk:

```
NDays = Days + 1
```

```
[{{'Antibiotica', '1mg'}, NDays, am}]
```

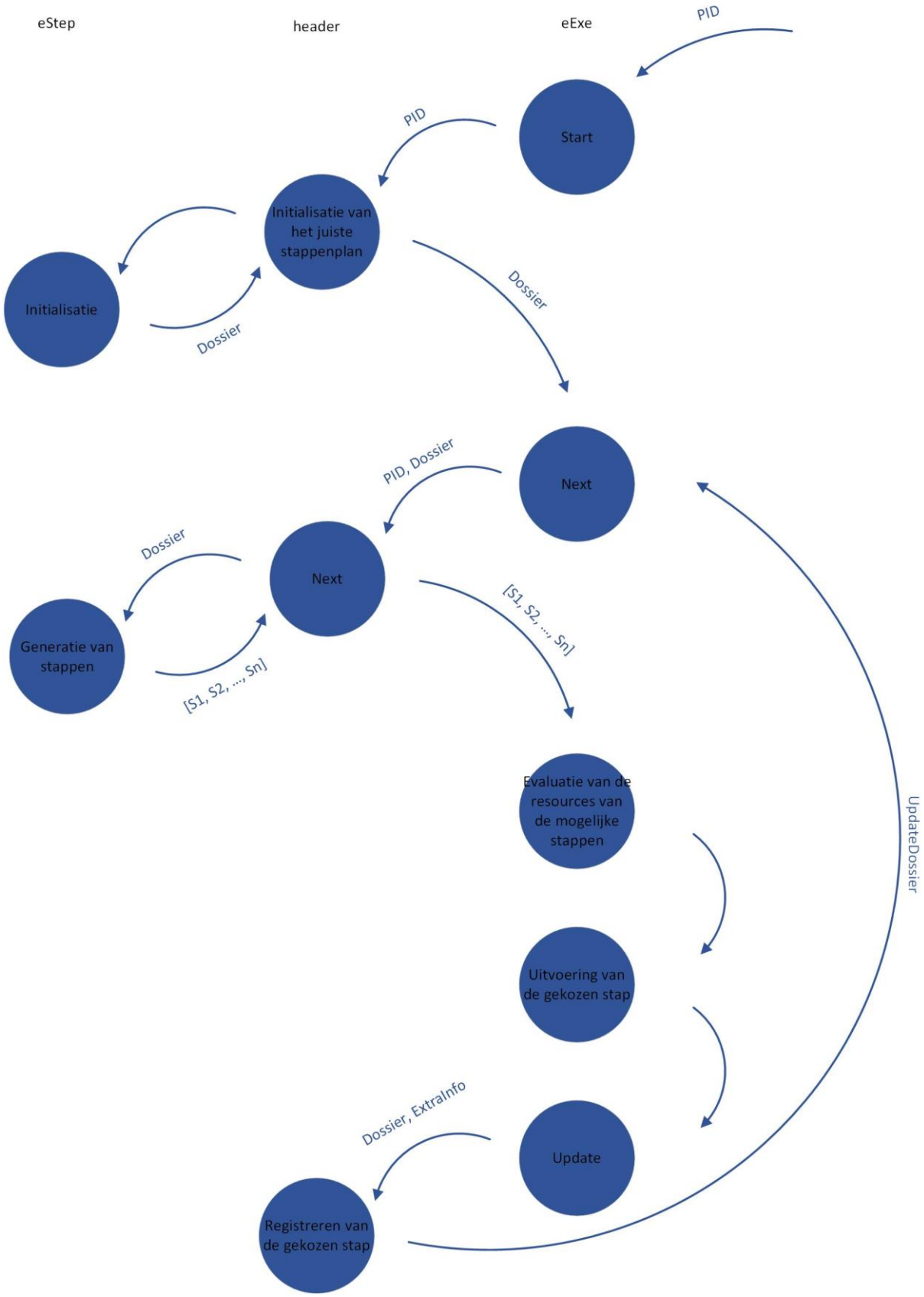
Na het uitvoeren van deze stap zal de willekeurige generatie opnieuw kiezen tussen `takenDose` en `notTakenDose`. Aangezien de patiënt zijn medicatie niet heeft ingenomen zal `notTakenDose` gekozen worden. *eCureExe* zal dan het dossier, via het *hrl*-bestand, door *eCure* laten updaten. Dit ziet er als volgt uit:

```
{{Disease, Meds, Dose, N}, NDays, am}
```

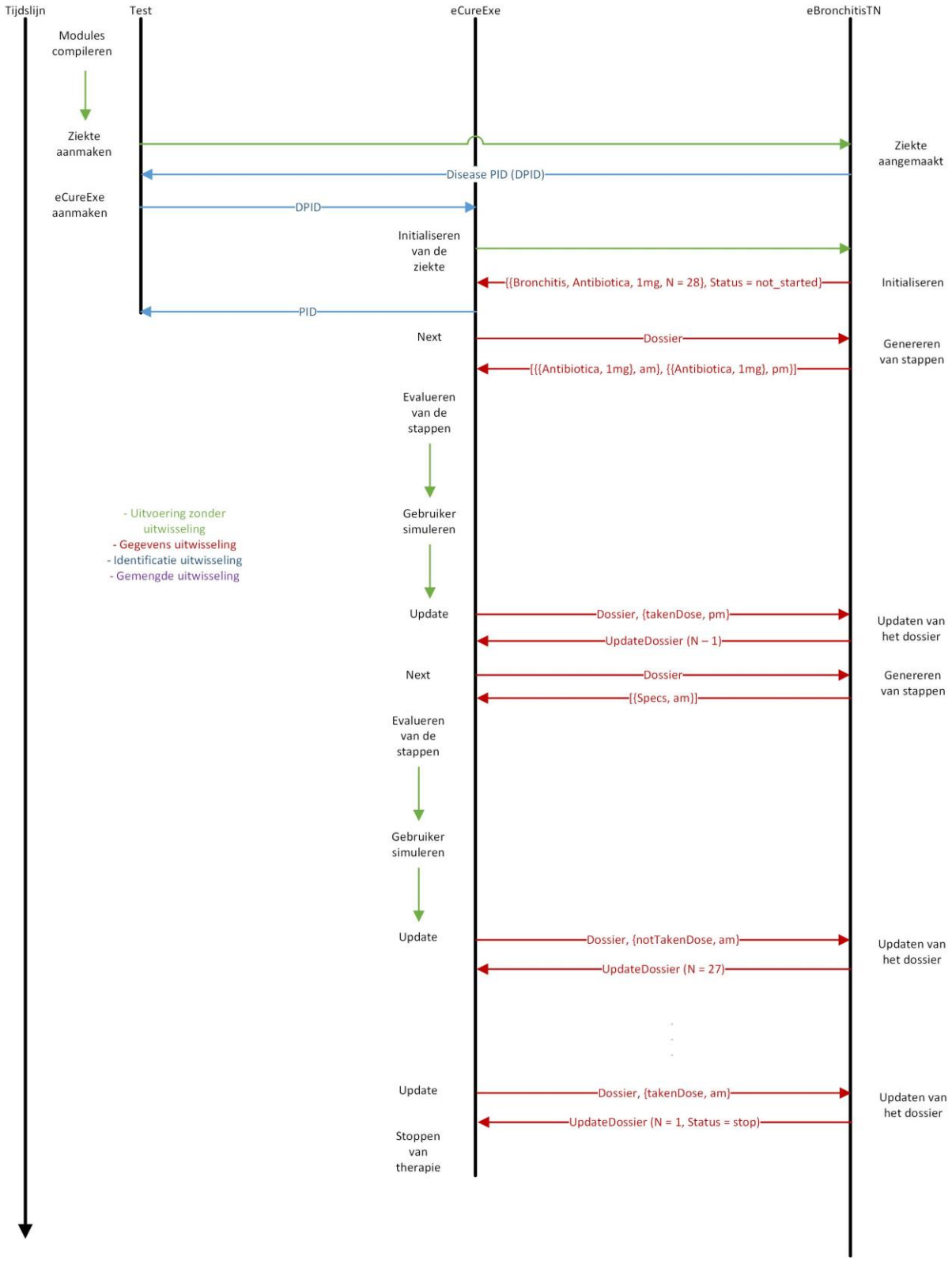
Doordat de medicatie niet is ingenomen wordt de totale hoeveelheid niet aangepast en wordt deze verplaatst naar de eerst volgende inname.

Aan het einde van de kuur zal de totale hoeveelheid zakken tot nul. *eCureExe* zal als volgende stap een `stop` krijgen. Deze mogelijkheid zal ervoor zorgen dat *eCureExe* zichzelf beëindigt en is het hele voorbeeld afgelopen.

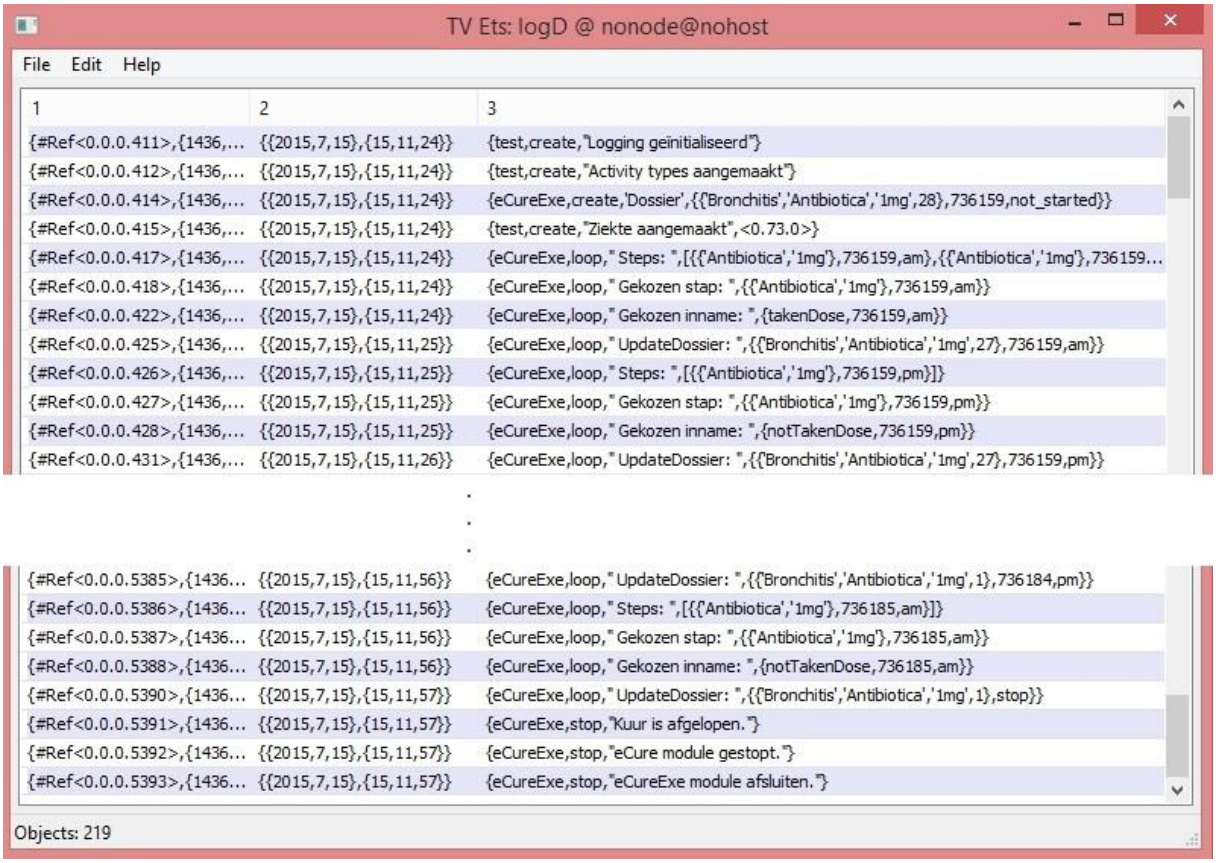
Voor elke code-stap die *eCureExe* doet: initialisatie, opvragen van verschillende mogelijkheden, *resources* nakijken (in deze fase nog niet van toepassing), gebruikersinteractie en het updaten van het dossier, is er een *loop* functie. Op deze manier wordt de broncode gestructureerd in plaats van deze in continuïteit te schrijven, zodat men later snel bepaalde stukken kan terugvinden om eventueel wijzigingen aan te brengen. Op deze wijze kan men eenvoudiger naar bepaalde segmenten gaan.



Figuur 6: Fase1 communicatieschema van het NEU-protocol met header.



Figuur 7: Fase1 communicatieschema van het NEU-protocol in detail.



Figuur 8: Fase1 output.

4.2 Tweede activity type of tweede therapie

In deze sectie wordt de extra ziekte (Diabetes Type1) kort aangehaald, het functioneren is in gelijke mate hetzelfde als voor de ziekte bronchitis. Het kort aanhalen van deze therapie is om de reden dat alle fases worden getest met zowel één als twee ziektes om steeds de uitbreid- en schaalbaarheid, in verband met meerdere aandoeningen, te testen. Er is in elke fase, behalve voor Fase1, extra code voorzien om zelf de test uit te kunnen uitvoeren.

Figuur 9 toont enkel de *output* enkel van de aandoening Diabetes in tegenstelling tot Figuur 10 waar beide ziektes simultaan gedemonstreerd worden. Op deze manier bekomt men een beeld van de uitbreid- en schaalbaarheid in deze broncode. In de volgende fases wordt de tweede ziekte niet meer aangehaald, wel kan men de gegevens bekijken door Bijlage B te volgen.

`{{Disease, Meds, Dose, Boundary}, Days, not_started}`

- `Disease`, de eigenlijke ziektenaam, die in dit voorbeeld 'Diabetes' is.
- `Meds` staat voor medication en bevat de naam van de te nemen medicatie, bv. 'Insuline'.
- `Dose` bevat de dosis van de medicatie, bv. 1 mg. Later zal dit een variabele worden zodat de dosis aansluit bij de gemeten waarden.
- `Boundary` staat voor de grenzen van de patiënt vooraleer hij een bepaalde handeling dient te doen. *Boundary* is een *tuple* welke een onder- en bovengrenswaarde bevat `{Low, High}`. In onderstaande toelichting worden de grenzen verklaard:
    - `Low` staat voor de onderste grens, wanneer men onder dit niveau komt zal de patiënt suikerhoudende producten moeten eten of drinken om terug naar de normale waarden te stijgen.

    - `High` staat voor de bovenste grens, als men boven dit niveau komt zal de patiënt insuline moeten spuiten om terug binnen de aanbevolen grenzen te komen.
    - Tussen de twee bovenstaande grenzen in zal de patiënt niets moeten ondernemen.
- `Days` houdt op welke datum de inname moet gebeuren. Het formaat is een Gregoriaanse telling en zal later handig blijken bij het loggen van de complexere implementaties. De dagtelling wordt niet enkel bijgehouden in het dossier maar ook bij het updaten na uitvoering en later bij voorspelling en voortijdige inplanning. Zodoende kan het dossier aangepast worden met de dagtelling waarin de stap zich bevindt.
- Status dient om de nodige mogelijke volgende stappen te kunnen genereren.

Per dag wordt er één meting gedaan met de daarbij horende behandeling. Zoals reeds aangehaald is, is deze ziekte nog niet volledig op punt maar dient het enkel om een test te doen op uitbreid- en schaalbaarheid.

1	2	3
{#Ref<0.0.0.452>,{1437,...	{{2015,7,20},{10,35,48}}	{test,create,"Logging geinitialiseerd"}
{#Ref<0.0.0.453>,{1437,...	{{2015,7,20},{10,35,48}}	{test,create,"Activity types aangemaakt"}
{#Ref<0.0.0.455>,{1437,...	{{2015,7,20},{10,35,48}}	{eCureExe,create,'Dossier',{{'Diabetes','Insuline','2ml',{30,70}},736164,not_started}}
{#Ref<0.0.0.456>,{1437,...	{{2015,7,20},{10,35,48}}	{test,create,"Ziekte aangemaakt",<0.77.0>}
{#Ref<0.0.0.458>,{1437,...	{{2015,7,20},{10,35,48}}	{eCureExe,loop," Steps: ",[{{'Meettoestel'},736164,measure}]}
{#Ref<0.0.0.459>,{1437,...	{{2015,7,20},{10,35,48}}	{eCureExe,loop," Gekozen stap: ",{{'Meettoestel'},736164,measure}}
{#Ref<0.0.0.464>,{1437,...	{{2015,7,20},{10,35,48}}	{eCureExe,loop," Gekozen inname: ",{48,736164,measure}}
{#Ref<0.0.0.467>,{1437,...	{{2015,7,20},{10,35,49}}	{eCureExe,loop," UpdateDossier: ",{{'Diabetes','Insuline','2ml',{30,70}},736164,{48,meas...
{#Ref<0.0.0.468>,{1437,...	{{2015,7,20},{10,35,49}}	{eCureExe,loop," Steps: ",[{{nothing},736164,nothing}]}
{#Ref<0.0.0.469>,{1437,...	{{2015,7,20},{10,35,49}}	{eCureExe,loop," Gekozen stap: ",{{nothing},736164,nothing}}
{#Ref<0.0.0.470>,{1437,...	{{2015,7,20},{10,35,49}}	{eCureExe,loop," Gekozen inname: ",{takenDose,736164,nothing}}
{#Ref<0.0.0.473>,{1437,...	{{2015,7,20},{10,35,50}}	{eCureExe,loop," UpdateDossier: ",{{'Diabetes','Insuline','2ml',{30,70}},736164,nothing}}
{#Ref<0.0.0.474>,{1437,...	{{2015,7,20},{10,35,50}}	{eCureExe,loop," Steps: ",[{{'Meettoestel'},736165,measure}]}
{#Ref<0.0.0.475>,{1437,...	{{2015,7,20},{10,35,50}}	{eCureExe,loop," Gekozen stap: ",{{'Meettoestel'},736165,measure}}
{#Ref<0.0.0.476>,{1437,...	{{2015,7,20},{10,35,50}}	{eCureExe,loop," Gekozen inname: ",{60,736165,measure}}
{#Ref<0.0.0.479>,{1437,...	{{2015,7,20},{10,35,50}}	{eCureExe,loop," UpdateDossier: ",{{'Diabetes','Insuline','2ml',{30,70}},736165,{60,meas...
{#Ref<0.0.0.480>,{1437,...	{{2015,7,20},{10,35,50}}	{eCureExe,loop," Steps: ",[{{nothing},736165,nothing}]}
{#Ref<0.0.0.481>,{1437,...	{{2015,7,20},{10,35,50}}	{eCureExe,loop," Gekozen stap: ",{{nothing},736165,nothing}}
{#Ref<0.0.0.482>,{1437,...	{{2015,7,20},{10,35,50}}	{eCureExe,loop," Gekozen inname: ",{notTakenDose,736165,nothing}}
{#Ref<0.0.0.487>,{1437,...	{{2015,7,20},{10,35,51}}	{eCureExe,loop," UpdateDossier: ",{{'Diabetes','Insuline','2ml',{30,70}},736165,nothing}}
{#Ref<0.0.0.488>,{1437,...	{{2015,7,20},{10,35,51}}	{eCureExe,loop," Steps: ",[{{'Meettoestel'},736166,measure}]}
{#Ref<0.0.0.489>,{1437,...	{{2015,7,20},{10,35,51}}	{eCureExe,loop," Gekozen stap: ",{{'Meettoestel'},736166,measure}}
{#Ref<0.0.0.490>,{1437,...	{{2015,7,20},{10,35,51}}	{eCureExe,loop," Gekozen inname: ",{36,736166,measure}}
{#Ref<0.0.0.493>,{1437,...	{{2015,7,20},{10,35,52}}	{eCureExe,loop," UpdateDossier: ",{{'Diabetes','Insuline','2ml',{30,70}},736166,{36,meas...
{#Ref<0.0.0.494>,{1437,...	{{2015,7,20},{10,35,52}}	{eCureExe,loop," Steps: ",[{{nothing},736166,nothing}]}

Figuur 9: Fase1.2 output Diabetes.

1	2	3
{#Ref<0.0.0.1126>,{1437...	{{2015,7,20},{9,55,45}}	{eCureExe,create,'Dossier',{{'Diabetes','Insuline','2ml',{30,70}},736164,not_started}}
{#Ref<0.0.0.1127>,{1437...	{{2015,7,20},{9,55,45}}	{test,create,"Ziekte aangemaakt",<0.112.0>}
{#Ref<0.0.0.1128>,{1437...	{{2015,7,20},{9,55,45}}	{eCureExe,loop," Steps: ",[{{'Antibiotica','1mg'},736164,am},{{'Antibiotica','1mg'},736164...
{#Ref<0.0.0.1129>,{1437...	{{2015,7,20},{9,55,45}}	{eCureExe,loop," Gekozen stap: ",{{'Antibiotica','1mg'},736164,am}}
{#Ref<0.0.0.1130>,{1437...	{{2015,7,20},{9,55,45}}	{eCureExe,loop," Gekozen inname: ",{takenDose,736164,am}}
{#Ref<0.0.0.1132>,{1437...	{{2015,7,20},{9,55,45}}	{eCureExe,loop," Steps: ",[{{'Meettoestel'},736164,measure}]}
{#Ref<0.0.0.1133>,{1437...	{{2015,7,20},{9,55,45}}	{eCureExe,loop," Gekozen stap: ",{{'Meettoestel'},736164,measure}}
{#Ref<0.0.0.1138>,{1437...	{{2015,7,20},{9,55,47}}	{eCureExe,loop," Gekozen inname: ",{27,736164,measure}}
{#Ref<0.0.0.1141>,{1437...	{{2015,7,20},{9,55,47}}	{eCureExe,loop," UpdateDossier: ",{{'Bronchitis','Antibiotica','1mg',27},736164,am}}
{#Ref<0.0.0.1142>,{1437...	{{2015,7,20},{9,55,47}}	{eCureExe,loop," Steps: ",[{{'Antibiotica','1mg'},736164,pm}]}
{#Ref<0.0.0.1143>,{1437...	{{2015,7,20},{9,55,47}}	{eCureExe,loop," Gekozen stap: ",{{'Antibiotica','1mg'},736164,pm}}
{#Ref<0.0.0.1144>,{1437...	{{2015,7,20},{9,55,47}}	{eCureExe,loop," Gekozen inname: ",{notTakenDose,736164,pm}}
{#Ref<0.0.0.1149>,{1437...	{{2015,7,20},{9,55,47}}	{eCureExe,loop," UpdateDossier: ",{{'Bronchitis','Antibiotica','1mg',27},736164,pm}}
{#Ref<0.0.0.1150>,{1437...	{{2015,7,20},{9,55,47}}	{eCureExe,loop," UpdateDossier: ",{{'Diabetes','Insuline','2ml',{30,70}},736164,{27,meas...
{#Ref<0.0.0.1151>,{1437...	{{2015,7,20},{9,55,47}}	{eCureExe,loop," Steps: ",[{{'Antibiotica','1mg'},736165,am}]}
{#Ref<0.0.0.1152>,{1437...	{{2015,7,20},{9,55,47}}	{eCureExe,loop," Gekozen stap: ",{{'Antibiotica','1mg'},736165,am}}
{#Ref<0.0.0.1153>,{1437...	{{2015,7,20},{9,55,47}}	{eCureExe,loop," Gekozen inname: ",{notTakenDose,736165,am}}
{#Ref<0.0.0.1154>,{1437...	{{2015,7,20},{9,55,47}}	{eCureExe,loop," Steps: ",[{{'Sugar Food','Sugar Drinks'},736164,sugar}]}
{#Ref<0.0.0.4241>,{1437...	{{2015,7,20},{9,56,19}}	{eCureExe,loop," UpdateDossier: ",{{'Bronchitis','Antibiotica','1mg',1},stop}}
{#Ref<0.0.0.4242>,{1437...	{{2015,7,20},{9,56,19}}	{eCureExe,stop,"Kuur is afgelopen."}
{#Ref<0.0.0.4243>,{1437...	{{2015,7,20},{9,56,19}}	{eCureExe,stop,"eCure module gestopt."}
{#Ref<0.0.0.4244>,{1437...	{{2015,7,20},{9,56,19}}	{eCureExe,stop,"eCureExe module afsluiten."}
{#Ref<0.0.0.4245>,{1437...	{{2015,7,20},{9,56,19}}	{eCureExe,loop," Steps: ",[{{'Meettoestel'},736190,measure}]}
{#Ref<0.0.0.4246>,{1437...	{{2015,7,20},{9,56,19}}	{eCureExe,loop," Gekozen stap: ",{{'Meettoestel'},736190,measure}}
{#Ref<0.0.0.4247>,{1437...	{{2015,7,20},{9,56,19}}	{eCureExe,loop," Gekozen inname: ",{26,736190,measure}}
{#Ref<0.0.0.4250>,{1437...	{{2015,7,20},{9,56,19}}	{eCureExe,loop," UpdateDossier: ",{{'Diabetes','Insuline','2ml',{30,70}},736190,{26,meas...
{#Ref<0.0.0.4251>,{1437...	{{2015,7,20},{9,56,19}}	{eCureExe,loop," Steps: ",[{{'Sugar Food','Sugar Drinks'},736190,sugar}]}
{#Ref<0.0.0.4252>,{1437...	{{2015,7,20},{9,56,19}}	{eCureExe,loop," Gekozen stap: ",{{'Sugar Food','Sugar Drinks'},736190,sugar}}
{#Ref<0.0.0.4253>,{1437...	{{2015,7,20},{9,56,19}}	{eCureExe,loop," Gekozen inname: ",{takenDose,736190,sugar}}
{#Ref<0.0.0.4256>,{1437...	{{2015,7,20},{9,56,20}}	{eCureExe,loop," UpdateDossier: ",{{'Diabetes','Insuline','2ml',{30,70}},736190,sugar}}

Figuur 10: Fase1.2 output van Bronchitis en Diabetes simultaan.

5 Resources

In dit hoofdstuk zullen er verschillende *resources* aan de implementatie worden toegevoegd. Deze *resources* zullen ervoor zorgen dat het *NEU*-protocol de stappen kan evalueren om daarna een mogelijke volgende stap uit te voeren. Zo'n stap zal in de toekomst gekozen worden aan de hand van welke fase de meeste *resources* beschikbaar heeft met een zo klein mogelijk ecologische en economische voetafdruk. In de huidige implementatie wordt er enkel gekeken of de stap uitgevoerd kan worden en wat het resultaat is na uitvoering. De *resources* die in deze fase besproken zullen worden betreffen de patiënt en de verpleegkundige.

Naast de bespreking van de *resources* zal ook de virtualisatie behandeld worden. Er is een basis virtualisatie ontwikkeld welke, voor dit stadium, parallel langs de realistische uitvoering plaatsvindt en verder zal besproken worden in sectie 5.3 welke de patiënt *resource* zal gebruiken om het virtuele pad af te leggen en sectie 5.5 waar de verpleegkundige *resource* mee in wordt betrokken.

De *eCureExe module* is in deze fase lichtjes aangepast ten opzichte van Fase1, zie sectie 4.1. De module zal nu alle stappen in de lijst laten evalueren door de *resources*. Na evaluatie zullen alle stappen worden uitgevoerd en worden geregistreerd bij de *event manager*. Als alle handelingen zijn doorlopen zal de module wachten op *feedback*. Deze *feedback* zal een nieuwe lijst genereren en de *eCureExe* zal deze lijst op zijn beurt laten evalueren tot uiteindelijk de therapie is afgelopen. Hier zal de *Days* variabele bij het loggen zijn nut aantonen.

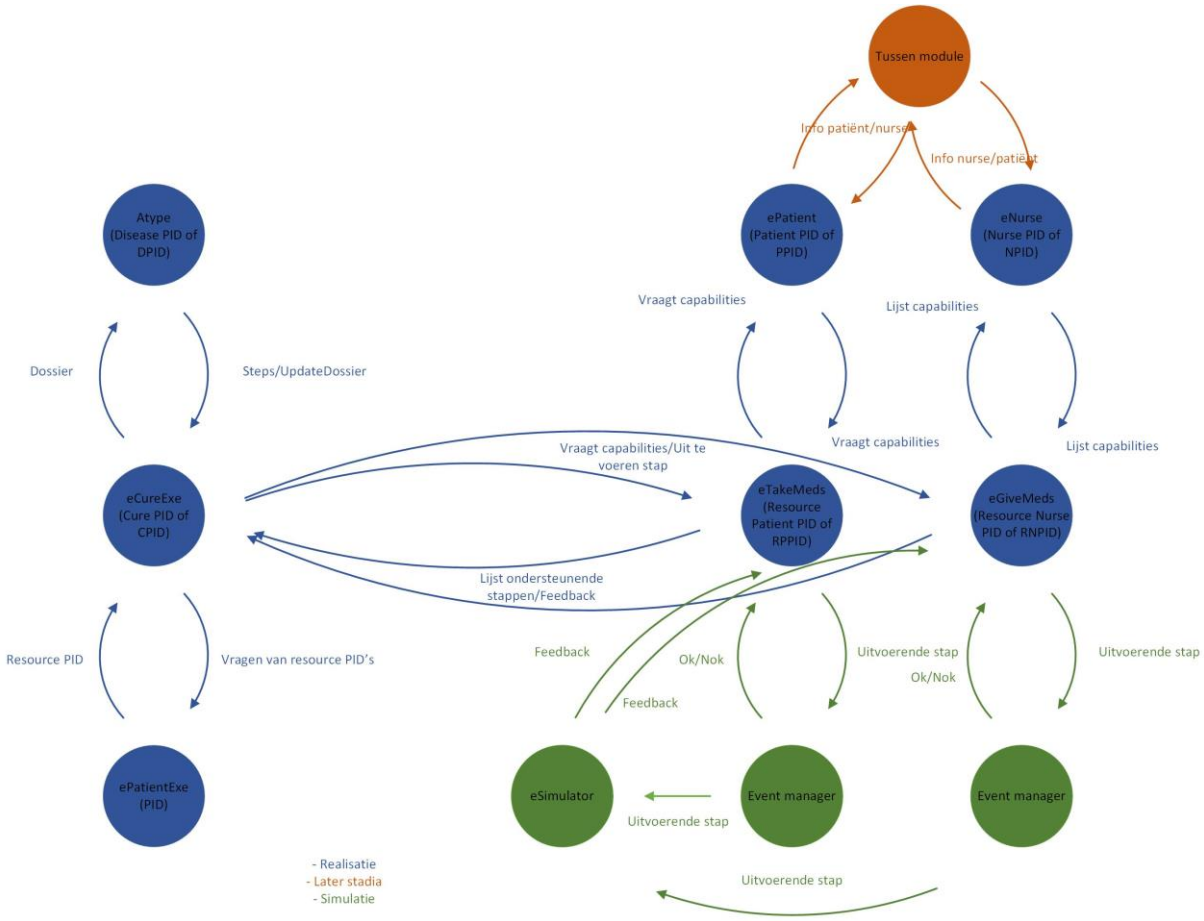
5.1 *Resource* implementatie

Op Figuur 11 is te zien hoe de *resources* in het totaalbeeld passen en hoe iedere module zijn bijhorende PID heeft. De *ePatientExe module* onthoudt alle *PID*'s van de verschillende therapieën en de belangrijkste *resource PID*'s. Er wordt telkens één *ePatientExe module* aangemaakt per patiënt. De belangrijkste *resource PID*'s zijn afkomstig van de patiënt en verpleegkundige. Eenvoudig gezegd kent *ePatientExe* alle *resources* en ziektes van één enkele patiënt maar heeft geen kennis over deze specifieke modules. De *eCureExe module* zal aan de *ePatientExe module* de nodige *PID*'s van de *resources* opvragen voor evaluatie. Als *eCureExe* de *PID*'s heeft dan zal het de communicatie opstarten met de desbetreffende *resource modules*.

De *event manager* registreert alle uit te voeren stappen, zoals een agenda systeem, en zal de patiënt tijdig waarschuwen als er een volgende stap dient uit gevoerd te worden. De *eSimulator module* vervangt tijdelijk de gebruiker en zal aangesproken worden door de *event manager* wanneer het tijd is om een stap uit te voeren. De *eSimulator module* zal dan een *feedback* opstellen aan de hand van de uit te voeren stap en dit doorsturen naar de *eTakeMeds* of *eGiveCare module*. Deze modules versturen op hun beurt de *feedback* verder naar de juiste *eCureExe module* voor verdere afhandeling.

De *event manager*, welke gebruikt worden door de *eTakeMeds* en *eGiveCare*, zijn gelijkaardige modules en zijn apart getekend in Figuur 11 voor meer duidelijkheid te bieden.

In de volgende punten zullen de *resource modules* worden uitgelegd. De 'tussen' module zal in een later stadium geïmplementeerd worden en zal dienen om de beschikbaarheid van zowel thuiszorg als patiënt na te gaan en zo afspraken vast te leggen. Dit wordt in een aparte module geplaatst zodat de modules *ePatient* en *eNurse* geen logica hoeven te bezitten om onderhandelingen tussen deze twee modules te kunnen afhandelen.



Figuur 11: Huidige implementatie met resources en 'tussen module'.

5.2 Patiënt

Deze fase voegt de patiënt *resource* (*ePatient module*) toe aan de applicatie. De *eTakeMeds module* (*resource* instance) zal van de *eCureExe* een lijst met mogelijke handelingen krijgen om na te gaan welke de patiënt zelf kan uitvoeren zonder externe hulp. *eTakeMeds* zal de lijst stap per stap (modulair) overlopen en doorsturen naar de *ePatient module* (*resource type*) om via *pattern matching* de mogelijke uitvoering te testen. Om dit verstaanbaar te maken wordt onderstaand voorbeeld gebruikt:

```
[{{antibiotica, 1mg}, Days, am}, {{'syringe'}, Days, 'blood
sample'}, {{'sphygmomanometer'}, Days, 'blood pressure'}]
```

In de lijst staat dat er antibiotica van 1mg, een bloedstaal en bloeddruk dient genomen te worden. In deze fase zijn er dus stappen die niet uitvoerbaar zijn voor de patiënt zoals het nemen van een bloedstaal en bloeddruk, hiervoor is hulp van een verpleegkundige nodig (sectie 5.4). In tegenstelling tot bovenstaand voorbeeld geeft dit volgende wel stappen weer die de patiënt zelf zou kunnen uitvoeren:

```
[{_Specs, _Days, am}, {_Specs, _Days, pm}]
```

De `_Specs` zorgt ervoor dat er in de *ePatient module* geen rekening dient gehouden te worden met de specificaties. In later verloop, kan dit wel een vereiste zijn. Denk maar aan een inspuitbare medicatie die de patiënt zelf niet mag plaatsen, bijgevolg zijn de specificaties wel nodig om *pattern matching* te kunnen toepassen. De patiënt kan dus enkel, zoals in sectie 4.1, medicatie innemen.

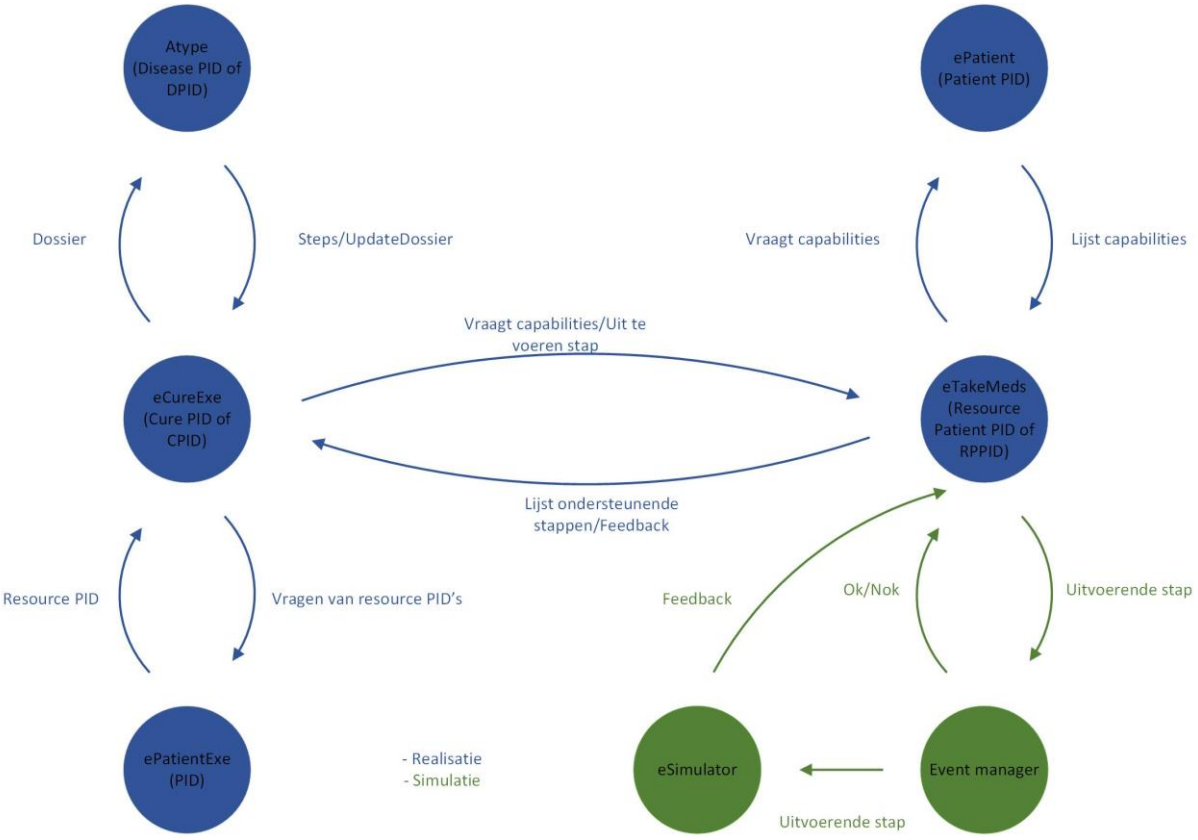
Als de patiënt de stap zelf kan uitvoeren zal de *ePatient module* een `true` naar de *eTakeMeds module* terugsturen, zo niet is dit een `false`. De `_Days` variabele is voor de module niet van belang, deze is enkel relevant bij het innemen en loggen van de stappen.

In de *eTakeMeds module* zal een nieuwe lijst worden opgesteld door middel van de `true` en `false` antwoorden. Als `true` wordt teruggegeven zal de stap in een nieuwe lijst worden geplaatst, waarin alle stappen komen welke de patiënt zelf kan uitvoeren, de uitvoerbare lijst. Als er een `false` wordt geretourneerd zal de stap niet in de uitvoerbare lijst komen te staan, deze stap zal door een andere *resource* worden afgehandeld, zoals besproken in sectie 5.4. Merk op dat de specificaties van de stappen wel van belang zijn voor eventuele verdere evaluatie (niet van toepassing in deze thesis) en/of de uitvoering van de desbetreffende stap. Daarom zal in de uitvoerbare lijst de specificaties blijven staan. De uitvoerbare lijst voor dit voorbeeld ziet er als volgt uit:

```
[{{antibiotica, 1mg}, Days, am}]
```

Nu de patiënt *resource* bekend is met zijn functie zal de implementatie uit de doeken gedaan worden.

De implementatie van een *resource*, zie Figuur 12, bevat een *type* en een *instance* zoals dit het geval is bij het uitvoeren van een ziekte, zie sectie 4.1. Het *type* is de *ePatient module* en heeft een lijst met alle mogelijkheden die de patiënt zal ondersteunen. De *instance* is de *eTakeMeds module* en zal een aanvraag ontvangen voor een controle van de mogelijke stappen, alsook de uit te voeren stap registreren bij de *event manager* waarbij een `ok` wordt teruggezonden als dit is gelukt, bij een foutmelding zal er een niet of *not* ok (`nok`) terug gestuurd worden.



Figuur 12: Fase2 implementatie van het NEU-protocol en een patiënt resource.

De controle van deze mogelijke stappen gebeurt als volgt, zoals te zien is op Figuur 12 en de communicatie in detail in . Op Figuur 14 is de *output* te zien of de patiënt weldegelijk zijn medicatie inneemt. Figuur 15 toont de *debug output* van de stiptheid en zorgvuldigheid van de patiënt. In het verdere verloop van deze thesis zullen de goede voorbeelden niet meer getoond worden, enkel de willekeurig gegenereerde keuzes. Op Figuur 16 is de output te zien wanneer de patiënt een willekeurig gedrag vertoont bij het al dan niet innemen van de medicatie. Figuur 17 toont de *debug output*, van het willekeurige gedrag van de patiënt, van de *eCureExe module* die deze uitleg vergezelt:

- *eCureExe* vraagt een lijst van mogelijke stappen op aan het stappenplan van de ziekte (*activity type*).

  ```
  Steps = eCureNext({{'Bronchitis', 'Antibiotica', '1mg', 28}, Days, not_started}).
  ```

  ➔ `Steps = [{{'Antibiotica, '1mg'}, Days, am}, {{'Antibiotica, '1mg'}, Days, pm}]`

- Daarna zal aan de *ePatientExe module* de *PID* van de patiënt *resource* worden opgevraagd.

  ```
  RPid = ePatientExeGetRPid().
  ```

- Met behulp van deze *PID* kan de *eCureExe module* de lijst doorsturen naar *eTakeMeds* voor de evaluatie.
- De module zal deze lijst element per element doorsturen naar de *ePatient module*.
- De *ePatient module* zal het binnen gekomen *item* in zijn lijst nakijken. Als de patiënt dit zelfstandig kan uitvoeren zal de module een *true* terug geven op dit element, zo niet zal dit een *false* zijn.
- *eTakeMeds* zal alle *items* die een *true* gekregen hebben in een nieuwe lijst plaatsen. In deze fase zullen alle stappen een *true* krijgen en in deze nieuwe lijst geplaatst worden.
- Als alle elementen zijn overlopen zal de *eTakeMeds module* de nieuwe lijst terugsturen naar de *eCureExe module*.

  ```
  ExecutableSteps = [{Specs, Days, am}, {Specs, Days, pm}]
  ```

- *eCureExe* zal de eerste beschikbare stap kiezen.

  ```
  {Specs, Days, am}
  ```

- De gekozen stap zal via *eTakeMeds* naar de *event manager* gestuurd worden. Bij het registreren van de stap zal er nagegaan worden of dit is gelukt. Waanneer dit zo is zal er een `ok` terug gestuurd worden naar de *eCureExe module* zo niet is dit een `nok`.

  ```
  Ok = eventmanagerPost(Ref, From, Step).
  ```

  - `Ref` is de referentie van de therapie.
  - `From` is de *PID* van de desbetreffende ziekte (*CPID*).
  - `Step` is de uit te voeren stap.
- Deze `ok` en `nok` worden gebruikt om de stap te updaten als "*pending*" in het stappenplan. Een `ok` zal de geregistreerde stap uit de algemene lijst met alle volgende stappen verwijderen. Een `nok` zal de stap in de lijst laten staan totdat deze is uitgevoerd.
  - Bij `ok`:

`[{Specs, Days, am}, {Specs, Days, pm}]` ➔ `[{Specs, Days, pm}]`

- Bij `nok`:

  `[{Specs, Days, am}, {Specs, Days, pm}]` ➔ `[{Specs, Days, am}, {Specs, Days, pm}]`

- Zodra alle stappen zijn geregistreerd bij de *event manager* zal de *eCureExe module* in een wachtmodus komen en wachten tot een stap is uitgevoerd.

  `Steps = []`

- Op het moment dat een stap dient uit gevoerd te worden zal de *event manager* de stap doorsturen naar de *eSimulator module* voor het creëren van een *feedback*. Deze *eSimulator module* vervangt weliswaar de gebruikersinteractie tijdens het ontwikkelen.
- *eSimulator* zal de *feedback* sturen naar de *eTakeMeds module* die dit bericht op zijn beurt doorstuurt naar de juiste *eCureExe module*.

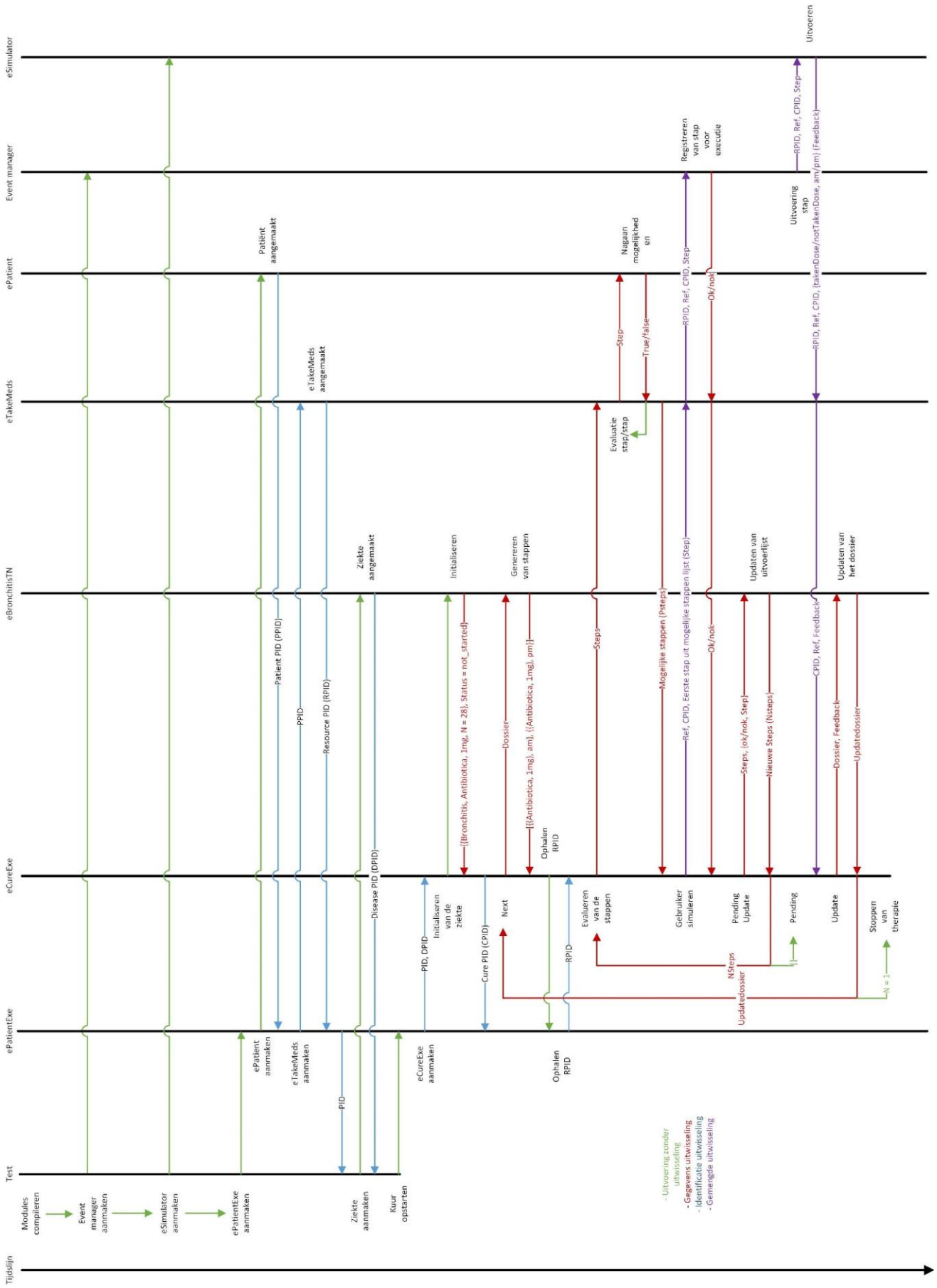
  `Feedback = {takenDose, Days, am}`

  of

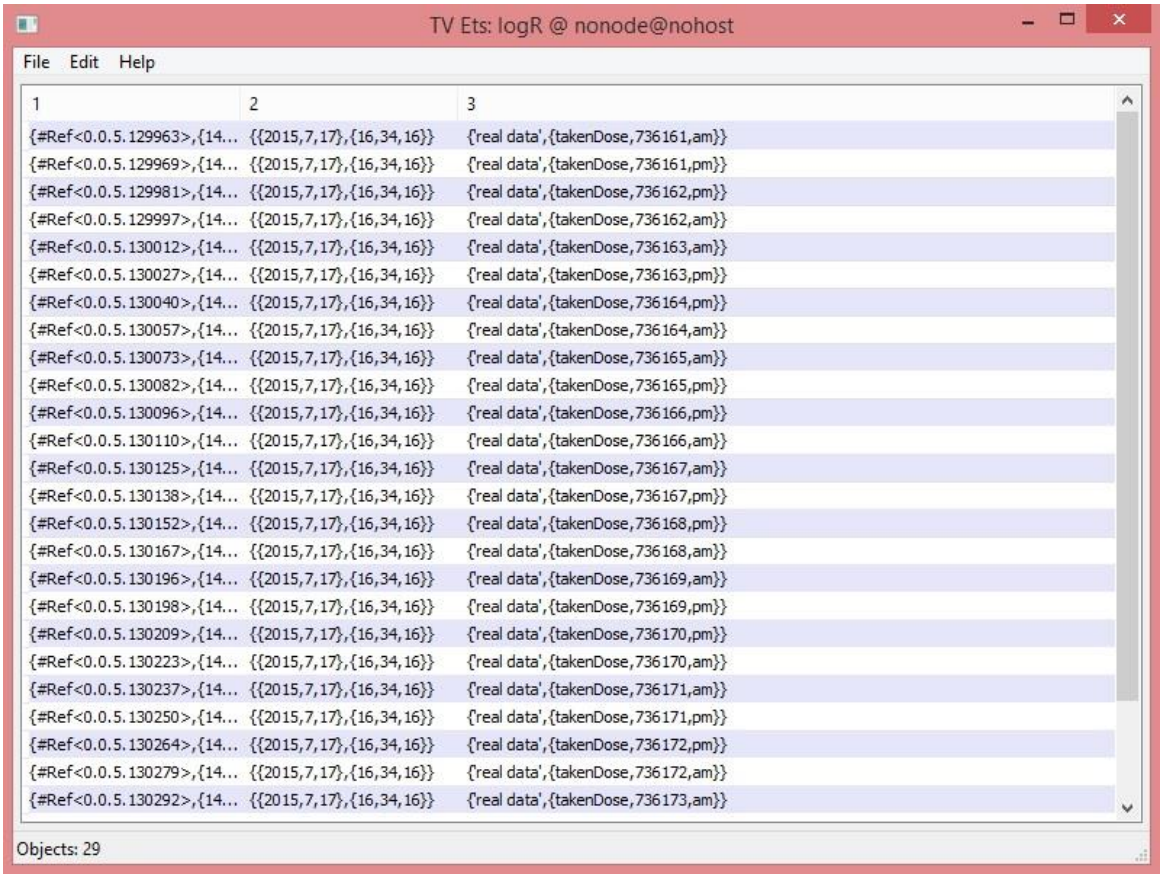
  `Feedback = {notTakenDose, Days, am}`

- De *eCureExe module* zal het dossier updaten en de volgende stappen opvragen. Alles zal opnieuw vanaf het begin overlopen worden. Op deze wijze zullen toekomstige stappen worden behandeld.

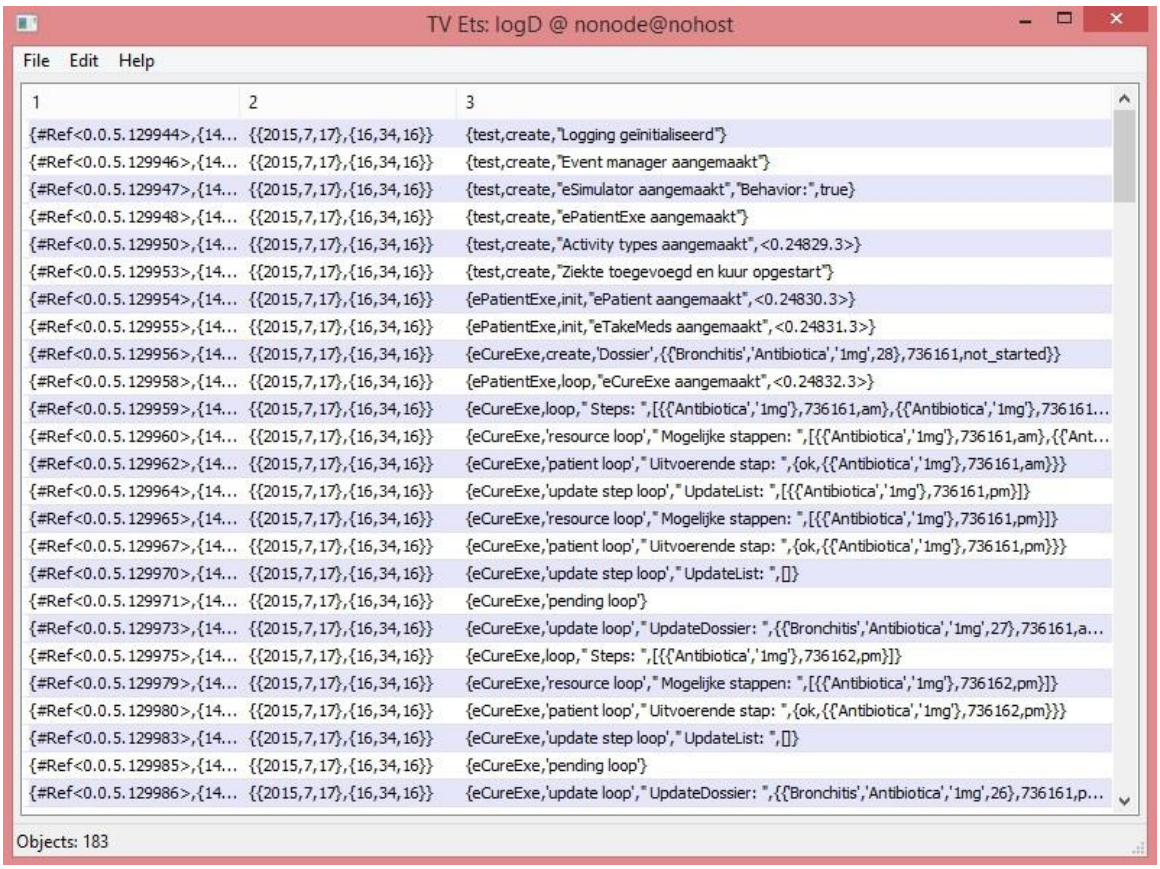
  `Steps = [{Specs, Days + 1, pm}]`



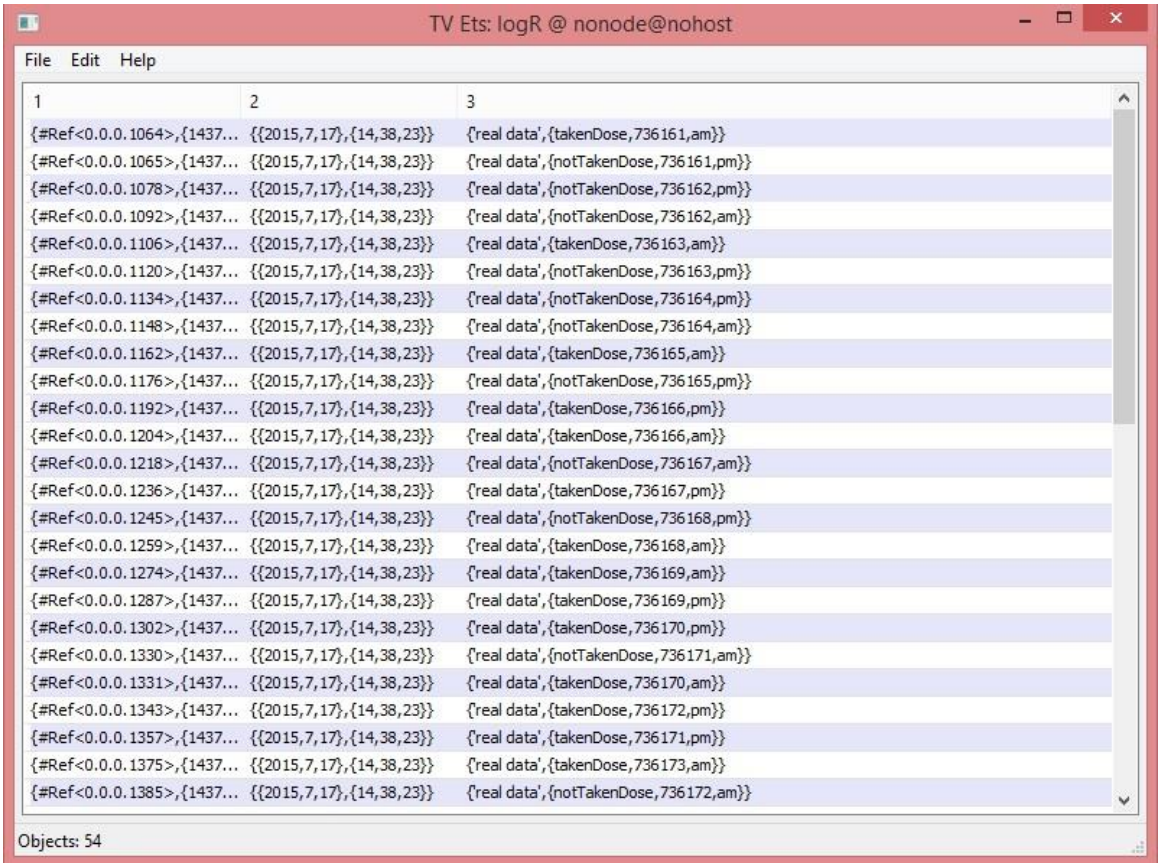
Figuur 13: Fase2 implementatie van het NEU-protocol en een patiënt resource detail.



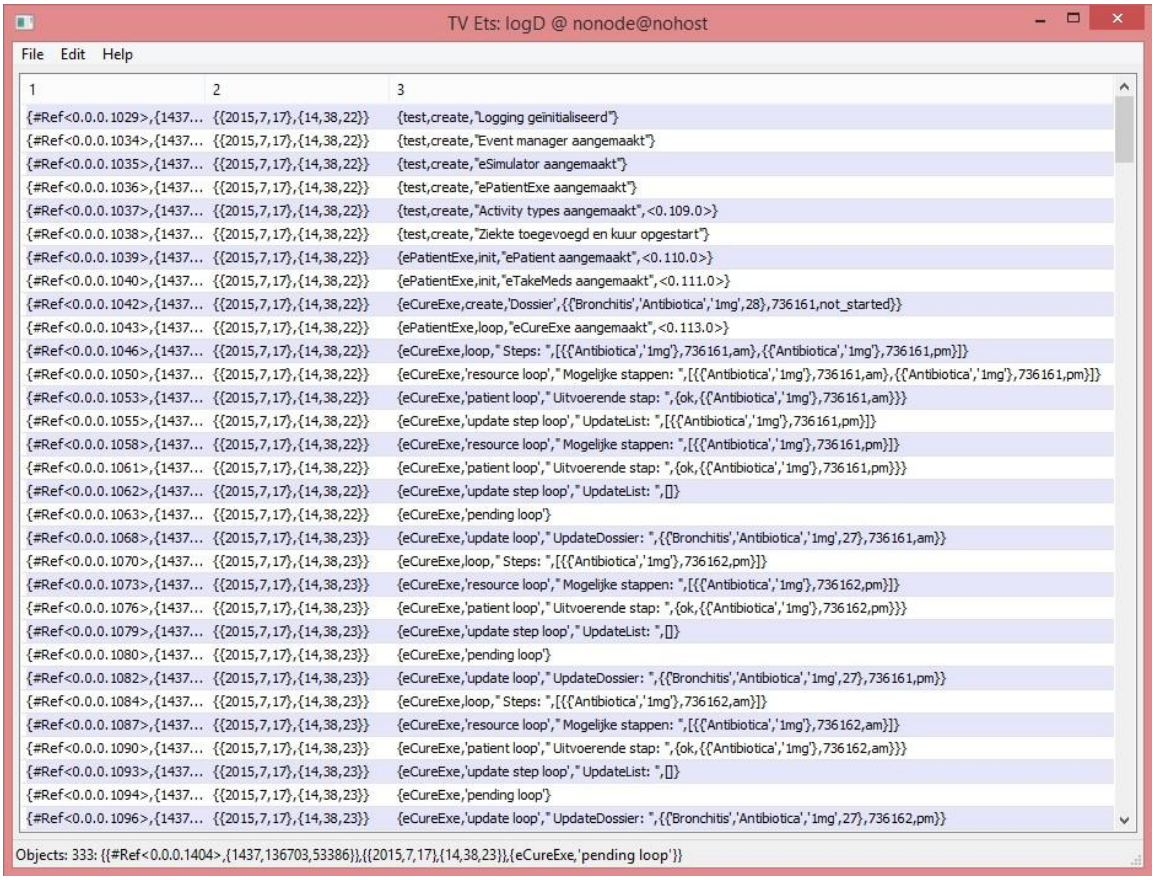
Figuur 14: Fase2 goede inname output.



Figuur 15: Fase2 goede inname debug output.



Figuur 16: Fase2 willekeurig reële output.



Figuur 17: Fase2 willekeurig debug output.

5.3 Virtualisatie met patiënt

Deze fase betreft de basis van de virtuele verkenning in de huidige implementatie, dit is te zien op Figuur 18. De bijgekomen module, in het grijs genaamd *eCureExeV*, heeft een werking gelijkaardig aan deze van de gewone *eCureExe*. De 'V' achteraan de naam staat voor *virtual* of virtueel en dient als aanduiding dat het om de virtuele verkenning gaat.

Er is voor een aparte module gekozen zodat er parallel met de realiteit gewerkt word en niet serieel. Op deze manier heeft de *eCureExe module* meer tijd om vragen te beantwoorden en *feedback* af te handelen, het werk wordt verlicht en de module blijft eenvoudig in code.

De virtuele verkenning vindt plaats bij elke rust van de *eCureExe module*. Het zal voor elke stap een patrouille maken doorheen de verschillende *resources* opzoek naar complicaties waarbij de therapie en/of planning veranderd dient te worden. Op deze manier kan er naar eventuele maatregelingen of alternatieven worden gezocht en kunnen deze worden besproken met de betrokken arts en patiënt. Dit met de bedoeling om in de toekomst ernstige complicaties te vermijden en bezorgde patiënten en omstaanders gerust te stellen. Tevens zullen de kosten dalen vanwege minder consultaties betreffende kleine bezorgdheden, omdat deze via dit systeem op afstand eveneens opgelost kunnen worden. Zo zal de arts ook meer beschikbare tijd krijgen voor andere patiënten.

De huidige versie is een uitlijning van wat hierboven wordt vermeld en er zal enkel parallel langs de realistische uitvoering een virtuele uitvoering plaatsvinden. Zo kan er met behulp van het vergelijken van de logberichten in een later stadium, via experimenten en uitbreidingen, de reeds vermelde toekomstvisie worden bekomen.

Het communicatiedetail van de huidige implementatie waar aangeduid wordt wanneer de virtualisatie gebeurd, is te zien in . toont de communicatiedetail van de *eCureExeV module*. Hieronder wordt kort de werking van *eCureExeV*, omdat dit grotendeels hetzelfde is als *eCureExe*, aangehaald:

- *eCureExe* heeft geen uit te voeren stappen meer en gaat in zijn wachtmodus waar het de *eCureExeV module* oproept te samen met het huidige dossier.

  ```
  eCureExeV:create({PatientPid, DiseasePid}, Dossier)
  ```

- *eCureExeV* gaat te samen met het dossier de volgend mogelijke stappen opvragen aan de ziekte.
- Nadat de stappen zijn opgevraagd zal er aan de *ePatientExe module* de *patient PID* (*PPID*) opgevraagd worden.
- Met de mogelijke stappen en de *PPID* zal de door de patiënt uitvoerbare lijst opgevraagd worden.
- De *eTakeMeds module* zal element per element nagaan of de stappen door de patiënt uitgevoerd kunnen worden. Omdat er enkel maar één *resource* beschikbaar is zullen alle stappen uitgevoerd kunnen worden.
- Als *eTakeMeds* de uitvoerbare lijst opgesteld heeft, wordt deze lijst terug gestuurd naar de *eCureExeV module*.
- Uit deze uitvoerbare lijst zal de eerste stap gekozen worden ter virtuele executie.
- *eCureExeV* zal de gekozen stap doorsturen naar de *eTakeMeds module*.

  ```
  eResourceTypeExecuteV(Pid, Ref, Step)
  ```

- De *eTakeMeds module* zal een `ok` terugsturen zodat *eCureExeV* zijn lijst kan laten updaten door het stappenplan van de ziekte (*activity type*). eTakeMeds zal in plaats van de stap te registreren bij de *event manager* de stap doorgeven aan de *ePatient module*.

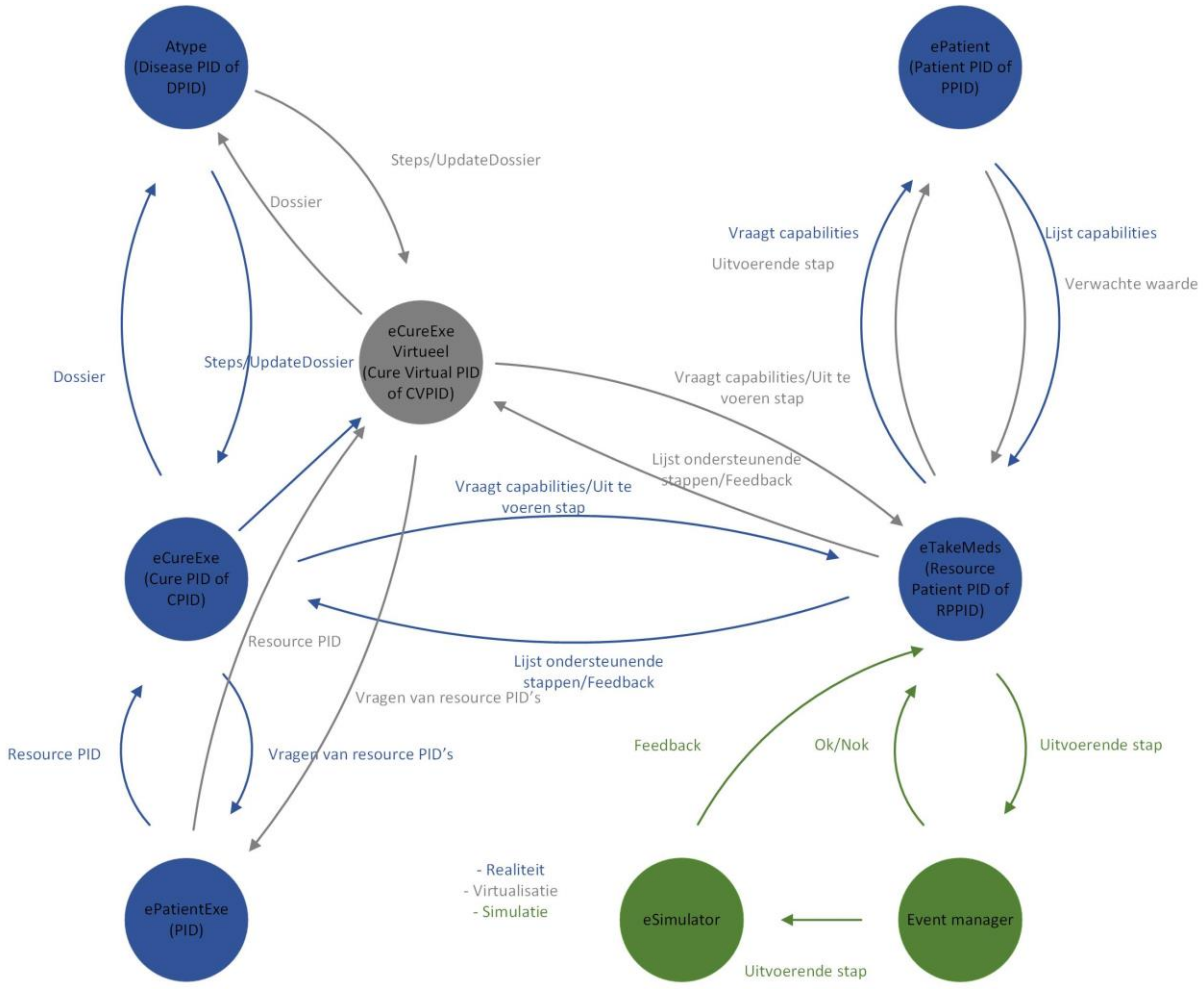
  ```
  Feedback = ePatientExecuteV(PatientPid, Ref, Step)
  ```

- De *ePatient Module* zal, in deze fase, de stap altijd uitvoeren en een *feedback* teruggeven aan de *eTakeMeds module*, de *feedback* zal in deze thesis een goed gedrag simuleren.

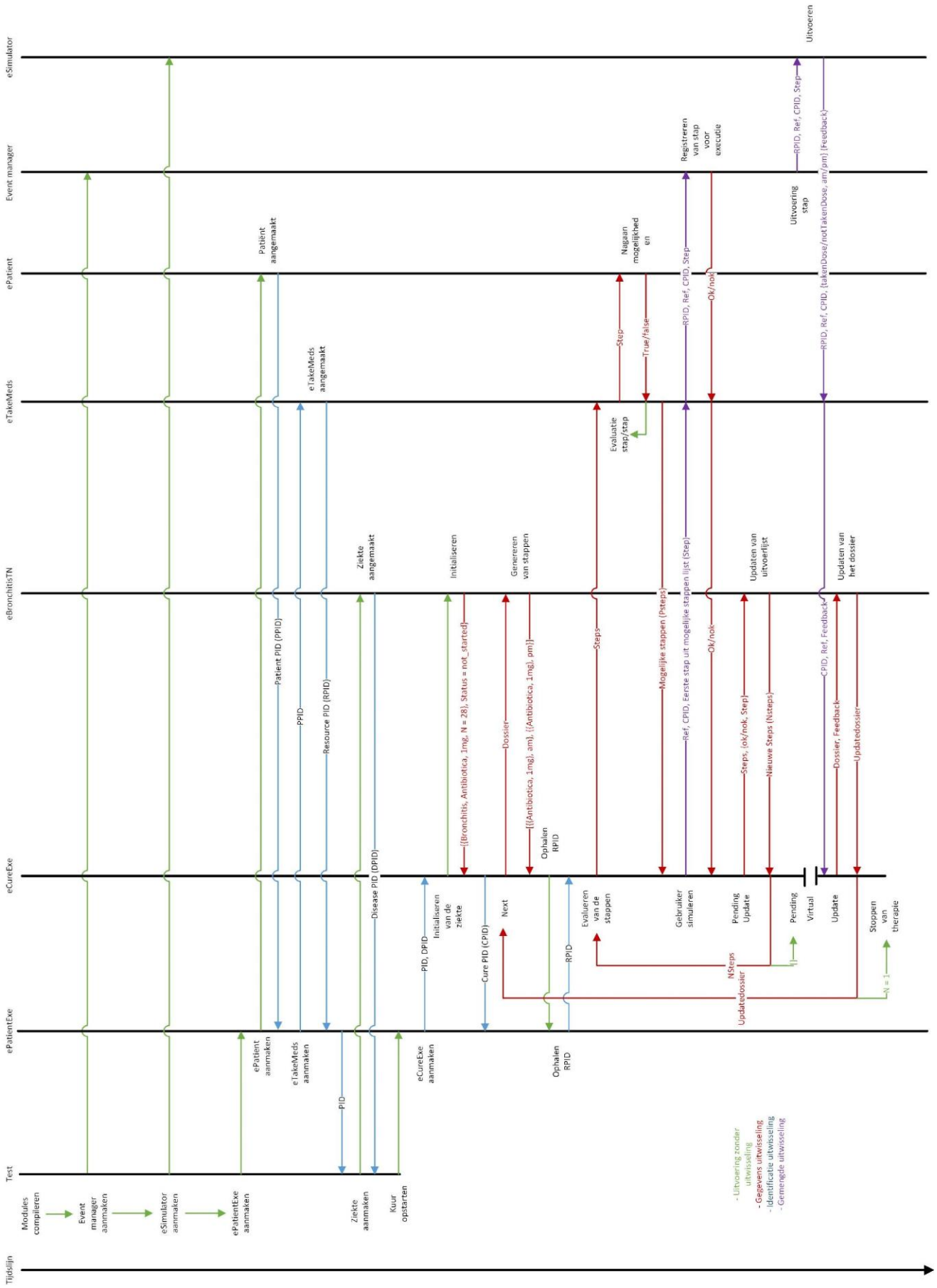
  ```
  {takenDose, Days, Time}
  ```

- *eTakeMeds* zal op zijn beurt de *feedback* doorsturen naar de *eCureExeV module*.
- *eCureExeV* zal daarna het dossier updaten met de virtueel uitgevoerde stap en zijn *feedback*.
- *eCureExeV* zal vervolgens de stappen opnieuw doorlopen totdat de therapie virtueel ten einde is gelopen en zichzelf zal afsluiten.

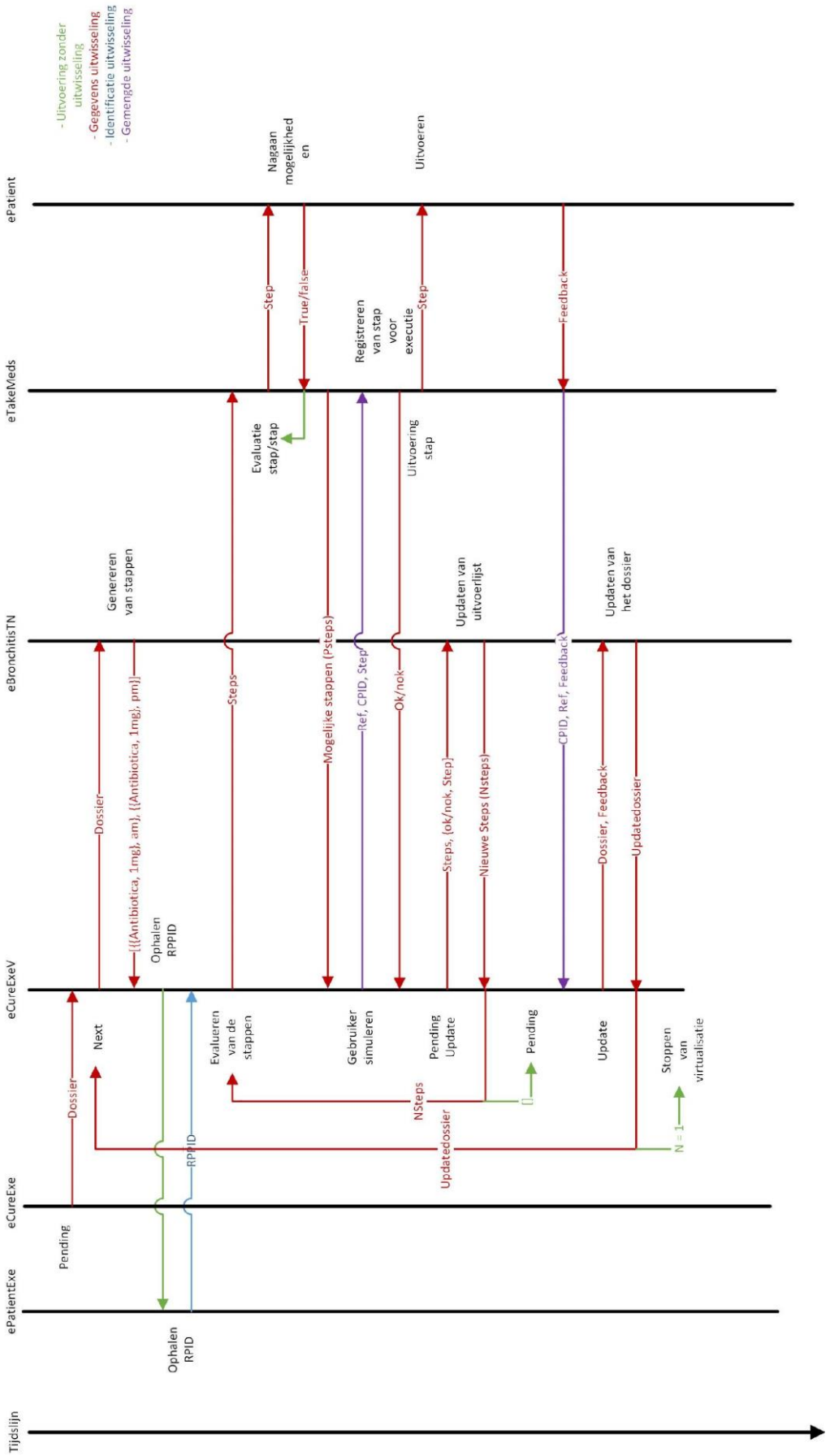
Alle uitgevoerde stappen met de bijhorende *feedback* zal in de logV *ETS* tabel, zie sectie 4.1, worden opgeslagen.



Figuur 18: Implementatie met virtualisatie.



Figuur 19: Implementatie met virtualisatie, realistisch deel detail.



Figuur 20: Implementatie met virtualisatie, virtueel deel detail.

5.4 Verpleegkundige

In deze masterproefthesis is er een tweede *resource* namelijk een verpleegkundige (*eNurse module*). De *resource* heeft als doel taken uit te voeren welke de patiënt niet zelf kan volbrengen. In deze thesis zal de verpleegkundige alle stappen kunnen uitvoeren mits voorschrift van de betrokken medicus. In de praktijk kan dit echter anders zijn, zoals vaardigheden van de verpleegkundige, niet de juiste machines of materialen, etc., maar dit is buiten de scope van deze thesis.

Bij de ziekte Bronchitis is er een extra stap bijgekomen welke de patiënt niet zelf kan uitvoeren maar wel de verpleegkundige, namelijk het meten van de bloeddruk van de patiënt. De bloeddrukmeting vindt plaats de dag nadat de `pm` medicatie is ingenomen. Er volgt geen nieuwe reeks stappen na de meting.

Aan de ziekte Diabetes is er niets aangepast maar de bloedglucosemeting en Insuline injectie worden nu gedaan door de verpleegkundige en niet meer door de patiënt zelf. De patiënt zal wel nog de stappen `nothing` (niets doen omdat de bloedglucose binnen de grenzen ligt) en `sugar` (suikerhoudende producten nuttigen) uitvoeren.

De volledige implementatie is te zien in Figuur 21, de communicatie in detail in .

De opbouw en werking van de *eNurse module* is gelijkaardig aan die van de *ePatient*. Er is dus een *resource instance* gelijkaardig aan de *eTakeMeds module* namelijk de *eGiveCare module*. De keuze om een andere *resource instance* te gebruiken is zodat er geen verwarring kan ontstaan tussen eigen en de externe zorg. De *header eResourceType* blijft voor beide *instances* behouden en onaangepast.

De aanpassing in de *ePatientExe module* omvat de aanmaak van een tweede *resource*, naast de patiënt *resource*, en onthoudt twee *PID*'s (patiënt en verpleegkundige) in een lijst van *tuples*:

```
[{patient, PatientPID (PPID)}, {nurse, NursePID (NPID)}]
```

De aanpassing bij de *eCureExe module* is dat van zodra de uitvoerbare lijst van de patiënt leeg is, maar de stappen nog niet allemaal uitgevoerd zijn, er een verpleegkundige wordt ingeschakeld. Als de lijst volledig leeg is zal *eCureExe* in zijn wachtmodus gaan. Hieronder zal de werking uitgelegd worden (Figuur 23), de stappen die hetzelfde blijven worden enkel kort aangehaald:

- Haal de volgende stappen op bij het stappenplan.

  ```
  Steps = [{Specs, Days, am}, {Specs, Days, pm},
  {{'sphygmomanometer'}, Days, 'blood pressure'}]
  ```

- Haal de patiënt *PID* op bij de *ePatientExe module*.

  ```
  {patient, PPID}
  ```

- Ga na welke stappen de patiënt zelf kan uitvoeren.

  ```
  PatientExecutableList = [{Specs, Days, am}, {Specs, Days, pm}]
  ```

- Voer zo één stap uit en vang de plaatsing *feedback* op. Dit is een `ok` of `nok` (niet ok of *not ok*) en wordt terug gegeven door de *eTakeMeds module* als de uitvoerende stap is geplaatst in de *event manager*. Een `ok` als de plaatsing gelukt is en bij een foutmelding een `nok`.
- Update het dossier dat een stap in uitvoering is. Bij een `ok` zal deze stap worden verwijderd uit de algemene lijst. Anders blijft deze in de lijst staan totdat dit is geplaatst. Bij een `nok` zal *eCureExe* blijven proberen om deze stap uit te laten voeren door de patiënt. In een later

stadium kan ervoor gezorgd worden dat bij het herhaaldelijk uitblijven van de uitvoering van een bepaalde stap door de patiënt, de verpleegkundige deze taak zal toegewezen krijgen.

De *eCureExe module* zal deze stappen blijven herhalen tot alle fases die door de patiënt uitgevoerd kunnen worden geplaatst zijn. Van zodra de stappenlijst voor de patiënt is overlopen en alle stappen zijn geregistreerd, bij de *event manager*, zal de module de volgende acties ondernemen:

- Haalt de overgebleven volgende stappen op bij het stappenplan.

  ```
  Steps = [{{'sphygmomanometer'}, Days, 'blood pressure'}]
  ```

- Haalt de patiënt *PID* op bij de *ePatientExe module*.

  ```
  {patient, PPID}
  ```

- Gaat na welke stappen de patiënt zelf kan uitvoeren. Hier zullen geen stappen meer volgen en zal er dadelijk worden overgegaan tot de volgende stap.

  ```
  PatientExecutableSteps = []
  ```

- Haalt de verpleegkundige *PID* op bij de *ePatientExe module*.

  ```
  {nurse, NPID}
  ```

- Gaat na welke stappen de verpleegkundige kan uitvoeren, dit zullen alle stappen zijn.

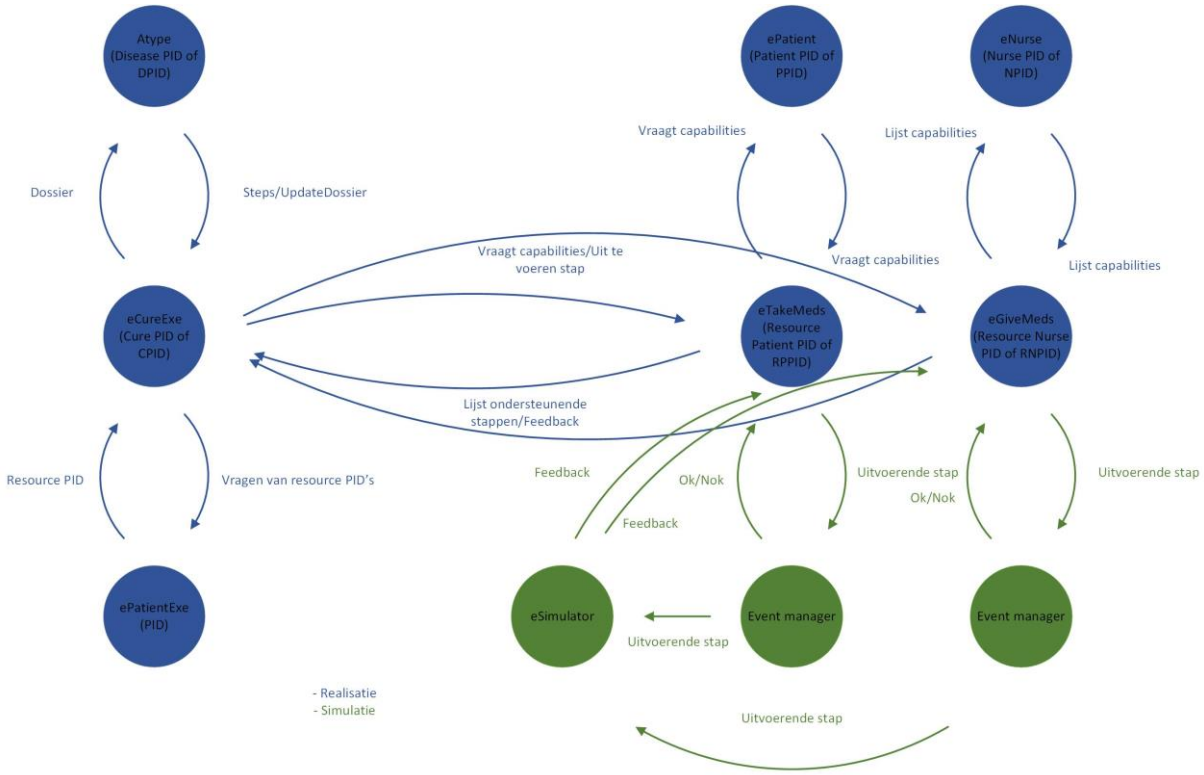
  ```
  NurseExecutableList = [{{'sphygmomanometer'}, Days, 'blood
  pressure'}]
  ```

- Voert zo één stap uit en vangt de plaatsing *feedback* op.

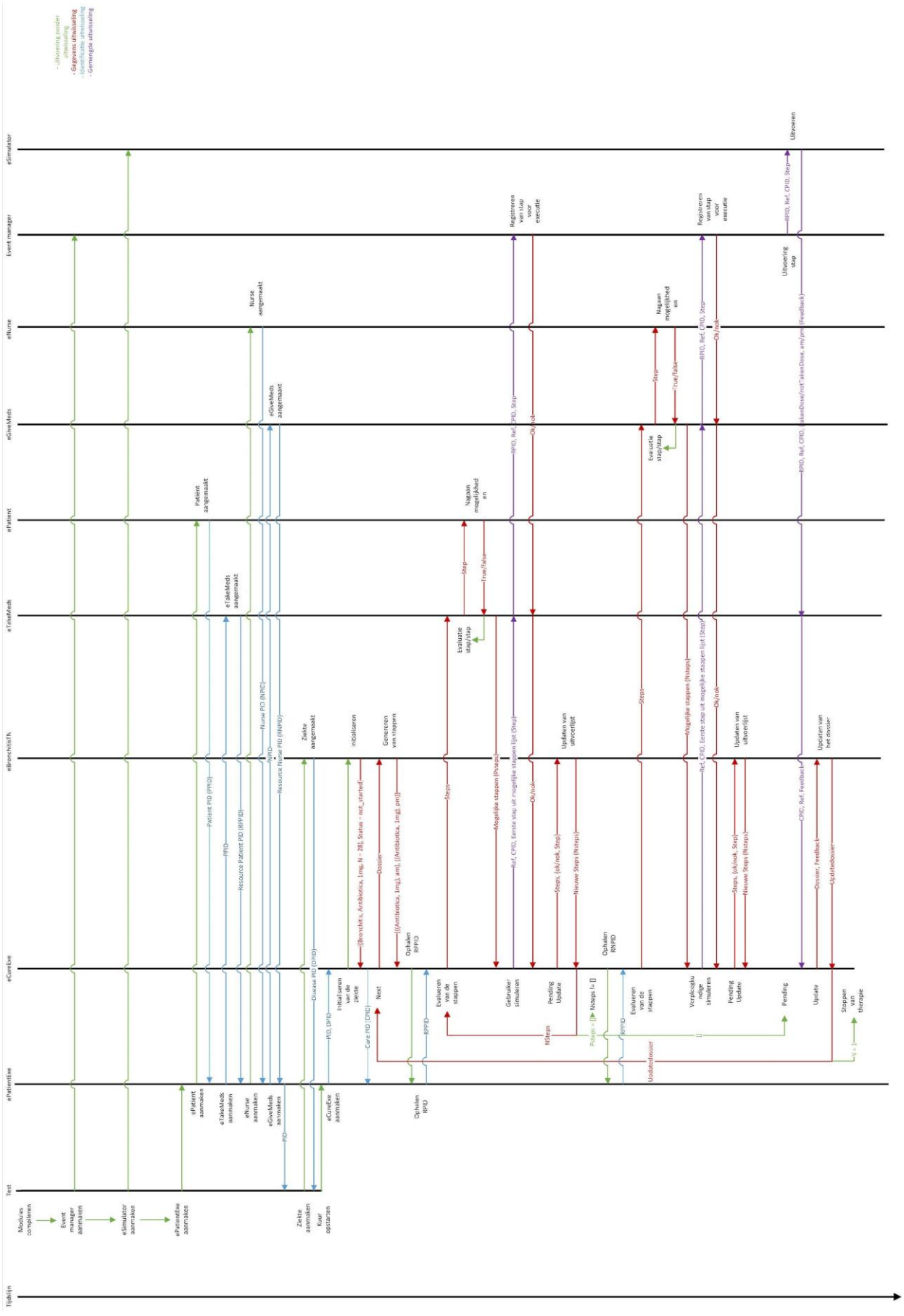
  ```
  Ok = eResourceTypeExecute(NPID, Ref, Step)
  ```

  - `NPID` is de *PID* van de verpleegkundige *resource*.
  - `Ref` is de referentie van de therapie.
  - `Step` is de uit te voeren stap.
- Update het dossier dat een stap in uitvoering is.

Bovenstaand proces zal volledig opnieuw worden doorgenomen totdat de lijst volledig leeg is. Bij een lege lijst zal de *eCureExe module* in een wachtmodus geplaatst worden. In deze modus zal er, zoals reeds aangehaald in de vorige hoofdstukken, gewacht worden tot een uitvoering is afgelopen. Met de *feedback* zal het dossier worden geüpdatet en de volgende stappenlijst worden opgevraagd waarna alles van vooraf aan begint.



Figuur 21: Huidige implementatie met verpleegkundige resource.



Figuur 22: Implementatie met verpleegkunde module detail.

1	2	3
#Ref<0.0.7.182963>	{{2015,7,20},{14,25,37}}	{eCureExe,'update loop',{notTakenDose,736164,pm}}
#Ref<0.0.7.182965>	{{2015,7,20},{14,25,37}}	{eCureExe,'update loop'," UpdateDossier: ",{{'Bronchitis','Antibiotica','1mg',27},736164,pm}}
#Ref<0.0.7.182968>	{{2015,7,20},{14,25,37}}	{eCureExe,loop," Steps: ",[{{'Antibiotica','1mg'},736165,am},{{sphygmomanometer},736165,'blood pressure'}]}
#Ref<0.0.7.182971>	{{2015,7,20},{14,25,37}}	{eCureExe,'resource loop'," Mogelijke stappen: ",[{{'Antibiotica','1mg'},736165,am}]}
#Ref<0.0.7.182975>	{{2015,7,20},{14,25,37}}	{eCureExe,'update step loop'," UpdateList: ",[{{sphygmomanometer},736165,'blood pressure'}]}
#Ref<0.0.7.182979>	{{2015,7,20},{14,25,37}}	{eCureExe,'resource loop'," Mogelijke stappen: ",[]}
#Ref<0.0.7.182983>	{{2015,7,20},{14,25,37}}	{eCureExe,'nurse loop'," Mogelijke stappen ",[{{sphygmomanometer},736165,'blood pressure'}]}
#Ref<0.0.7.182987>	{{2015,7,20},{14,25,37}}	{eCureExe,'update step loop'," UpdateList: ",[]}
#Ref<0.0.7.182988>	{{2015,7,20},{14,25,37}}	{eCureExe,'update loop',{notTakenDose,736165,pm}}
#Ref<0.0.7.182990>	{{2015,7,20},{14,25,37}}	{eCureExe,'update loop'," UpdateDossier: ",{{'Bronchitis','Antibiotica','1mg',27},736165,pm}}
#Ref<0.0.7.182992>	{{2015,7,20},{14,25,37}}	{eCureExe,loop," Steps: ",[{{'Antibiotica','1mg'},736166,am},{{sphygmomanometer},736166,'blood pressure'}]}
#Ref<0.0.7.182997>	{{2015,7,20},{14,25,37}}	{eCureExe,'resource loop'," Mogelijke stappen: ",[{{'Antibiotica','1mg'},736166,am}]}
#Ref<0.0.7.183001>	{{2015,7,20},{14,25,37}}	{eCureExe,'update step loop'," UpdateList: ",[{{sphygmomanometer},736166,'blood pressure'}]}
#Ref<0.0.7.183006>	{{2015,7,20},{14,25,37}}	{eCureExe,'resource loop'," Mogelijke stappen: ",[]}
#Ref<0.0.7.183009>	{{2015,7,20},{14,25,37}}	{eCureExe,'nurse loop'," Mogelijke stappen ",[{{sphygmomanometer},736166,'blood pressure'}]}
#Ref<0.0.7.183013>	{{2015,7,20},{14,25,37}}	{eCureExe,'update step loop'," UpdateList: ",[]}

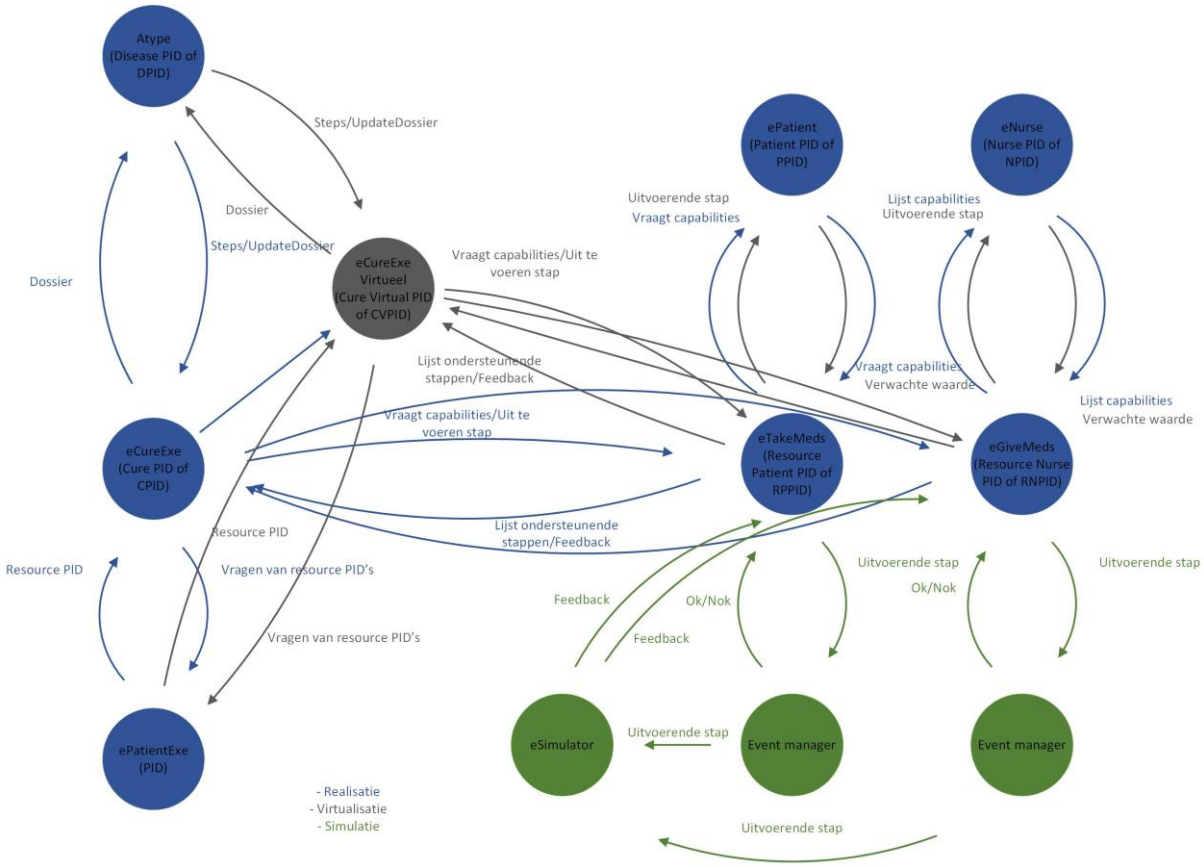
Figuur 23: Fase3 output debug.

5.5 Virtualisatie met verpleegkundige

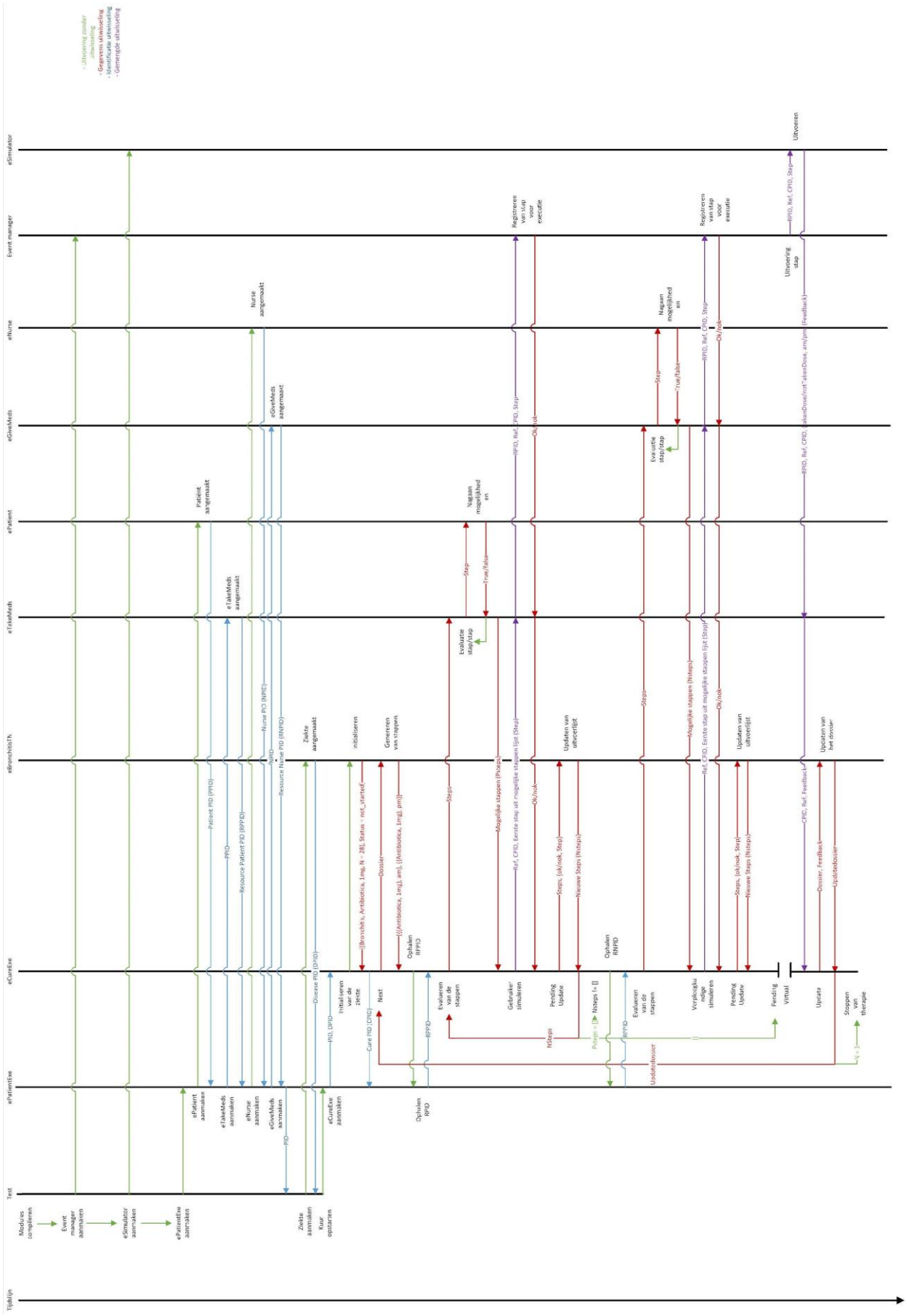
In deze fase wordt de virtuele verkenning uitgebreid met de verpleegkundige uit sectie 5.4. De implementatie is te zien in Figuur 24, de huidige communicatie in detail in waar de start van de virtualisatie wordt aangeduid. In staat de communicatie in detail van de virtuele verkenning zelf.

De werking voor *eCureExeV* is geheel hetzelfde als *eCureExe* in sectie 5.4. De lichte aanpassing voor de *eGiveCare module* is dat er een virtuele registratie bijkomt en dat de *eNurse module* deze registratie opvangt om het daarna virtueel uit te voeren en *feedback* te geven. De stap wordt, in deze fase, altijd uitgevoerd.

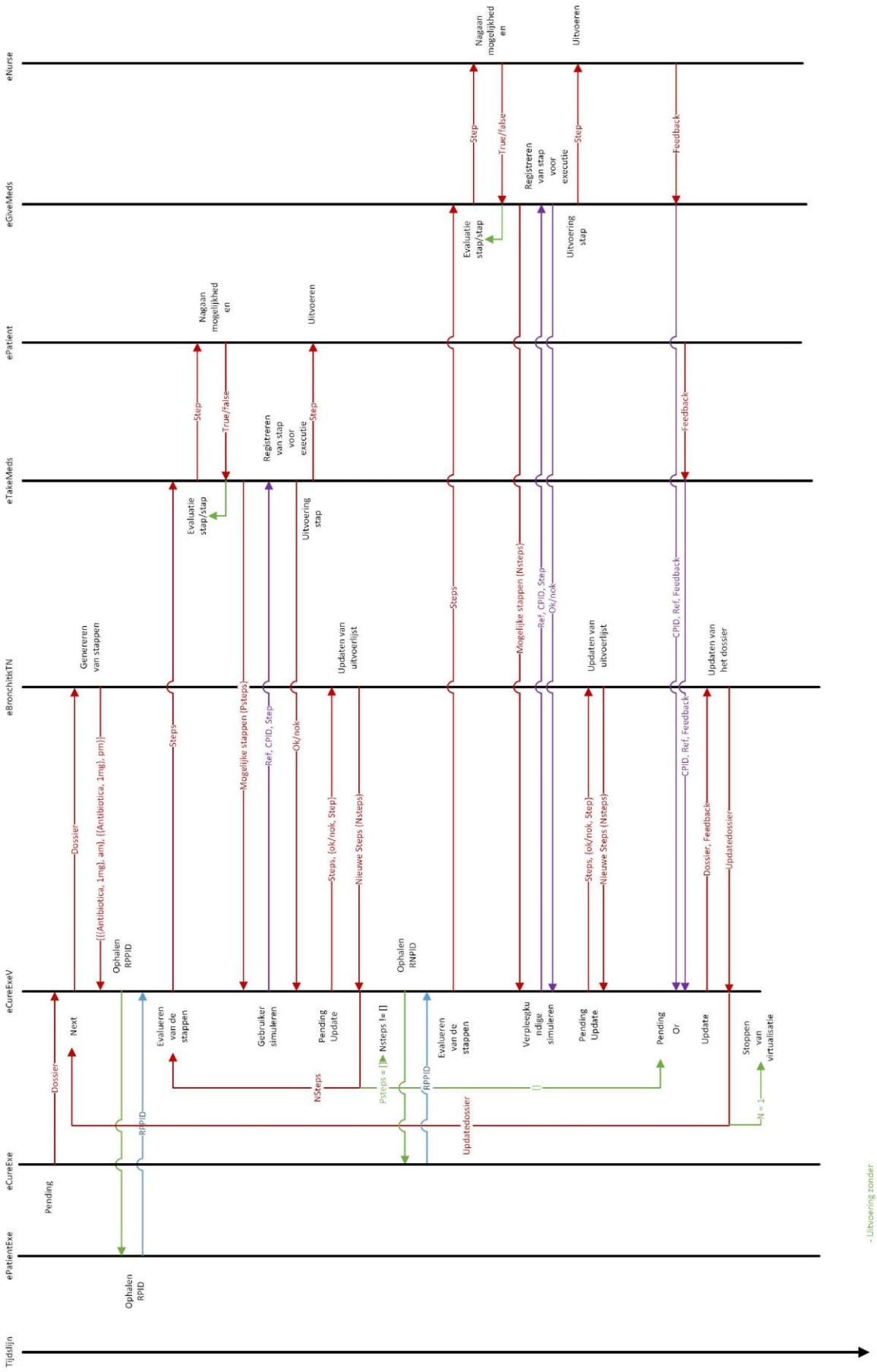
De werking wordt niet meer aangehaald, er wordt hiervoor doorverwezen naar de secties en 5.4 en de Figuur 27 voor de virtuele waarden.



Figuur 24: Huidige implementatie met virtualisatie en verpleegkundige resource.



Figuur 25: Virtualisatie met een verpleegkundige resource, realistisch deel detail.



Figuur 26: Virtualisatie met een verpleegkundige resource, virtueel deel detail.

1	2	3
{#Ref<0.0.10.220723>,{1...	{{2015,7,20},{15,48,22}}	{eCureExeV,create,'Dossier',{{'Bronchitis','Antibiotica','1mg',28},736164,not_started}}
{#Ref<0.0.10.220724>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736164,am}}
{#Ref<0.0.10.220725>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736164,pm}}
{#Ref<0.0.10.220726>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736165,pm}}
{#Ref<0.0.10.220727>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736165,am}}
{#Ref<0.0.10.220728>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual nurse data',{45,736165,'blood pressure'}}
{#Ref<0.0.10.220729>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736166,am}}
{#Ref<0.0.10.220730>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual nurse data',{128,736166,'blood pressure'}}
{#Ref<0.0.10.220731>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736166,pm}}
{#Ref<0.0.10.220732>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736167,pm}}
{#Ref<0.0.10.220733>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736167,am}}
{#Ref<0.0.10.220734>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual nurse data',{49,736167,'blood pressure'}}
{#Ref<0.0.10.220735>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736168,am}}
{#Ref<0.0.10.220736>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual nurse data',{196,736168,'blood pressure'}}
{#Ref<0.0.10.220737>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736168,pm}}
{#Ref<0.0.10.220738>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736169,pm}}
{#Ref<0.0.10.220739>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736169,am}}
{#Ref<0.0.10.220740>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual nurse data',{133,736169,'blood pressure'}}
{#Ref<0.0.10.220741>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736170,am}}
{#Ref<0.0.10.220742>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual nurse data',{198,736170,'blood pressure'}}
{#Ref<0.0.10.220743>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736170,pm}}
{#Ref<0.0.10.220744>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736171,pm}}
{#Ref<0.0.10.220745>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736171,am}}
{#Ref<0.0.10.220746>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual nurse data',{125,736171,'blood pressure'}}
{#Ref<0.0.10.220747>,{1...	{{2015,7,20},{15,48,22}}	{#Ref<0.0.10.220391>,'virtual patient data',{takenDose,736172,am}}

Figuur 27: Fase3.1 output virtueel.

6 Conclusie

Deze masterproef is een verkenning geweest omtrent de werking van het *NEU*-protocol en de implementatie ervan in een *ICT*-platform voor de zorgsector. Het *NEU*-protocol is zeer flexibel en kan overal, niet enkel in de zorgsector, gebruikt worden. Ook het feit dat dit protocol sterk overeenstemt met de realiteit speelde een belangrijke factor in de keuze binnen deze applicatie, een klein voorbeeld om de gelijkenis aan te tonen:

> In alle situaties, zie hoofdstuk 3, bekijkt men eerst alle mogelijke stappen per optie, om daarna ze één voor één te bestuderen en met elkaar te overwegen (Next). Eenmaal dit is gebeurd zal de meest geschikte optie gekozen en uitgevoerd worden (Execute). Na het uitvoeren zal men reflecteren over de gekozen optie en zal men rekening houden met het resultaat voor dezelfde situatie in de toekomst (Update).

Om bovenstaande realiteitsbeeld voort te trekken moet de applicatie zo worden ontwikkeld dat er eenvoudig en snel uitbreidingen kunnen worden toegevoegd. Om deze reden wordt er gebruikt gemaakt van *header* bestanden voor de uiteindelijke functieoproepen waardoor de nieuwe of eigenlijke modules niet moeten worden aangepast.

Uiteindelijk werd er met modulair programmeren een basisapplicatie ontwikkeld waarmee men, in latere stadia, eenvoudig en snel kan experimenteren alsook uitbreiden. Bij elke genomen stap werd er een kopie gemaakt zodat men samen met de teksten in de thesis eenvoudig het protocol kan begrijpen en gebruiken, met andere woorden deze stappen vormen een *tutorial*.

De virtuele verkenning begeeft zich in de beginfase wat in latere stadia, via experimenten met het zoeken van alternatieve wegen bij (on)voorziene complicaties (bv.: operatie, arm gebroken, etc.), verder uitgebreid kan worden tot een volwaardig generisch systeem welk gepast kan reageren op elke situatie binnen de door de arts opgegeven grenzen of in samenspraak met hem. Dit zal op zijn beurt bijdragen aan de medisch digitaal persoonlijke assistent voor alle eindgebruikers. Deze assistent zal automatisch taken verlichten en afspraken regelen zodat het voor elke betrokken partij het goed uitkomt en zo zal zorgen dat de kosten zullen dalen.

Bibliografie

[1] „Waarom studeren aan UC Leuven-Limburg?," UC Leuven-Limburg, [Online]. Available: http://www.ucll.be/waarom-studeren-aan-de-uc-leuven-limburg/. [Geopend 25 maart 2015].

[2] „KIC - Health Caring IT," KHLeuven, [Online]. Available: http://ehealth.khleuven.be/. [Geopend 22 oktober 2014].

[3] L. P. Fumagalli, G. Radaelli, E. Lettieri, P. Bertele en C. Masella, „Patient Empowerment and its neighbours: Clarifying the boundaries and their mutual relationships," Elsevier Inc., 05 november 2014. [Online]. Available: http://www.healthpolicyjrnl.com/article/S0168-8510%2814%2900281-4/abstract. [Geopend 13 juli 2015].

[4] „PALANTE," Europese commissie, [Online]. Available: https://www.palante-project.eu/. [Geopend 10 mei 2015].

[5] M. Duman, Patient Information Forum, 2014.

[6] „Erlang-verdeling," 13 mei 2014. [Online]. Available: http://nl.wikipedia.org/wiki/Erlang-verdeling. [Geopend 06 augustus 2014].

[7] „Agner Erlang," Wikipedia (nl), 8 maart 2013. [Online]. Available: http://nl.wikipedia.org/wiki/Agner_Erlang. [Geopend september 2014].

[8] „Technology," Maximonster, [Online]. Available: http://maximonster.com/en/page/352/technology. [Geopend 2015].

[9] „Erlang (programmeertaal)," 30 juli 2014. [Online]. Available: http://nl.wikipedia.org/wiki/Erlang_(programmeertaal). [Geopend 06 augustus 2014].

[10] „Build massively scalable soft real-time systems," [Online]. Available: http://www.erlang.org/. [Geopend 06 augustus 2014].

[11] „Erlang (programming language)," Wikipedia (en), 20 maart 2015. [Online]. Available: http://en.wikipedia.org/wiki/Erlang_(programming_language). [Geopend 29 maart 2015].

[12] „Where is Erlang used and why?," StackOverflow, 2009. [Online]. Available: http://stackoverflow.com/questions/1636455/where-is-erlang-used-and-why. [Geopend 19 maart 2015].

[13] „What are the advantages of Erlang over other programming languages?," Quora (blogsite), 2009. [Online]. Available: http://www.quora.com/What-are-the-advantages-of-Erlang-over-other-programming-languages. [Geopend 29 maart 2015].

[14] E. Miller, „Why I Program In Erlang," evanmiller.org, 20 oktober 2012. [Online]. Available: http://www.evanmiller.org/why-i-program-in-erlang.html. [Geopend 29 maart 2015].

[15] F. Hébert, „Learn Yous Some Erlang for Great Good," [Online]. Available: http://learnyousomeerlang.com/. [Geopend 2014].

[16] „StackOverflow," StackExchange, [Online]. Available: http://stackoverflow.com/.

[17] P. Valckenaers en H. V. Brussel, Design for the Unexpected, 1st red., 2015, p. 226.

[18] K. Casteels, „Kinderdiabetologie," [Online]. Available: http://www.kuleuven.be/uzschool/download/studiedag/casteels_tekst.pdf. [Geopend 7 mei 2015].

[19] „Yaws," [Online]. Available: http://yaws.hyber.org. [Geopend 04 augustus 2014].

[20] L. Hoguin, „99s," Nine Nines, 2012. [Online]. Available: http://ninenines.eu. [Geopend 04 augustus 2014].

[21] „Simple Network Management Protocol," Wikipedia (nl), 09 januari 2015. [Online]. Available: http://nl.wikipedia.org/wiki/Simple_Network_Management_Protocol. [Geopend 03 april 2015].

[22] „Erlang basics," [Online]. Available: http://wiki.blender.org/index.php/User:Ansimionescu/Notes/7. [Geopend 29 maart 2015].

[23] „Open Telecom Platform (OTP)," Erlang, [Online]. Available: http://www.erlang.org/documentation/doc-5.0.1/pdf/. [Geopend 15 juli 2015].

[24] „Releases," 1997. [Online]. Available: http://www.erlang.org/doc/design_principles/release_structure.html. [Geopend september 2014].

[25] „Rebar," Basho, [Online]. Available: https://github.com/basho/rebar. [Geopend 30 maart 2015].

[26] A. Castro, „Erlang App. Management with Rebar," 01 mei 2010. [Online]. Available: http://alancastro.org/2010/05/01/erlang-application-management-with-rebar.html. [Geopend 20 februari 2015].

[27] R. Jones, „Erlang rebar tutorial: generating releases and upgrades," Metabrew, 26 maart 2011. [Online]. Available: http://www.metabrew.com/article/erlang-rebar-tutorial-generating-releases-upgrades.

[28] „Build Erlang releases with erlang.mk and relx," ninenines, [Online]. Available: http://ninenines.eu/articles/erlang.mk-and-relx/. [Geopend 17 09 2014].

[29] L. Rutten, „Intro Erlang: Een applicatie bouwen met Erlang.mk en Relx," UCLL, 2014. [Online]. Available: http://eaict.technologiecampusdiepenbeek.be/~lrutten/cursussen/tagp/intro-erlang.html#een-applicatie-bouwen-met-erlang.mk-en-relx. [Geopend 17 09 2014].

Bijlagen

A Een release maken

Om het project te consolideren en te kunnen distribueren moet het gecompileerd en uitvoerbaar worden gemaakt. Er zijn drie mogelijkheden om dit te doen namelijk: de basis handelwijze *OTP release*, de twee nakomende omgangsvormen en veel eenvoudiger in gebruik namelijk erlang.mk met relx en rebar.

Bij het compileren van Erlang bestanden ontstaan er *BEAM* bestanden. Een *BEAM* bestand is een *bytecode* versie van de module, zoals dit gebeurt bij Java, en wordt geïnterpreteerd door de Erlang *Virtual Machine* (VM).

A.1 OTP release

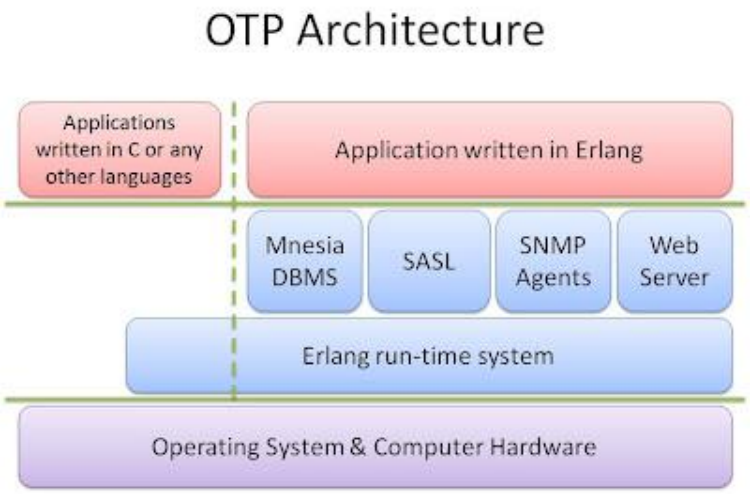
Een *Open Telecom Platform* (*OTP*) *release* is de oudste en meest omslachtigste manier van werken. Men verwijst hiernaar vermits dit de basis is en nog steeds gebruikt wordt door erlang.mk met relx en rebar in de achtergrond.

OTP wordt met *EShell* geïnstalleerd.

Figuur 28 schept een duidelijk beeld van de architectuur van Erlang/*OTP*. Op de onderste laag bevindt zich de *hardware* en *Operating System* (*OS*) van de computer van de gebruiker. Daar boven situeert zich het Erlang *run-time system* (*ERTS*) wat zorgt dat alle benodigde operaties voor Erlang werken en draaien zodat een applicatie kan worden gecompileerd, er aan *type checking* kan gebeuren, etc. Op de bovenste laag bevindt zich de eigenlijke applicatie geschreven in de programmeertaal. Het blokje ernaast dient voor interoperabiliteit met Erlang.

Hieronder worden de verschillende onderdelen, welke zich op de derde laag bevinden in het schema, kort aangehaald:

- Voor het webserver onderdeel wordt er verwezen naar de webservers Yaws [19] en Cowboy [20].
- Het *System Architecture Support Library* (*SASL*) is nodig zodat er in *run-time* een *upgrade* of *downgrade* van een hele *release* kan gebeuren zonder de gehele applicatie stil te leggen, *hot code swapping*.
- Mnesia DBMS is het database *management* systeem welke gebruik maakt van de Mnesia database. In Erlang wordt er naast deze database *ETS* en *DETS* tabellen gebruikt:
    - *Erlang Term Storage* (*ETS*) is een tabel welke de data bijhoudt zolang de tabel blijft bestaan. Met andere woorden de data wordt niet opgeslagen en gaat verloren na het afsluiten van de tabel.
    - *Disk Erlang Term Storage* (*DETS*) is een tabel welke de data wel opslaat in een bestand en het terug kan oproepen na het heropenen van de tabel.
- *SNMP Agents* is een afkorting voor *Simple Network Management Protocol Agents*. Deze *Agents* zorgen voor een goede en eenvoudige overdracht van de berichten tussen verschillende modules. Dit *SNMP* protocol maakt deel uit van het internet protocol *Transmission Control Protocol/Internet Protocol* (*TCP/IP*) [21].



Figuur 28: Overzicht van Erlang/OTP architectuur [22].

A.1.1 Release Concept

Voor het maken van een *release* moet men een *Release Resource* bestand creëren, dit bestand definieert welke applicaties en subsets van Erlang/*OTP* er moeten geïntegreerd worden in de *release*.

Het genereren van de *boot scripts* en *release packages* wordt gedaan met het *Release Resource* bestand. Een systeem dat getransporteerd wordt naar en geïnstalleerd wordt op een andere plaats, ook wel *target* systeem genoemd.

Voor meer informatie omtrent deze sectie wordt er verwezen naar [23] evenals het specifiek maken en distribueren volgens de *OTP* handelwijze [24].

A.2 Rebar

Rebar is een Erlang *tool* voor het eenvoudig compileren en testen van Erlang applicaties en *releases*. Het is dus een Erlang *script*, dat eenvoudig kan ingebed worden in een project. Rebar werd door Dave Smith van Basho, ook bekend als de ontwikkelaars van Riak, ontwikkeld voor het bouwen en genereren van Erlang applicaties [25].

Rebar werkt volgens de *OTP design* principes. Wat wil zeggen dat de applicatie dezelfde structuur moet hebben zoals bij *OTP*. Rebar zit niet standaard in het Erlang pakket en zal dus apart geïnstalleerd dient te worden, zie site https://github.com/basho/rebar. De enige vereiste is dat de Erlang *Shell* (*EShell*) op zijn minst versie R13B03 moet zijn.

Het creëren en compileren van een applicatie met Rebar wordt in deze thesis niet besproken, referenties [26] en [27] bevatten een uitgebreide *tutorial* hierover. Referentie [27] bespreekt ook de nieuwe *features* van de Rebar versies 2 en 3.

A.3 Erlang.mk met relx

Deze methode van werken wordt door Cowboy gebruikt [28]. Ook werde deze werkwijze toegepast in de cursus van het gevolgde vak “Toepassingen en algoritmes van geavanceerde programmeertalen” van Ing. Leo Rutten, om alzo Cowboy onder de knie te krijgen voor deze thesis [29]. Een *release* maken, zoals eerder aangehaald, vereist twee stappen namelijk:

- Ten eerste het bouwen van verschillende *OTP* applicaties die toegevoegd worden aan de *release*.
- Ten tweede moet de *release* zelf opgebouwd worden door middel van het toevoegen van het *ERTS*, er is een *boot script* nodig voor het starten van alle *nodes* en al de applicaties en configuratie bestanden.

Deze stappen worden vergemakkelijkt door Erlang *make* met Relx (Erlang.mk en Relx), deze twee *tools* worden hieronder toegelicht:

- Erlang.mk is een bijkomend bestand voor GNU Make. Het toevoegen van de *Makefile* aan de applicatie is al voldoende. Deze *Makefile* zorgt ervoor dat bij het compileren en het maken van een *release* alle afhankelijkheden gaat ophalen en gaat compileren. Met andere woorden Erlang.mk leidt de compilatie en moet toegevoegd worden in het project. Dit zorgt ervoor dat de eerste stap, hierboven beschreven, vergemakkelijkt of vereenvoudigd wordt. Dit bestand moet men eerst downloaden van volgende site: https://raw.github.com/extend/erlang.mk/master/erlang.mk
- Relx is een *release creation tool* en is uitgebracht als een *single* uitvoerbaar bestand. Dit is de uiteindelijke *build tool* en zorgt voor de *release*, welke de laatste stap vereenvoudigt.

De structuur van de applicatie ziet er als volgt uit:

```
mysample
|-- Makefile
|-- rel
    `-- vm.args
|-- erlang.mk
|-- relx.config
`-- src
    |-- mysample.app.src
    |-- mysample.erl
    `-- mysample_app.erl
```

Om te kunnen compileren moeten er een paar regels in de *Makefile* komen te staan. In deze *Makefile* worden alle *includes* en afhankelijkheden toegevoegd zodat tijdens het bouwen van de *release* de juiste afhankelijkheden kunnen gedownload, gecompileerd en toegevoegd worden met de *release*.

Hieronder staat een voorbeeld van het kleinste erlang.mk gestuurde *Makefile*. De enige vereiste is de naam van het project en de erlang.mk *include*.

```
PROJECT = mysample

include erlang.mk
```

Het is mogelijk dat het project afhangt van applicaties of modules die op bv.: github staan. Om te zorgen dat het project deze mee integreert moeten deze ook worden meegegeven in de *Makefile*.

Na de project naam worden alle afhankelijkheden opgesomd. Om alle afhankelijkheden te kunnen downloaden, in dit geval van Github, moet er voor iedere afhankelijkheid een regel worden gemaakt met daarin de *Uniform Resource Locator* (*URL)*. De *URL* vermeldt waar de juiste afhankelijkheid is opgeslagen samen met het versienummer dat nodig is voor het project.

```
PROJECT = mysample

DEPS = cowboy erlydtl eventmanager

dep_eventmanager =
http://ontwerpen1.khlim.be/gitblit/r/erlang1415/eventmanager.git
master

dep_cowboy = https://github.com/extend/cowboy.git 0.8.5

dep_erlydtl = https://github.com/evanmiller/erlydtl.git 4d0dc8fb

include erlang.mk
```

Om de applicatie te compileren en de *release* te maken dient men enkel volgend commando in te geven:

```
make
```

Nadat de applicatie is gecompileerd zijn er twee *directories* bijgekomen, namelijk ebin en _rel.

- In de map ebin bevinden zich alle *BEAM* bestanden
- In de map _rel staat de volledige *release*.

Zodoende heeft men alle bestanden van de applicatie en alle bestanden die er nodig zijn om Erlang op een platform te laten werken.

B Instructies

- Download en installeer de Erlang *Shell* (*EShell*) op de site http://www.erlang.org/download.html.
- Download de broncode naar de gewenste plaats. Vb.: c:\Users\Gebruiker\Desktop
- Open in de *window manager* de *EShell*.
- Type in de *EShell* `cd('PATH').` waar `PATH` de locatie is waar de code zich bevindt, bv.: `cd('c:/Users/Gebruiker/Desktop').`. Vergeet niet het "`.`" na elk commando en worden de "\" vervangen door "/".
    - Fase 1 bevat een eenvoudig voorbeeld van het *NEU*-protocol met de ziekte Bronchitis.
    - Fase 1.2 bevat een eenvoudig voorbeeld van het NEU-protocol met een extra ziekte Diabetes type1.
    - Fase 2 is uitgebreid met één *resource*, namelijk de patiënt en een gebruiker simulatie.
    - Fase 2-1 bevat Fase 2 met een eenvoudige versie van de virtuele zorg.
    - Fase 3 is een uitgebreidere versie van Fase 2 met een extra *resource* namelijk een verpleegkundige.
    - Fase 3-1 bevat Fase 3 met een eenvoudige versie van de virtuele zorg.
- Type in de *EShell* `observer:start().` Er zal een *window manager* geopend worden.
- Ga naar het tabblad *Table Viewer.* Hierin zullen, na het starten van de applicatie, alle *ETS* tabellen komen te staan. Door een tabel te openen is het mogelijk om de data te bezichtigen. `CTRL + R` dient om een *reload* te doen van de *table viewer* of de data in de *ETS* tabel.

Vanaf nu wordt de daadwerkelijke code gestart, voor meer uitleg wordt er verwezen naar hoofdstuk 4.

- Type in de *EShell* `c(test).` om de test module te compileren. De andere modules moeten niet op deze manier handmatig gecompileerd worden, dit zal automatisch gebeuren bij het starten.
- Type in de *EShell* `test:create().` om het programma te starten. Opgepast bij de versies Fase2 en hoger moet een *behaviour* worden meegegeven aan de `test:create` functie. Er dient `true` meegegeven te worden als de simulatie zich als een voorbeeldige patiënt moet gedragen, een `false` zorgt er dan weer voor dat er een willekeurig patroon de keuze van het al dan niet innemen van medicatie beslist. Men zal een *tuple* terug krijgen waarin het volgende staat:

  ```
  {ok, Pid, [CPid1, CPid2, …]}
  ```

    - *Pid* staat voor het volledige ziektebeeld van de patiënt en wordt gebruikt voor het stoppen van het volledige ziektebeeld of een aparte kuur.
    - De *CList*, staat voor *Cure List*, en bevat alle *Pid*'s van de elke kuur die een bepaalde patiënt heeft.
- Ga naar de *table viewer* om de gegevens in de *ETS* tabellen na te gaan. Op Figuur 6 en Figuur 7 is de werking grafisch te volgen.

Onderstaande instructies overlopen de stopinstructies van het testprogramma:

- Voor het stoppen van een bepaalde ziekte maakt men gebruik van het volgende commando:

  ```
  test:stopDisease(Pid, CPid).
  ```

    - De *Pid* is de *ID* van het ziektebeeld van de patiënt, dit is verkregen na het starten van de testmodule.
    - *CPid* is een *ID* verkregen uit de *CList* welke men ook verkrijgt na het starten van de test.
- Voor het totaal stoppen van het ziektebeeld, alle therapieën en *resources*:

```
test:stop(Pid).
```

    - De *Pid* is de *ID* van het ziektebeeld.

Als men de Diabetes ziekte gebruikt om te testen moet men bovenstaande stop instructies volgen om de ziekte te stoppen of via het alternatief in de *EShell* CTRL + G typen. Dan komt men in een apart gedeelte van deze *shell*, om dan verder te gaan moet men `k` (*kill process*) intypen, vervolgen met `s` (*start new process*) en om te eindigen en verder te kunnen testen `c` (*connect to process*) intypen. Dit omdat de tijden zeer kort gehouden worden, zodat de testen niet te lang duren, en hierdoor de computer kan crashen omwille van de oneindig continue van deze ziekte.

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
ICT-platform voor geïntegreerde gezondheidszorg

Richting: **master in de industriële wetenschappen: elektronica-ICT**
Jaar: **2015**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Mouha, Daan

Datum: **18/08/2015**