Afstudeerverslag
Ontwikkelen van een intelligente meter

Afstudeerverslag

Auteur	Tom de Vroomen
Datum	09-06-2006
Locatie	Rotterdam
Versie	1.0

Bedrijf
LogicaCMG Nederland NV
Programma: Working Tomorrow
Contactpersoon: drs. E. van der Laan
Email: eric.van.der.laan@logicacmg.com
Adres: Kralingseweg 241-249
3061 CE ROTTERDAM
Tel: +31 102537339

Afstudeerder
Naam: Tom de Vroomen
Studentnummer: 20031022
Email: tom.de.vroomen@logicacmg.com
Tel: +31 610251897

Afstudeerverslag
Ontwikkelen van een intelligente meter

Afstudeerverslag

Auteur	Tom de Vroomen
Datum	09-06-2006
Locatie	Rotterdam
Versie	1.0

Bedrijf
LogicaCMG Nederland NV
Programma: Working Tomorrow
Contactpersoon: E. de Ridder
Email: eddy.de.ridder@logicacmg.com
Adres: Kralinseweg 241-249
3061 CE ROTTERDAM
Tel: +31 102537211

Afstudeerder
Naam: Tom de Vroomen
Studentnummer: 20031022
Email: tom.de.vroomen@logicacmg.com
Tel: +31 610251897

Versie beheer

Datum	Versie	Wijzigingen
30-03-2006	0.8	Inhoud gegeven aan verslag
01-05-2006	0.9	Lay-out netjes gemaakt
09-06-2006	1.0	Eindversie

Intelligente Meter
Afstudeerverslag

Referaat

In dit verslag wordt ingegaan op de ontwikkeling van een intelligente stroommeter in het kader van het afstuderen van Tom de Vroomen bij LogicaCMG te Rotterdam.

Kernwoorden:
LogicaCMG
Stroommeter
Stroom management
JAVA
Prepaid stroom
Energie

LogicaCMG Haagse Hogeschool

Voorwoord

Voor u ligt het afstudeerverslag van Tom de Vroomen. Dit verslag is tot stand gekomen door het uitvoeren van de afstudeeropdracht "Intelligente Meter" bij LogicaCMG te Rotterdam. De afstudeeropdracht is uitgevoerd ter afsluiting van de opleiding Technische Informatica aan de Haagse Hogeschool te Den Haag.

Tijdens de afstudeerstage hebben een aantal personen en bedrijven ervoor gezorgd dat deze opdracht tot een succesvol einde is gebracht. Deze personen zijn mijn bedrijfsbegeleiders Eric van der Laan en Eddy de Ridder. Daarnaast heeft de Architect van het Working Tomorrow programma, Marco Pas, mij geholpen met het ontwerp van de software.
De bedrijven die mij geholpen hebben met de hardware voor mijn demo-opstelling zijn 4Top voor de stroommeter en Kadex Domotica voor de schakelaars.

Via deze weg wil deze personen en bedrijven bedanken voor hun hulp en inzet tijdens mijn afstudeerstage.

Ook wil ik u nog even wijzen op een quote die ik op Internet tegenkwam en wel toepasselijk vind:

"Once upon a time, before the advent of electricity, home automation had a different name: servants"
Yahoo Internet Life Magazine, July 2002

Tom de Vroomen
9 juni 2006
Rotterdam

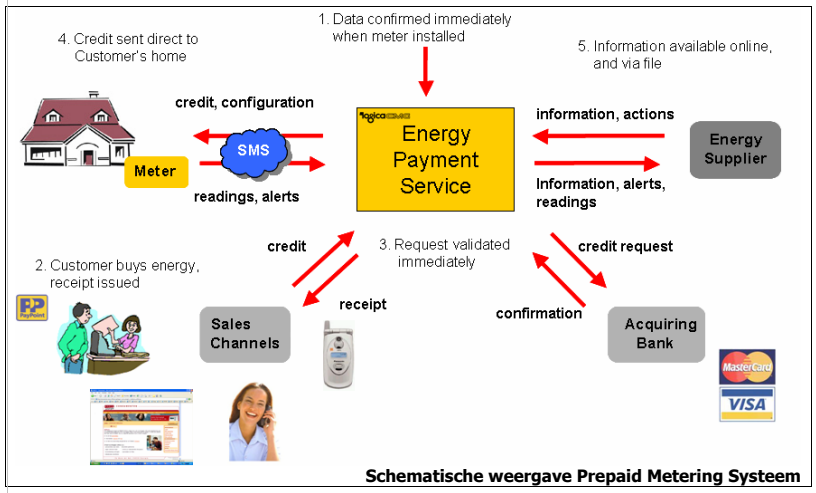
Intelligente Meter
Afstudeerverslag

Inhoudsopgave

LogicaCMG Haagse Hogeschool

1 Inleiding

De laatste tijd wordt door de energieleveranciers en de politiek gesproken over het invoeren van energie op basis van vooruitbetaling. Prepaid energie is niet nieuw. In Groot-Brittannië wordt energie al jaren op prepaid basis geleverd. LogicaCMG is bezig met het ontwikkelen van een nieuw prepaid systeem, namelijk Prepaid Metering.



Schematische weergave Prepaid Metering Systeem

Prepaid Metering is het systeem waarbij alle partijen die te maken hebben met het aanbieden van stroom met elkaar in verbinding staan. Op deze manier heeft de leverancier up-to-date informatie over het stroomverbruik, de afnemer kan op een makkelijke manier het tegoed opwaarderen door middel van SMS of het Internet.

Ook is Italië al een eind op weg met energie op prepaid basis. In Nederland wordt over enkele jaren energie geleverd op prepaid basis. Dit om de kosten voor de leverancier te minderen, maar ook om de afnemers bewuster met energie om te laten gaan. De prepaid meters die gebruikt zullen worden, zijn uit te breiden met software. Een uitbreiding zou de "Intelligente Meter" kunnen zijn. De "Intelligente Meter" helpt met het reduceren van de kosten voor de afnemer en de belasting op het milieu. Gezien het rapport van het KNMI over het toekomstige klimaat in Nederland, zal dit ook een issue zijn. Het KNMI voorspelt namelijk een warmer en natter klimaat. Deze belastingen zullen minder worden door het stroomverbruik niet boven een bepaald punt te laten komen. Deze uitbreiding zal de klant er eerder toe overhalen om gebruik te maken van prepaid energie.

In dit verslag wordt beschreven hoe de "Intelligente Meter" tot stand is gekomen. Er wordt ingegaan op de ontwikkeling van de meter. Dit wordt gedaan vanuit het oogpunt van de functionaliteiten, maar ook het technische gedeelte. Ook worden er keuzemomenten toegelicht.

1.1 Leeswijzer

Dit verslag is bedoeld om de examencommissie van de Haagse Hogeschool inzicht te geven in de activiteiten van de student tijdens de afstudeerstage. Voor LogicaCMG om inzicht te geven of het concept van de "Intelligente Meter" verder uitgewerkt moet worden naar een eindproduct welke interessant is voor klanten van LogicaCMG.

Hierbij een korte routebeschrijving van het verslag. In de Inleiding wordt het doel en het kader van de opdracht beschreven, zo ook de doelgroep van het verslag.
In het hoofdstuk Bedrijf wordt een beschrijving van LogicaCMG en de positie van de opdracht en de student weergegeven. Het hoofdstuk De Opdracht beschrijft de tot standkoming van de opdracht, de onderzoeksvragen die gehanteerd zijn en de aanpak en scope van de opdracht. In de hoofdstukken Inception fase, Elaboration fase, Construction fase en Transition fase worden de activiteiten beschreven gedaan tijdens deze fasen. In het hoofdstuk Aanbevelingen geeft de student een aantal aanbevelingen met betrekking tot de verdere ontwikkeling van de "Intelligente Meter" voor LogicaCMG. In de evaluatie zijn de proces- en productevaluatie te lezen.

2 Bedrijf

Het bedrijf LogicaCMG is een belangrijke speler in de ICT-wereld. In dit hoofdstuk wordt beschreven wat de activiteiten van LogicaCMG zijn en hoe het bedrijf georganiseerd is. Ook komt aan bod hoe mijn opdracht in dit plaatje past.

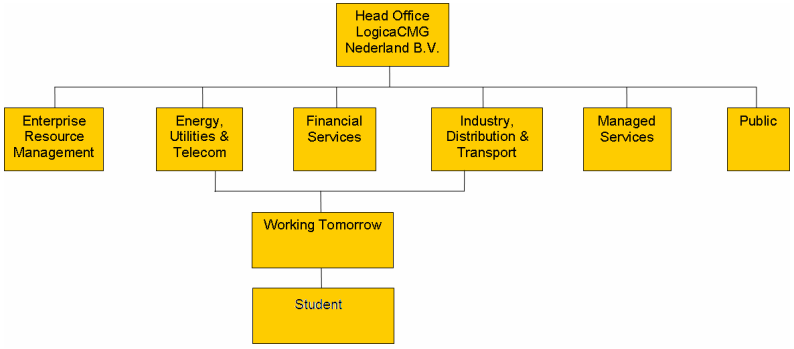
2.1 Situatieschets afstudeerbedrijf

LogicaCMG is een belangrijke internationale speler in de ICT dienstverlening en draadloze telecommunicatie. Zij levert diensten op het gebied van management en ICT consultancy, systeemintegratie en outsourcing. LogicaCMG's klanten zijn actief in diverse markten, zoals telecommunicatie, bank- en verzekeringswezen, energie en utilities, industrie, distributie, transport en de overheid. Na de fusie tussen Logica en CMG in december 2002 heeft het bedrijf ongeveer 20.000 medewerkers in dienst en heeft kantoren in 34 landen. Het bedrijf heeft bijna 40 jaar ervaring in de ICT-dienstverlening. LogicaCMG heeft haar hoofdkantoor in het Verenigd Koninkrijk en is genoteerd aan de beurzen van Londen en Amsterdam.
Het afstudeerproject valt binnen het programma "Working Tomorrow" van LogicaCMG. Binnen dit programma houdt men zich bezig met het toetsen van nieuwe technologische ontwikkelingen op haalbaarheid en mogelijkheden. Working Tomorrow ontwikkelt prototypes en demo's ter verkenning van de technologie en methodiek. Daarnaast wordt er een indicatie van de factoren gegeven die bij een daadwerkelijke implementatie een rol kunnen gaan spelen.

2.2 Positie van de student

De projecten binnen het Working Tomorrow programma worden onder andere uitgevoerd door studenten die hiermee hun afstudeeropdracht kunnen vervullen. Studenten krijgen op deze manier de mogelijkheid een uitdagende en innovatieve afstudeeropdracht in een bedrijfsmatige omgeving uit te voeren. Het hier beschreven project maakt deel uit van dit Working Tomorrow programma en wordt uitgevoerd door Tom de Vroomen onder begeleiding van bedrijfsbegeleiders Eric van der Laan en Eddy de Ridder. De studenten binnen het Working Tomorrow programma zitten in een gezamenlijke ruimte en delen opgedane kennis en ervaringen.

2.3 Organogram LogicaCMG



2.4 De organisatie

LogicaCMG is een wereldwijd actieve dienstverlener op het gebied van management en ICT consultancy, systeemintegratie en outsourcing. LogicaCMG heeft bovendien een uitgebreide expertise in draadloze datacommunicatie, waarmee LogicaCMG haar klanten ondersteunt in diverse markten, zoals telecommunicatie, bank- en verzekeringswezen, energie en utilities, industrie, distributie, transport en de overheid.

LogicaCMG heeft de volgende doelstelling:
'*To help leading organisations worldwide achieve their business objectives through the innovative delivery of information technology and business process solutions*'.

Naast deze hoofddoelstelling richt LogicaCMG zich ook op toekomstige werknemers d.m.v. het Working Tomorrow project. Binnen dit project zijn vele afstudeerders van verschillende universiteiten en HBO-opleidingen werkzaam.

Het Working Tomorrow project heeft 2 hoofddoelen en dit zijn:
- Verhogen van de reputatie van LogicaCMG op het gebied van innovatie.
- Het vinden van toekomstige werknemers.

3 De opdracht

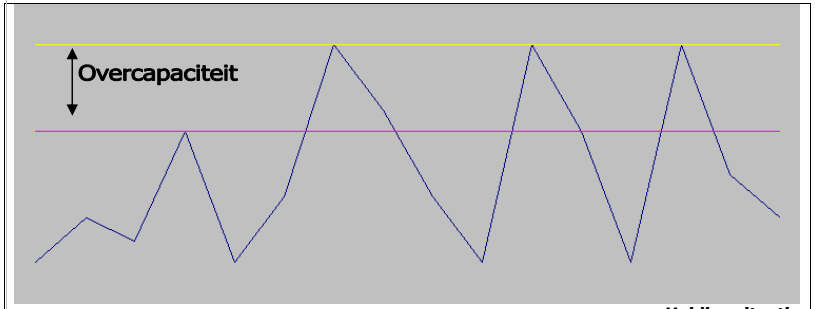
3.1 Aanleiding

De Nederlandse politiek en energieleveranciers zijn de mogelijkheden van prepaid stroom aan het bestuderen. Er worden mogelijkheden besproken om elk huishouden en bedrijf uit te rusten met een prepaid stroommeter. Het doel hiervan is de afnemer bewuster om te laten gaan met energie. Dit bewustzijn kan ook geregeld worden door extra functionaliteit in de prepaid meter.
Op het moment dat alle prepaid stroommeters uitgerust zijn met een manager die het stroomverbruik beïnvloedt, zal de leverancier minder overcapaciteit op het netwerk hoeven leveren. Dit houdt vervolgens in dat er minder stroom opgewekt hoeft te worden met als gevolg een lagere belasting van het milieu.

3.2 Probleem- en doelstelling

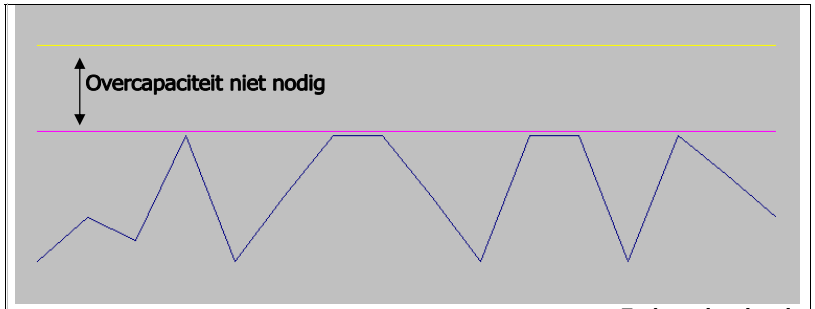
3.2.1 Probleemomschrijving

De laatste tijd is de prijs voor stroom enorm gestegen. Het wordt voor bedrijven interessant om oplossingen te bedenken om het stroomverbruik te reduceren. Eén van deze oplossingen kan gezocht worden in het voorkomen van verbruikspieken op het stroomnetwerk.



Huidige situatie

In de afbeelding hierboven is de huidige situatie geschetst. De afnemer heeft pieken in het verbruik. De leverancier moet ervoor zorgen dat deze pieken worden opgevangen. Het gebied tussen de paarse en gele lijn is de overcapaciteit die de leverancier op het stroomnetwerk moet zetten. Dit is het gebied waarin de stroom duur is. Stroom kan namelijk niet opgeslagen worden.



Toekomstige situatie

De toekomstige situatie zal er uit zien in de schets hierboven. Op het moment dat er geen verbruikspieken op het stroomnetwerk ontstaan, hoeft de leverancier hiervoor ook geen extra elektriciteit op te wekken. Als de leverancier geen extra stroom hoeft op te wekken om pieken op te vangen, zal dit voor de leverancier goedkoper zijn. Dit kan vervolgens doorberekend worden aan de klant. De leverancier is namelijk bereid om een lager tarief aan de klant te rekenen op het moment dat deze verbruikspieken zal weten te voorkomen.

3.2.2 Doelstelling

"Het detecteren van pieken in het stroomverbruik. Dit detecteren gebeurt op een prepaid stroommeter, die met deze functionaliteit wordt uitgebreid."

Met deze extra functionaliteit zal de meter beslissingen nemen op basis van het verbruik, welke de meter intelligent zal maken.

3.2.3 Onderzoeksvragen

Om de doelstelling te specificeren zijn er onderzoeksvragen opgesteld.

De hoofdonderzoeksvraag van deze opdracht is:
Op welke wijze kan de intelligente meter worden ontwikkeld met behulp van de "prepaid meter", gegeven de vraag en aanbod in het elektriciteitssysteem c.q. netwerk?

Hieruit kunnen de volgende onderzoeksvragen worden herleid:

1. **Wat is de gewenste functionaliteit van de intelligente meter?**
   Er wordt ingegaan op wat een prepaid meter is en hoe deze nu en in de toekomst toegepast kan gaan worden.
   Ook wordt er gekeken naar hoe een piek gedetecteerd kan worden op het energienetwerk.
2. **Hoe kan deze toegevoegd worden aan de prepaid meter?**
   Er zal onderzocht worden of dit een hardware of een software oplossing wordt. In het geval van een hardware oplossing zal er een extra apparaat toegevoegd worden aan de prepaid meter. Vervolgens zullen deze met elkaar moeten kunnen communiceren.
   In het geval van een software oplossing zal er software worden toegevoegd aan het bestaande pakket van software die op de meter aanwezig is.
3. **Wat is de positie van de intelligente meter binnen het totale systeem en vraag en aanbod van het elektriciteitssysteem in huis, gebouw, etc.?**
   Er zal bepaald worden op welke plaats de piekdetectie gaat komen en waar de verantwoordelijkheid ligt of een apparaat of groep aan-/uitgeschakeld gaat worden.

3.3 Aanpak

Om het project in goede banen te leiden is er gebruik gemaakt van een projectmethode. Aangezien de stage is gelopen bij een grote onderneming, is er een vaste projectmethode die gebruikt wordt. Bij LogicaCMG wordt gebruik gemaakt van Rational Unified Process (RUP). De keuze voor deze methode stond al vast bij het begin van het project.

In de eerste week van de stage is een presentatie gegeven hoe RUP toegepast kan worden. De RUP-methode schrijft een viertal fases voor waarin het project opgedeeld wordt. Dit zijn de Inception, Elaboration, Construction en Transition fase. In de eerste fase wordt de haalbaarheid van een project bekeken en de scope van het project bepaald. Daarna worden in de Elaboration fase de requirements bepaald en wordt er een technisch en functioneel ontwerp gemaakt. De derde fase, de Construction fase, wordt gebruikt voor het maken van het eindproduct, ofwel het schrijven van de software. Tijdens de laatste fase, de Transition fase, wordt het project overgedragen aan de klant en zal het geïmplementeerd worden. In het geval van dit project zal het inhouden dat de demo en de software overgedragen wordt aan LogicaCMG.

3.4 Scope

De scope van het project is bepaald met behulp van een MoSCoW-analyse. Door gebruik van de MoSCoW-analyse worden de requirements opgedeeld in onderdelen die niet mogen ontbreken en onderdelen die eventueel later verwerkt worden. Hiervoor worden 4 niveaus gehanteerd, dit zijn Must, Should, Could en Won't (at this time). Het niveau "Won't" zal dus niet uitgevoerd worden.

De scope van dit project is terug te vinden in het Initiatiedocument en in het Software Requirements Specification document waarin de functionaliteiten van de scope uitgebreid beschreven staan.

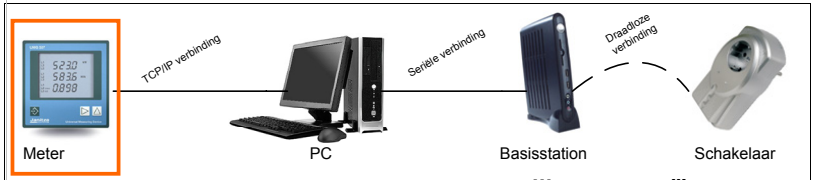
4 Inception fase

In dit hoofdstuk wordt uiteen gezet hoe de Inception fase is doorlopen. In de Inception fase wordt er een Initiatiedocument geschreven. Het Initiatiedocument is terug te vinden in BIJLAGE A: Initiatiedocument. Ook het bepalen van de te gebruiken hardware en software vallen binnen de Inception fase.

4.1 Bepalen te gebruiken hardware

Voor dit project zijn naast de PC twee hardwarecomponenten belangrijk geweest. Dit zijn een stroommeter en schakelaars voor de apparaten. In dit hoofdstuk wordt ingegaan op de keuze van deze componenten.

4.1.1 Stroommeter kiezen



Weergave opstelling apparaten

Aan het begin van de afstudeerperiode had LogicaCMG al een te gebruiken meter beschikbaar in het kader van een LogicaCMG propositie. Dit was al door LogicaCMG geregeld. Deze meter, die gefabriceerd wordt door Iskraemeco, wordt in Groot-Brittannië gebruikt voor de ontwikkeling van een nieuwe prepaid stroom applicatie (Prepaid Metering) voor de Britse markt.

Nadat er contact gezocht was met de Britse locatie van LogicaCMG, zou het geen probleem zijn de meter en de applicatie op te sturen. De student zou vervolgens om deze meter heen het intelligente gedeelte bouwen. LogicaCMG in Nederland is bezig een applicatie te bouwen die werkt met Interactive Voice Response (IVR) in samenwerking met Prepaid Metering uit Groot-Brittannië. IVR is spraakherkenning door de computer. Op deze manier kan een klant het tegoed opwaarderen door het te zeggen en hoeft het niet in te toetsen op de telefoon. Voor een demo van het IVR was een werkende versie van Prepaid Metering nodig. Om dit te kunnen realiseren is de student voor 5 dagen vertrokken naar Edinburgh om de Prepaid Metering applicatie op te halen, samen met 2 stroommeters. Deze reis stond oorspronkelijk gepland in week 4 van de afstudeerperiode. Nadat duidelijk was geworden dat deze reis pas veel later gemaakt zou worden, is er gezocht naar een alternatief voor de meter.

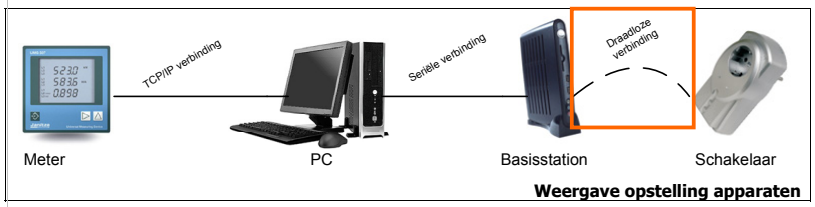
Het alternatief is gevonden in de digitale stroommeter UMG507 van Janitza. Deze meter is gekozen vanwege de vele functionaliteiten. Een van deze functionaliteiten is het kunnen uploaden van JAVA-applets in de meter. Dit betekende dat het mogelijk zou zijn om de software op de meter te kunnen uitvoeren. Bij de andere meters was deze functionaliteit niet aanwezig. Ook was het voor LogicaCMG interessant deze meter aan te schaffen vanwege een fikse korting op de meter mogelijk gemaakt door 4Top. Ook kunnen studenten met andere opdrachten die te maken hebben met stroomverbruik gebruik maken van deze meter.

Een ander alternatief voor een meter zou geleverd kunnen worden door het bedrijf Alectryon. Maar vanwege de vele mogelijkheden die de UMG507 biedt, is er geen contact gezocht met dit bedrijf. Op de website van Alectryon waren geen meters beschikbaar met de functionaliteiten van de UMG507.

Janitza UMG507 **Iskraemeco MT172** **Alectryon MICO-serie**

De student is in de 9e week van de afstudeerperiode wel vertrokken naar Edinburgh voor het ophalen van de Prepaid Metering applicatie. Om de applicatie te kunnen uitvoeren is er gebruik gemaakt van Red Hat Linux Enterprise en Oracle 10g. De applicatie is geschreven in de taal JAVA. Tevens moest de JAVA Runtime voor Linux geïnstalleerd worden.

4.1.2 Communicatieprotocol tussen meter en apparaten kiezen



Weergave opstelling apparaten

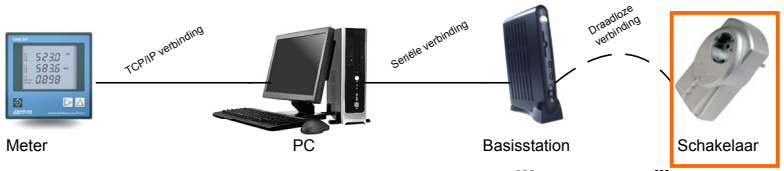
Om te kunnen communiceren tussen meter en de apparaten is er een medium nodig. Dit kan kabel zijn of door de lucht. Bij kabel kan, in dit geval, gedacht worden aan UTP-kabel of de stroomkabel. Bij communicatie door de lucht is gekeken naar WiFi en radiofrequenties. Er is ook gekeken naar radiofrequenties, omdat voor het schakelen van apparaten door middel van een schakelaar in een stopcontact meerdere oplossingen zijn. De keuze hierin wordt in het volgende hoofdstuk beschreven.
Het nadeel van het gebruik van UTP-kabel is dat door het gehele huis naar elk apparaat een kabel getrokken moet worden en moet er netwerkapparatuur aangeschaft worden om de huishoudelijke apparaten te koppelen met de meter.

De ideale oplossing leek in eerste instantie PowerLine Communication (PLC). Dit omdat de communicatie verloopt via het stroomnetwerk. Maar ook hiervoor moet apparatuur geïnstalleerd worden. Een ander nadeel is dat een netfilter geplaatst moet worden bij de kabel naar buiten toe. Het zou anders mogelijk kunnen zijn dat het verkeer bij bijvoorbeeld de buren terecht komt met als gevolg dat bij de buren apparaten geschakeld worden.

Een andere oplossing kwam naar voren tijdens een gesprek met dhr. Peters, Competence Manager bij LogicaCMG en Domotica expert. Dhr. Peters adviseerde gebruik te maken van een draadloos systeem. Voorbeelden van een dergelijk systeem zijn www.klikaanklikuit.nl of apparatuur van Kadex Domotica. Hoe het systeem van www.klikaanklikuit.nl technisch werkt is via de website niet te achterhalen. Het systeem van Kadex werkt via de radiofrequentie van 868,3Mhz. Deze frequentie is Europees genormeerd, storingsvrij en bedrijfszeker.

De keuze is gevallen op een draadloze variant, zodat men zo min mogelijk aanpassing hoeft te doen aan de huidige infrastructuur.

4.1.3 Schakelaars van de apparaten kiezen



Weergave opstelling apparaten

Aangezien de keuze voor de communicatie tussen meter en apparaat gevallen was op een draadloze variant, werd de keuze voor schakelaars al kleiner. Het ging daarbij om de keuze tussen het systeem van klikaanklikuit of van Kadex. Dhr. Peters heeft goede contacten met Kadex Domotica en deze heeft het mogelijk gemaakt dat de benodigde apparatuur voor de tijd van de stage geleend kon worden.

Kadex Wandcontactdoos

Klikaanklikuit wandcontactdoos

4.2 Bepalen te gebruiken software

Er moest worden bepaald in welke programmeertaal de software geschreven zou worden. Ook moest er een keuze worden gemaakt voor de Integrated Development Environment (IDE).

4.2.1 Programmeertaal bepalen

In de eerste fase van het project was nog niet duidelijk of de oplossing hardwarematig of softwarematig aangepakt zou worden. Toen duidelijk was dat de meter UMG507 gebruikt zou worden, die JAVA-applets ondersteund, heeft de student gekozen voor de taal JAVA. De taal is ook gekozen omdat er binnen LogicaCMG veel kennis over JAVA aanwezig is. Er is een gehele afdeling die zich bezig houdt met het programmeren in de taal JAVA. Als er voor een hardwarematige oplossing gekozen zou zijn, was de keuze waarschijnlijk C of C++ geweest.

4.2.2 Integrated Development Environment kiezen

Er zijn vele verschillende IDEs. Zo zijn er commerciële pakketten beschikbaar, maar ook een aantal open source pakketten. De functies in commerciële pakketten verschillen maar weinig met die van de open source pakketten. Een aantal voorbeelden van commerciële pakketten zijn van Borland JBuilder, JCreator Pro of IntelliJ IDEA. Open source tegenhangers zijn Sun NetBeans, Eclipse of BlueJ.

Voor afstudeerders is het erg makkelijk als de software die gebruikt wordt open source is. Dit betekent dat er niet betaald hoeft te worden voor pakketten die dezelfde functionaliteit bieden als commerciële producten.

De student heeft, net als de afdeling Working Tomorrow, gekozen voor het pakket Eclipse. Dit omdat het open source is, en omdat dit de standaard van Working Tomorrow is. Het voordeel van Eclipse is ook dat er plugins voor ontwikkeld kunnen worden. Zo is er voor elke student altijd wel een plugin beschikbaar die hij of zij kan gebruiken bij het programmeren van de software.

4.2.3 Unified Modelling Language tooling kiezen

Voor het maken van de functionele en technische ontwerpen wordt er gebruik gemaakt van de Unified Modelling Language. Net als IDEs zijn hiervoor ook vele verschillende software pakketten beschikbaar. Zowel commercieel als open source. Een commercieel product is bijvoorbeeld IBM Rational Rose, Microsoft Office Visio. Poseidon for UML van Gentleware, mits niet commercieel gebruikt, en ArgoUML.

De student heeft voor Poseidon for UML Community Edition gekozen. Ook omdat deze standaard gebruikt wordt bij Working Tomorrow. Deze versie is gratis, mits het voor niet commerciële doeleinden wordt gebruikt. Dit is voor de studenten handig, want dan kan het ook thuis gebruikt worden.

4.3 Requirements bepalen

LogicaCMG werkt voor het bepalen van requirements met een standaard document in het kader van het gebruik van RUP binnen LogicaCMG. Dit Software Requirements Specification (SRS) document wordt opgebouwd in twee fases. In de eerste fase worden er keuzes gemaakt ten aanzien van de gewenste oplossing. Daarna worden deze keuzes verder uitgewerkt.

- In de eerste fase wordt met behulp van een MoSCoW-analyse bepaald welke functionaliteiten, welke prioriteit krijgen.

- In de tweede fase wordt van elke functionaliteit een korte omschrijving gegeven en de actie en gevolgen van deze functionaliteit.

Functionaliteiten zijn vastgesteld aan de hand van gesprekken met de opdrachtgever en door brainstormen over de vraag aan welke eisen de software moet voldoen.

In de eerste versie van het SRS-document zijn er functionaliteiten beschreven die te maken hebben met het hebben van stroomgroepen. Deze functionaliteiten zijn verplaatst naar "Won't" omdat het hebben van stroomgroepen technisch niet mogelijk was bij de demo-opstelling.

Nummer	Omschrijving	Prioriteit
FEAT01	- Piekdetectie (als dit niet gedetecteerd kan worden op het stroomnetwerk, wordt dit gesimuleerd) - Beslissingen maken op basis van prioriteiten groep/apparaten	Must
	- Verbinding met apparaten - Prioriteiten per apparaat - Aan- / uitzetten per apparaat	Should
	- Beheer via internet/webinterface - Prioriteiten per groep - Aan- / uitzetten per groep - Beheergedeelte	Could
	- Beveiligde communicatie	Won't

Functionaliteiten EnergyManager v1.0

Als eerste functionaliteit staat Piekdetectie beschreven. In de implementatie is deze functionaliteit gesimuleerd. Het is technisch niet mogelijk om de hoeveelheid overcapaciteit op het stroomnetwerk te meten met de meter waarmee gewerkt wordt. Dit wordt gesimuleerd doordat er een functie in de software is gebouwd die geactiveerd wordt door de gebruiker. Dit om het concept van piekdetectie te kunnen bewijzen.

Nummer	Omschrijving	Prioriteit
FEAT01	- Piekdetectie (als dit niet gedetecteerd kan worden op het stroomnetwerk, wordt dit gesimuleerd) - Beslissingen maken op basis van prioriteiten van de apparaten - Beheergedeelte	Must
	- Verbinding met apparaten - Prioriteiten per apparaat - Aan- / uitzetten per apparaat	Should
	- Beheer via internet/webinterface	Could
	- Beveiligde communicatie - Prioriteiten per groep - Aan- / uitzetten per groep	Won't

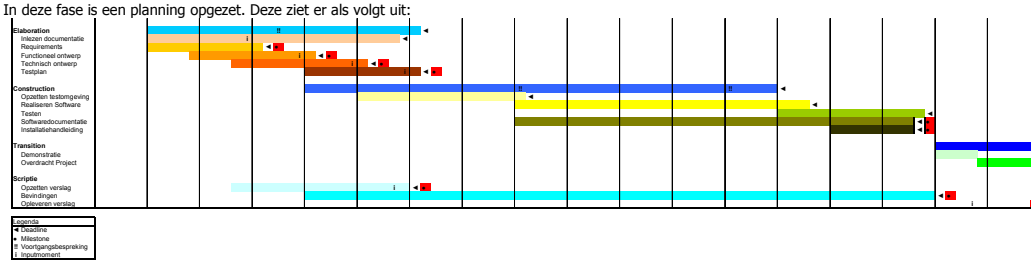
Functionaliteiten EnergyManager v1.1

De functionaliteit Beheergedeelte is verplaatst van Could naar Must, omdat het hebben van een beheergedeelte in de vorm van een gebruiksinterface érg belangrijk is voor een demo. De gebruiker moet kunnen 'spelen' met de demo. Als de gebruiker niet kan 'spelen' met de demo, zal de gebruiker minder snel overtuigd zijn van het concept.

De functionaliteit "Beheer via internet/webinterface" is niet uitgewerkt. Hiervoor zou de software nog verder uitgebreid moeten worden. Daar is helaas geen tijd voor gevonden.

4.4 Planning

In deze fase is een planning opgezet. Deze ziet er als volgt uit:



5 Elaboration fase

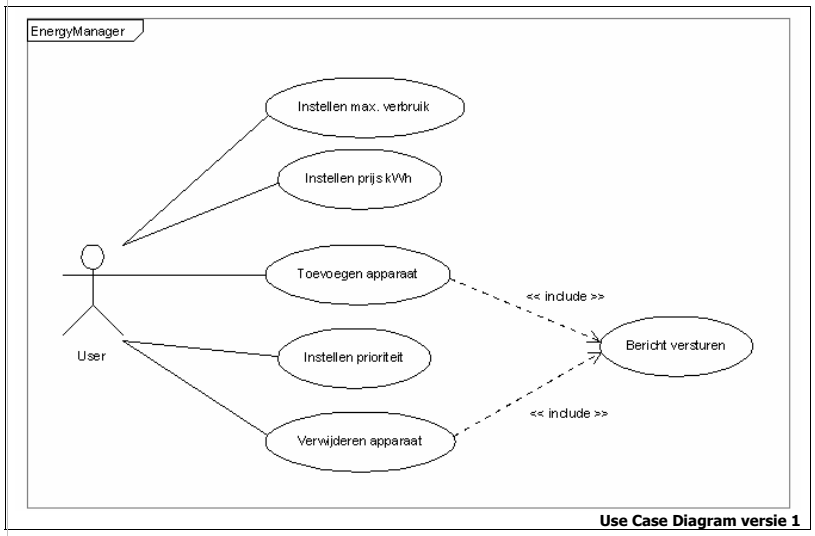
In de Elaboration fase is het ontwerp van de software gemaakt. Dit houdt in dat er een functioneel en een technisch ontwerp is gemaakt. In het functionele ontwerp is vooral terug te vinden wat de functionaliteiten van de software zijn. In het technisch ontwerp is beschreven hoe de software geschreven wordt. Het functioneel en technisch ontwerp zijn terug te vinden in Bijlage B (Functioneel Ontwerp EnergyManager) en Bijlage C (Technisch Ontwerp EnergyManager).

5.1 Functioneel ontwerp EnergyManager

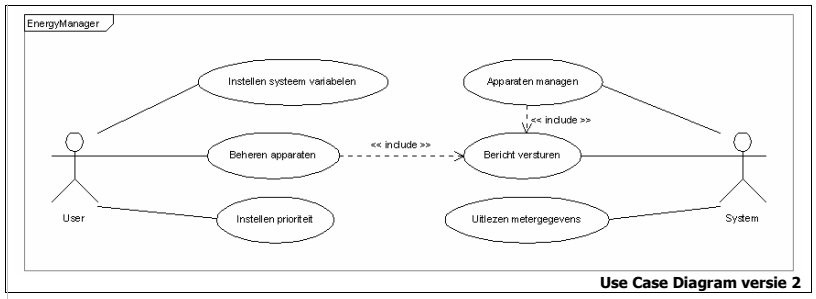
In het functioneel ontwerp staan de requirements beschreven. Dit zijn de eisen waaraan de software moet voldoen. Ook zijn er Use Cases gemaakt van enkele functionaliteiten. Dit is om duidelijk te maken wat de resultaten van deze verschillende functionaliteiten zijn.

5.1.1 Use Cases beschrijven

Voor het beschrijven van de use cases zijn de gebruikersfunctionaliteiten en een enkele functionaliteit van het interne systeem meegenomen. Dit om aan te geven wat de mogelijkheden voor de gebruiker zijn en wat er globaal intern plaats vindt.



Use Case Diagram versie 1



Use Case Diagram versie 2

In de eerste versie van het use case diagram is er uitgegaan van de gebruikersfunctionaliteiten. Na een expert-review van dit diagram zijn er een aantal wijzigingen doorgevoerd. In de tweede versie is als actor ook het systeem opgenomen. Dit om duidelijk te maken dat het systeem een duidelijke rol speelt in de acties die de gebruiker activeert. Ook zijn er een aantal use cases gegeneraliseerd. Zoals "Instellen maximaal verbruik" en "Instellen prijs kWh" kunnen neergezet worden als één use case, namelijk "Instellen systeem variabelen". Op deze manier hoeft het diagram niet aangepast worden op het moment dat er een variabele bijkomt.

De use cases "Toevoegen apparaat" en "Verwijderen apparaat" kunnen samengevoegd worden onder de use case "Beheren apparaten", omdat de acties die uitgevoerd worden niet veel van elkaar verschillen.

Bij de actor "System" zijn 3 use cases verbonden. Dit zijn "Apparaten Managen", "Bericht versturen" en "Uitlezen metergegevens". Deze waren eerst niet meegenomen in het use case diagram, omdat was uitgegaan van de gebruikersfunctionaliteiten. De student heeft besloten om deze use cases wel in het diagram op te nemen, omdat deze 3 functionaliteiten onmisbaar zijn voor het totale systeem. De kracht van het systeem zit vooral in de use case "Apparaten managen". Hier ligt de intelligentie van het systeem.

Zo is de use case "Uitlezen metergegevens" ook onmisbaar voor het systeem. Op basis van deze gegevens wordt het verbruik berekend. Op basis van het verbruik wordt vervolgens besloten of de use case "Apparaten managen" geactiveerd wordt.

De use case "Bericht versturen" is eveneens onmisbaar voor het systeem. Deze use case zorgt ervoor dat de schakelaars op de apparaten geactiveerd worden.

Ondanks dat deze use cases interne functionaliteiten zijn, is het belangrijk om deze wel op te nemen in een use case diagram, omdat het ontwerp meestal wordt gedaan door een ander team dan het bouwen van het systeem. Als deze use cases er niet in zouden staan, kan het gebeuren dat deze vergeten worden bij het bouwen van het systeem.

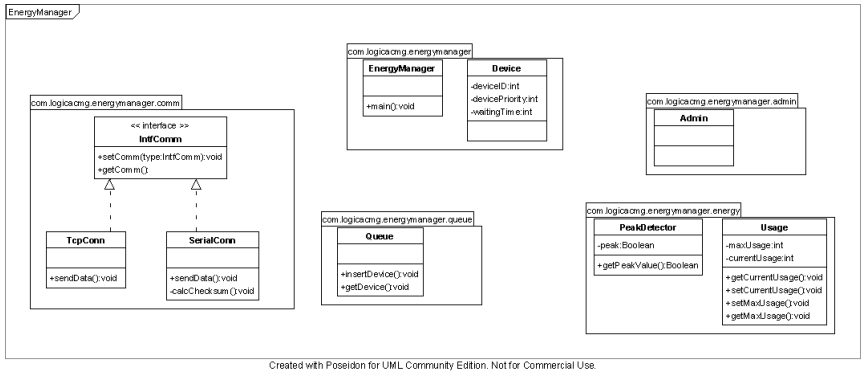
5.2 Technisch ontwerp EnergyManager

Het technisch ontwerp van EnergyManager bestaat uit het klassendiagram.

5.2.1 Klassendiagram ontwerpen

Voordat er met programmeren begonnen kon worden, is eerst een klassendiagram gemaakt om een overzicht te krijgen hoe de software opgebouwd zou worden.

Er is gebruikt gemaakt van packages om klassen die qua functionaliteit bij elkaar horen te groeperen. Bijvoorbeeld de package "comm" bevat de klassen die zorgen voor de communicatie. Dit kan een TCP-communicatie, maar ook een seriële communicatie zijn. Het gebruikt van packages is een onderdeel van het object geörienteerd programmeren.

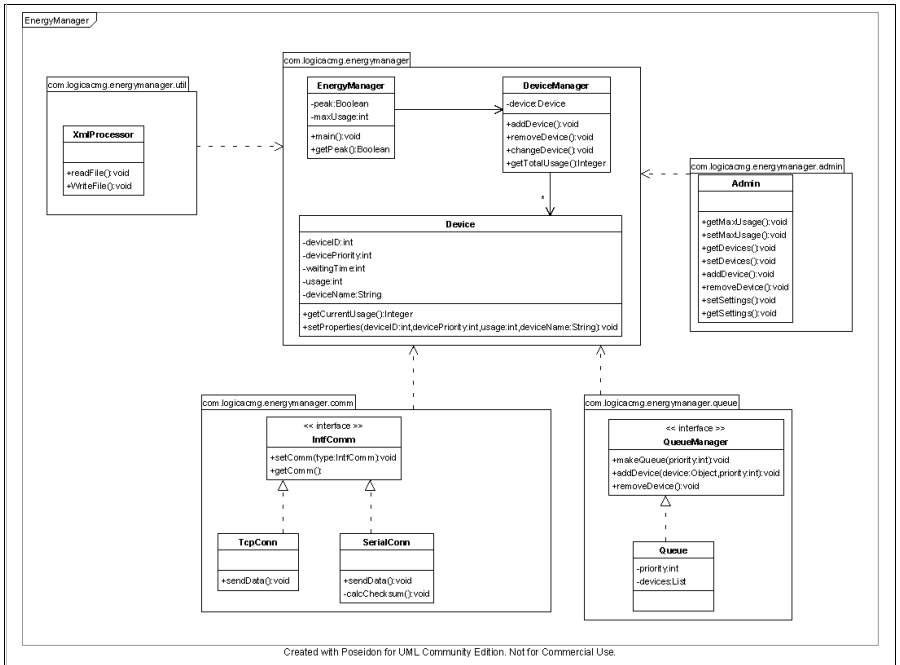


Eerste versie klassendiagram

In de tabel hieronder worden de namen van de packages verklaard.

Package	Functionaliteit
com.logicacmg.energymanager	Entry-point van de software
com.logicacmg.energymanager.admin	Grafische Gebruikersinterface
com.logicacmg.energymanager.comm	Communicatie
com.logicacmg.energymanager.energy	Klassen die verbruiksgegevens bevatten
com.logicacmg.energymanager.queue	Wachtrij

Na een expert-review is er besloten om een aantal wijzigingen door te voeren. Er is een DeviceManager toegevoegd aan de package energymanager. Deze klasse zorgt ervoor dat apparaten beheerd worden. Ook is er een package "util" bijgekomen. In deze package zitten de klassen die niet horen in een van de andere packages. Voor het opslaan van apparaten en instellingen is er gebruikt gemaakt van XML. De apparaten worden in een XML-bestand opgeslagen, zo ook de instellingen als het IP-adres van de meter. Voor de verwerking van de XML-bestanden is een XmlProcessor opgenomen. Ook heeft de klasse Device de eigenschap usage erbij gekregen. Deze eigenschap is nodig om te bepalen of een apparaat wel of niet aangezet kan worden. Hoe de software precies is geworden wordt besproken in hoofdstuk 6.



Tweede versie klassendiagram

5.3 Wachtrij-algoritme ontwerpen

Het intelligente gedeelte van de Intelligente Meter betreft de wijze waarop apparaten aan- en uitgeschakeld worden. Om dit te bewerkstelligen wordt er gebruik gemaakt van een wachtrij. Er zijn verschillende soorten wachtrijen waarvan de First In First Out (FIFO) en Last In First Out (LIFO) de bekendste zijn. Een andere bekende is het Priorityqueuing.

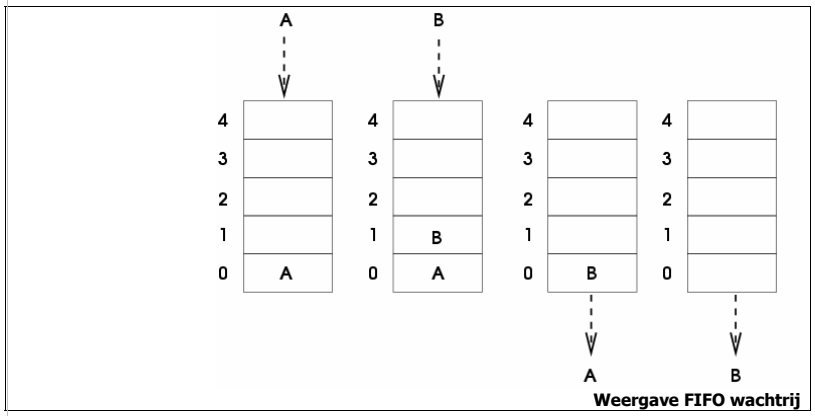
Voor het bepalen van het algoritme is een aantal aannamen gedaan. Zo hebben apparaten die in de software aangemeld worden de volgende eigenschappen die bepalend zijn bij de beslissing welk apparaat aangezet wordt:

- Prioriteit
- Verbruik in Watt
- Wachttijd in pulsen

Deze drie eigenschappen bepalen welk apparaat aangezet zal worden. Eerst wordt beschreven wat voor mogelijke keuzes er waren, vervolgens zal beschreven worden voor welk algoritme van de Intelligente Meter is gekozen.

5.3.1 FIFO

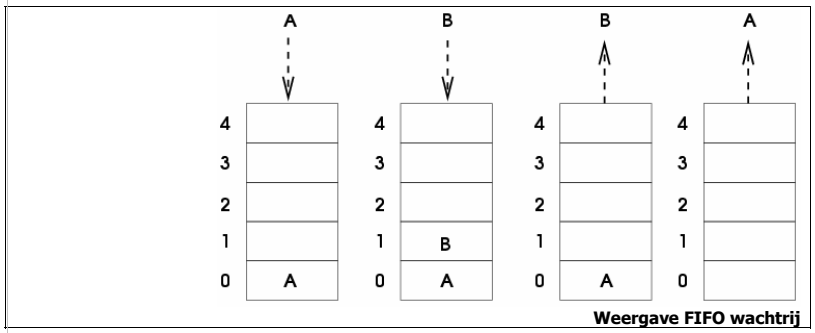
Zoals de naam al duidelijk maakt werkt FIFO volgens het First In First Out principe. Dit kan het beste vergeleken worden met een rij voor de kassa bij een supermarkt. Iedereen sluit achteraan en wacht op zijn beurt, totdat hij vooraan staat. Wie het eerst komt, is het eerste aan de beurt. In het figuur hieronder wordt schematisch weergegeven hoe een FIFO wachtrij werkt.



Weergave FIFO wachtrij

5.3.2 LIFO

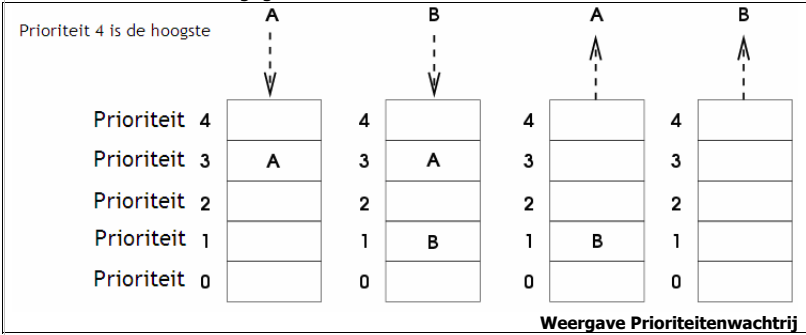
LIFO werkt volgens het Last In First Out principe. De werking van dit algoritme kan het beste vergeleken worden met een stapel dienbladen. Het dienblad dat het laatste op de stapel wordt gelegd, wordt als eerste weer gepakt. In het figuur hieronder wordt dit weergegeven.



Weergave FIFO wachtrij

5.3.3 Prioriteitenwachtrij

Bij een priorityqueue wordt een wachtrij aan de hand van de prioriteit van de elementen afgewerkt. Dit is het beste te vergelijken met processen in een systeem. Het proces met de hoogste prioriteit zal het eerste verwerkt worden. In het figuur hieronder wordt dit weergegeven.



Weergave Prioriteitenwachtrij

5.3.4 Algoritme Intelligente Meter

Voor het algoritme van de Intelligente Meter is gekozen voor een combinatie van het FIFO-algoritme en een zelf bedacht algoritme dat rekening houdt met de prioriteit, het verbruik en de wachttijd van de apparaten.

Het LIFO-algoritme is afgevallen omdat dit geen eerlijke manier is om de apparaten te schakelen. Met het LIFO-algoritme is de kans dat een apparaat nooit aan de beurt komt erg hoog. Dan zou bijvoorbeeld de wasmachine die zich het eerst heeft aangemeld nooit aan gaan.

Ondanks dat prioriteit een eigenschap van een apparaat is, is het priorityqueing afgevallen. Dit om dezelfde reden als bij het LIFO-algoritme. Het kan gebeuren dat apparaten met een lage prioriteit nooit aan de beurt komen.

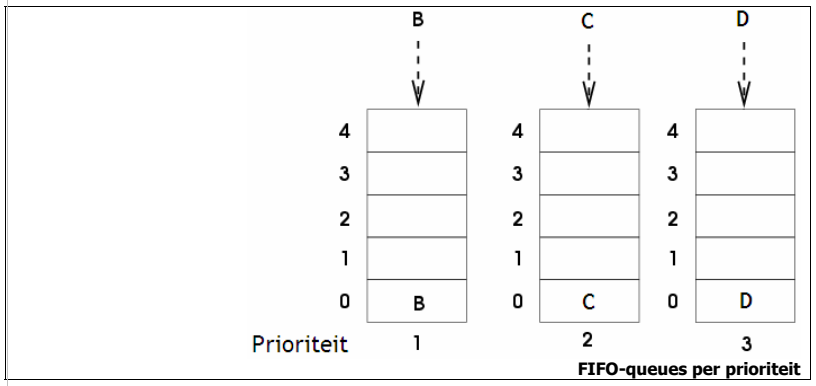
Voor het algoritme van de Intelligente Meter is gekozen voor een FIFO-wachtrij voor elke prioriteit. Een apparaat kan één van de volgende prioriteiten hebben:

Prioriteit	Functie
0	Zal nooit uitgeschakeld worden
1	Staat gelijk aan prioriteit 'Hoog'
2	Staat gelijk aan prioriteit 'Middel'
3	Staat gelijk aan prioriteit 'Laag'

Er is gekozen voor een prioriteit 0, omdat er apparaten zijn die niet uit mogen. Een goed voorbeeld hiervan is de ketel van de CV. Als er warm water gevraagd wordt, moet er ook warm water komen. Dit mag niet tegengehouden worden door de Intelligente Meter. De beslissing welke apparaten welke prioriteit krijgen is aan de gebruiker. De gebruiker geeft ook aan hoeveel Watt het apparaat verbruikt.

Op de volgende pagina's wordt aan de hand van flowcharts uitgelegd hoe het algoritme werkt. Op pagina 28 is een totaaloverzicht te zien.

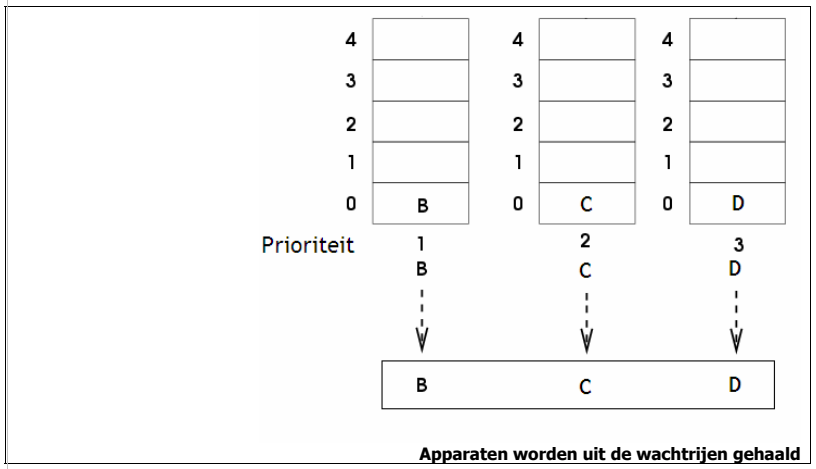
In de figuur hieronder staan de letters A, B, C en D voor apparaten die per prioriteit in een FIFO-queue worden gezet. Op het moment dat een apparaat in een queue staat, zal elke seconde de wachttijd van elk apparaat in de queue met 1 worden verhoogd.



FIFO-queues per prioriteit

Bij de beslissing welk apparaat aan de beurt is, zal prioriteit 0 niet meegenomen worden, omdat dit apparaten zijn die altijd aan staan. Deze prioriteit is dan ook niet meegenomen in de flowcharts en figuren.

Vervolgens zal aan de hand van prioriteit, wachttijd en verbruik bepaald worden welk apparaat aangezet zal worden. Vanuit elke wachtrij wordt het eerste apparaat in een beslissingsboom gezet.



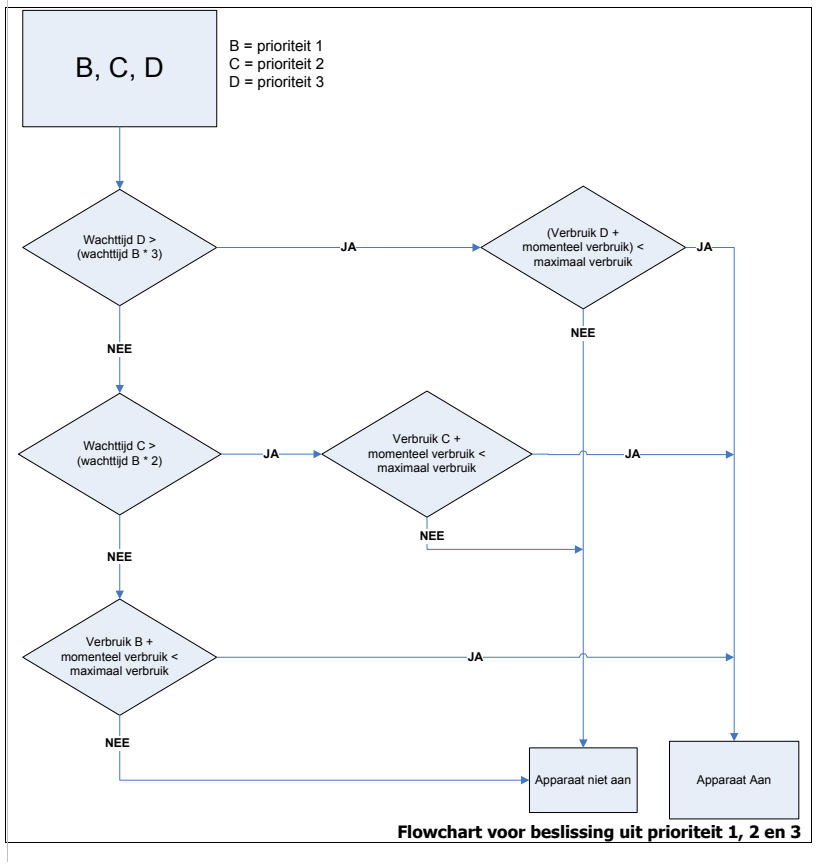
Apparaten worden uit de wachtrijen gehaald

Voor de beslissing zijn er vier mogelijkheden waaruit bepaald kan worden welk apparaat aangezet wordt. Dit heeft te maken met het aantal apparaten in de wachtrijen. De verschillende mogelijkheden op basis waarvan besloten wordt welk apparaat aan gaat zijn de volgende:

- B,C en D
- B en C
- C en D
- B en D
- B óf C óf D

Naast de prioriteit en wachttijd van de apparaten wordt het momenteel verbruik van alle apparaten ook meegenomen bij de beslissing. Het momenteel verbruik wordt berekend door middel van het voltage en de ampère die uit de meter worden gelezen.

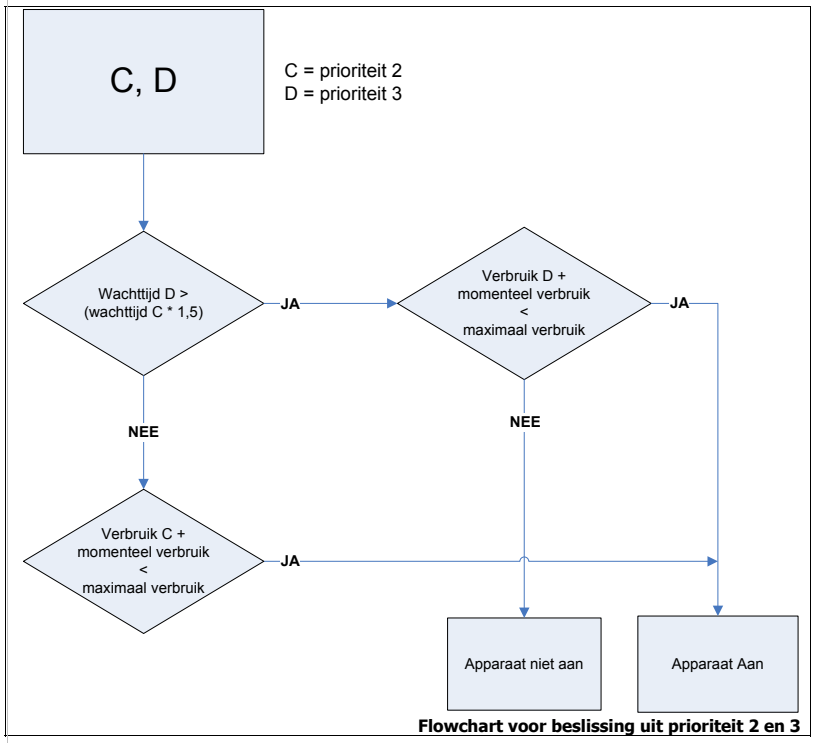
In geval er van elke prioriteit apparaten in de wachtrij staan, geldt de volgende flowchart:



Flowchart voor beslissing uit prioriteit 1, 2 en 3

Om te voorkomen dat een apparaat met prioriteit 3 nooit aan de beurt komt, is er besloten dat als de wachttijd van een apparaat met prioriteit 3, 3x hoger is dan dat van het apparaat van prioriteit 1, deze voorrang krijgt op prioriteit 1. Dit gebeurt ook zo met prioriteit 2, echter dan wordt de vermenigvuldigingsfactor 2 gebruikt.

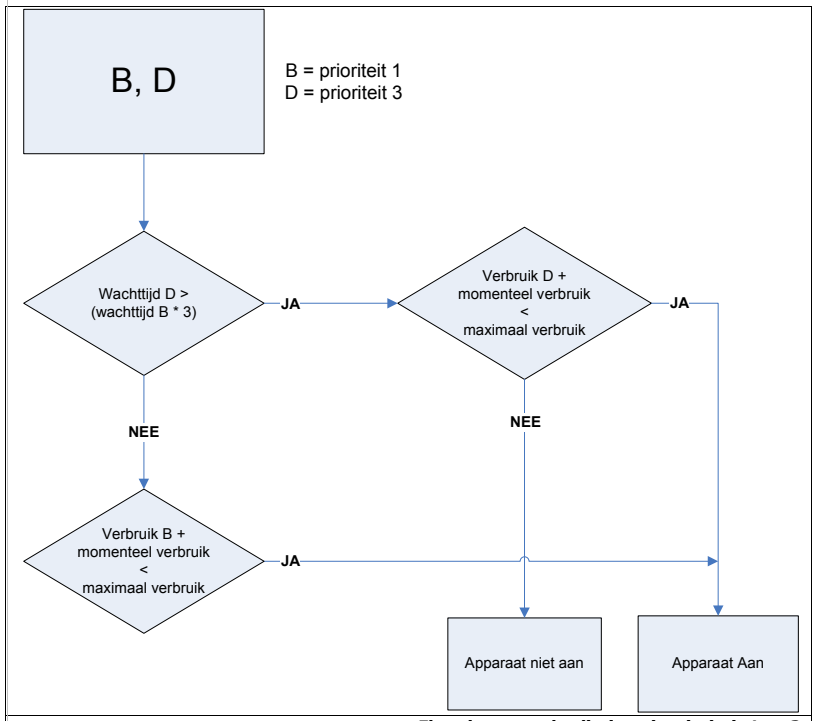
Als er besloten moet worden uit apparaten uit prioriteit 2 en 3 geldt de volgende flowchart:



Flowchart voor beslissing uit prioriteit 2 en 3

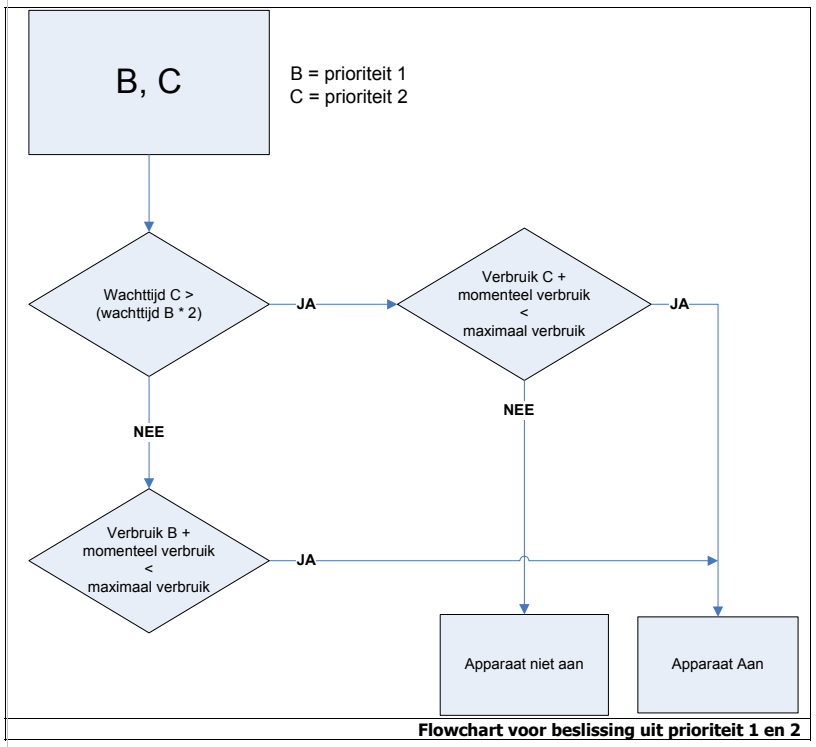
In het geval dat er besloten moet worden uit prioriteit 2 en 3, kan de wachttijd niet vergeleken worden met een apparaat uit prioriteit 1, omdat deze er niet is. Dan wordt de wachttijd vergeleken met prioriteit 2 vermenigvuldigd met 1,5.

Als er besloten moet worden uit apparaten uit prioriteit 1 en 3 geldt de volgende flowchart:



Flowchart voor beslissing uit prioriteit 1 en 3

De flowchart voor een beslissing uit prioriteiten 1 en 2 ziet er als volgt uit:

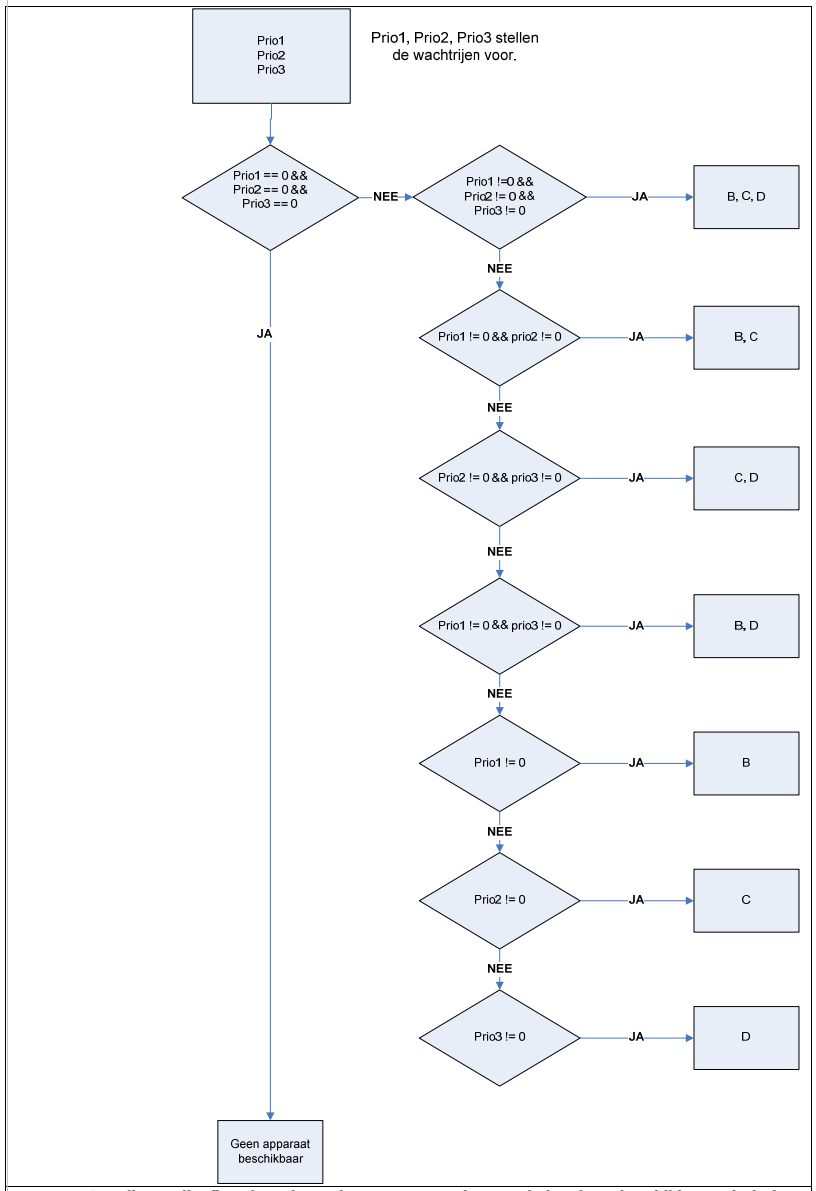


Flowchart voor beslissing uit prioriteit 1 en 2

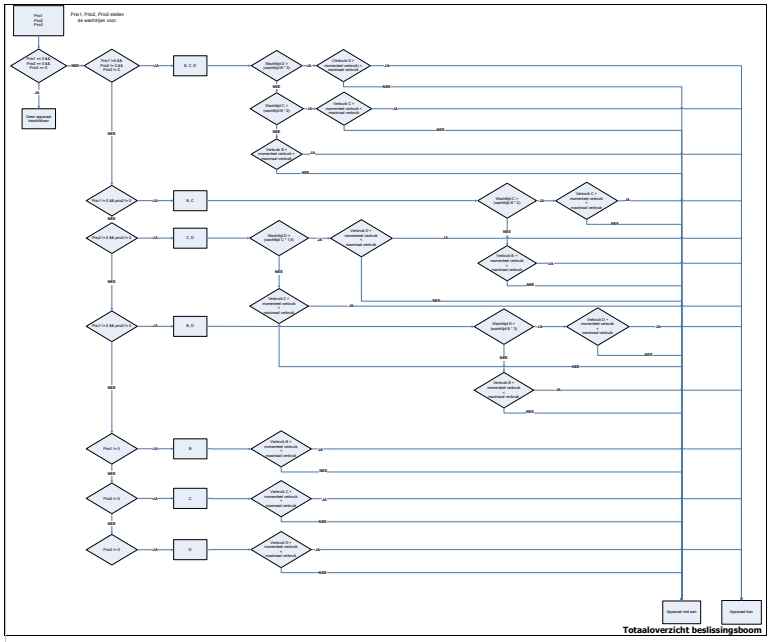
Zoals te zien is in de flowcharts, wordt er eerst besloten op basis van wachttijd in vergelijking met prioriteit 1, daarna wordt er berekend of het verbruik van het apparaat plus het momenteel verbruik niet boven het maximaal verbruik komt. Als deze laatste stap niet gedaan zou worden, dan is de kans groot dat apparaten maar voor heel korte tijd aan gaan, omdat het verbruik dan boven het maximale verbruik komt.

Als het momenteel verbruik boven het maximaal verbruik komt, is het nodig dat er één of meerdere apparaten uitgeschakeld worden. Het uitschakelen gebeurt op prioriteit. Er wordt niet gekeken hoe lang het apparaat aan staat of hoeveel het apparaat verbruikt. Er worden vanaf prioriteit 3 naar prioriteit 1 apparaten uitgeschakeld totdat het gewenste niveau bereikt is.

Om te bepalen welke flowchart doorlopen moet worden, wordt gekeken naar de grootte van de wachtrijen. Bijvoorbeeld als prioriteit 1 != 0 en prioriteit 2 != 0 en prioriteit 3 != 0 dan zal de flowchart voor B, C, D worden doorlopen.



Bepaling welke flowchart doorgelopen moet worden aan de hand van beschikbare prioriteiten



Totaaloverzicht beslissingsboom

6 Construction fase

In de Construction fase is de software geschreven en is de demo opgebouwd. In deze fase zijn alle ideeën van de voorgaande fases verwerkt in een werkende demo.

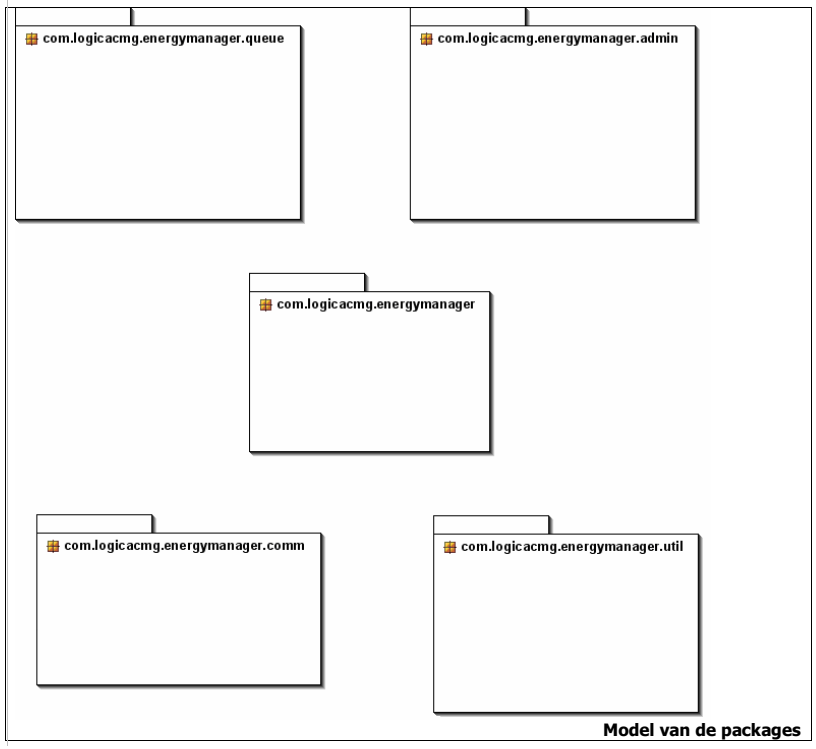
De afstudeeropdracht heeft de titel "Intelligente Meter". Dit omdat de software een meter 'intelligent' maakt. De titel van de software heeft daarentegen niet "Intelligente Meter" gekregen, omdat de software op zichzelf geen meter is. De software heeft de titel "EnergyManager" gekregen. Deze naam is gekozen omdat de software de apparaten aanstuurt (managed). De software zorgt ervoor dat de energie gemanaged wordt, door het aan- en uitschakelen van apparaten.

6.1 Ontwikkelen EnergyManager

In dit hoofdstuk zal worden beschreven hoe de software technisch in elkaar zit. Hierbij is kennis van JAVA benodigd. Een aantal termen die in dit hoofdstuk worden genoemd zijn terug te vinden in de woordenlijst.

6.1.1 Packages

De software is opgebouwd uit verschillende packages. De packages bevatten klassen (objecten) die een relatie met elkaar hebben. Zo is er een package `Comm` waarin alle klassen zitten die samen de communicatie verzorgen.



Model van de packages

Package	Functionaliteit
com.logicacmg.energymanager	Entry-point van de software en klassen voor devices, verbruik en apparaatbesturing
com.logicacmg.energymanager.admin	Grafische Gebruikersinterface
com.logicacmg.energymanager.comm	Communicatie
com.logicacmg.energymanager.energy	Klassen die verbruiksgegevens bevatten
com.logicacmg.energymanager.queue	Wachtrij

In de tabel hierboven worden de functionaliteiten van de verschillende packages kort beschreven.

6.1.2 Klassen

Bij het programmeren zijn 3 momenten geweest waar de student moeilijkheden heeft ondervonden. Dit zijn het programmeren van XmlProcessor, SerialComm en QueueManager.

XmlProcessor
Het probleem bij het maken van de XmlProcessor was dat er twee manieren zijn om XML-bestanden in te lezen en daar operaties op uit te voeren. Uit deze twee manieren moest een keuze gemaakt worden

XML
Zoals de naam al doet vermoeden wordt er gebruik gemaakt van XML.
XML staat voor eXtensible Markup Language. XML is een bepaalde manier om gegevens gestructureerd vast te leggen. Deze manier is gedefinieerd en mag iedereen gebruiken. Het is zo ontworpen dat het zowel door een programma als door een mens kan worden gelezen. XML is niet alleen geschikt om gegevens in op te slaan maar wordt de laatste tijd ook meer en meer gebruikt om gegevens via het internet te versturen.

De ondersteuning van XML in JAVA is aanwezig. Hiervoor zijn er in JAVA verschillende parsers opgenomen. De Java/ Api for XML Parsing (JAXP) maakt dit mogelijk. Binnen JAVA zijn er meerdere manieren om XML te verwerken.
Dit zijn Simple API for XML (SAX) en Document Object Model (DOM).

Verschillende parsers
Een parser is een software die bestanden kan inlezen, zodat het gebruikt kan worden door de applicatie die geschreven wordt.

Het verschil tussen de SAX API en DOM API is de manier waarop een document wordt gelezen. De SAX API leest een document stukje voor stukje, waar de DOM API een document in één keer in zijn geheel inleest. Vervolgens is de boomstructuur voor de software beschikbaar voor lezen en schrijven. Eenvoudig gezegd, is het verschil tussen SAX en DOM het verschil tussen opeenvolgende, read-only toegang (SAX), en willekeurige, lees-schrijftoegang (DOM).
Het SAX-model leest een document waarbij bij elke verandering van de opmaak een event handler moet worden aangeroepen.

De DOM API maakt gebruik van interfaces `Document` en `Node`. Een `Document` is een top-level object dat een XML-document vertegenwoordigt. Het `Document` bevat de data al een boom van `Nodes`. Een `Node` is een basistype wat een element, een attribuut of een ander soort inhoud kan zijn. De `Nodes` zijn een stukje data in de boom van het `Document`.
`Nodes` kunnen gevraagd worden om de ouders (`ParentNodes`), kinderen (`ChildNodes`) of andere aanvragen te verwerken. Het `Document` kan ook worden gewijzigd door het toevoegen of verwijderen van `Nodes`.

Keuze parser
De API waarvoor gekozen is, is de DOM API. De DOM API leest een XML-document in één keer in zijn geheel en de elementen van het document zijn te bereiken via Nodes. Vervolgens kunnen de Nodes gewijzigd worden. Voor het gebruik in de software van de EnergyManager is dit nodig. Elke keer dat een apparaat gewijzigd, toegevoegd of verwijderd wordt, moet dit verwerkt worden in het XML-document. Dat moet ook willekeurig mogelijk zijn. De ene keer wordt het 2e apparaat gewijzigd, de andere keer het 6e.

Gebruik binnen EnergyManager
De EnergyManager heeft twee XML-documenten in gebruik. Dit zijn `devices.xml` en `settings.xml`.
`devices.xml` wordt gebruikt om apparaten in op te slaan. Als de EnergyManager wordt afgesloten, hoeft de gebruiker niet opnieuw apparaten in te voeren.
`settings.xml` wordt gebruikt om instellingen in op te slaan.

De opbouw van `devices.xml` is als volgt:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceList>
  <device>
    <name>Blender</name>
    <priority>2</priority>
    <usage>50.0</usage>
    <address>0</address>
  </device>
</DeviceList>
```

Van `settings.xml` is dit de opbouw:

```
<?xml version="1.0" encoding="UTF-8"?>
<Settings>
  <priceKWh>0.05</priceKWh>
  <ipMeter>10.17.22.5</ipMeter>
  <maxUsage>250</maxUsage>
</Settings>
```

Bij `devices.xml` is alles tussen `<DeviceList>` en `</DeviceList>` het Document. Waar `<device>` en `</device>` de ParentNodes zijn en alles daartussen de ChildNodes.

Door deze manier van opslaan van instellingen in `settings.xml` is het toevoegen of verwijderen van instellingen niet veel werk. Op twee plaatsen in de software moeten aanpassingen worden gedaan: in XmlProcessor en Settings. Settings is de klasse waar de instellingen worden opgeslagen tijdens de werking van de software.

SerialComm

De klasse SerialComm verzorgt de seriële communicatie. Er is seriële communicatie nodig om de schakelaar via het basisstation aan te sturen. Er moet een commando verstuurd worden naar het basisstation, vervolgens stuur het basisstation een commando naar de schakelaar.
Het probleem hierbij is dat het basisstation gebruik maakt van ASCII. JAVA maakt echter gebruik van Unicode. Er moest een vertaalslag worden gemaakt om de commando's goed te versturen. Unicode is net als ASCII en soort alfabet voor het vertalen van een bepaalde code naar karakters. Bijvoorbeeld het karakter '2' is in ASCII '32', maar in Unicode is dit 'U+0032'.

Commando's en checksum
De commando's die naar het basisstation verstuurd moeten worden bestaan uit een reeks cijfers. Een voorbeeld van een bericht is als volgt:

```
02:26:3device:30:31:30:30:30:30:30:30:30:41:30:00:00:00:00:checksum:03
```

In het bericht staat `device` voor het apparaat welke geschakeld moet worden, en `checksum` wordt gebruikt als controle. De waarden die gebruikt worden zijn ASCII-waarden. De waarden worden hexadecimaal gepresenteerd. Zo betekent 10h het getal 10 hexadecimaal, ofwel 16 decimaal. In het voorbeeld is de eerste waarde 02h. Dit is 2 decimaal, en staat voor het karakter STX (Start of Header).

De checksum wordt berekend door alle waarden bij elkaar op te tellen. Dit resultaat modulo 100 (x % 100) geeft de checksum.

Bijvoorbeeld apparaat 1 moet geschakeld worden. Dan is de berekening als volgt:

```
02+26+31+30+31+30+30+30+30+30+30+30+41+30+00+00+00+00+03 = 24E % 100 = 4E
```

Het bericht dat verstuurd wordt, ziet er uiteindelijk als volgt uit:

```
02:26:31:30:31:30:30:30:30:30:30:30:41:30:00:00:00:00:4E:03
```

Verzenden commando
Aangezien JAVA werkt met unicode karakters kan het bericht niet zomaar verstuurd worden. De ASCII waarde van hexadecimaal 02 is namelijk anders dan de unicode van 02.
De decimale waarde van 2 in ASCII is het karakter SOT (Start of Text). Terwijl de unicode van decimaal 2 is U+0002.

Als het bericht wordt verstuurd door de JAVA-software zullen dit de standaard unicode karakters zijn. Hiervoor moest een oplossing bedacht worden.

In de software zijn de berichten opgebouwd uit een `String`. Om de verschillende waarden te kunnen scheiden van elkaar zijn er tussen de waarden ":" geplaatst.

De `String` wordt uit elkaar gehaald door de `String` te splitsen bij elke ":". Het resultaat wordt in een `String array` gezet. Om er uiteindelijk ASCII-karakters van te maken is het nodig om elke `String` in de `String array` te converteren naar een `Integer`. Op dit moment is de tekst omgezet naar een getal, een decimaal getal. Om er een hexadecimaal getal van te maken wordt het getal "ge-AND-ed" met 0xFF.
Hierna wordt het getal gecast naar een `char`. Nu is de waarde gereed om verzonden te worden naar het basisstation.

De broncode van de conversie is als volgt:

```
String mssg = "02:26:31:30:31:30:30:30:30:30:30:30:41:30:00:00:00:00:4E:03"
String temp[] = mssg.split(":");
        for (int i = 0 ; i < temp.length ; i++) {
             int n = Integer.parseInt(temp[i], 16);
             n &= 0xFF;
             char ch = (char)n;
             try {
                 os.write(ch);
             }
             catch (Exception e) { System.out.println(e); }
        }
```

Een ander probleem van de seriële communicatie is dat JAVA dit standaard niet ondersteund. Hiervoor moest een aparte API geïnstalleerd worden. Sun heeft voor JAVA wel een Communications API voor seriële verbindingen, maar sinds de nieuwste versie van de JAVAcomm API is deze niet meer beschikbaar voor Windows-systemen.

De oplossing is gevonden in een open source API die de JAVAcomm API vervangt. Deze open source API ondersteunt seriële en parallelle communicatie. Via deze API wordt de verbinding opgezet met het basisstation.

QueueManager
De QueueManager verzorgt de beslissing welk apparaat aan of uit gaat. In JAVA SE 1.4.2 is er geen mogelijkheid tot queuing, of het moet zelf geschreven worden. In JAVA 1.5 is er wel queuing mogelijk. In JAVA 5, zo wordt JAVA 1.5 ook wel genoemd, is een interface `Queue<E>` opgenomen. De `<E>` staat voor het element waaruit de queue bestaat. In het geval van de EnergyManager wordt dit dus `Queue<Device>`. Een Queue bestaande uit Devices.

Er zijn verschillende soorten queues. De interessante queue implementaties voor de EnergyManager zijn de LinkedList en de PriorityQueue. De andere soorten queues in JAVA 5 zijn de AbstractQueue, ArrayBlockingQueue, ConcurrentLinkedQueue, DelayQueue, LinkedBlockingQueue, PriorityBlockingQueue en de SynchronousQueue, hier wordt verder niet op ingegaan.

De PriorityQueue is een queue die de elementen in volgorde van prioriteit zet. Voor de EnergyManager zou dit de juiste queue implementatie zijn. Er wordt bij de apparaten gewerkt met prioriteiten.

Toch is de keuze gegaan naar de LinkedList. Deze implementatie van de queue zet de apparaten in volgorde van toevoeging in de rij. Het FIFO-algoritme dus.
De LinkedList is gekozen, omdat er voor elke prioriteit een aparte queue is gemaakt. Dit is gedaan omdat er niet alleen op prioriteit wordt bepaald welk apparaat aan mag. De beslissing ligt uiteindelijk bij de wachttijd en het vermogen van het apparaat.

6.2 Bouwen van de demo-opstelling

Om te laten zien dat de software werkt, is er een demo-opstelling gemaakt. Deze bestaat uit een aantal schakelaars, een meter, een basisstation voor de schakelaars en een aantal lampen en dimmers. En het centrale punt van de demo, de computer waarop de software draait.

Er is gebruik gemaakt van dimmers, om te laten zien dat de software reageert op veranderingen in het verbruik. Het verbruik wordt gelezen uit de meter, vervolgens wordt op basis van het verbruik besloten of er een apparaat aan mag, of uit moet.

7 Transition fase

In het Initiatiedocument staat beschreven dat in deze fase het product geïmplementeerd gaat worden. Gezien alle wijzigingen die hebben plaatsgevonden tijdens de afstudeerperiode met het product, is het product niet geïmplementeerd. De Intelligente Meter zou in het Living Tomorrow huis in Amsterdam geïmplementeerd worden tijdens de afstudeerperiode. Er wordt wel gekeken of dit in een later stadium kan plaatsvinden.

8 Evaluatie

In dit hoofdstuk zal worden beschreven wat goed ging en wat minder goed ging tijdens deze afstudeerstage. Dit is opgedeeld in een procesevaluatie en een productevaluatie.

8.1 Procesevaluatie

Procesmatig gezien is dit project minder goed verlopen.
In het begin leek het erop dat alles al geregeld was, en dat er meteen begonnen kon worden met het ontwikkelen van de intelligente meter.
Dit was helaas niet het geval. De te gebruiken meter zou de meter zijn welke in de UK gebruikt wordt om het Prepaid Metering systeem te ontwikkelen. Deze meter moest nog geregeld worden. Na twee weken werd duidelijk dat dat erg moeilijk werd, omdat de producent van de meter deze niet op tijd zou leveren. Gedurende die tijd heeft de student zich bezig gehouden met het uitzoeken wat de beste UML-tool is voor de afdeling. Ook heeft de student de "WT-CD" gemaakt. Dit is een CD waarop alle producten van de afstudeerders komen, dat gebruikt wordt als archief als de afstudeerperiode voorbij is. Ook heeft de student kennis opgedaan van seriële communicatie en het gebruik van interface binnen JAVA.

De student zal zich de volgende keer meer bezig houden met het bijstellen van de planning. Omdat de opgestelde planning na al het wachten op een meter geen overzicht heeft gegeven, heeft de student deze ook niet meer bijgewerkt. Hij had er vertrouwen in dat het zonder planning ook wel goed zou komen. Ondanks dit vertrouwen zal er bij een volgend project de planning worden bijgesteld. Zo krijgt de opdrachtgever een beter inzicht hoe de vorderingen verlopen.

Doordat pas in de 7e week van de afstudeerperiode écht is begonnen aan het ontwikkelen van de Intelligente Meter, is er ook geen Testplan opgezet. Ook als de planning wel bijgewerkt zou worden, zou dit hierop geen invloed hebben gehad.

Het ontwikkelen van de software is goed verlopen. Dit komt omdat ik voor mijzelf duidelijke milestones heb gesteld voor de ontwikkeling. De eerste milestone was het werkend krijgen van het programma zonder algoritme. Dit houdt in dat het programma functionaliteiten heeft zoals het beheren van apparaten en het wijzigen ervan. Pas toen dit af was kon er begonnen worden met het implementeren van het algoritme. Het algoritme is afhankelijk van apparaten, en kan dus niet onafhankelijk werken. Na deze milestone is het algoritme geïmplementeerd. Nadat deze bleek te functioneren is de simulatie van piekdetectie geschreven.

8.2 Productevaluatie

Na het lange wachten op de hardware waarmee gewerkt zou worden, de meter en de schakelaars, is de ontwikkeling van de Intelligente Meter voorspoedig verlopen. Gezien de requirements, opgesteld in de Initiation fase, voldoet de software daaraan. De Must en Could onderdelen zijn geïmplementeerd en er is nog extra functionaliteit toegevoegd. Zo kan er ook van prioriteit gewisseld worden.
Aan de hand van een interview is bepaald of het product voldoet aan de eisen en wensen van de opdrachtgever.

9 Verklarende Woordenlijst

Intelligente Meter	Stroommeter welke pieken in het verbruik voorkomt
RUP	Rational Unified Process. Projectmethode
Prepaid Meter	Stroommeter welke werkt op basis van vooruitbetaling
MoSCoW-analyse	Analyse waarbij functionaliteiten worden opgedeeld in prioriteiten. Volgens deze prioriteiten worden de functionaliteiten geïmplementeerd
SRS	Software Requirements Specification. Document waarin de eisen van het te maken product zijn verwerkt
Prepaid Metering	Applicatie die in de UK wordt ontwikkeld voor de prepaid stroommarkt
JAVA	Programmeertaal
JAVA-applet	JAVA-programma dat in een internetbrowser werkt
WiFi	Wireless Fidelity. Draadloos netwerk technologie
PLC	PowerLine Communication. Computernetwerk via het stroomnetwerk
Domotica	Verzamelnaam voor slimme elektronische voorzieningen in woonhuizen die het wooncomfort, de veiligheid enz. vergroten (bron: www.vandale.nl)
IDE	Integrated Development Environment. Applicatie waarin software geprogrammeerd wordt
Working Tomorrow	Afstudeerprogramma van LogicaCMG
UML	Unified Modelling Language. Taal waarmee schematisch de opbouw van een applicatie wordt weergegeven
FIFO	First In First Out. Wachtrijalgoritme
LIFO	Last In First Out. Wachtrijalgoritme
XML	eXtensible Markup Language
JAXP	JAVA API for XML Parsing
SAX	Simple API for XML
DOM	Document Object Model
API	Application Program Interface

10 Bronnen

Boeken
Praktisch UML 2e Editie

Websites
http://www.javaalmanac.com
http://java.sun.com
http://www.rxtx.com
http://forum.java.sun.com
http://www.javaworld.com

BIJLAGEN

Intelligente Meter
BIJLAGEN

BIJLAGE A
Initiatiedocument

Logica CMG
Haagse Hogeschool

BIJLAGE B
Software Requirements Specification

BIJLAGE C
Functioneel Ontwerp EnergyManager

BIJLAGE D
Technisch Ontwerp EnergyManager

BIJLAGE E
Onderzoeksverslag

BIJLAGE F
Gebruikershandleiding