MAS-ENERGY simulator

Masteroppgave Av: **Eirik Bremnes**, student nr. 194986

Veileder: Thomas Ågotnes
professor

Institutt for informasjons- og medievitenskap – Universitetet i Bergen, Norge

Abstrakt

Forskning på energi-besparende intelligente hus (heretter smart-house) er blitt mer aktuelt, blant annet fordi tilgang til nåværende energi resurser kan avta i nær fremtid. Forskningsprosjekter i Europa og ellers i verden ønsker å finne frem til nye teknologier som kan implementeres i nåværende konstruksjoner og fremtidige smart-house. Spesielt er energi-besparing med og uten alternative energikilder, automatisering, bygningsmaterialer, innemiljø, temperatur kontroll og personlig komfort mye i fokus.

Denne oppgaven handler om agenter som lærer og kommuniser i et smart-house kan føre til at energiforbruket minskes. For å vise til dette er det laget en simulator (heretter MAS-ENERGY) som bruker forskjellige type instansieringer av simulatoren. Type en hvor agenter ikke er sofistikerte (refleks-agenter, heretter kalt NO-MAS) kontra type to, et sofistikert system hvor agenter lærer og kommuniserer med hverandre (multi agent system, heretter kalt MAS).

Forskjellige parametre brukes, som historiske værdata for å etterligne ute-temperaturer for mer nøyaktige måling av bygningens varme-gjennomgangkoeffisienter eller bruk av komponenter for blant annet oppvarming av inne-temperatur. Parametre brukes i scenarioer som eksperimenterer med data og forskjellige kontekster for å finne et resultat. Ved å sammenligne resultatene kan et scenario for et MAS kontra et NO-MAS vise til en eventuell besparelse på energiforbruket.

Oppgaven konkluderer med at agenter som lærer og kommunisere ved å bruke maskinlære teknikker og observasjon, er med til å minske energiforbruket sammenlignet med agenter som bare bruker kondisjonelle regler og som ikke kommuniserer med hverandre i et smart-house.

Innhold

1 Innledning

Definisjonen på smart-house kan være bygninger som har installert autonome komponenter med programvare, også kalt agenter, som kan automatisere eller påvirke funksjoner til disse i bygninger, ved å kunne forandre på tilstanden, gjort ved hjelp av observasjoner og ved bruk av maskinlære teknikker.

Tema for denne masteroppgaven er å finne svar på om hvordan læring og kommunikasjon mellom agenter og miljøet i et smart-house kan gi god kontroll over energiforbruket. Med god kontroll menes om agentene vil kunne være i stand til å behandle problemstillinger rundt energiforbruket i reell tid når tilstanden forandrer seg. Oppgaven ønsker å vise til at bruk av MAS gir bedre kontroll over energiforbruket ved å bruke forskjellige maskinlære teknikker, kommunikasjon og parametre.

Til dette formålet er det laget en simulator i Java kode som med maskinlære teknikker finner forskjellige utfall ved å utforske med observasjon og lagrer data i eksterne filer for senere testing. Tester lages i et scenario og benytter spørsmål som skal til slutt gi en konklusjon på om det er bedre å bruke agenter som lærer og kommuniserer, MAS, eller bruk av agenter som ikke kommuniserer og bruker kondisjonelle regler, NO-MAS.

Denne masteroppgaven starter etter innledningen med først å gi i kapitel 2 bakgrunns informasjon om hva som er motivasjonen for denne masteroppgaven og sier noe om hva som er analysert før definisjon av problemstilling. Definisjon av problemstilling kommer i neste kapitel 3 og sier noe om hypotesene, forskningsspørsmål knyttet til disse og metoder for analyse og teknologi valg. Dette kapitlet danner grunnlaget for alle scenario som senere blir fremstilt i kapitel 8 om resultater fra simuleringen og videre til kapitel 9 om denne oppgavens konklusjon. Kapitlene 4 viser til forskjellen mellom MAS og NO-MAS, og hvordan systemet er planlagt. Med planlagt menes her utforming av UML, type design mønstre, kommunikasjon mellom agenter og valg av type objekter som systemet består av. Kapitel 5 handler om parametre, variabler og konstanter i simulatoren og beskriver hvordan utregning av blant annet varmegjennomgangskoeffisienten eller hvordan attributter brukes sammen med variabler og konstanter. For kapitelet 6 om AI implementasjonen handler det om hvilke maskinlære teknikker som er brukt i MAS-ENERGY. Spesielt er det lagt mye vekt på beslutnings-tre,Stuart Russel og Peter Norvig (2003, side 653) og J. Ross Quinlan (1975),og læringsprosessen som er simulatorens hoved AI implementasjon. Selv om det i denne versjonen av MAS-ENERGY ikke er lagt mye vekt på brukertilfredsstillelse som verdi for beregninger av

komfort og for energi-besparing, er det ment som et viktig parameter for fremtidige versjoner og kapitel 7 er derfor lagt til for å si noe om dette. Etter resultater, kapitel 8 og konklusjon, kapitel 9 som nevnt tidligere, kommer til slutt i oppgaven et kapitel 10 som handler om hva autor mener er viktige aspekter for fremtidige versjoner av MAS-ENERGY.

2 Bakgrunn

Fire prosjekter ble høsten 2010 studert under analysefasen for å hente inn informasjon om state of the art for smart-house simulatorer. EU prosjektet The SmartHouse/SmartGrid Vision (2010) samt Boman et. al (1999) om et smart-house prosjekt i Rønneby (Sverige) ble begge brukt som motivasjon for selve arkitekturen av hvordan et smart-house fungerer innvendig, hvilket vil si om hvordan strømtilførsel, komponenter, sensorer og andre attributter benyttes av maskinlære teknikker for automatisering av huset. To andre prosjekter ble undersøkt for å finne ut hvordan selve kildekoden til en simulatoren kunne bygges eller benytte deres kode, Cyryl Krzyska (2006) om hvordan bruke sensorer til å trigge aksjoner og Kevin Bouchard et. al (2009) som gav ideer til den grafiske bruker grensesnittet.

2.1 Smart-house og Smart-grid

Det tradisjonelle kraftnettet er i dag basert på sentralisert styring som forsyner sluttbrukere via et veletablert, enveis-transmisjon og distribusjonssystem. Prosjekter, som The SmartHouse/SmartGrid Vision (2010) ønsker å bidra med et nytt nettverksystem som skal revolusjonere energi-flyten mellom kilde og sluttbruker. Forestill som eksempel at et kraftverk i et smart-grid får problemer og må stenge tilførslen til nettet. Ved hjelp av agenter som kommuniserer med andre kraftverk i systemet, vil disse kunne erstatte leveransen av elektrisitet. Det tradisjonelle sentraliserte blir da en del av et smart-grid nettverk som forhåpentligvis skal kunne redusere karbon utslippet innen 2020. Sitat fra info brosjyren til ERA-NET (2010, side 1):

«*The proposed EU targets to be met by 2020 include the need to reduce carbon dioxide levels, improve energy efficiency, raise the share of renewable energy, and increase the level of bio-fuels in transport fuel.*»

Smart-grid er store, kompliserte systemer som krever mye kunnskap og som dermed behøver flere resurser en hva en enkelt person kan klare i løpet av en mastergrad periode. Denne oppgaven fokuserer dermed kun på hva et smart-house kan gjøre for å redusere forbruk ved hjelp av agenter som lærer og dermed bidra til energi-besparing som igjen fører til mindre karbon utslipp. Årsak for å nevne smart-grid her i oppgaven er fordi smart-house ofte blir nevnt i sammenheng med smart-grid og det er naturlig å inkludere en kort forklaring, selv om det i denne masteroppgaven dessverre ikke ble mulig å implementer i simulatoren.

Definisjonen på et smart-house er som sagt i innledningen, bygninger som har installert funksjonelle komponenter med programvare som automatiserer disse funksjonene og kan forandre

tilstanden på dem. I MAS-ENERGY er dette funksjoner slik som kontroll av temperatur i rom, automatisk lys av og på eller informasjon om varmegjennomgangskoeffisientene. Dette betyr for eksempel ved forandring av utendørstemperaturen, at individer innendørs automatisk får tilpasset romtemperaturen etter deres behov eller at romtemperatur senkes når ingen er tilstede. Videre betyr det også mulighet for kontroll av hvordan bygningsmateriale oppfører seg under bruk og viten om verdiene av transmisjons-tap eller ventilasjons-tap. Det pågår en omfattende forskning på dette området og for Europas del kan det nevnes EU prosjektet SEEDS (2011), som har formål å finne teknologivalg for fleksible systemer, som kan innpasses i eksisterende bygninger (modifikasjon og/eller implementasjon) eller i nye konstruksjoner slik som smart-house. På deres nettside sies det blant annet, sitat: "*It will aim to develop an energy management system that will allow buildings to continuously learn to maintain user comfort whilst minimizing energy consumption and CO2 emission*". Per april 2012 er prosjektet kommet over startfasen og partnere over hele Europa, hovedsaklig i Spania, England, Tyskland og Norge jobber tett sammen. Prosjektet har to pilot bygninger som brukes som laboratorium, hvor det ene ligger i Norge, Stavanger og består av flere bygninger med åpen plass mellom dem og det andre i Spania, Madrid i et større 6 etasjers kontorbygg. Det er interessant å nevne den store klima forskjellen som eksisterer mellom disse to lokalitetene, noe som gir ekstra utfordringer med tanke på blant annet varmegjennomgangs-koeffisienten.

2.2 MAS systemer

MAS systemer består av flere agenter som kommuniserer med hverandre for å løse et problem, som en enkeltstående agent eller et monolittisk system ikke kan løse. Med monolittisk system menes en programvare som ikke består av separate objekter og har en sterk kobling innad i koden. I MAS-ENERGY er betegnelsen for en *agent*, en programvare eller en del av en programvare, som *skal* brukes under simuleringen for kunne observere og eventuelt forandre på tilstanden til objekter i MAS-ENERGY. Stephen D. J. McArthur et. Al, (2007) sier om MAS, sitat:

«*MAS kan bestå av flere agenter som kommuniserer med hverandre i et program eller i flere uavhengige systemer. Felles for alle agenter, er forestillingen om at det er en agent, dets miljø og egenskapene som styrer dens autonomi* ».

Forestillingen om en programvare i MAS-ENERGY er en agent eller ikke, vises ved å implementere brukergrensesnittet for agenter. Dette vil i MAS bety det samme som å bruke en merkelapp for å identifisere et objekt, men uten å fortelle til objektet selv hva den er. Dette kan fra MAS-ENERGY vises med eksemplet for en Random agent (se 4.4.5 Random), som er en polymorfi av både et individ (se 4.5 Populasjon) og en agent (se 4.6 Agenter). Random oppfører seg som et

hvilket som helst annet individ under simulering, men kan aktiveres som en agent for å forpurre tilstander. Autonomi i MAS får en agent ved å selvstendig kunne observere sin egen verden og kommunisere persepsjonen til andre agenter i et nettverk. Hvert objekt i et slikt nettverk, eller oop har noen egenskaper med verdier som forteller noe om hva den er og hvilken nytte den har i systemet. Objektenes egenskaper og verdier danner i MAS-ENERGY miljøet omkring agenter.

2.3 MAS for smart-house: state of the Art

M. Boman, P. Davidsson og H.L. Younes (1999) beskriver et smart house prosjekt i Rønneby, Sverige, hvor et MAS er blitt installert med LonWorks (ECHELON, 2010), et system som kombinerer intelligens og kommunikasjon til diverse enheter via et elektronisk hjerteslag over strømforsyningsnettet. Om dette prosjektet sier Boman et. al. (1999,side 1) sitat :

«Our test site Villa Wega in Rønneby, Sweden, is a three-story research laboratory equipped with LonWorks and devices for communicating on the electric grid....We report here on an attempt at improving results previously obtained, by letting agents use ***automated decision support*** *when faced with situations in which* ***uncertainty*** *plays a vital role.»*

Boman et. al. (1999, side 1) beskriver multi-agenter i en intelligent bygning:

«Our hypothesis is that by equipping certain agents in a MAS controlling an intelligent building with automated decision support, two important factors will be in-creased. The first is ***energy saving*** *in the building. The second is customer value how the* ***people*** *in the building* ***experience the effects*** *of the actions of the agents.»*

Boman et. al.s (1999) rapport konkluderer med at det er mulig for agenter å operere under usikkerhet i eller deler av et smart-house og flere verktøy ble brukt for å finne gode løsninger på beslutninger, slik som Netica (2011), SMILE (2011) og Data Interactive (1968), heretter Netica, SMILE og Data. Usikkerhet i MAS-ENERGY kan være å ha for lite informasjon om hvilke utfall eller konsekvenser det finnes for en gitt handling. I prosjektet ønsket Boman et. al. (1999) å ikke begrense omfanget av en spesiell regel for å oppnå en god beslutning for en agentens oppførsel (se neste avsnitt om optimal justering) og stiller blant annet spørsmål ved hva som menes med en god beslutning, Boman et al. (1999, side 5) sitat:

"*What is meant by the best action is determined in part by the nature of the social space (sometimes called an artificial ecosystem) that the agent is in. For instance, one agent can be part of several coalitions, each of which constrains its actions considerably, while an other agent is more individualistic.....It must always be possible to over-rule the decisions of the agents in the MAS by physical interaction with the electrical equipment. For instance, even if an agent has decided*

that the light in a room should be on, it must be possible for a person to turn off the light using the switch in the actual room.»

Menneskelig oppførsel eller rasjonalitet er vanskelig å etterligne og dermed vanskelig å implementere i maskinlære. En av årsakene for å velge en induktiv maskinlære algoritme for MAS-ENERGY, beslutningstre, var muligheten for å bruke eksempler til å kunne la agenter få lære i et usikkert miljø. Boman et. al. (1999) kommer ikke med noen konklusjon i deres oppgave og sier at dette er et tema som er under forsking og i MAS-ENERGY er menneskelig oppførsel derfor lagt inn med attributter som sier noe om hvordan individet lever sin hverdag og hvor agentene i denne versjonen kun har som oppgave å spare energi uten å ta hensyn til menneskelige behov.

Om å spare energiforbruket sier Boman et. al (1999) at det er mulig å gjøre dette ved at agentene i sanntid får velge autonomt og rasjonelt mellom forskjellige alternativer (se forrige avsnitt om å begrense omfang av regler) for handling i stedet for bare å bruke regler i koden. Med regler i koden menes at det innenfor en gitt problemstilling kun finnes en løsning som er basert på kondisjonelle regler, som for eksempel en if-then setning. I følge rapporten oppnådde Boman et. al (1999) simulator i enkelte simuleringer hele 40% besparing på energi ved denne fremgangs-metoden.

Et klassisk problem som kan oppstå under en simulering som Boman et. al (1999) gjorde, er når to agenter prøver å nå sitt optimale simultant. For å finne svar på denne problemstillingen konkluderer Boman et. al. (1999) med at denne og lignende problemer løses ved å analysere og evaluere forskjellige handlings alternativer og lage en uttalelse (beskrivelse av tilstanden). En slik uttalelse kaller de en «pronouncer» og implementeres slik at den inneholder forskjellige forhåndsdefinerte modeller som skal si noe om en tenkt verden. På denne måten kan en agent bruke modellen til å forhandle med andre agenter og det kan oppnås en «enighet» om riktig avgjørelse.

Når to agenter prøver å oppnå sitt optimale og befinner seg i samme tilstand eller plassering, kan dette føre til en konflikt i beslutningsprosessen og er en av de store utfordringene i et MAS. Hvordan løser man to motstridende beslutninger? En mulighet er å lage analyser av hvordan verden ser ut omkring seg og bruke modeller som evolusjonært eller deduktivt forteller noe om sannsynligheten for et gitt resultat. Andre muligheter kan være å la en agent eksperimentere med data fra persepsjonen og under sanntid, hvor utfallet av hver test som har høyest sannsynlighet for å være beste tenkelig utfall, velges som neste tilstand.

Sitat fra Boman et. al (1999, side 5) artikkel der de snakker om å la agenter formulere beslutninger i sanntid

«*In our implementations we do not allow the agents to formulate decision situations on the*

fly, since it would be extremely difficult for them to do so. Instead, the pronouncer contain template models designed in advance for decision situations that can be presumed to occur. Each template determines the structure of a certain problem, while it is up to the agents to specify the values. How these values are set can vary with the implementation. In addition to the usual way of adding value nodes to an influence diagram, various models of prediction can be adopted»

Dette fungerer bra så lenge det **ikke** er konflikt mellom agentene eller at det **finnes** en modell som kan benyttes i en gitt situasjon, men hva skjer hvis det oppstår en konflikt i sanntid der to agenter prøver å forandre på verdiene på samme objekt eller at det ikke finnes noen modell for et gitt problem? Sitat fra Boman et. al (1999, side 2):

«*Usually, the goal of a Room agent and agents realizing user preferences in the room are conflicting: The Room agent tries to maximize energy savings while other agents try to maximize customer value. In the intelligent building domain, this is the main trade-off*».

Deres løsning inkluderer bruk av forhåndsdefinerte modeller, kalt «pronouncer», som lar agentene bruke disse modellene til å legge inn data, slik at det lages en beskrivelse av problemet og i en tenkt situasjon hvor det kan oppstå en konflikt kan agentene reforhandle og justere seg etter behov. Dette kan være i en situasjon der to individer har forskjellige verdier for temperatur og agenten for rom temperatur reforhandler med individets komfort agent for å finne passende middelverdi. Boman et. al (1999) modeller er statiske filer samlet i mapper og har ikke mulighet for å forandres under sanntid, derfor gir bruken av slike statiske modeller noen ganger ikke en løsning. Boman et. al. (1999) foreslår et alternativ til forhåndsdefinerte modeller, ved å la agentene implementere separate beslutnings-moduler, som inneholder funksjonalitet for å foreta beslutninger, men gikk vekk fra dette fordi agentenes sanntid ble for lang og programvaren for kompleks. Om hva som skjer hvis det ikke finnes modeller for en tilstand nevner Boman et. al. (1999), sitat:
«*Each template determines the structure of a certain problem, while it is up to the agents to specify the values. How these values are set can vary with the implementation. In addition to the usual way of adding value nodes to an infuence diagram, various models of prediction can be adopted*».

2.4 SIMACT

Denne 3D simulatoren er utviklet ved universitet i Quebec (Canada) og er egnet som verktøy for forskere som ønsker å lage simuleringer i et smart-house miljø. Fra abstraksjonen i Kevin Bouchard et. al. (2009), heretter SIMACT, sitat:

«*Smart home technologies have become, in the last few years, a very active topic of research. However, many scientists working in this field do not possess smart home infrastructure*

allowing them to conduct satisfactory experiments in a concrete environment with real data. To address this issue, this paper presents a new flexible 3D smart home infrastructure simulator developed in Java specifically to help researchers working in the field of activity recognition.»

Simulatorens er delt inn i tre grensesnitt, det første viser en 3D grafikk som tillater bruker å bevege seg inn og rundt i rommet, det andre er en skript sone, hvor bruker kan kontroller skript koden i sanntid og den tredje sonen som benyttes av tredje parts applikasjoner for å hente ut data fra simuleringen. Simact bruker XML filer (eXtensible Markup Language) til å lage små skript av forskjellige scenarioer som er valgt av bruker og 3D simulatoren grafikk som er svært detaljert, utfører komplekse scenario fra filene som sees på dette eksempelet:

```
<step time="5">
        <name>Open refrigerator door</name>
                <description>Open the refrigerator door to get the milk</description>
        <before>
        </before>
        <after>
                <rotation>
                        <object>Refrigerator_Door</object>
                                <database_description>
                                The refrigerator door is open</database_description>
                        <angle>-0.5</angle>
                        <axe>y</axe>
                </rotation>
        </after>
</step>"
```

Kevin Bouchard et. al. (2009, side 529) har utført forskjellige kliniske eksperimenter med forskjellige grupper individer for innsamling av empiriske data, blant annet med Altzheimer pasienter, for å få kunnskap om hvordan gjenskape situasjoner for denne gruppen individer. Disse kliniske studiene er derfor en av denne simulatorens viktigste bidrag til samfunnet, sitat:

"*Functionality is surely important, but one of SIMACT most significant contribution to the community is to come with a well defined set of scripts of activities, based on real case scenarios extracted from clinical trials.*".

Hovedsakelig er dette en applikasjon som kan hjelpe en bruker med å finne ut hvordan, for en gruppe individer, et hus kan innredes og gjøres mest komfortabelt.

2.5 Smart House Simulation Tool

I likhet med SIMACT er Cyryl Krzyska (2006) simulator, heretter Smart House Simulation Tool, en simulator med grafiske egenskaper og benytter seg av input fra bruker for å lage scenarioer. Med input menes her at bruker tegner sitt eget smart house og legger inn sensorer på ønsket posisjoner, som reagerer på bevegelser innenfor huset. Sensor kan trigge forskjellige aksjoner som er lagt inn i

et scenario som bruker har forhåndsprogrammert i simulatoren. Sitat Cyryl Krzyska (2006, side 33):

«*Scenarios may be created by an end user by means of the application, saved to external files and replayed somewhere in future. As scenarios are persisted in a user friendly XML format there may be created in any text editor and then loaded into the application.*»

Hovedsakelig er dette en applikasjon som hjelper en bruker med en parametrisering å visualisere tenkte bevegelser og handlinger i et scenario for et smart-house.

2.6 Konklusjon

Boman et. al (1999) prosjekt har vært nyttig som bakgrunn for denne simulatoren og selv om det er anvendt andre formuleringer, attributter og variabler i MAS-ENERGY er det tatt høyde for deler av deres notater. I deres simulator er det tatt i betraktning ventilasjons-tap i miljøet og brukt tilfeldig valgte verdier for beregning av utendørstemperatur, som gav tre mulige utfall for rom temperatur, sitat Boman et. al (1999, side 4):

«*The future outside temperature, which constitutes the uncertainty in this situation, is modeled as a random variable with five possible outcomes ranging from high positive difference between the outside temperature and the desired room temperature to a high negative difference. Finally, the chance node representing the final outcome of the room temperature has three possible outcomes: The temperature is higher than desired, the temperature is desirable, or the temperature is lower than desired*».

I likhet med Boman et. al (1999) er det i MAS-ENERGY tre forskjellig utfall for rommets temperatur; 1) under minimums temperatur, 2) gjennomsnittlig temperatur for alle individer tilstede og 3) under maksimumstemperatur, forskjellen er hvordan temperaturen oppnås. I deres simulasjon blir utendørstemperatur og varme fra sollys ignorert når agentene skal kontrollere temperatur, dette er ikke tilfelle for MAS-ENERGY hvor temperatur i et rom er avhengig av flere faktorer, slik som vinduer, dører og ikke minst varierende utendørstemperatur, som blant annet fra varme soldager. For å gi en mer reel simulering i MAS-ENERGY ble det inkludert transmisjons-tap, slik at sammen med ventilasjons-tap ble til parameter for varmegjennomgangs-koeffisienten. I MAS-ENERGY er det dessuten brukt værdata fra eKlima (2010) med historiske værdata og åpnet for muligheten til å variere til andre datasett eller data i sanntid fra internettet, dette er en klar forbedring fra Boman et. al (1999). som beregnet utendørstemperatur med sannsynlighetsberegning. Til sammen gir varmegjennomgangs-koeffisientene og klimadata en bedre og mer fleksibel måling og fører til mer troverdige resultater sammenlignet med Boman et. al (1999) simulator.

Boman et. al (1999) sier om konflikter mellom agenter i et MAS at det er et hoved *trade-off (her:*

negativ årsak) for smart-house og årsak for analyse og evolusjon av alternative beslutninger med mer bruk av autonome og rasjonelle agenter. Hans prosjekt viser til gode resultater med bruk av en «pronouncer» (se 2.3 MAS for smart-house: state of the Art) hvor data modellering og agenter brukes for å lage input til modellene. Dette gir beslutninger raskt og kan videre brukes ved konflikter til å reforhandle med andre agenter i systemet. Ulemper ved hans løsning er forhåndsdefinerte statiske modeller som ikke kan forandres under sanntid og dermed heller ikke brukes til uforutsette hendelser. Dette kan gi konflikter som ikke alltid kan reforhandles og dermed blir enten systemet blokkert eller datastøy oppstår (data som ikke kan tolkes). Han kommer med forslag om å lage et alternativ til disse forhåndsdefinerte modellene, ved å la agentene implementere separate moduler med funksjonaliteter som kan foreta beslutninger, men gikk fra ideen fordi programvaren ble for kompleks. Dette er et interessant tema for MAS-ENERGY som delvis har implementert dette, ved at hver agent bruker et lært beslutningstre. Dette treet bruker eksempler som er laget med logiske operatorer og disse operatorene kan ved behov forandres og nye eksempler lages, basert på tidligere tilstander og utility-funksjoner (se 6.2.1 Utility function). Forskjeller mellom disse løsningene er i Boman et. al. (1999) simulering hastigheten i sanntid, hvor forhåndsdefinerte modeller gir rask beslutning fra en «pronouncer» i motsetting til MAS-ENERGY Controller som bruker lengre tid på å beregne verdier deduktivt fra sannhets tabeller, men hvor MAS-ENERGY ikke bruker et mellomlag som Boman et. al (1999) «pronouncer» og er konfliktfri per se .

Bruk av SIMACT ble fort ekskludert fordi hvert scenario inneholder svært mange detaljer om hvordan elektriske komponenter oppfører seg under menneskelig interaksjon og MAS-ENERGY som skal vise til en energi besparelse generelt vil ved å implementere SIMACT bruke for mye datakraft for unødvendige beregninger av grafikk. En annen problemstilling med SIMACT er at selv om kildekoden er tilgjengelig så er klasser, attributter og alle kommentarer i koden skrevet på fransk og oversettelse for å sette seg inn i arkitekturen ble beregnet til å ta for lang tid for denne masteroppgaven. Utfordringen ved Smart House Simulation Tool har vært å finne kildekoden som ifølge prosjektet er free-ware og etter rimelig mange søk på nettet ble dette gitt opp som grunnlag for MAS-ENERGYs kildekode. Felles for SIMACT og Smart House Simulation Tool er at begge er verktøy som blir implementert sammen med et MAS eller annet agent basert system og har som formål å gi en visualisering av scenarioer programmert av bruker, for å teste menneskelig interaksjon med komponenter..

3 Problemstilling

Målet med oppgaven er å evaluere i hvilken grad multiagent-teknologi egner seg til å overvåke og kontrollere energiforbruket i et smart-house miljø. Med energiforbruk menes hvor mye energi som forbrukes til oppvarming, belysning og bruk av elektriske komponenter under et periode på 540 dager med forskjellige utendørs temperaturer.

Grunnhypotesen (H0) er at bruk av intelligente agenter som kommuniserer og lærer, ikke vil være medvirkende til å senke totalt energiforbruk sammenlignet med mindre sofistikerte systemer.

Hypotesen vil bli testet vha. simuleringer med en simulatoren som vil kunne parametrisere ulike typer agent system og ulike brukertyper.

> Det totale energiforbruket skal vise til hvor mye et hus forbruker enten per iterasjon eller etter endt simulering. Slik håpes det å kunne svare nei på H0 og argumentere til fordel for H1:
>
> H1 - er at bruk av intelligente agenter som kommuniserer og lærer kan senke totalt energiforbruk sammenlignet med mindre sofistikerte systemer.

For å bygge opp under H1 er det formulert noen spørsmål som det ønskes å kunne svare ja på:

Simulatoren skal inneholde agenter som vil være i stand til å observere tilstander, vurdere situasjoner ved bruk av maskinlære og kommunikasjonen. Ut fra dette formuleres spørsmål 1:

- *Spørsmål 1- Agenter som kan observere og kommunisere vil klare utfordringer slik som læring, eksperimentering og problemløsning bedre en agenter som ikke kommuniserer og er basert på kondisjonelle regler og som dermed fører til lavere energiforbruk.*

Hvordan vil et multi-agent system raskest mulig finne løsninger som gir bedre kontroll over energiforbruket? Kan det være bedre å benytte seg av konstanter og ikke variabler for å få et godt resultat og kan dette føre til uønskete situasjoner? Ut fra dette formuleres spørsmål 2:

- *Spørsmål 2 - Agent systemer med modeller som kun benytter seg av konstanter vil*

ha mindre kjøretid og gi raskere beslutninger, men vil ikke kunne returnerer fullstendig status på tilstanden for objekter slik som vedlikehold.

I avsnittet om metode diskuteres hypotesene H0-H1, spørsmål 1-2.

3.1 Metode

Resultatet fra simuleringen skal fremskaffe kvantitative data fra de ulike modellene, med data menes variablenes verdier slik som tid, tilstand, effektivitet og ytelse m.m. For å finne variablene og dermed danne grunnlaget for hva som er målbart, vil denne oppgaven foreta en operasjonalisering av hver modell. I kapitel 8 presenteres resultater fra simuleringene fra forskjellige scenarioer.

En slik operasjonalisering av hva som er målbart kan trekkes ut fra grunnhypotesen H0 og lage verdier for teknologivalg, energiforbruk og brukertilfredsstillese.

Teknologivalg I - **intelligente agenter som kommuniserer og lærer** kan motta, sende og behandle meldinger. Disse meldingene vil kunne for hver iterasjon i agentens metode-/er kunne si noe om forandring i variablene og føre til at agenten lærer noe. Teknologivalg I er en uavhengig variabel som kan manipuleres og studeres virkning av.

Teknologivalg II – agenter med regelbasert oppførsel mottar meldinger og handler etter hvilken regel som passer for gitt parameter. Teknologivalg II er en uavhengig variabel som kan manipuleres og studeres virkning av.

Teknologivalg III - mindre sofistikerte systemer er konvensjonelle hus med NO-MAS. Teknologivalg III er en uavhengig variabel som kan manipuleres og studeres virkning av.

Teknologivalg IV - Verdien av teknologien er å kunne **senke totalt energiforbruk**, som kan oppnås ved måling av energien som smart-house forbruker. Dette er en avhengig variabel som kan måles virkning på.

Det er laget en simulator som parametrisere med forskjellige agent modeller i et smart-house, et hvor kontroll styres med mindre sofistikerte agenter, NO-MAS og et annet mer sofistikerte, MAS (se 4.1 MAS eller NO-MAS en beskrivelse av to modeller). Ved å bruke scenarioer kan data fra

parametriseringen av begge typer føre til en konklusjon som kan si noe om hvilken hypotese som er korrekt. Følgende diskuteres her hvordan grunnhypotese og hypotese 1 kan besvares:

Bruk av kondisjonelle regel baserte agenter kan gi et resultat som sier noe om det totale energiforbruket og dermed gi god brukertilfredsstillese. Med god brukertilfredsstillelse menes her om individet har en økonomisk besparelse på energiforbruket eller har en god opplevelse av å oppholde seg i huset. Når disse agentene blir koblet sammen i et system og kan kommunisere med hverandre og dette fører til læring, uten at energi-forbruket minskes, kan det konkluderes med at H0 er korrekt, er tilfelle motsatt med at energiforbruket spares og spørsmålene 1 og 2 i tillegg kan besvares med et ja, er H1 korrekt.

Det er i MAS-ENERGY ikke brukt agenter med eget definert språk, men maskinlære teknikker hvor attributter og kondisjonelle regler er utført med boolsk algebra og logiske operatorer. Dette fører til bruk av deduktive metoder (hvor læring foregår med hjelp av eksempler) som beslutnings-tre (se 6.3.1 Attributter og klasser i beslutnings-tre) i MAS-ENERGYS maskinlære implementasjon (se 6.3 beslutningstre og ID3). Kommunikasjon mellom agenter utføres under sanntid med et observasjons-mønster (se 4.3.2.1 Observer) som har metoder for oppdatering av persepsjonen. Persepsjon sier noe om hvordan verden ser ut akkurat nå og agentene bruker data fra denne i maskinlære teknikker for å oppnå bedre kontroll gjennom læring. Hvis data fra MAS simuleringen viser til bedre kontroll over forbruket av energi og leder til mindre forbruk, fordi agentene lærer med kommunikasjon og maskinlære teknikker, er det mulig å svare ja på spørsmål 1.

NO-MAS agentene bruker kondisjonelle regler med konstanter og variabler for å komme frem til best mulig beslutning uten bruk av kommunikasjon. Dette kan gi svært raske beslutninger, men vil føre til tilstander som ikke er fullstendig observerbare, med dette menes at agenten ikke vet noe om andre agenter og kjenner ikke noe til utfallet av deres handlinger. MAS bruker et observasjons-mønsteret som oppdaterer objektenes tilstand under hver iterasjon og det observerbare objektet vet noe om tilstanden til alle andre objekter. Fornuftige beslutninger tas ved å bruke persepsjon og maskinlære teknikker. Fordi alle observerte objekter i MAS, kan kommunisere sin tilstand, er det mulig å overvåke alle forandringer, også forandringer knyttet til vedlikehold. Hvis NO-MAS simuleringen gir raskere resultat en MAS , men har mindre kunnskap over hvordan utføre vedlikehold med agenter, vil det være mulig å svare ja på spørsmål 2.

4 Utforming

MAS-ENERGY programvare består av mange forskjellige typer objekter som sammen simulerer et smart-house. Prosessen frem til ferdig programvare er en iterativ prosess med innsamling av kildemateriale og analyse av lignende prosjekt, kravspesifikasjon, design av klasser og attributter, skriving av kode og til slutt utføre tester. Dette kapitelet beskriver de forskjellige prosessene frem til første ferdige versjon av MAS-ENERGY med unntak av koden som er tilgjengelig som vedlegg (se 12.1 Kode) og tester av programvaren.

4.1 MAS eller NO-MAS en beskrivelse av to modeller

For å vise til besparelse av energi med bruk av systemer hvor agenter kommuniserer og lærer, er det nødvendig å kunne sammenligne forskjellige parametriseringer i samme kontekst. Med dette menes hvordan egenskapene til objektene i MAS-ENERGY varierer under iterasjonen med forskjellige typer variabler og statiske verdier. Med kontekster menes type agent eller agenter i MAS-ENERGY miljøet og deres rolle. Agentene i MAS-ENERGY kan bli instansiert med to forskjellige roller, avhengig av hvilken modell som velges for smart-house, NO-MAS eller MAS.

Hvis simuleringen ønskes testet med det mindre sofistikerte systemet NO-MAS, innebærer dette parametrisering med refleks agenter (se 6.1 Refleks-agent) som foretar endringer i miljøet ved å teste parametre med kondisjonelle regler. Kondisjonelle regler i NO-MAS er hardkodet, hvilket vil si at en *beslutning* er resultatet av tester hvor kondisjoner, slik som *if-then* setninger, tester ulike verdier med konstanter og variabler i koden. Med dette menes at regler er tenkte situasjoner for en tilstand, som for eksempel temperatur kontroll. I dette eksemplet vil konstanter være minimum eller maksimum temperatur for et rom ifølge hva termostaten er programmert til og variabler vil være aktuell romtemperatur og antall individer tilstede. En slik kondisjonelle regel med en *hviss* setning (hvis og bare hvis) kan i pseudokode se slik ut : *«HVIS romtemperatur er under maksimum OG HVIS romtemperatur er over minimum returner FALSE»*, og medfører i dette tilfelle at termostaten ikke foretar seg noe for å forandre romtemperatur.

Velges test for det mer sofistikerte systemet MAS vil samme parametrisering forekomme som med forrige modell NO-MAS, men Controller (se 4.7.1 Controller) som er hovedagent kan observere alle agenters persepsjoner uten at disse kjenner til andre agenter, slik at beslutninger som vil forandre smart-house tilstand impliserer persepsjon, læring og kommunikasjon ved å bruke maskinlære teknikker.

4.1.1 MAS

Hus som har et system med agenter som kan kommuniserer med og påvirker hverandre har i MAS-ENERGY benevnelse MAS. I MAS skjer kommunikasjon med et observerbart objekt som inneholder en liste av objekter som implementerer et observasjons-mønster (se 4.3.2.1 Observer) og metode for oppdatering av persepsjonen. Mønsteret for observasjon fungerer enkelt forklart som en samling av objekter som er knyttet sammen i et nettverk (som et spindelvev) hvor objektet i sentrum av nettverket sender ut oppdatering til alle medlemmer under hver iterasjon. I MAS-ENERGY har Controller denne rollen i dette nettverket og vet noe om persepsjonen til objektene som er knyttet til simuleringen. MAS skal foreta fornuftige beslutninger som er basert på persepsjonen og felles mål. Hva menes med fornuftige beslutninger? I MAS-ENERGY er en beslutning som ikke er motstridende med et felles mål, som besparelse av energi, en fornuftig beslutning. Dette kan vises med et enkelt eksempel være hvor et rom med et høyt transmisjons-tap og en lav romtemperatur, inneholder agenter som kun prøver å varme opp rommet men ikke foretar seg noen alternativ handling på bakgrunn av verdiene for transmisjons-tapet, som kan være så enkelt som å lukke et vindu og dermed heve romtemperaturen, uten å forbruke energi på oppvarming. Beslutninger fås ved å bruke maskinlære teknikker som i MAS-ENERGY er beslutnings-tre (se 6.3 Beslutningstre og ID3) som ved bruk av eksempler og reelle observasjoner deduktiv kommer frem til fornuftige valg. Alle valg i MAS hører til en type kategori, som er avhengig av hvilket eksempelsett (se Tabell 1:lightOnOff i 6.3.1 Attributter og klasser i beslutnings-tre) og logisk modell som er benyttet til dannelsen av beslutnings-treet, slik at agenter kan blir varslet og fornuftige handlinger utført. Kort oppsummert er MAS et system hvor kommunikasjon av persepsjoner gjøres ved bruk av et observasjons-mønster og dens oppdaterings-metode, hvor fornuftige beslutninger tas av hovedagenten ved å bruke maskinlære teknikker og data fra persepsjonen.

4.1.2 NO-MAS

Hus som har agenter som ikke kjenner til andre agenter eller kan kommunisere med dem har i MAS-ENERGY benevnelse NO-MAS (NO = ikke + MAS som ovenfor). Disse agentene bruker samme persepsjon som omtalt i forrige avsnitt om MAS, men kjenner ikke andre objekters persepsjon. NO-MAS bruker kondisjonelle regler og tar dermed noen ganger beslutninger som ikke alltid er fornuftige eller det best tenkelige utfallet. Hva er kondisjonelle regler i NO-MAS? En kondisjonell regel tester verdier ved å stille spørsmål med logiske operatorer (IF, ELSE, AND, OR osv.), slik som «hva skjer hvis x er større en y» og hvor operatorene tester variabler som gir et utfall eller en beslutning. Disse beslutningene er ikke alltid fornuftig da den kan stride mot felles mål,

fordi regelen ikke tar hensyn til andre variabler i miljøet.

I MAS-ENERGY er alle regler laget i klassen for Rules (se 12.2 javadoc *agents.interactions.Rules*) og velges ved å klassifisere type objekt og type persepsjon (se 12.2 javadoc *agents.interactions.Percept*). Rules klassifiserer hvert objekt ved å se på objektets navn og bruker dertil en liste over alle lovlige objekter i systemet. Når et objekts identitet er funnet, sendes den inn sammen med persepsjonen og blir testet av forskjellige kondisjonelle regler før neste iterasjon. Kort oppsummert er NO-MAS et system hvor beslutninger fra persepsjonen tas ved hjelp av kondisjonelle regler, uten å kommunisere dette til andre agenter eller objekter. Dette fører til noen ganger at beslutninger ikke er fornuftige eller motstridene og kan også omtales som egoistiske.

4.2 Analyse og Design

Under prosessen med kravspesifikasjonen for MAS-ENERGY ble det funnet lite litteratur eller artikler med kombinasjonen av MAS og smart-house. Det finnes prosjekter som The SmartHouse/SmartGrid Vision (2010), men dette handler mer om hvordan strømforsyning kan kombineres med alternative energikilder. Andre handler om hvordan verktøy for simulering i et smart-house kan utformes, men ikke om implementering av agenter som kontrollerer dette, slik som SIMACT, et verktøy for aktivitets gjenkjennelse i et grafisk 3D miljø.

4.2.1 Analyse

Kravspesifikasjonen har under analyse prosessen vært under kontinuerlig forandring. Forskjellige eksperimenterer med data og metoder i programkode, forandret synet på hvilke parametre som til slutt ble valgt (se 5 Parametre, variabler og konstanter i MAS-Smarthouse simulatoren) og satt sammen med ideer fra andre prosjekter som ble studert. Slik kom ideer som hvordan strukturere mapper for programkode og datainnsamling i form av XML filer eller hvordan implementere komponenter i MAS-ENERGY simulatoren.

Bruk av evolusjonær utviklings modell har gjort det mulig i utformingen av MAS-ENERGY å oppdage hvordan faser i design- eller testprosessen kan gjøres anderledes. En fase kan være en problemstilling om hvordan utforme programvaren som kontrollerer et objekt, til hvordan analyse av kvantitative data kan brukes i et scenario på en slik måte at det blir lett å overskue for en ny bruker. Et eksempel på hvordan utvikling fører til nye ideer, var et meldingssystem som opprinnelig var tenkt benyttet for å løse utfordringen med kommunikasjon av tilstander. Det ble først valgt å lage en server hvor alle objekter ble registrert ved å bruke *Java ServerSocket,* Java (2011) og sende meldinger til alle i simuleringen via server ved hjelp av små tekstfiler i Json format, Json (2011).

Json objektet inneholdt opplysninger om avsender, mottaker og en tekst streng som sa noe om tilstand for avsender objektet. I praksis fungerte dette bra, men var vanskelig å implementere i sanntid fordi objektene, som er under kontinuerlig forandring måtte generere et serialisert (et objekt som kan bli instansiert fra en fil) Json objekt for hver iterasjon og agenten måtte lese inn disse filene før beslutning kunne tas. Dette mellomlaget gjorde systemet til noe veldig uoversiktlig på grunn av antall filer og langsomt fordi prosessen med å lese inn filer tok for lang tid. Da det ble brukt iterativ metode, kunne derfor dette meldingssystemet forkastes og erfaringene bli brukt videre på å forbedre systemet som førte til observasjons-mønsteret. En god ide for å unngå unødvendig tid på ulike problemstillinger forbundet med kjente løsninger er å bruke mønstre i programmeringen. Disse mønstrene er laget for å løse problemer i en utviklingsprosess og har grunnlag i reelle situasjoner som hjelper designer til å finne løsninger (se 4.2.2.1 Observer og 4.2.2.2 AbstractFactory) som utviklingsverktøy ikke kan løse alene.

4.2.2 Design

Under utvikling av applikasjoner oppstår ofte problemer som er relatert til hvordan funksjonene i programvaren fungerer i den virkelige verden, hvilket vil si om input og output kan brukes til å gi nytte eller ikke. Design mønstre er løsninger til maskinvare design problemer og beskriver metoder for hvordan løse i en gitt kontekst et eller flere programmering problem. Disse mønstrene skal gi en utforming av rammeverket for gjenbrukbare løsninger. Mønstre kan brukes enkeltstående for et prosjekt eller kan kombineres med andre mønstre. For å bruke et design mønster, bør man kjenne til et abstrakt nivå til problemet, slik at riktig velges tidlig. I følge Erich Gamma et. al. (1995-2005, side 3), består et mønster av fire elementer,

1) **mønster navn** som skal beskrive selve problemet, løsning og konsekvensen med et ord eller to,
2) **problemstilling** som sier når man bør anvende mønsteret (forklarer problemet og konteksten),
3) **løsningen** beskriver elementet som designet består av og
4) **konsekvensen** av bruken av mønsteret. MAS-ENERGY simulator bruker to mønstre, **Observer** og **AbstractFactory**

4.2.2.1 Observer

Konsistente systemer i et objekt orientert miljø er avhengig av riktig oppdatering av objektenes tilstand og at data ikke er motsiende. Ifølge Erich Gamma et. al. (1995-2005, side 293) er Observer mønsteret, sitat: «*Define a one-to-many dependency between objects so that when one object*

change state, all its dependents are notified and updated automatically». For MAS-ENERGY må data som sier noe om tilstanden til et objekt være observerbart. For å implementere dette er det i MAS-ENERGY brukt brukergrensesnittet Java Observable (2011) som har tre viktige metoder: **addObserver, notifyObservers**,og **update**. Java Observable (2011) bruker tråder for å binde objekter sammen med hverandre via det observerbare objektet, som i et nettverk eller hierarki ved å bruke *addObserver* metoden. Tråder i Java er en struktur som den virtuelle java maskinen bruker for å eksekverer programkode etter prioritet prinsippet og i Java Observable (2011) kalles alle medlemmer fra det observerbare objektet for å be om en ny tilstand fra deres *update* metode. Det observerbare objektet i MAS-ENERGY simulatoren er Controller (se 6.4.1 Controller). Hvordan kaller Controller alle medlemmer i systemet? Objekter er bundet sammen i et hierarki hvor Controller er forbundet med dem via tråder, som i en server. Når metoden *notifyObservers* blir eksekvert vil Controller sende ut et kall til alle medlemmer som har implementert brukergrensesnittet for Java Observable (2011) og er lagt til listen over objekter i Controller *addObserver* metode som gir en tett kobling innad. Erich Gamma et. al. (1995-2005, side 294) sier da også om bruk av observer-mønsteret og tett kobling, sitat: *«When an object should be able to notify other objects without making assumptions about who these objects are. In other words you don't want these objects tightly coupled»* og for Java Observable (2011) er dermed alle objekter knyttet til Controller som er det eneste objektet som vet noe om eksistenes til andre objekter.

4.2.2.2 AbstractFactory

Behovet for å lage forskjellige instansieringer av samme klasse, eller lage en gruppe av objekter som tilhører samme familie i MAS-ENERGY, gjør AbstractFactory til et mønster som passer fint til dette formålet. Ifølge Erich Gamma et. al. (1995-2005, side 87) er AbstractFactory, sitat: *«Provide an interface for creating families of related or dependent objects without specifying their concrete classes».* MAS-ENERGY har flere typer familierelaterte objekter, slik som individer, agenter og komponenter. Dette er en fordel når forskjellige scenario skal programmeres, fordi man kan ha en fast ramme rundt hvert smart-house uten å måtte skrive om koden (re-fakturere) og dermed benytte gjenbruk av klasser. I MAS-ENERGY er en ramme kalt plot (engelsk: tomt) og klassen Plot (se 12.2 javadoc *plotmatrice.Plot*) brukes for å etterligne en byggegrunn, med fire hjørner og et smart-house med 4 rom. Klassen holder også alle abstrakte forekomster av vindu, dører, komponenter og individer. Dette mønsteret gir forskjellige fordeler for et smart-house som spart tid ved gjenbruk av kode, mulighet for lettere utvidelse til mere komplekse bygninger eller kobling av flere smart-house i en større modell og isolering av bestemte klasser som gir bedre manipulasjon fordi all handling av disse klassene skjer gjennom abstraksjonene.

4.3 Kommunikasjon mellom agenter, objekter og persepsjon

Persepsjonen er en tolkning av hvordan verden ser ut akkurat nå og agenter kan ved å kjenne til persepsjoner utføre handlinger. Percept klassen (se 12.2 javadoc agents.interactions.Percept) inneholder metoder for instansieringer av persepsjonen (se illustrasjon 1) og inneholder informasjon om objektet.

```
Percept [percept=model.houses.vestlandshus@80fa6f,
rules=agents.interactions.Rules@1b9ce4b,
behaviourPercepts=[]]
```

Illustrasjon 1: Persepsjon fra vestlandshus

I MAS-ENERGY er det Controller som har oppgave med å be om oppdatering fra medlemmene i nettverket og gjøres en gang per iterasjon (se 4.2.2.1 Observer). Hvordan fungerer så denne formen for kommunikasjon i praksis? Et eksempel med forandring av romtemperatur, hvor rommets oppdatering metode sender en persepsjon som inneholder verdien for romtemperatur. Controller mottar oppdateringen sammen med alle andres objekter persepsjoner og sender ut en notifikasjons-melding til medlemmene. Termostat agenten (se 12.2 javadoc components. ImplementedAgentBehaviour. ThermostatActuator) som befinner seg i rommet mottar melding om notifikasjon og utfører en sjekk for å se om den bør foreta justeringer av temperatur, ved å aktivere eller ikke oppvarming. Alle verdier som blir justert, vil ved neste iterasjon igjen bli oppdatert og ny persepsjon sendes ut. Kommunikasjon mellom agenter i MAS-ENERGY foregår kun fra Controller og med to mulige typer. Type en er NO-MAS hvor Controller ikke kobler inn maskinlære teknikker, men sender direkte ut beskjed om notifikasjon og oppdatering til alle medlemmer. Medlemmer som er refleks-agenter bruker persepsjon til å foreta justeringer med kondisjonelle regler (se 4.1.2 NO-MAS). Type to er MAS hvor Controller kobler inn maskinlære teknikker og setter tillatelser direkte til alle impliserte refleks-agenter i systemet, avhengig av utfallet fra maskinlære algoritmen. Først etter at notifikasjon og oppdatering er utført vil agentene foreta handling og sende ut ny persepsjon.

4.4 Utforming av et smart-house

I MAS-ENERGY er utforming lagt opp etter hva et vanlig bolighus bør ha av rom. Inndelingen er derfor lagt inn i et hierarki hvor hus er på topp, fulgt av rom, vegger, dører, vinduer og komponenter. Det er brukt AbstractFactory mønster for å lage abstrakte klasse for gjentakende moduler i huset.

4.4.1 House og plot

Hus er bygget på en tomt og House klassens (se 12.2 javadoc model.abstracts.House) metode

setPlot (se 12.2 javadoc plotmatrice.Plot) holder på et koordinatsystem med x (bredde) og y (høyde) og dermed husets plassering i på tomten. Plot.java klassen (se 12.2 javadoc plotmatrice.Plot) inneholder forekomster som sier noe om hvordan huset er bygget, hvilket vil si slik som yttervegger, hjørne-pæler og rom. Det er viktig å påpeke at MAS-ENERGY simulatorens husmodeller ikke har takkonstruksjon spesifisert fordi parametre som skal måle varme- og ventilasjons-tap er lagt inn i vegg, vindu og dører.

4.4.2 Room

Abstrakt klasse som beskriver et rom med fire vegger, dør og vindu. Alle rom blir instansiert med en varmegjennomgangskoeffisient (les u-verdi) på 0.415 W/m2K. som følger forskrift fra Lovdata (2010), *Forskrift om krav til byggverk og produkter til byggverk (TEK)*. U-verdier måles i W/(m²K) (W = Watt, m²K = grader Kelvin per kvadratmeter) og angir mengde varme som pr. tidsenhet skal passere en kvadratmeter av konstruksjonen med en temperaturforskjell på én grad kelvin mellom konstruksjonens to sider. Fra forskriftens paragraff 8-21 brukes verdiene:

- U-verdi yttervegg: 0,18 W/m2K.
- U-verdi glass/vinduer/dører: 1,2 W/m2K som gjennomsnittsverdi inkludert karm/ramme.

Under simuleringen kalkuleres varmegjennomgangskoeffisienten med formlene:

«Transmission loss through buildings materials PT = A × U × (tRom – DUT)»

Hvor **A** er konstanten for rommets kvadratmeter, **U** konstanten er u-verdien for rommet, **tRom** er rommets nåværende temperatur og **DUT** er utendørstemperaturen.

«Ventilation loss through isolation Pv = q × c × r × (trom – DUT)»

Hvor **q*** er dimensjonert utendørs luftstrøm, **c*** er varmekapasitet, **r** er luft tetthet (satt fast til 1.5) **tRom** er rommets nåværende temperatur og **DUT** er utendørstemperaturen.

* MAS-ENERGY bruker sannsynlighetsberegning for denne variablen.

Varmegjennomgangskoeffisienten bruker metodene *transmitionLosses()* og *ventilationLosses* () til å utføre disse kalkylene (se 12.2 javadoc service.interfaces Interface.EnergyLosses)

4.4.2.1 Bedroom

Individer velger ved første iterasjoner sitt soverom, hvor det skal befinne seg når det sover (ikke aktiv og hjemme), når det våkner (aktiv og hjemme) og når det bader . Soverommet har belysning, vindu, dør og ovn.

4.4.2.2 Kitchen

Individene lager mat på forskjellige tidspunkt avhengig av parametriseringen av deres spise variabler (frokost, lunsj og middag). Kjøkken har belysning, vindu, dør, ovn og diverse komponenter.

4.4.2.3 Living room

Individene oppholder seg utenom spise og sove aktiviteter i husets stue. Stue har belysing, vindu, dør, ovn og fjernsyn.

4.4.2.4 Hall

Individers bevegelser foregår fra rom til andre rom via gang. Gang har belysning og ovn. Merk at midtgang som i simulatoren er instansiert som død sone, hvilket vil si at når individer forlater huset, så blir disse plassert i dette punktet og hvor ikke noen form for energibehov kalkyler blir utført.

4.4.3 Exterior Window

Vindu fra soverommet, stue eller kjøkken ut mot gateplan. Kan åpnes, lukkes og forandre u-verdiene til rommet eller seg selv og temperatur mellom ute og inne.

4.4.4 Internal Door

Dør fra rom til gang. Kan åpnes, lukkes og forandre temperatur mellom rom.

4.5 Populasjon

Definisjonen på en populasjon i statistikk er en samling av individer eller objekter som kan bli satt i en menneskelig kontekst under et studie. I MAS-ENERGY er individer abstraksjoner av isPerson klassen (se 12.2 javadoc population.abstracts.isPerson) og har attributter som sier noe om komfort, behov og rutiner for en person. Alle forekomster av et hus har fra 2 til 4 individer som er valgt slumpmessig mellom 5 typer individer: Terra, Working9 to 5, Unemployeed, Lazy og Random.

Noen individer kan spille rolle som «korrupte» agenter og med dette menes at disse kan være motstridene mot felles mål som individer kan ha i et hjem. Med felles mål i MAS-ENERGY menes å kunne spare energiforbruket. Et eksempel på en slik «korrupt» agent er Random (se 4.5.5 Random).

4.5.1 Terra

Terra er et individ som har alle egenskaper som forventes av en god beboer i et smart-house. Hva er

en god beboer? For å kunne sammenligne resultater individ mot individ, er det nødvendig med en beboer som oppfyller alle krav om god helse og får høy verdi ved hver iterasjon. Terra blir nesten aldri missfornøyd, eller mister jobben ved dårlig helse.

4.5.2 Working 9 to 5

Working 9 to 5 har faste morgen rutiner og forlater huset tidlig for å gå på arbeid. Etter endt arbeidsdag får dette individet enkelte privilegier som gir ekstra helse verdi. Hvis derimot helse verdien synker kan jobb status forandres til arbeidsløs. I likhet med morgen, har kveld og natt sine faste rutiner som ikke bør forandres under iterasjonen.

4.5.3 Unemployeed

I likhet med Working 9 to 5 har dette individet faste rutiner for hverdagen, men kan komme hjem til lunsj etter jobb søk. Unemployeed finner aldri en jobb, men prøver å ha en behagelig tilværelse i smart-house. Unemployeed kan ha rutiner om natten som fraviker fra Working 9 to 5, slik som å stå opp og føle seg sulten.

4.5.4 Lazy

Dette individet har liten tiltakslyst og er ute av smart-house i korte perioder. Den legger seg sent etter å ha sett fjernsyn neste hele dagen og lar ofte komponenter være aktive. Det verste den vet er å bli vekket for tidlig og i et kaldt soverom.

4.5.5 Random

Random er ikke bare et individ men kan også få rolle som «korrupt» agent (se 4.6.3 Random). Denne agenten kan ha som oppgave er å forandre verdier i MAS-ENERGY som strider mot felles mål. Den sover nesten aldri, kan bevege seg fritt rundt og går aldri ut. Den kan forandre på smart-house modellenes verdier eller på temperaturer i et rom. Random kan ikke deaktiveres, men kan settes som ikke aktiv eller miste rettighet til å agere av Controller agenten. Metoder som den bruker kan være å justere på rommets vindu slik som vist her i dette kodeeksemplet:

```
if (rand.nextDouble() < .001)
            {
            room.getWindow().open();
            return Action.ON;
            }
if (rand.nextDouble() < .001)
            {
            room.getWindow().close();
            return Action.OFF;
```

}

4.6 Agenter

MAS-ENERGY har agenter som agerer rasjonelt og en definisjon på en rasjonell agent er ifølge Stuart Russel et. al. (2003, side 34), en som fungerer for å oppnå det beste resultatet, eller ved usikkerhet, det beste *forventede* resultatet. I smart-house er en agent en programvare som kan observere en tilstand, som eksempel seg selv via en persepsjon eller andres persepsjoner og utføre en handling som skal føre til det best oppnåelige resultat.

4.6.1 Controller

Controller er en agent som er observerer smart-house og kan oppdatere tilstander ved å kalle på medlemmer i observer-mønsteret med deres oppdaterings metode. I et MAS bruker den maskinlære teknikker for å finne rett beslutning. Den mottar alle persepsjoner fra objekter som er registrert i huset og avgjør om hvilken handling som gir best oppnåelig resultat for en gitt tilstand. Controller i et smart-house som ikke har MAS aktivert, fungerer kun som observerbart objekt for alle observerbare objekter (se avsnitt 4.2.2.1 Observer). Controller kan gi eller fjerne rettigheter for alle type agenter, til at disse selv kan utføre handlinger.

4.6.2 Actuator

Refleks-agent (se 6.1 Refleks-agent) som utfører i en gitt tilstand og med data fra en persepsjon, en serie handlinger som er basert på kondisjonelle regler, med aktuatorer for å oppnå det beste oppnåelige resultatet.

4.6.3 Random

Har som oppgave å forandre på objektene i simuleringen. Denne type agent skal slumpmessig gå inn å forandre på komponenter i et smart-house og innvirke på forandringer ved å utgi seg for å være et individ (se 4.5.5 Random).

4.7 Komponenter

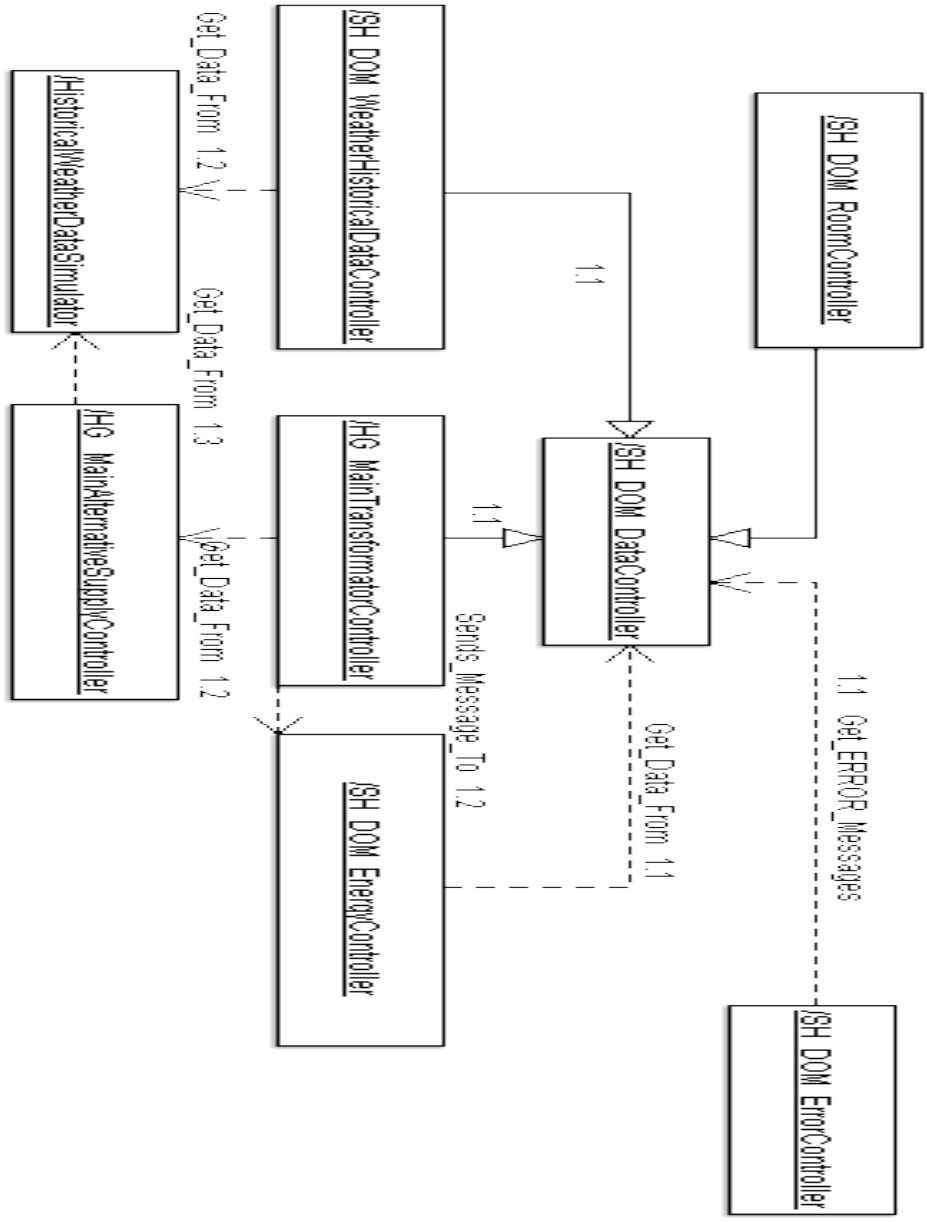
Komponenter konsumerer energi eller fungerer som et mellomlag for en handling utført av en agent. Alle komponenter med forbruk av energi har en variabel for kilowatt per time forbruk og metoder for å være aktivert eller deaktivert. Ikke alle komponenter er brukt i denne versjonen av MAS-ENERGY.

- Airconditioner aktiveres av Controller hvis temperaturen i et rom er av en slik karakter at vanlig utluftning ikke er tilstrekkelig for å oppnå en behagelig temperatur.

- AlarmClock er laget for fremtidige simuleringer og skal fortelle noe om når et individ skal gjøre en handling som krever en tidskontroll.
- CoffeeMaker er en maskin som lager kaffe og brukes når et individ lager måltid etter parametrene frokost, lunsj eller middag.
- DishWasher aktiveres av Controller hvis det registreres at et individ har laget mat.
- Frigde aktiveres av Controller hvis det registreres at et individ lager mat.
- Heater plasseres i rom og hvor rommet kan inneholde flere komponenter av denne typen. Heater har som funksjon å varme opp rommet og aktiveres av en ThermostatActuator agent enten direkte eller på kommando fra Controller agenten.
- Lamp_40W er små lamper som oftest finnes i soverom eller i stuen. Disse aktiveres av Controller agenten hvis det er et individ tilstede og behovet for lys eksisterer.
- Microwave aktiveres av Controller hvis det registreres at et individ lager mat.
- RoofLamp er lamper som oftest finnes i gang eller i kjøkken. Disse aktiveres av Controller agenten hvis det er et individ tilstede og behovet for lys eksisterer.
- Shower aktiveres av Controller hvis et individs ønsker et bad.
- Stove aktiveres av Controller hvis det registreres at et individ lager mat.
- Television aktiveres av Controller hvis et individs ønsker å se fjernsyn.
- WashingMachine aktiveres av n iterasjoner i MAS-ENERGY for å simulere bruk av vaskemaskin.

4.8 UML

I oppgaven er det brukt use-case og klasse-diagrammer. Samarbeids diagrammer (illustrasjon 2) ble brukt for å lage modellering av dataflyten og for å legge grunnlag for utviklingen av klasse-



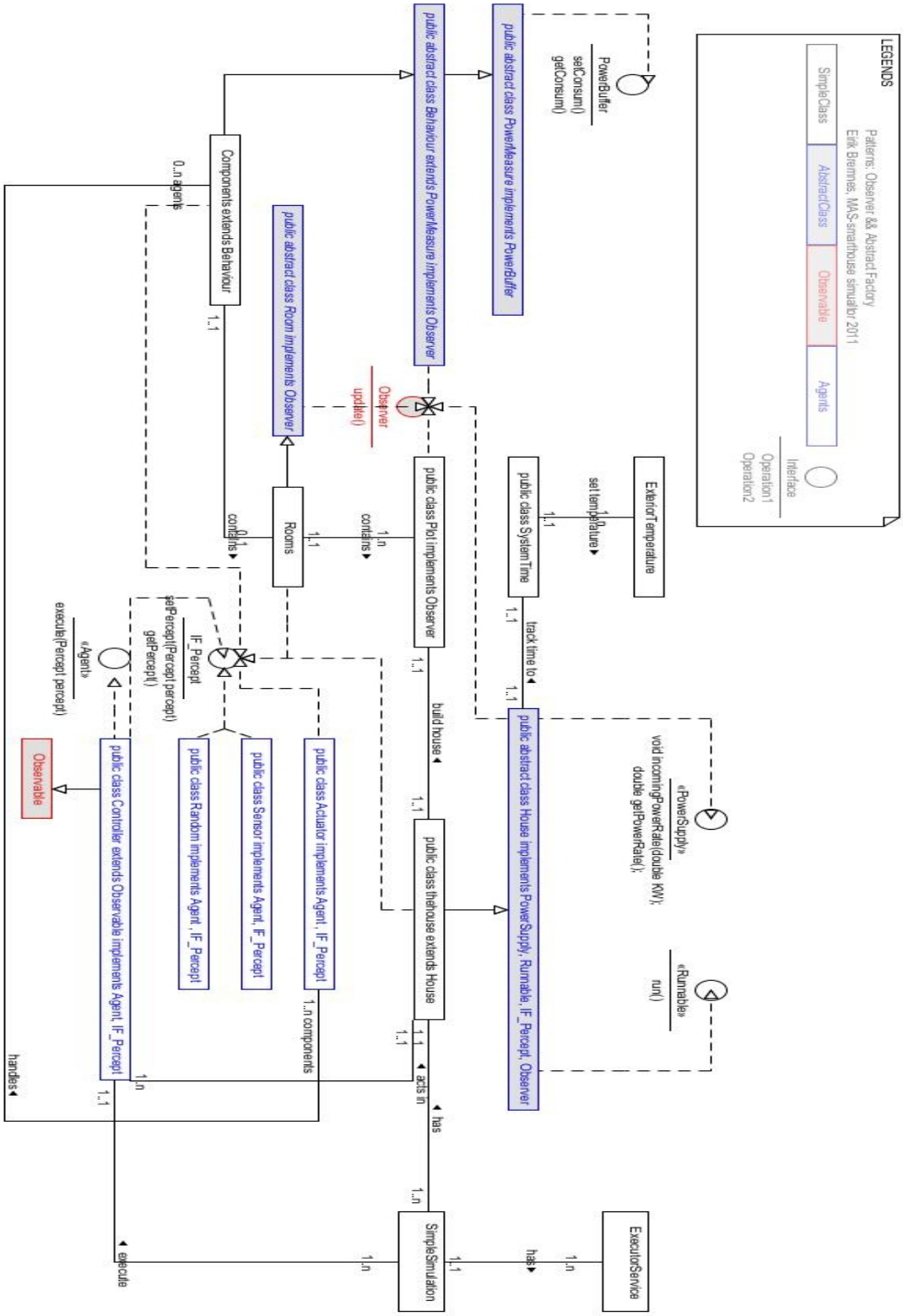
Illustrasjon 2: Figur som viser SD for Controller agenter

diagrammer.

I januar/februar 2011 ble samarbeids diagrammene forkastet fordi det førte til komplisering av systemet, årsaken til dette kan sees på illustrasjon 2 hvor antall Controller agenter er oppe i hele 8 instansieringer, mot 1 i dagens system!

Klasse diagrammene ble produsert for å kunne visualisere hvordan systemet ville løse de

forskjellige use-case scenarioene som kom frem etter hvert som parametre ble tydeligere. Følgende forklares eksplisitt klassediagrammene og det gis noen eksempler på use-case. Innledningsvis starter forklaring med eksekvering av systemet (se illustrasjon 3) hvor SimpleSimulation.java eksekverer en forekomst av Controller.java ved asynkront å bruke Java grensesnittet ExecutorService. ExecutorService har metoder for tråd basseng (engelsk: thread pool) slik at oppgaver fordeles på datamaskinen minne og åpner muligheter for innvirking på simuleringen under sanntid. En Controller opererer i et hus men skal ikke bestemme hva dette huset inneholder eller hvilken type styringssystem det har. Noen hus er MAS mens andre kan være mindre sofistikerte NO-MAS, i enkelte tilfeller er huset nøytraliserte, hvilket vil si de ikke inneholder noen form for agent systemer (verdien NULL brukes)(illustrasjon 4).



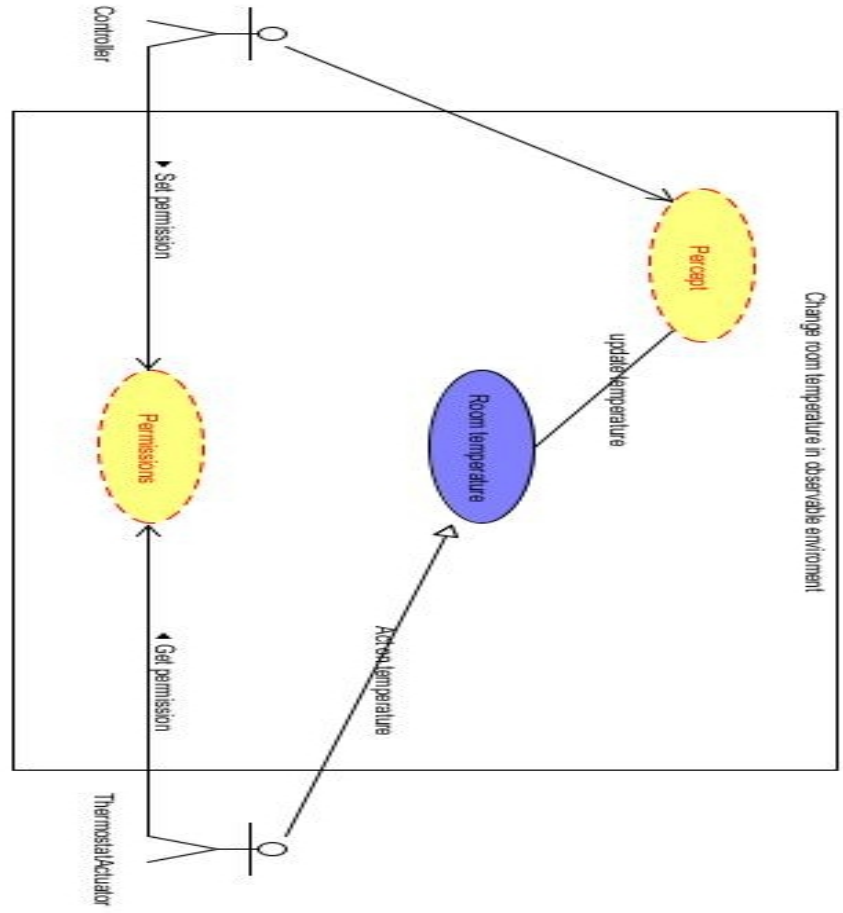
Illustrasjon 3: Klasse diagram MAS-ENERGY

Som illustrasjon 4 viser er Controller eksplisitt satt inn i to modeller av House (hus1 og hus2) mens en modell (hus 3) ikke har noen forekomst av klassen. Det oppstår derfor en uavhengighet mellom

```
Controller non_softicated_house = new Controller(false);
Controller softicated_mas_house = new Controller(true);
// vestlandhus er forekomst av House
vestlandshus hus1 = new vestlandshus(non_softicated_house, 0.0, 0.0,0.0, 0.0, new SystemTime());
vestlandshus hus2 = new vestlandshus(softicated_mas_house, 0.0, 0.0,0.0, 0.0, new SystemTime());
vestlandshus hus3 = new vestlandshus(NULL, 0.0, 0.0,0.0, 0.0, new SystemTime());
```

Illustrasjon 4:Kode fra SimpleSimulation klassen viser avhengighet mellom Controller og House

SimpleSimulation og Controller samt en uavhengighet mellom SimpleSimulation og House. Som illustrasjon 3 viser har Controller implementert grensesnittet for Observable (se 4.3.2.1 Observer) og metoden *setHouse(House)*(se 12.2 javadoc model.abstracts.House) i Controller lager en

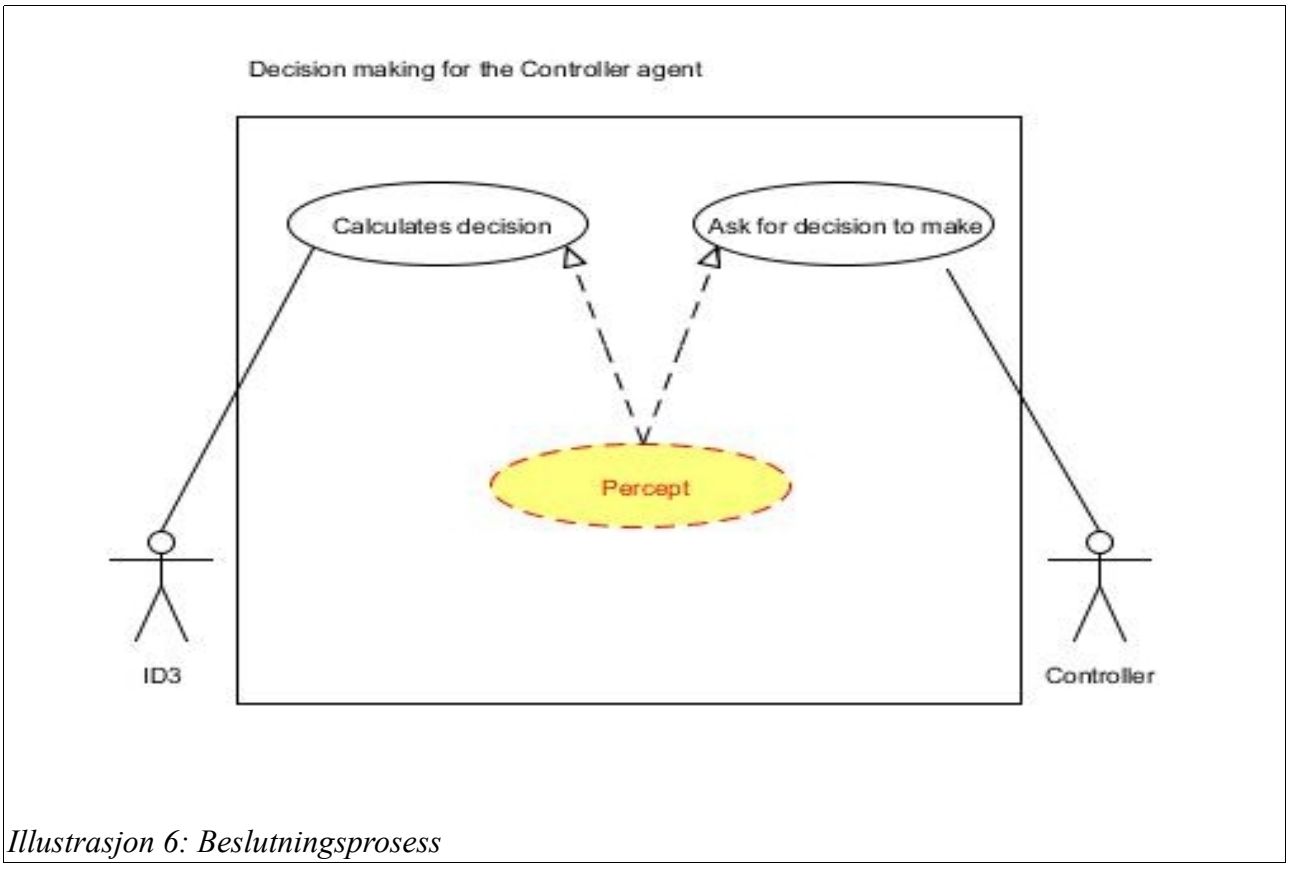


Illustrasjon 5: Temperatur kontroll

avhengighet mellom husets persepsjon og agentens mulighet for å observere forandringer i huset.

Persepsjon er et sentralt begrep i MAS-ENERGY og gjennomgående for alle objekter som forandrer tilstanden. Uten å vite hva som skjer akkurat nå kan ikke MAS-ENERGY utføre noen form for operasjonalisering og illustrasjon 5 viser et eksempel med use-case over temperatur kontroll, hvor persepsjonen forteller noe om tilstanden til rommets temperatur. Agenten vil alltid prøve å få forandret tilstanden til objektet ettersom den er regelbasert, mens det er Controller som gir den tillatelse til å bruke disse reglene eller bruke en annen beslutning. En slik beslutning er et utfall av persepsjon som Controller har mottatt fra objektets oppdaterings metode og returnert fra beslutningstre eller kondisjonelle regler. Illustrasjon 6 viser hvordan Controller lager en spørring i systemet ved å bruke persepsjon fra objekter og hvor beslutninger kommer fra ID3 (se 6.3 Beslutningstre og ID3) .

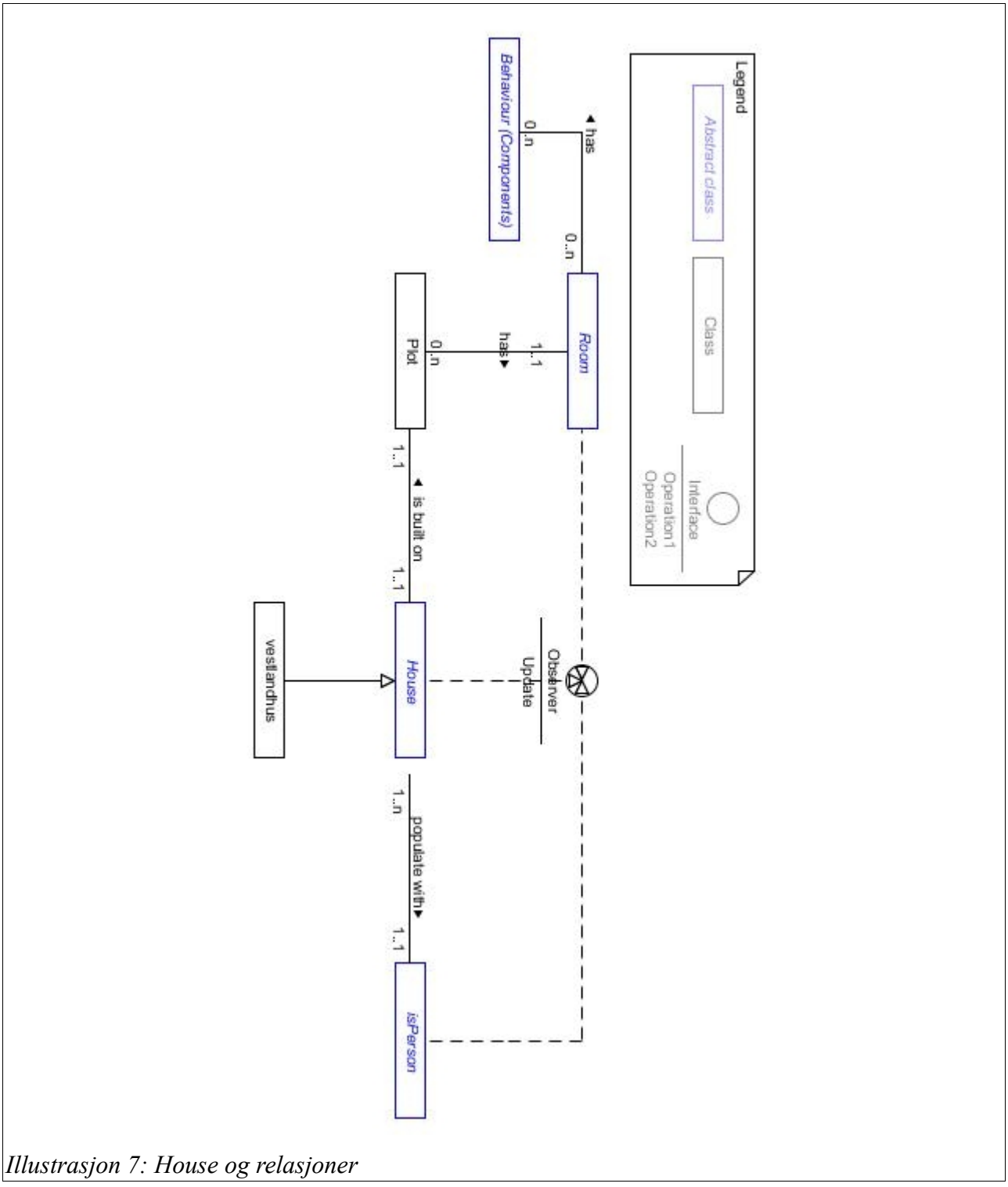
Et hus bygges på en tomt og har rom med dører, vinduer, komponenter og ganger mellom disse. Klassene Plot, Room og Behaviour (se 12.2 javadoc) representerer i MAS-ENERGY disse



Illustrasjon 6: Beslutningsprosess

aspektene. I simuleringen inneholder disse klassene variabler som sier noe om tilstanden til huset. Illustrasjon 7 viser klasse-diagrammet for House klassen og de mest sentrale relasjonene. Som observeres er forekomsten av Vestlandshus (se 12.2 javadoc model.houses.vestlandshus) en polymorfi (arv) av House og holder under simuleringen navn og verdi for en modell. House instansierer en forekomst av Plot hvor alle rom implementeres statisk og som Vestlandshus implementer i House ved å bruke metoden *setRooms(aRoom)* for å lage rom. Også komponenter

som er hardkodet for hver type rom, blir implementert i hver enkelt polymorfi av Room. Individer som simulerer en befolkning blir laget med sannsynlighetsberegning av antall og type i House metoden *populate()* som bruker polymorfisme av typen isPerson (se javadoc 12.2 population.abstracts.isPerson).

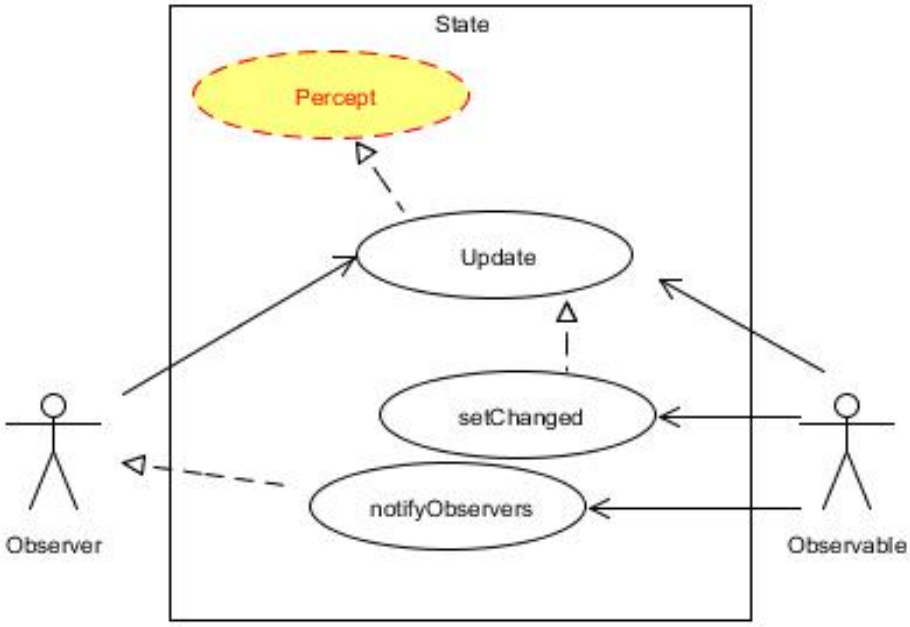


Illustrasjon 7: House og relasjoner

Bruk av abstraksjoner (se 6.2.2.2 AbstractFactory) er sammen med Observer mønsteret (se 6.2.2.1 Observer) utbredt i MAS-ENERGY og illustrasjon 7 gir et godt eksempel på hvordan klasse-

diagram kan vise sammenhengen mellom klasser i et observasjons-mønster.

Alle klasser som har en observerbar tilstand og med uavhengig parametrisering i MAS-ENERGY er knyttet til grensesnittet for Observer og metoden for oppdatering *update()*. Illustrasjon 8 viser use-case for hvordan systemet oppdaterer seg selv under sanntid, ved at objekter bruker metoden med data fra persepsjonen og Controller setter alle forandringer og varsler alle medlemmer om ny tilstand. Legg merke til at også Controller har sin egen metode for oppdatering for å kunne delegere oppgaver til andre agenter i miljøet.



Illustrasjon 8: Use case Observer mønster

5 Parametre, variabler og konstanter i MAS-Smarthouse simulatoren

For å oppnå en forståelse av et problem i en kontekst som inneholder variabler og konstanter, brukes betegnelsen parameter. I MAS-ENERGY blir disse brukt under scenarioene (se 8.1 Scenarioer) for å finne svar på spørsmål som stilles og her presenteres noen av de mest sentrale:

5.1 Gjennomsnittlig romtemperatur

Gjennomsnittlig temperatur i et rom basert på antall individers ønsket temperatur konstant fordelt på antall individer tilstede. Må ikke forveksles med rommet temperatur.

$\overline{TRom} = \sum INDtemp \ / \ \sum INDroom$

Hvor TRom er gjennomsnittlig temperatur, INDtemp er et individs gjennomsnitt ønsket temperatur og INDroom er individet som befinner seg i rommet.

5.2 Transmisjons-tap

Varmetap som skyldes bygningsmateriale er beregnet med formelen

$PT = A \times U \times (tRom - DUT)$

Hvor *A* er antall m2 rom flate, *U* er rommets u-verdi, *tRom* er rommets temperatur og *DUT* er utendørs temperatur.

5.3 Ventilasjons-tap

Varmetap som skyldes ventilering, slik som åpne dører eller vinduer er beregnet med formelen

$Pv = q \times c \times r \times (trom - DUT)$

Hvor *q* er dimensjonert utendørs luftstrøm, *c* er varmekapasitet, *r* er tetthet, *tRom* er rommets temperatur og *DUT* er utendørs temperatur.

5.4 Energiforbruk og system-tid

Hvor mye energiforbruk er det per iterasjon (`Iteration`) regnet i kilowatt per time, målt for hver hele time ($\sum(kWh * 60minutes * 24hours)$) i *t* periode (`t`).

$kWh = \sum(kWh * 60 * 24) \ / \ Iteration \ ti...tn$

5.5 Oppvarming og energiforbruk

Hvor stor mengde energi forbrukes på oppvarming av et rom målt i kilowatt per time.

$kW/h \ M2 = \ M2 * heatcapacity * ALFAtemp \ / \ T$

Hvor M2 er antall kvadratmeter, heatcapacity er elementets varmekapasitet, ALFAtemp er differansen i temperatur og T er tid (her er T = 3600).

5.6 Variabler

Verdier som kan manipuleres under sanntid er å regne som variabler. Alle objekter i MAS-ENERGY inneholder variabler som forteller noe om tilstanden til objektet. Under oppdatering vil variablene kunne gi informasjon som i MAS-ENERGY brukes til modeller i et scenario og gi resultater på hvilke parametre som ønskes testet. Variabler i MAS-ENERGY er enten boolske (true,false,0 eller 1) eller kontinuerlige (0,0 – *n*,*n*).

5.7 Konstanter

Verdier som er lagt inn i systemet og som ikke forandrer seg under sanntid er å regne som konstanter. Konstanter i MAS-ENERGY er mye benyttet både som statiske klasse variabler, lokale verdier eller som filer lastet inn i systemet under sanntid.

6 AI implementasjon

For å vise til besparing av energi i smart-house ble det i simuleringen implementert agenter, søkemetoder og maskinlære algoritmer. Agenter skal observere verden omkring seg med sensorer og bruke aktuatorer for handlinger i dette miljøet for å forandre tilstanden. Søk brukes for å finne verdier for data, enten det er empiriske data fra værdata filer eller det er data som kommer i sanntid fra programmet, som ved bruk av algoritmer med data minering-teknikker for tolkning av mønstre og dermed finne troverdige datamengder.

6.1 Refleks-agent

Mottar en persepsjon av verden slik den ser ut akkurat nå via sensorer og bruker kondisjonelle regler til å foreta en handling med en eller flere aktuatorer. I MAS-ENERGY er hver refleks-agent underlagt observasjons-mønsteret (se 4.2.2.1 Observer) og mottar persepsjonen fra sin oppdaterings metode. Eksempler på refleks-agenter i MAS-ENERGY er ThermostatActuator (se 12.2 javadoc components.implementedAgentBehaviour.ThermostatActuator) og WindowActuator (se 12.2 javadoc components.implementedAgentBehaviour.WindowActuator).

6.1.1 kondisjonelle baserte regler

En kondisjonell regel er enkelt forklart, en regel hvor verdier blir testet med forskjellige kondisjoner og hvis utkom er resultatet. Agenter basert på kondisjonelle regler, som refleks-agenter, bruker et sett med regler for å avgjøre hvilken beslutning som er den best oppnåelige for gitt

persepsjon og tilstand. For MAS-ENERGY er alle kondisjonelle regler hardkodet *hviss* setninger *(hviss* = hvis og bare hvis*)* som eksemplene i tekst 1 og tekst 2 viser. Her vises hvordan MAS-ENERGY bruker persepsjonen og sender en forespørsel til en kondisjonell regel:

```
// case 4 ACTUATOR -> get observer

      if(percept instanceof Actuator) {
 action = rules.doAction( (Actuator) percept, ((Actuator) percept).getType());
      }
```

Tekst 1: Kode eksempel fra Pecept.java i MAS-ENERGY

Persepsjon her er fra en ThermostatActuator agent som får returnert en konstant verdi hvis suksess eller null hvis ikke (se tekst 2):

```
 if (room.getTemprature() <= ThermostatActuator.min_temprature) {
            room.setTemprature(rand.nextDouble());
                  for(Heater heater: room.getHeaters()) {
                              heater.on();
                  }
 }

 if (room.getTemprature() > ThermostatActuator.min_temprature
 && room.getTemprature() < ThermostatActuator.max_temprature) {
            room.setTemprature(-rand.nextDouble());
                  for(Heater heater: room.getHeaters()) {
                              heater.off();
                  }
 }
 return Action.ADJUSTTO;
.... code omitted
return NULL;
```

Tekst 2: Kondisjonell regel i Rules.java i MAS-ENERGY

Fordelene med denne løsningen for kondisjonell regel er tredelt, den første sparer tid mellom forespørsel og svar fordi persepsjonen har all informasjon fra systemet som er nødvendig for å kunne forandre på parametrene og den andre er at nye verdier settes i selve regelen direkte. Den tredje er at kun agenter med lovlig type definisjon har adgang til reglene. Med lovlig type menes om agenten er definert i settet for lovlige refleks-agenter (se 12.2 javadoc model. Enums. LegalReflexAgents) og defineres som `((Actuator) percept).getType()` i metoden som vist i tekst 1.

6.2 Utility based agent

Hvordan velge det beste alternativet når det oppstår en hendelse med flere alternative løsninger? En utility agent prøver på å løse dette ved å bruke parametre og persepsjon om hvordan verden ser ut

akkurat nå. Den kan velge forskjellige alternativer ved å bruke funksjoner og variabler slik som lykke (ikke som lykke i menneskelig emosjonell følelse) og sammenligne tilstander med gitte verdier (se 6.2.1 Utility function).

6.2.1 Utility function

Funksjonen for utility, eller nytte funksjon, brukes av agenten for å finne en verdi ved eksperimentering som kan beskrive et utfall i en gitt tilstand, som «hva vil skje hvis jeg gjør x handling?» og ser på om handlingen fører til en bedre tilstand (les nyttig) eller ikke (les ikke nyttig). Som eksempel fra MAS-ENERGY brukes *health* (se 12.2 javadoc population. abstracts.isPerson setHealth()) variabelen til å si noe om hvor tilfreds individet er med energiforbruket i forhold til økonomi og kan brukes av en utility-agent for å finne best mulige handling gitt persepsjonen. Først lages en mål-node ved å bruke empiriske data som sier noe om forventet forbruk i en tidsperiode og dermed noe om det økonomiske utfallet. Agenten eksperimenterer med forskjellige handlinger og ser på hvilke node verdier som har minst differanse til mål-noden og hvor den beste handlingen blir utvalgt for å kunne gi en bedre tilstand for variabelen og dermed forhåpentligvis bedre økonomi.

6.3 Beslutningstre og ID3

Beslutningstre brukes ofte innen induktiv læring, som betyr å lære ved hjelp av eksempler Russel et. al. (2003, side 653) og Giorgio Ingargiola (u.å.). Modellen for beslutningstre brukt i MAS-ENERGY er ID3, en maskinlære algoritme som bruker eksempler til å søke i en trestruktur fra rot til blad ved lage nye subtre og itererer inntil best oppnåelige tre kan returneres. ID3 gjør dette ved å bruke eksempelsett (se 6.3.2 Læreprosess) og i MAS-ENERGY er dette logiske modeller med boolske operatorer, laget med Lawrence Turner (2012) web-baserte generator og som tillater nærmest ubegrenset utvidelse av attributtene. Data til eksempelsett blir fremstilt ved å skrive boolske utrykk, som skal si noe om en tenkt tilstand. Utrykkene i Lawrence Turner (2012) generator blir til sannhets tabeller som kan brukes til å klassifisere settene (se 6.3.1 Attributter og klasser i beslutnings-tre) og er i MAS-ENERGY lagret som tekstfiler (se 6.3.2 Læreprosess, avsnitt 1). Sannhets tabellene (se tabell 1), eller eksempelsett er basert på proposisjoner om hva som er sant eller ikke, gitt et logisk bevis, som i disse eksemplene fra Controller agenten for oppvarming og vindus kontroll:

/// create dataset for heaterONOFF MODEL:

is_ext_temp_low | ~is_ext_temp_low & is_current_temp_low & ~is_avg_temp_low &

~window_isopen window_isopen & ~door_isopen & ~is_bot & is_active

// Create datasets for window MODEL:

openWindow -> (highCurrenRoomTemperature & ~highExteriorTemperature) |
openWindow -> (~highExteriorTemperature & highAverageRoomTemperature)

Ved hjelp av entropi og informasjons gevinst (se 6.3.1 Attributter og klasser i beslutnings-tre) velges det beste attributtet fra eksempelsettet som start noden i beslutningtreet og testes til neste forgrening, hvor det enten returneres en beslutning (blad node) eller det lages nytt subtre og prosessen gjentas. ID3 har derimot ikke mulighet for å håndtere kontinuerlige klasser, slik som temperatur eller varierende varmegjennomgangskoeffisienter.

Notat: Et alternativ til logikk var å bruke eget agentspråk for MAS-ENERGY og AgentSpeak (2010) ble vurdert for å kunne gi god kommunikasjonen mellom agenter og tydeligere persepsjoner. Dette ble forkastet fordi det var enklere å bruke logiske modeller og eksempelsett for beslutningstre fra filer.

6.3.1 Attributter og klasser i beslutnings-tre

For å finne beste attributt for klassifisering kan dette gjøres blant annet ved å finne riktig informasjons-gevinst (engelsk: Information Gain) og bruk av entropi verdier. Entropi måler renheten eller urenheten i et medlem av et datasett og returnerer* en verdi i bits som sier noe om renheten til klassen, hvis lav verdi uren og hvis høy verdi ren. Her et eksempel fra MAS-ENERGY treningsett lightOnOff (tabell 1) :

Entropien finnes ved å bruke følgende formel: Entropi(S) = n∑i -P*i* log2 P*i*.

Entropi for GOAL er: Entropi(GOAL) = -(9/16)log2(9/16) -(7/16)log2(7/16) =0.988

**hvis alle eksempler har samme klasse resulteres alltid 0*

Attributes			Goal
is_active	is_light_on	roof_lamp_on	turn_on
T	T	T	T
T	T	F	T
T	T	T	T
T	T	T	T
T	T	T	T
T	T	F	T
T	T	T	T
T	T	T	T
T	T	T	T
T	T	F	F
T	F	T	F
T	F	T	F
T	T	T	F
F	T	F	F
F	F	T	F
F	F	F	F

Tabell 1:lightOnOff (settet er modifisert for bedre lesbarhet)

Nå kan entropien brukes for å returnere attributtet med den høyest informasjon gevinst . Dette gjøres ved å måle reduksjonen i entropi ved å bruke formelen:

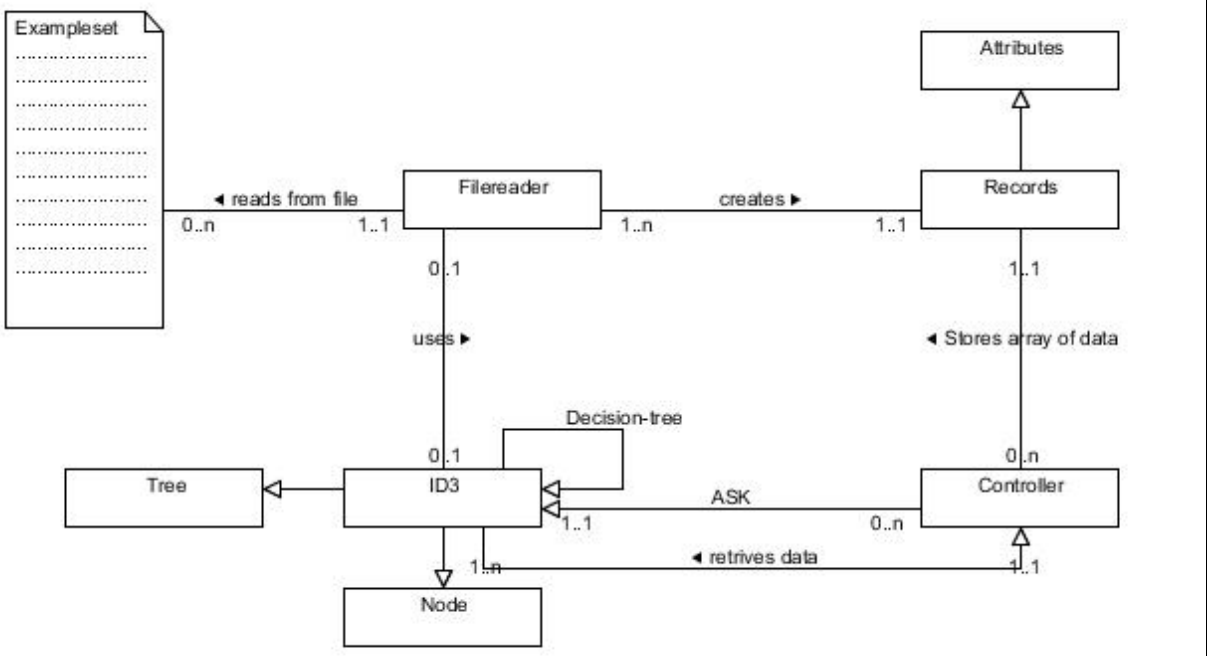
informasjons-gevinst(S,A) = Entropi(S) - n∑i -(Pi) log2 (Pi) , hvor S er settet, A er attributt og Pi er eksempler i settet. Attributtet med høyest gevinst blir valgt som rot node. Eksempel på utregning vises her med attributt for om individet er aktiv (is_active) (se 12.2 javadoc population .abstracts. isPerson) og for ordens skyld brukes det engelske gevinst, Gain:

$$\text{Gain}(\text{lightOnOff}, \text{is\_active}) = (0.988) - (13/16)\log_2(13/16) - (3/16)\log_2(3/16) = 1.684$$

6.3.2 Læreprosess

MAS-ENERGY bruker som nevnt i avsnitt 6.3 om beslutningstre og ID3, logiske modeller for å lage eksempelsett som sammen med parametre fra simuleringen trener opp beslutnings-treet inntil validering kan begynne.

Prosessen starter med først å transskribere eksempelsettet til en liste av Records (se 12.2 javadoc id3.Record) og lagret i Controller (se tegning 1). Records inneholder radene til eksempelsettet og består av attributtenes navn og deres verdier.



Tegning 1: Prosess for kreasjon av beslutnings-tre i MAS-ENERGY

For hver iterasjon lages det et nytt beslutnings-tre ved å lage noder fra listen av Records. Nodene er objekter fra Node klassen (se 12.2 javadoc id3.Node) og har egenskaper som sier noe om entropi, foreldrenode, barnnoder og test attributtet. Det er disse nodene som beslutnings-treet skal bygge på og dette gjøres ved å sendes dem som en liste inn i klassen Tree (se 12.2 javadoc id3.Tree) for å deretter bruke informasjons-gevinst til klassifisering av nodene.

Når et nytt tre er laget, skal det gjennom en iterativ prosess som bruker data fra parametre til å splitte opp dette i mindre subtre og prosessen forsetter inntil validering av data kan starte.

Det er forskjellige meninger eller anbefalinger på hvor mange prosent av eksempelsettet som skal brukes inntil læring er ferdig, noen mener 75% av eksemplene skal brukes til dette formålet og 25% til validering av data. I MAS-ENERGY skjer validering når treet har tilbake 3 noder uansett størrelsen på settet og mål nodens test attributt returneres til Controller som bruker denne verdien til å gi tillatelse eller ikke til agenter for å utføre handling. Det er ikke nødvendig i denne versjonen av MAS-ENERGY med større lister av noder til validering på grunn av eksempelsettene små størrelser (fra 16 til 128 rader).

Modellene som omtalt i kapitellets innledning brukes for hver iterasjon til å lage nye sett med data fra persepsjonen, som beslutnings-treet skal bruke i valideringen. Dette gjøres med kondisjonelle regler i Controller som i dette eksemplet (se 12.2 javadoc Controller.java linje 301):

```
if (!highExteriorTemperature) { // condition
heaterONOFF.add(new DiscreteAttribute("is_ext_temp_low",
```

```
DiscreteAttribute.T)); // attributes gives TRUE
} else if (highExteriorTemperature) { // condition
heaterONOFF.add(new          DiscreteAttribute("is_ext_temp_low",
DiscreteAttribute.F)); // attributtes gives FALSE
}
```

Gitt en kondisjon lages et attributt som i dette tilfelle er "is_ext_temp_low" og det tilegnes en verdi, enten TRUE eller FALSE. Sammen med alle attributtene i modellen utgjør dette et dataset som Controller bruker i ASK metoden som observert i tegning 1. Hvordan skjer validering? Som nevnt innledningsvis består eksempelsettet av tre noder når all læring er utført. Det er disse tre nodene som til sist blir brukt til å valideres imot datasettet som kommer inn fra hver iterasjon. Hver node har et test attributt (se 6.3.2 Læreprosess) hvis verdi sammenlignes med tilsvarende attritbutt fra datasettet. Noden som får høyest treff, hvilket vil har flest rikitge verdier for hvert attributt, blir til slutt valgt og verdien fra nodens test attributt blir returnert.

6.4 Informerte søk

I en simulering hvor objekter ikke har en fast plassering kan det være nødvendig med metoder for søk som finner frem til en mål-tilstand ved å bruke informerte søk (søk ved å utvide den minst kostbare noden) slik som nevnt i 6.4.1 A*. For eksempel hvordan et individ kommer seg fra A til B og samtidig unngå å komme i konflikter med andre objekter.

6.4.1 A*

Informerte søk fungerer ved å utvide nærmeste nabo som kan nås med en rett linje (f.eks. nord, vest, sør og øst) til en mål-node som søket kjenner til ved å kunne bruke en heuristisk funksjon. For hvert steg som tas, blir kostnaden for å nå noden økt og avstanden til goal noden fra nåværende blir målt. Ved å legge disse to verdiene sammen, kan en distanse lages for å si noe om hvor nærme målet noden befinner seg og hvor mye det koster å nå mål noden. Noden som er billigst og nærmest blir valgt for å utvide søket. A* søk er i MAS-ENERGY implementert i metoder for bevegelser i smart-house, men er ikke brukt i simuleringen fordi det ikke var behov for å la individene gå fra punkt A til B uten å møte forhindringer. Dette er ment for fremtidige behov for blant annet grafiske grensesnitt.

7 Brukertilfredsstillelse

I MAS-ENERGY er brukertilfredsstillelse et parameter som består romtemperatur, varme-gjennomgangskoeffisientene og energiforbruk variablene og skal si noe om hvor komfortabelt et individ har det i et smart-house miljø. Med komfort menes om individet har det behagelig med temperaturen i rommet det befinner seg i og om energi fører til at det spares på økonomi. varme

7.1 Beregne brukertilfredsstillelse

I MAS-ENERGY lages for hver iterasjon en persepsjon av alle objekter knyttet til observasjons-mønsteret (se 4.2.2.1 Observer) og sier noe om tilstanden til objektet. For alle individer brukes variabelen *health* (se 12.2 javadoc population.abstracts.isPerson setHealth()) som en variabel som forteller noe om graden av lykke (se 6.2 Utility based agent) og som ikke har noe med individets helse å gjøre, men er en representasjon av tilfredsstillelsen som individet får av å spare energi og ha det komfortabelt. For å beregne graden av lykke i MAS-ENERGY må tilstanden (eng.: *state*) fra flere objekter lagres i en liste etter hver iterasjon og kan bruke husets *states()* metode (se 12.2 javadoc model.abstracts.House) til å utføre dette. Disse tilstandene kan fortelle noe om temperaturer i rommet versa ønsket temperatur, om besparelse av energi eller varme-gjennomgangskoeffisientene siden forrige iterasjon. Metoden lagrer en liste av tilstander i en komma separert fil og brukes i nåværende versjon til dataanalyser i scenario (se kapitel 8.1 Scenario). Et eksempel vises her fra oppdaterings metoden til et rom, hvor variablene for varmegjennomgangskoeffisientene (ventilasjons-tap og transmisjons-tap) settes:

```
// set transmition losses to room
setTransmissionLoss(transmissionLosses(

                        getUpperPosition().distanceSq(lowerPosition),
                        getuValue(),
                        getTemprature(), exteriorTemperature));

// set ventilation losses to room
setVentilationLosses(
      ventilationLosses(random.nextDouble(), random.nextDouble()
                              * exteriorTemperature, 1.5, getTemprature(),
                              exteriorTemperature));
```

Listen av tilstander kan dermed benyttes for å si noe om husets tidligere persepsjoner og ny verdi for lykke kan beregnes ved å bruke disse data sammen med sanntids persepsjonen. Ved å bruke Bayes regel som vist i Kevin Murphy (2012) web-side kan det kalkuleres på at sannsynligheten for en variabelen fra sanntids persepsjonen gitt den empiriske persepsjonen, gir den riktige lykke verdien. Dette kan gjøres ved å lage to hendelser, empiriskLykke (*eL*) og sanntidLykke (*sL*), som

gir oss formelen:

$$P(eL|sL) = \frac{P(sL|eL)P(sL)}{P(sL)}.$$

Eksempel: Hvis 90% av alle iterajoner har en sunn sammenheng mellom verdiene for lykke (ingen data støy) og gitt at *eL* er 0.45 og *sL* er 0.48, vil dette gi oss

$P(eL|sL) = (0.9*0.45) / ((0.9*.45)+(0.1*0.48)) = 0.453.$

7.2 Konklusjon

I kapitel 2.6 nevnes Boman et. al. (1999) utsagn om konflikter mellom agenter i et MAS at det er et hoved *trade-off (her: negativ årsak)* for intelligente bygninger. Ved mange konflikter vil effektiviteten til systemet bli mindre og beslutninger kan bli feil-kalkulert som fører til en negativ effekt på brukertilfredsstillelsen. Med negativ effekt på brukertilfredsstillelsen menes her om graden av lykke som individer har i systemet med variabelen *health* minskes. I MAS-ENERGY er oppgaven å vise til at agenter som benytter seg av kommunikasjon kan lære bedre en agenter som kun er basert på regler og beregning av brukertilfredsstillelse er kun laget som en eventuell utvidelse av systemet og har ingen direkte innflytelse på resultatene. Med direkte innflytelse menes her at parametre for besparelse av energi ikke er avhengig av parametre for brukertilfredsstillelse og dermed kan utelukkes. Selv om det ikke er implementert i denne versjonen av MAS-ENERGY bør det inkluderes i fremtidige modifikasjoner av systemet, fordi brukertilfredsstillelse parametre kan si noe om hvordan individet opplever smart-house teknologien.

8 Resultater

Denne oppgaven ønsker å vise at man sparer energi ved å bruke et sofistikert agentsystem som lærer og kommuniserer. Til dette formålet er det laget en simulator som produserer en mengde data fra forskjellige objekter, som under simulering forsøker å etterligne hva som skjer i et smart-house. Simulering avsluttes med å lagre en fil med målbare data (se tabell 2) og brukes for å finne sammenhenger for parametre i et scenario. Med målbare data menes i MAS-ENERGY verdier fra objektenes attributter og lagret fra hver iterasjon hvor tidpunktet på døgnet er lik med hele timer. Tid i MAS-ENERGY blir kontrollert av SystemTime klassen (se 12.2 javadoc timer.SystemTime) og bruker antall registrerte dager i en xml fil fra meterologisk institutt, eKlima (2010). Dette gir en simulering på 540 dager med måling av temperaturer gjort på fire tidspunkter per dag, kl. 00:00, kl 07:00, kl 14:00 og kl. 19:00. Ved å kompansere manglende tidpunkt og temperaturer er det mulig å dele opp et døgn i timer og minutter. Iterasjonen til MAS-ENERGY er fastsatt til 1 gang per minutt for mer nøyaktig måling og som gir totalt 777.660 iterasjonener. Dette produserer en stor mengde data som krever mye datakraft for å lage presentasjoner av grafene og dermed vanskelig å vise grafisk, det ble derfor besluttet å kun bruke hele timer i simuleringen.

Time	Tconsum	ExtTemp	RTemp	Wopen	heaterOn	totalPrs	isBot	isActive	roomConsum
1	1,00043388	15,4	17,50304945	0	0	0	1	0	0
2	0,601161054	16,17259941	17,96554826	0	0	0	1	0	0,4
3	0,601292894	15,809164	17,25642184	0	0	1	0	0	0
4	1,000888984	16,29235868	17,60759171	0	0	0	0	0	0,1

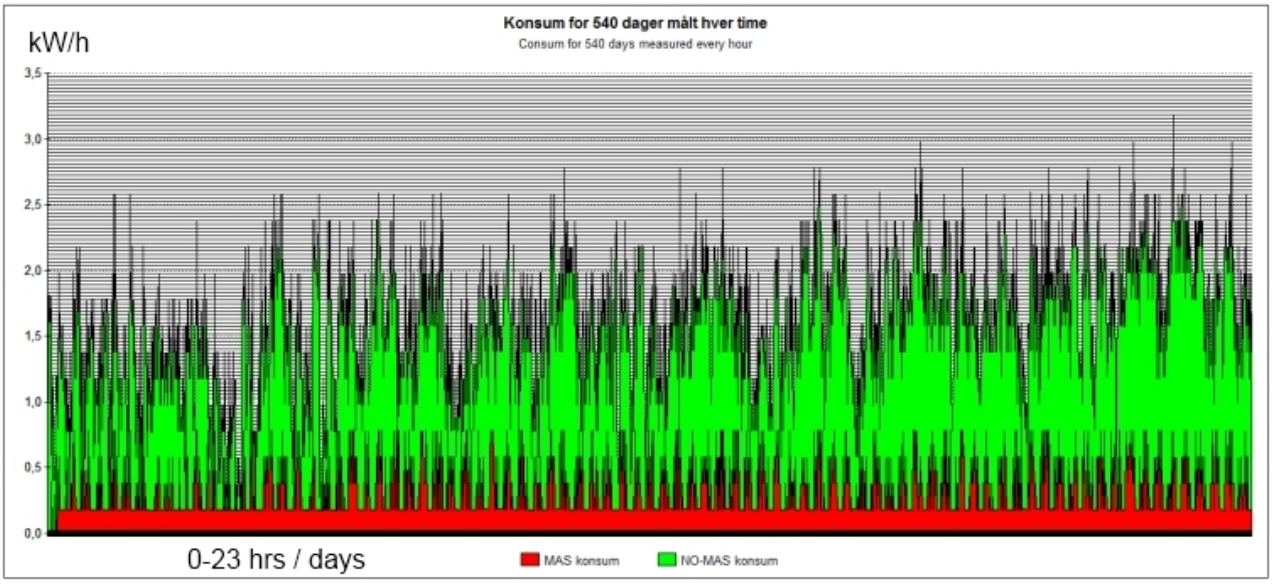
Tabell 2: Kvantitative data i regneark

8.1 Scenarioer

En hendelse kommer i sekvenser og disse kan si noe om tilstanden som var før hendelsen inntraff, hva som skjer under eksekvering, til den når sitt klimaks ved avslutting. Et scenario i MAS-ENERGY er forsøk på å gjenskape disse sekvensene ved å bruke datamodeller og kontekster. Datamodellene består av enkle boolske operatorer som AND eller XOR og brukes til å formulere spørsmål, eller test, til datasettene. Kontekster er sannsynlige situasjoner hvor det trolig vil inntreffe en hendelse, som kan si noe om agenter som lærer og kommuniserer, har bedre kontroll over energiforbruket en agenter som ikke lærer eller kommuniserer.

8.1.1 Utgangspunktet for målinger, scenario 0

Hvis resultatet fra simuleringen viser at for hus med MAS er energiforbruket mindre en hus med NO-MAS, konkluderes det med å ha et utgangspunkt for videre undersøkelser. Scenario 0 måler energiforbruket i MAS-ENERGY per hele time under 540 døgn, for både MAS og NO-MAS. Som illustrasjon 9 viser var resultatet av denne denne målingen positiv med hensyn til energi besparelse for MAS.



Illustrasjon 9: Differanse forbruk i MAS og NO-MAS.

Dette skal være utgangspunktet for kommende tre scenarioer som skal prøve å finne svar på hvorfor hus med MAS sparer mer energi en hus med NO-MAS.

- **Scenario 1 Gjennomsnitt forbruk MAS og ikke MAS klokken 00:00 og 08:00**

  Er det forskjell om det i et rom finnes til forskjellige tidspunkt aktive individer med hensyn til forbruk av energi?

  - Test 1: Energiforbruk klokken 00:00 AND antall individer aktive
  - Test 2: Hvordan er energiforbruk klokken 08:00 AND antall individer aktive?

- **Scenario 2 Energiforbruk og oppvarming MAS og NO-MAS**

  Hvordan er forbruk av energi og bruk av oppvarming på forskjellige tidspunkt på døgnet og har dette sammenheng med individer?

  - Test 1: Tilfeldig valgt tid på døgnet AND verdier for energiforbruk AND oppvarming
  - Test 2: XOR Individer AND oppvarming AND energiforbruk.

- **Scenario 3 Varmegjennomgangskoeffisientene og forbruk**

Hvilken effekt har forskjellen i varmegjennomgangskoeffisientene på energiforbruk?

- Test 1: Varmegjennomgangskoeffisienten under kalde dager?
- Test 3: Finnes det sammenheng mellom åpne vindu AND ventilasjons-tapet?
- Test 3: Finnes det sammenheng mellom transmisjons-tapet AND energiforbruk?

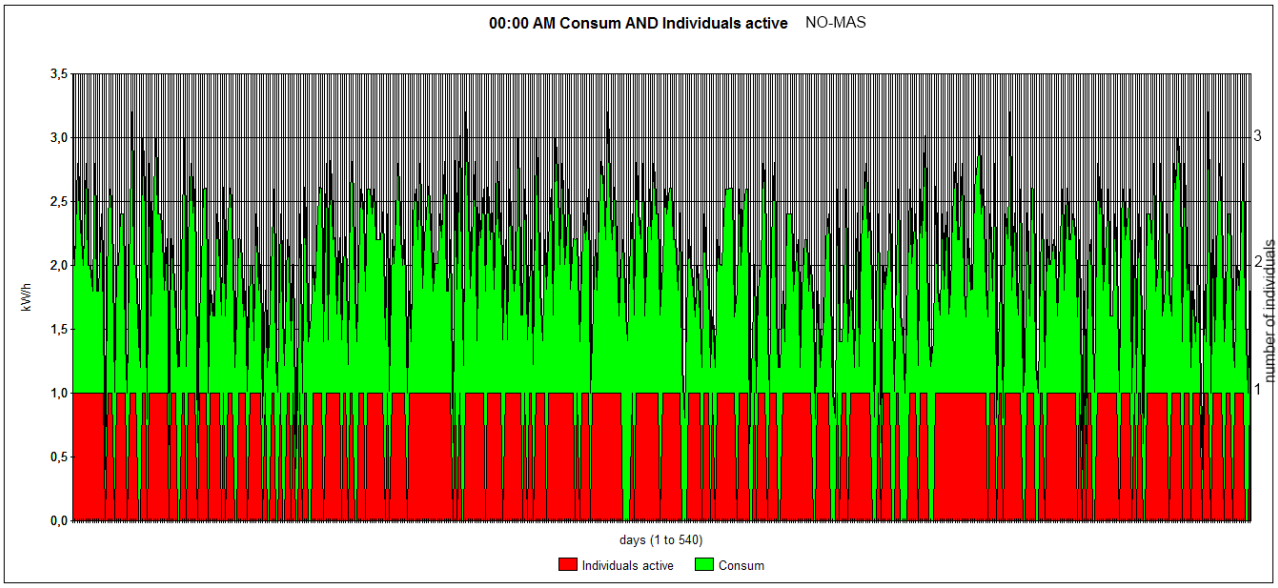
8.1.2 Scenario 1: Gjennomsnitt forbruk MAS og ikke MAS klokken 00:00 og 08:00

Er det forskjell om det i et rom finnes til forskjellige tidspunkt aktive individer med hensyn til forbruk av energi?

Har antall individer som er aktive noen merkbar innflytelse på energiforbruket i MAS og NO-MAS modellene? Dette scenarioet skal teste sammenhengen mellom et gitt klokkeslett og antall aktive individer som er registret på dette tidspunktet. I begge modeller finnes det også Random agenter (se 4.6.3 Random) som kan bevege seg fritt uavhengig klokkeslett og kan ha innvirking på forbruk av energi ved å justere på komponenter og dermed ha indirekte innvirkning på temperatur. Disse agentene er registret som individer og skal simulere naturlig tap av temperatur i rommet.

- Test 1a, og 1b: Energiforbruk klokken 00:00 AND antall individer aktive

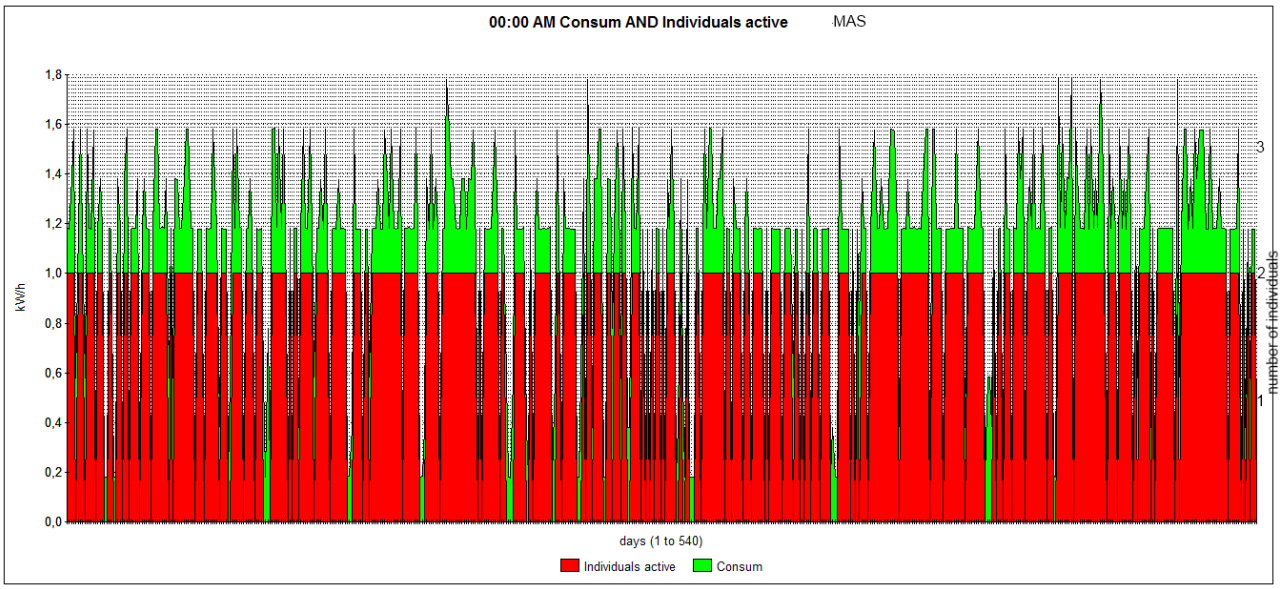
NO-MAS (540 dager) Illustrasjon 10:



Illustrasjon 10: Test 1a NO-MAS Energiforbruk klokken 00:00 AND antall individer aktive

Testen viser noen sammenheng mellom antall aktive og energiforbruk for NO-MAS. Utfra illustrasjonen sees hvordan sammenhengende blokker av individer som er aktive (rød farge) har innvirking på grafen for forbruk (grønn farge).

MAS (540 dager) Illustrasjon 11:

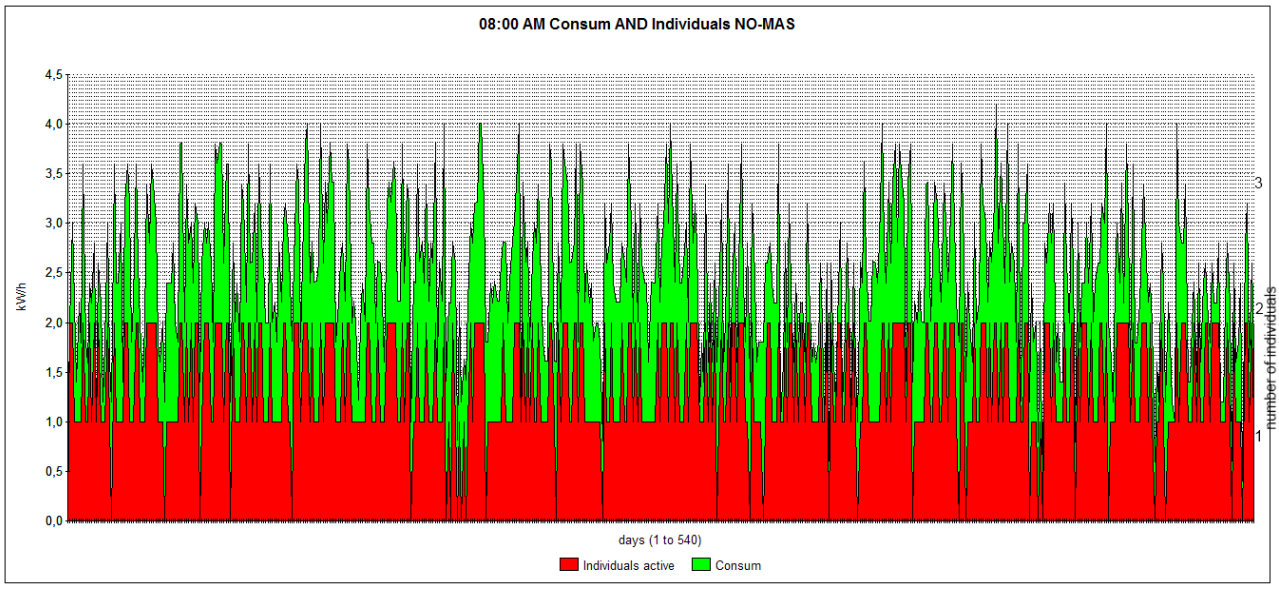


Illustrasjon 11: Test 1b MAS Energiforbruk klokken 00:00 AND antall individer aktive

Testen viser noen sammenheng mellom antall aktive og energiforbruk for MAS av samme årsak som test 1a. Test 2 i dette scenarioet skal bruke samme data for å se på om et senere tidspunkt har innvirking på resultatet.

- Test 2a og 2b: Hvordan er energiforbruk klokken 08:00 AND antall individer aktive?

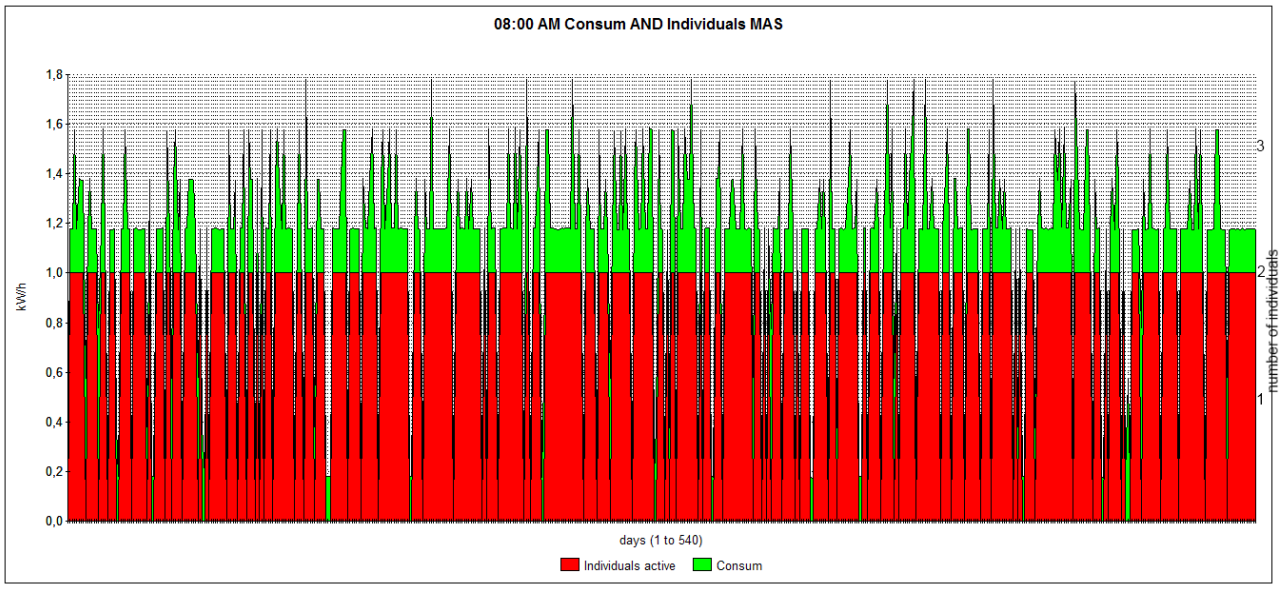
NO-MAS (540 dager) Illustrasjon 12:



Illustrasjon 12: Test 2a NO-MAS Energiforbruk klokken 08-00 AM AND antall individer aktive

Testen viser sammenheng mellom antall aktive og energiforbruk for NO-MAS, spesielt kan dette observeres på forbruket (grønn farge) hvor antall individer er over 1.

MAS (540 dager) Illustrasjon 13:



Illustrasjon 13: Test 2b MAS Energiforbruk klokken 08-00 AND antall individer aktive

Testen viser sammenheng mellom antall aktive og energiforbruk for MAS med samme årsak som test 2a.

Dette scenarioet konkluder med at det finnes sammenheng mellom antall aktive individer og energiforbruk i MAS og NO-MAS for husets hele totale energiforbruk. Med merkbar sammenheng menes her at det ser ut til alltid å øke energiforbruk når et individ kommer inn i et rom.

Dette kan være av flere årsaker, som:

1) Individer aktiverer komponenter som genererer energiforbruk.
2) Individer kan oppholde seg i et rom uten å påvirke forbruk av energi.
3) Forbruk av energi er ikke alltid avhengig om individer er aktive.
4) Forbruk av energi er ikke alltid avhengig om individer er til stede.

Neste scenario 2, ser på sammenheng mellom et individ, komponenter for oppvarming og energiforbruk lokalt i hvert rom (se 8.1.3. Scenario 2: Energiforbruk og oppvarming MAS og NO-MAS, *Test 2a og 2b: Individer, oppvarming og energiforbruk*).

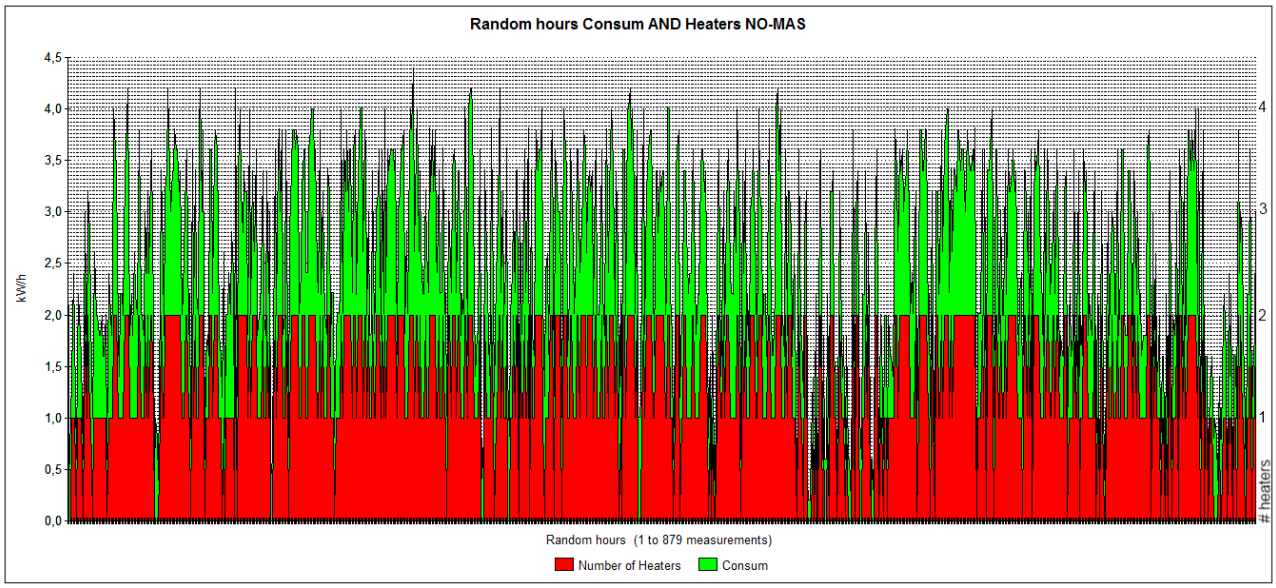
8.1.3 Scenario 2: Energiforbruk og oppvarming MAS og NO-MAS

Hvordan er forbruk av energi og bruk av oppvarming på forskjellige tidspunkt på døgnet og har dette sammenheng med individer?

Forrige scenario konkluderer med at det finnes sammenheng mellom antall aktive og husets totale energiforbruk. Som sees i illustrasjon 9, er det en merkbar besparelse av energiforbruk ved bruk av MAS og dette scenarioet skal se på om det finnes sammenheng mellom energiforbruk og oppvarming i smart-house for MAS og NO-MAS. Med oppvarming menes her om bruken av komponenten for oppvarming, *heater* (se 12.2 javadoc components.design.Heater), gir forskjellige resultater på variabelen for forbruk, *Consum* (se 12.2 javadoc model.abstracts.House getConsum) når *heater* har enten «*av modus*» (false) eller «*på modus*» (true). I test 1 måles dette ved å hente data for antall heater som er i true modus og energiforbruket fra MAS og NO-MAS ved tilfeldige valgte tidspunkt mellom klokken 00:00 og 23:00 (hele timer). Test 2 ser på antall aktive individer i rommet, om heater er i true modus og på energiforbruket for å finne en sammenheng mellom disse.

- Test 1a og 1b: Tilfeldig valgt tid på døgnet AND verdier for energiforbruk AND oppvarming

NO-MAS (879 målinger) Illustrasjon 14:



Illustrasjon 14: Test 1a NO-MAS Tilfeldig valgt tid på døgnet verdier for energiforbruk og oppvarming

Som illustrasjon 14 viser er det en sammenheng mellom oppvarming og forbruk av energi i NO-MAS med en gjennomsnitts energiforbruk tilsvarende 1,19kWh. Dette henger sammen med hvordan ThermostatActuator (se 12.2 javadoc components. ImplementedAgentBehaviour.

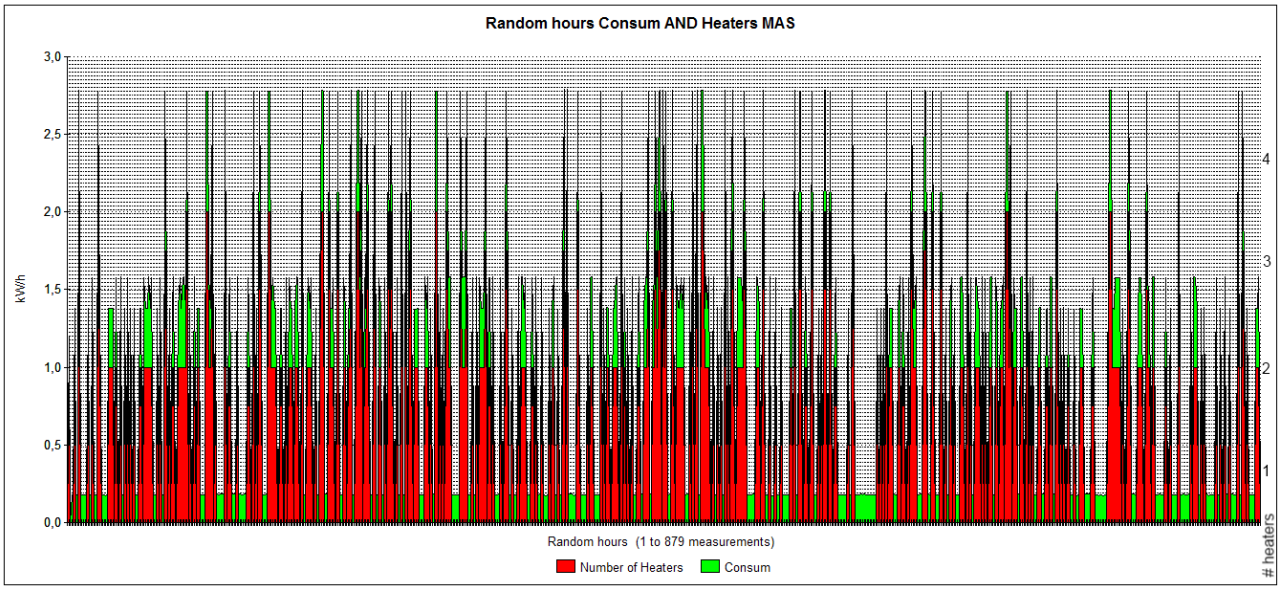
ThermostatActuator) bruker variablene for minimum eller maksimum temperatur for å justere romtemperaturen ved bruk av kondisjonelle regler. Ved å se nærmere på grafene kan det observeres hvordan forbruket (grønn farge) går opp når flere *heater* (rød farge) objekter er aktivert. Illustrasjon 15 viser koden hvor ThermostatActuator justerer på temperaturen.

```
if (room.getTemprature() <= ThermostatActuator.min_temprature) {
      room.setTemprature(rand.nextDouble());
            for(Heater heater: room.getHeaters()) {
                  heater.on();
            }
}

if (room.getTemprature() > ThermostatActuator.min_temprature
&& room.getTemprature() < ThermostatActuator.max_temprature) {
      room.setTemprature(-rand.nextDouble());
            for(Heater heater: room.getHeaters()) {
                  heater.off();
            }
}
```

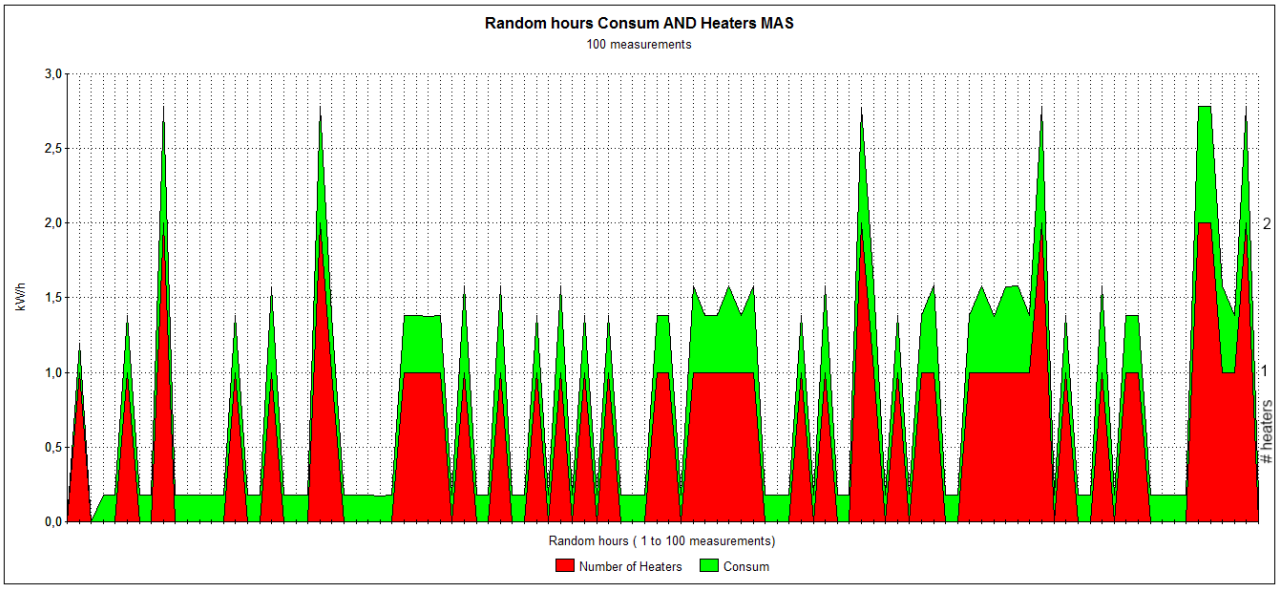
Illustrasjon 15: kondisjonell regel for setting av temperatur i et rom.

MAS (809 målinger) Illustrasjon 16:



Illustrasjon 16: Test 1b MAS Tilfeldig valgt tid på døgnet verdier for energiforbruk og oppvarming

Som ved test 1a (se illustrasjon 14) er det i test 1b merkbar sammenheng mellom oppvarming og energiforbruk, men med mindre gjennomsnittsforbruk på 0,34 kWh. Sammenlignes resultatet fra test 1a er besparelsen gjennomsnittlig på 0,84kWh eller ca. 70%. Illustrasjon 17 viser et mer tydeligere bilde av hvordan forbruks-variabelen *Consum* henger sammen med verdiene for oppvarming av rommet.



Illustrasjon 17: Test 1b MAS Detalje 100 målinger med tilfeldig valgte tider på døgnet med verdier for energiforbruk og oppvarming.

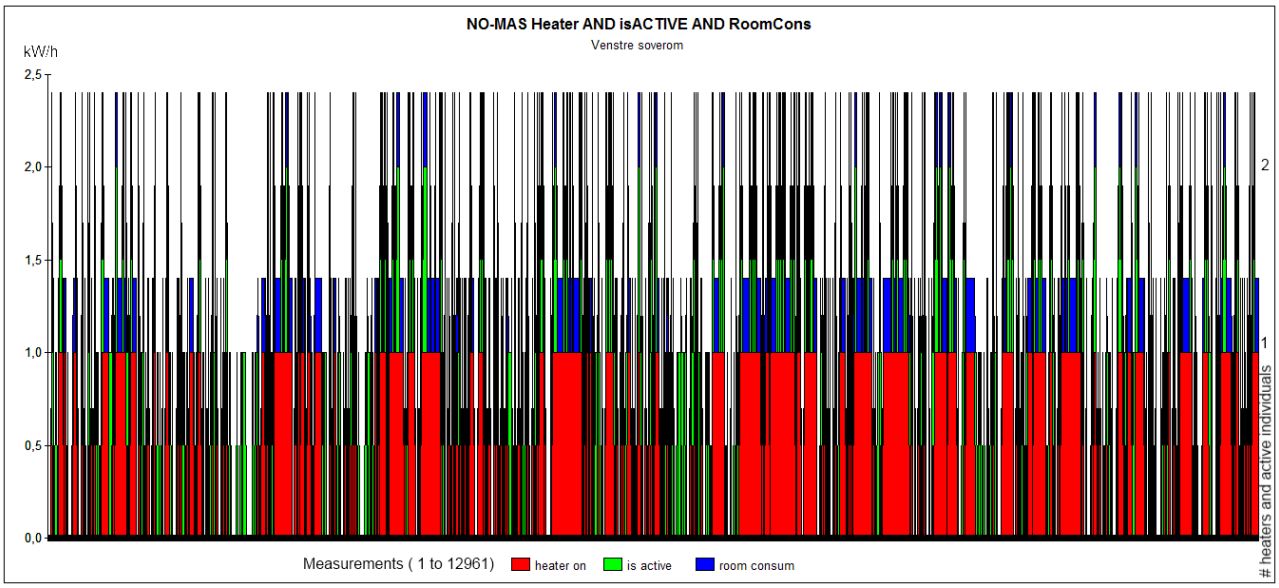
Notat:

Deler av forbruket som sees på y aksen skyldes andre variabler som belysning og varmegjennomgangs koeffisienten (opptil 0.3 kWh grensen). I neste scenario (8.1.4 Scenario 3: Varmegjennomgangskoeffisientene og forbruk) skal denne analyseres for å finne sammenheng mellom disse verdiene og energiforbruk.

- Test 2a og 2b: XOR Individer AND oppvarming AND energiforbruk.

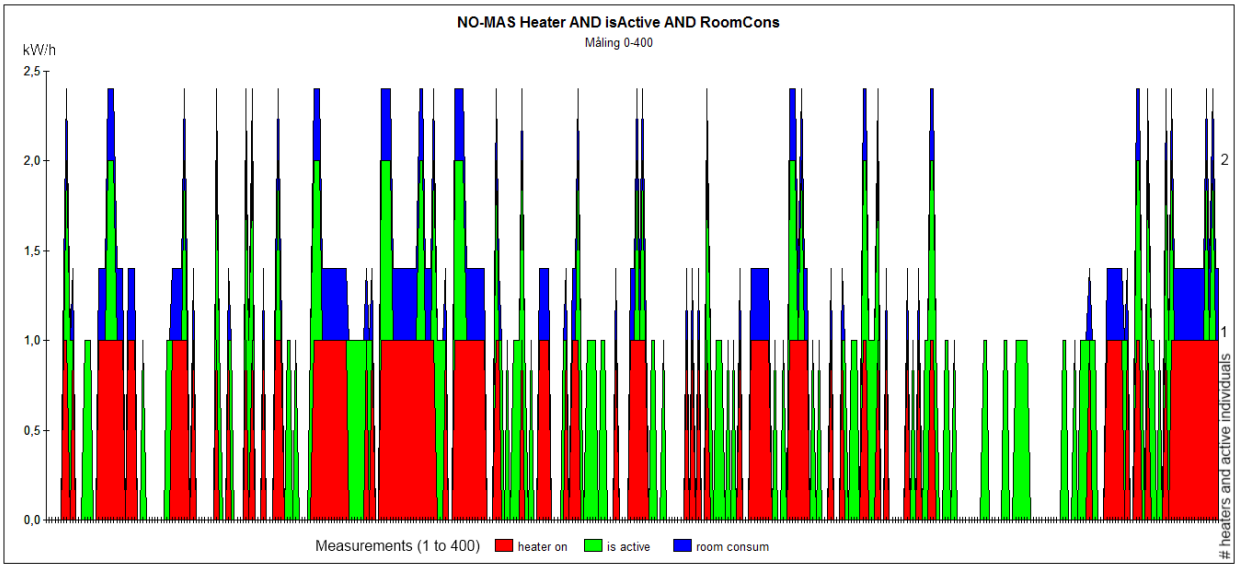
  NO-MAS Venstre soverom (12961 målinger) Illustrasjoner 18:

  merknad! illustrasjoner 19, 20 og 21 er utsnitt av illustrasjon 18

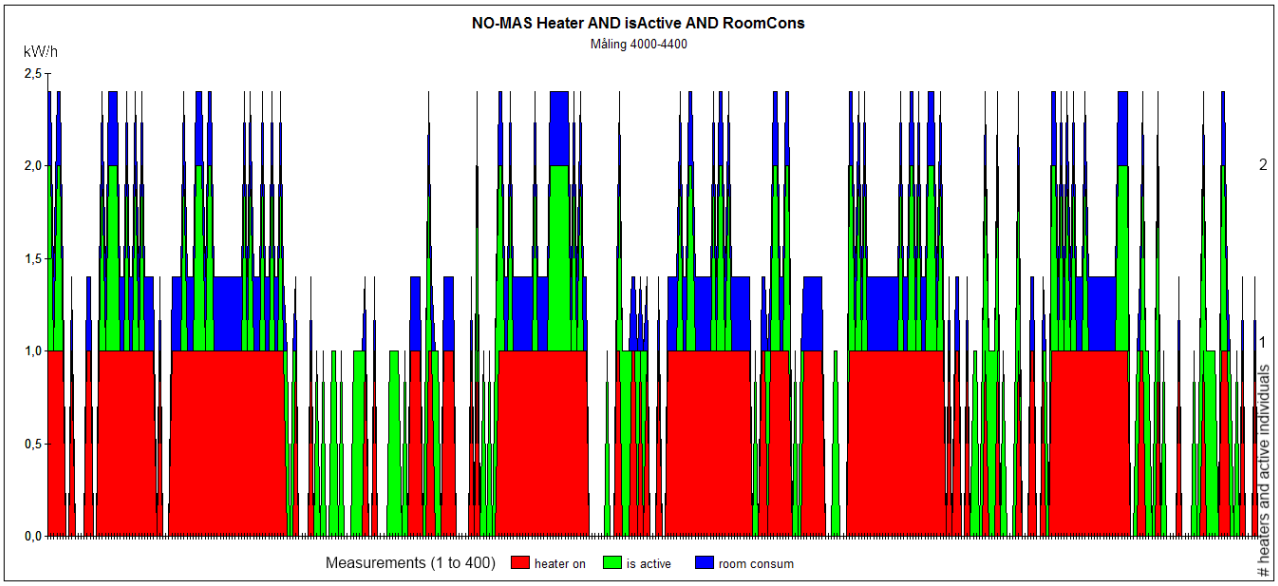


Illustrasjon 18: Test 2a venstre soverom, verdier for oppvarming, antall individer aktive og rommets forbruk.

Resultatet for test med venstre soverom i et NO-MAS hus viser at det finnes en sammenheng mellom aktive individer, oppvarming og rommets aktuelle energiforbruk. Datamengden er svært stor og det er derfor laget utsnitt (se illustrasjonene 19 til 21) med mindre mengder data. Dette er gjentatt også for soverom i et MAS hus (illustrasjoner 22-25). Forbruket som er målt her er lokalt forbruk, hvilket vil si at det er kun i dette rommet det er målt *heater* komponentens verdier. Ved å sammenligne resultatet fra dette scenarioet og scenario 1 kan det observeres hvordan dette tydelig viser til sammenheng mellom aktivitet og energiforbruk. I scenario 1 er bruken av totale verdier, slik som total energiforbruk og totalt antall individer, gjort det vanskelig å kunne å kunne observere hva som virkelig skjer i simuleringen, dette unngås i dette scenarioet ved å konsentrere seg om et rom.

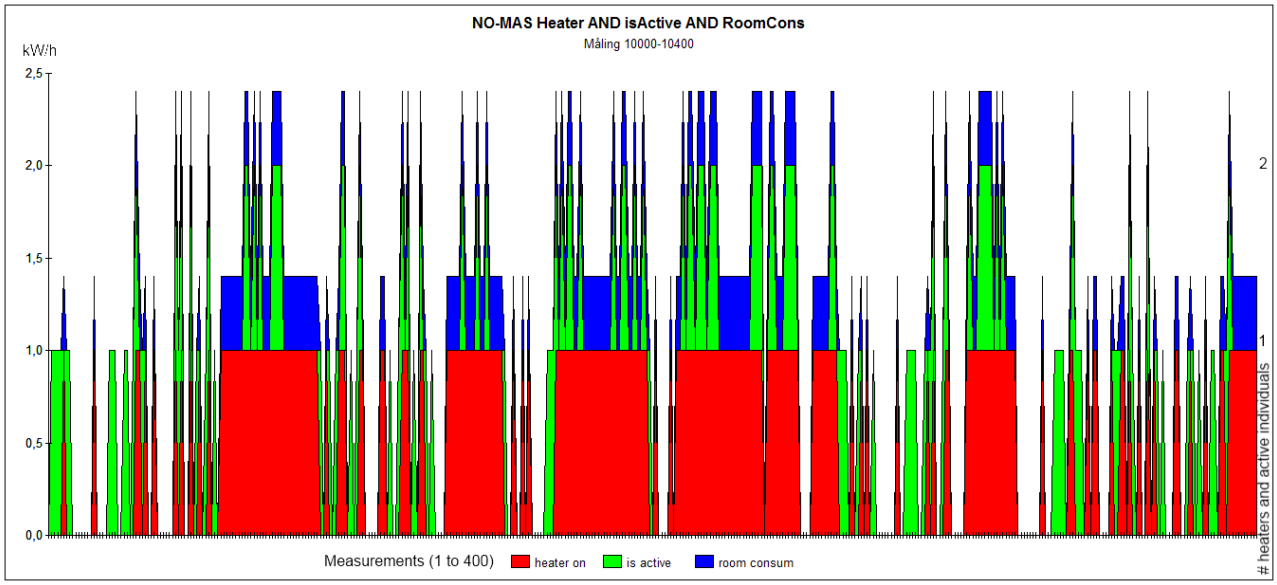


Illustrasjon 19: Venstre soverom, verdier for oppvarming, antall individer aktive og rommets forbruk, utsnitt måling fra 0 til 400



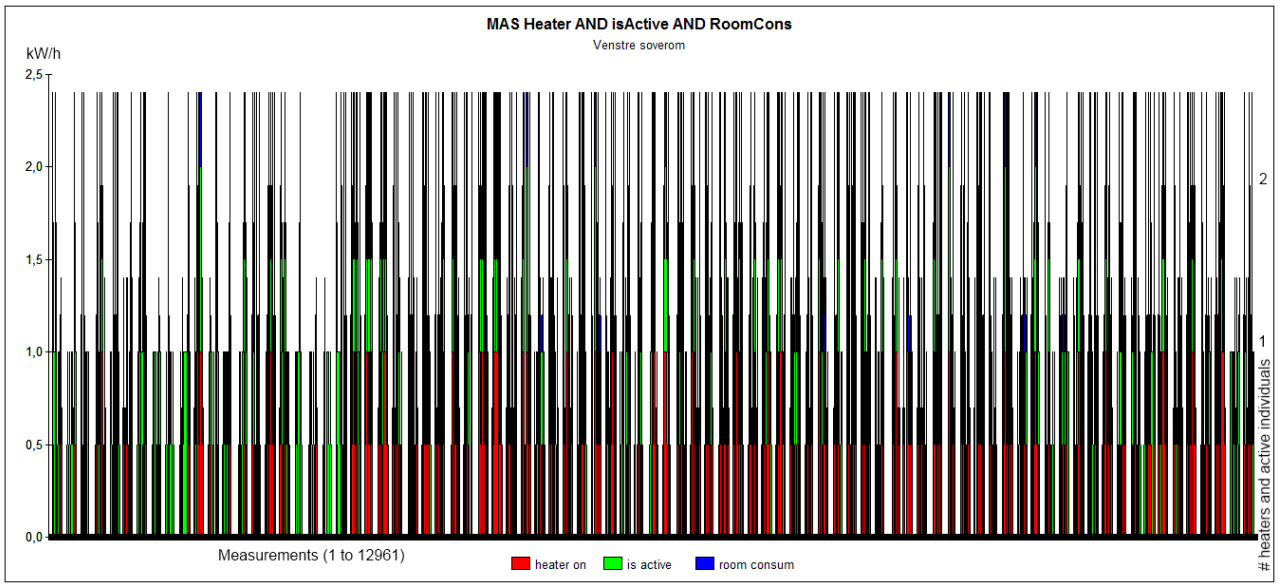
Illustrasjon 20: Venstre soverom, verdier for oppvarming, antall individer aktive og rommets forbruk, utsnitt måling fra 4000 til 4400

Illustrasjon 21: Venstre soverom, verdier for oppvarming, antall individer aktive og rommets forbruk, utsnitt måling fra 10000 til 10400



MAS Venstre soverom (12960 målinger) illustrasjon 22:

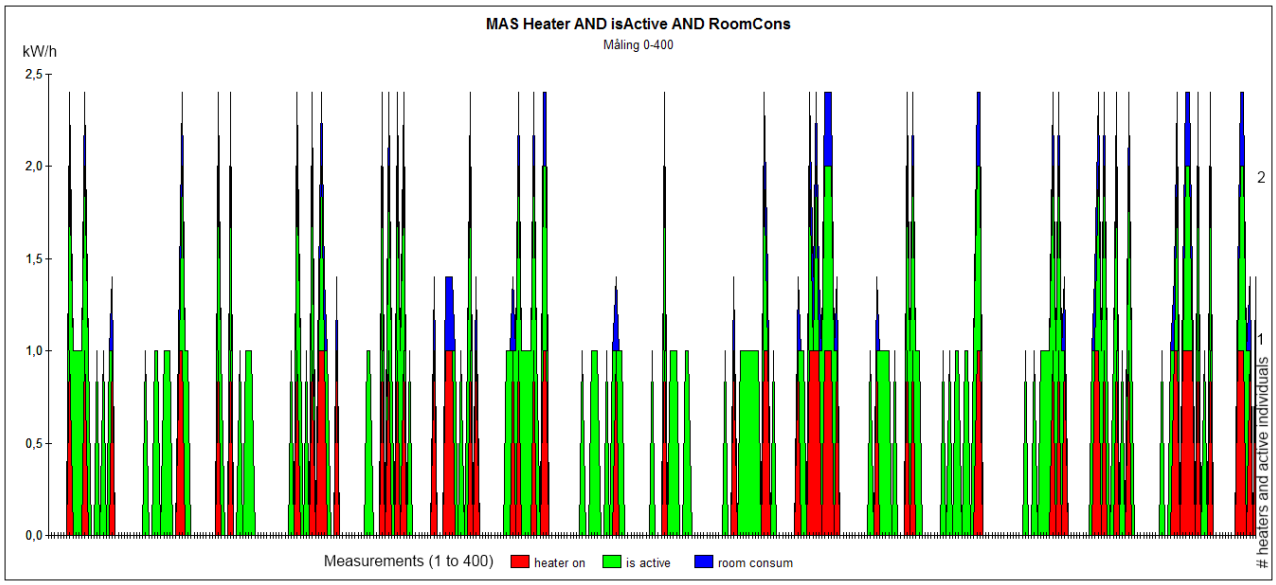
merknad! illustrasjoner 23, 24 og 25 er utsnitt av illustrasjon 22



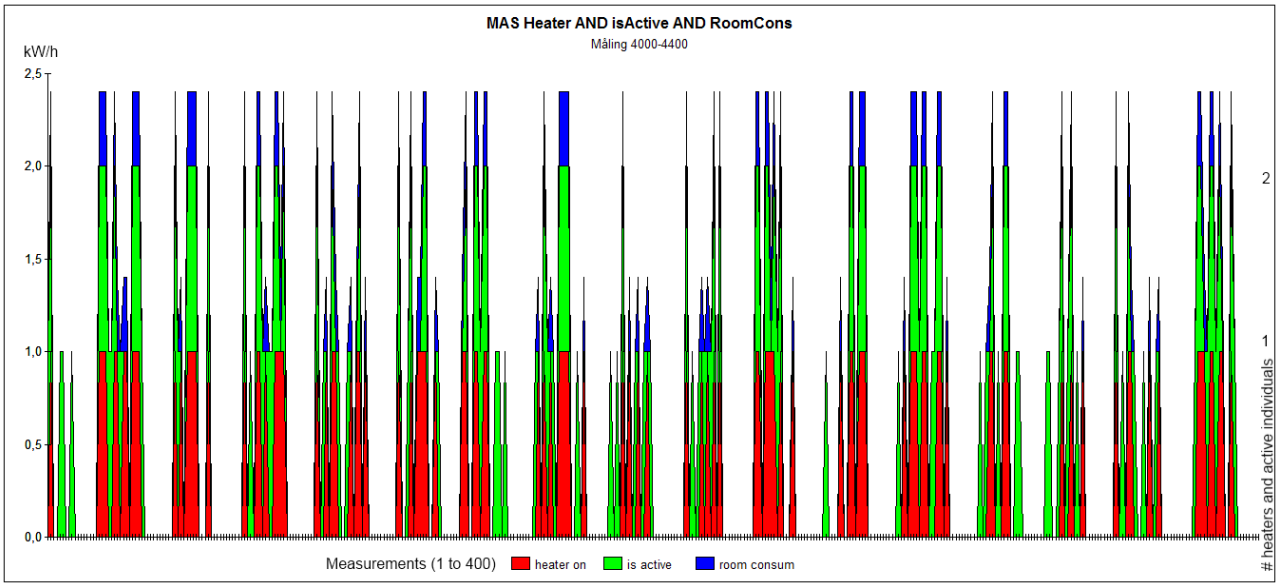
Illustrasjon 22: Test 2b venstre soverom, verdier for oppvarming, antall individer aktive og rommets forbruk.

I likhet med test 2a har denne testen gitt samme resultater som beskrevet tidligere og vil derfor ikke bli gjentatt her, men legges til scenarioets konklusjon. Ved å observere utsnittene for NO-MAS kontra MAS sees forskjellene markant. MAS modellen har kortere frekvens på true og false modus for *heater* komponenten, noe som gir et mer kontrollert energiforbruk og mindre forbruk. Dette henger derfor sammen med test 1 fra dette scenarioet, hvor det vises til en besparing på ca. 70% i

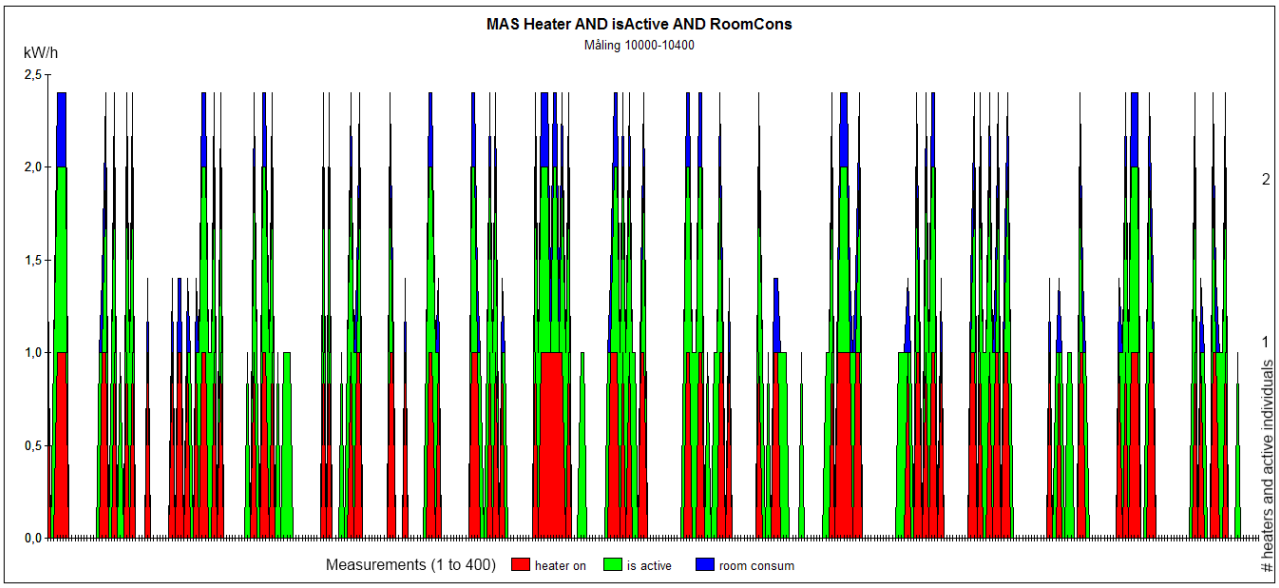
forhold til NO-MAS.



Illustrasjon 23: Venstre soverom, verdier for oppvarming, antall individer aktive og rommets forbruk, utsnitt måling fra 0 til 400



Illustrasjon 24: Venstre soverom, verdier for oppvarming, antall individer aktive og rommets forbruk, utsnitt måling fra 4000 til 4400



Illustrasjon 25: Venstre soverom, verdier for oppvarming, antall individer aktive og rommets forbruk, utsnitt måling fra 10000 til 10400

Dette scenarioet konkluderer med at det finnes en målbar sammenheng mellom energiforbruk og oppvarming. I test to konkluderes det med at frekvensen på true og false modus for heater er kortere i MAS en i NO-MAS, noe som forklarer hvorfor MAS har mindre energiforbruk på oppvarming, fordi beslutningen om aktivere eller ikke varmeelementer synes å bli bedre kontrollert med maskinlære teknikker. Det er også vist til en sammenheng i rom enkeltvis mellom aktive individer og energiforbruk, både for NO-MAS og MAS. I begge testene kan det vises til ca. 70% besparelse på energiforbruket i MAS sammenlignet med NO-MAS.

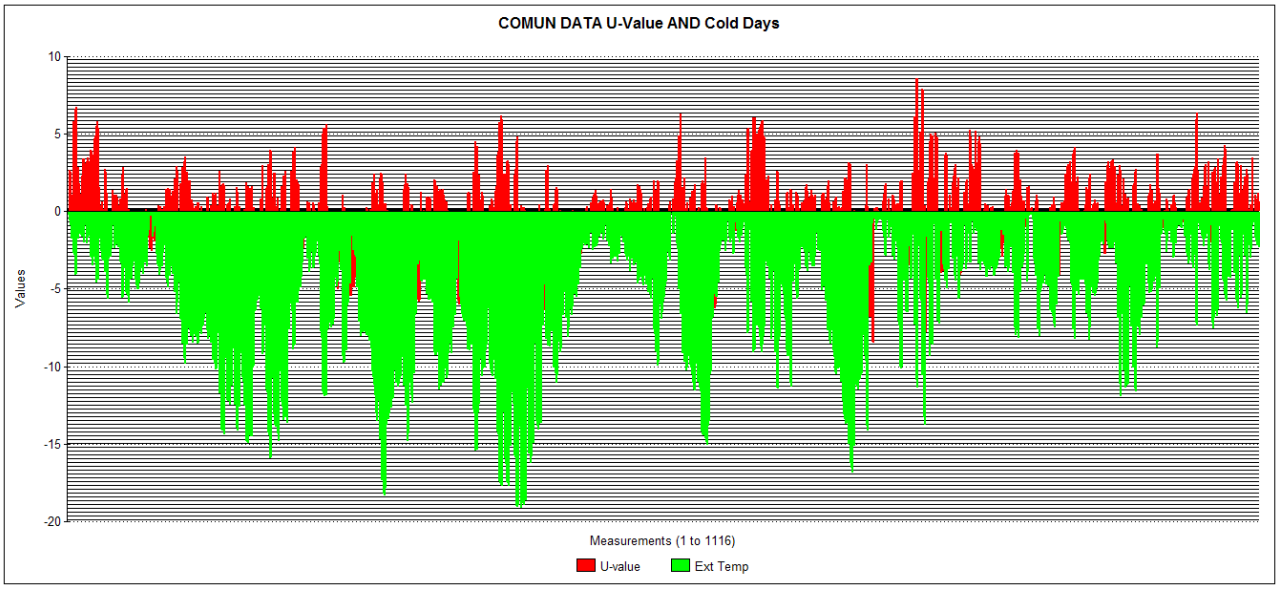
8.1.4 Scenario 3: Varmegjennomgangskoeffisientene og forbruk

Hvilken effekt har forskjellen i varmegjennomgangskoeffisientene på energiforbruk?

Dette scenario ser på verdiene for varmegjennomgangskoeffisientene og om det er en sammenheng mellom disse verdiene og energiforbruket.

- Test 1: Varmegjennomgangskoeffisienten under kalde dager?

NO-MAS og MAS (1116 målinger) Illustrasjon 26:



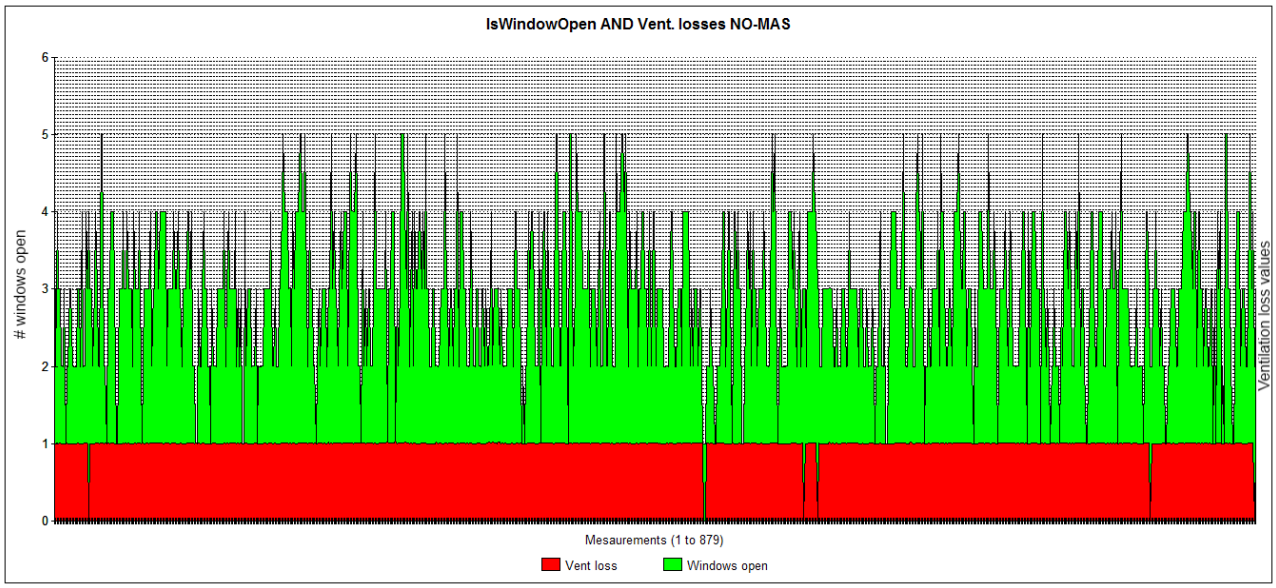
Illustrasjon 26: Varmegjennomgangskoeffisienten under kalde dager

Testen viser en viss, eller ikke merkbar, sammenheng mellom varmegjennomgangskoeffisienten (rød farge) og utendørstemperatur (grønn fagre) for kalde dager.

Fra scenario 2 ble det konkludert med at det fantes en sammenheng mellom oppvarming og energiforbruk, det er derfor grunn for å tro at transmisjons- (se 5.2 Transmisjons-tap) eller ventilasjons-tapet (se 5.3 Ventilasjons-tap) har en virkning på energiforbruket. Neste test ser på ventilasjons-tapet (Test 2a og 2b) og vindu i rommet, ved å måle ventilasjons-taps verdien når et vindu står åpent og når det er lukket. Siste test i dette scenarioet (Test 3a og 3b) ser på om det er noen effekt på transmisjons-tapet og energiforbruket, ved å overlappe verdier for transmisjons-tap med verdier for energiforbruk.

- Test 2a og 2b: Finnes det sammenheng mellom åpne vindu og ventilasjons-tapet?

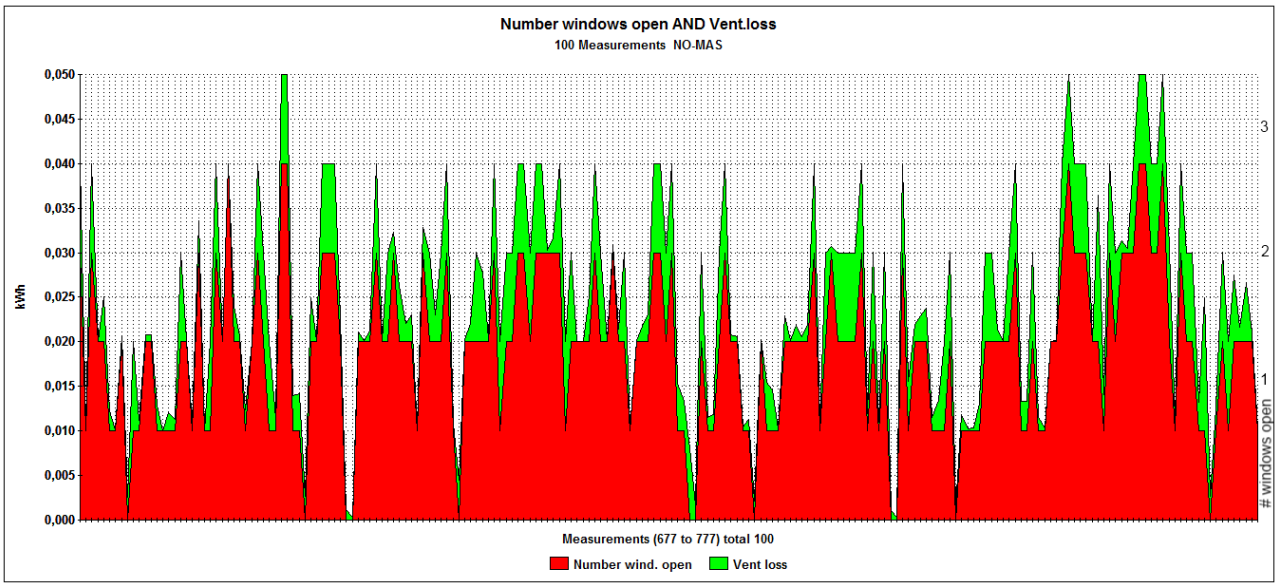
NO-MAS (879 målinger) Illustrasjon 27:



Illustrasjon 27: TEST 2a NO-MAS med variabler for åpent eller lukket vindu og ventilasjons-tap

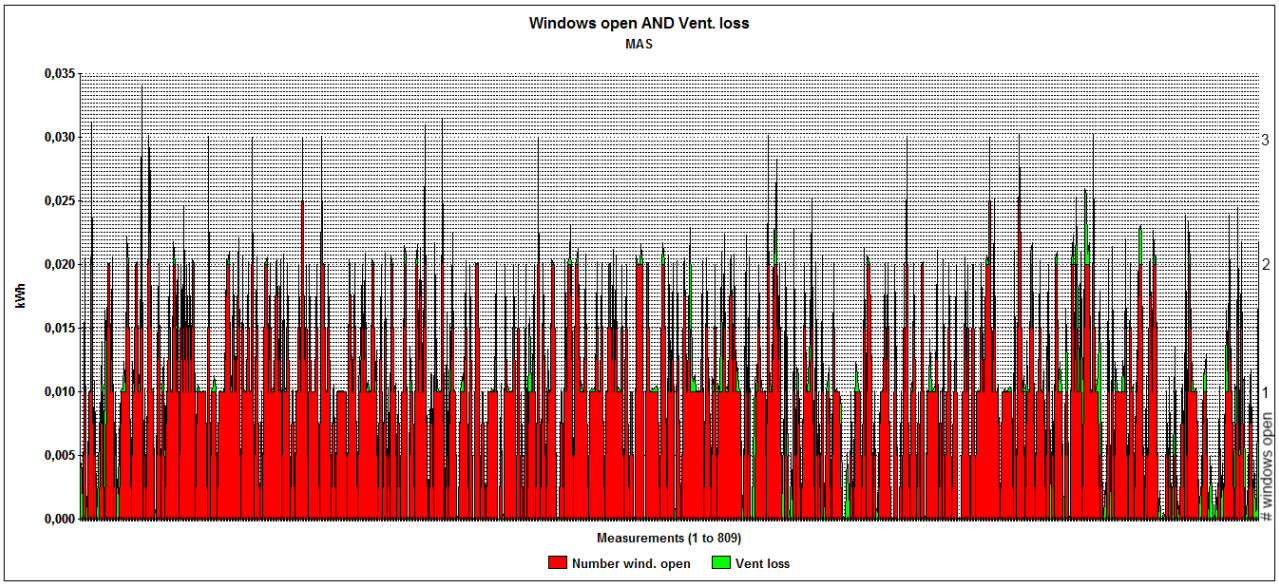
Det er vanskelig med store datamengder som i illustrasjon 27, å vise at ventilasjons-tap verdiene faktisk øker når vindu i rom står åpent og illustrasjon 28 har derfor utsnitt for 100 målinger for å fremstille dette tydeligere. Dette gjentar seg også for MAS i test 2b (illustrasjon 29 og 30).

Åpne eller lukke vinduer i NO-MAS og MAS simuleres ved bruk av Random agenter (se 4.6.3 Random) for å etterligne menneskelig oppførsel. For MAS finnes det en beslutningsprosess som tar kontroll over vinduet og dermed minsker også ventilasjons-tapet. Gjennomsnittsverdien for ventilasjons-tapet målt under 540 dager er i NO-MAS på 0,0033383 mot MAS -0,000342714 viser en forskjell på 0,002995586 og selv om dette er små verdier er det en målbar besparing på energien i form av mindre tap i romtemperaturen.



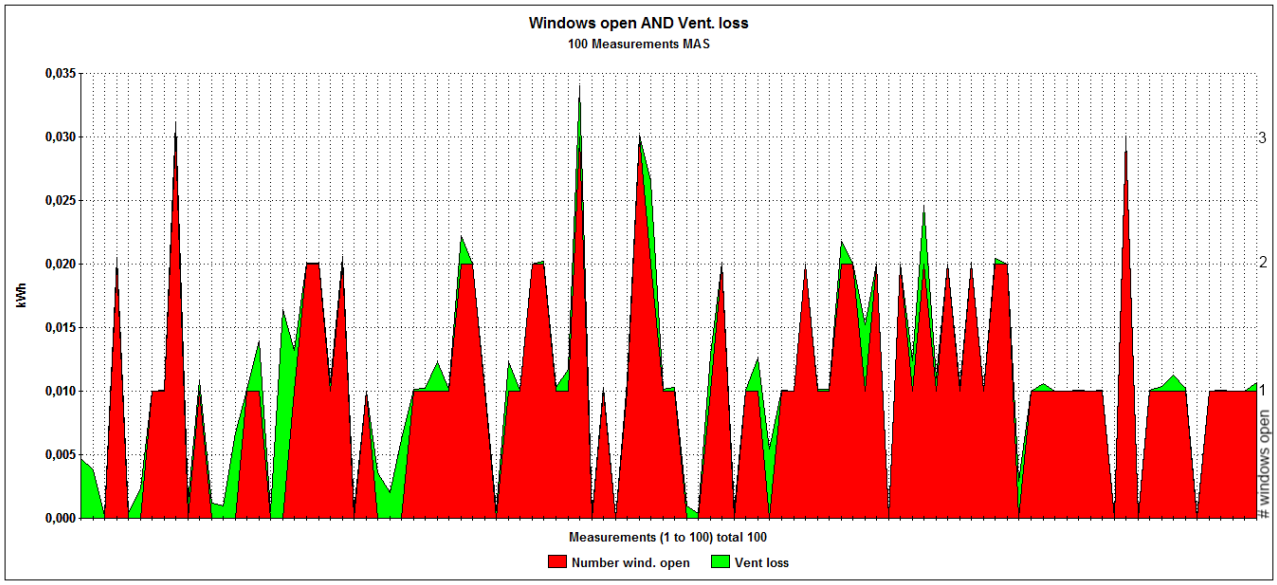
Illustrasjon 28:

MAS (808 målinger) Illustrasjon 29:



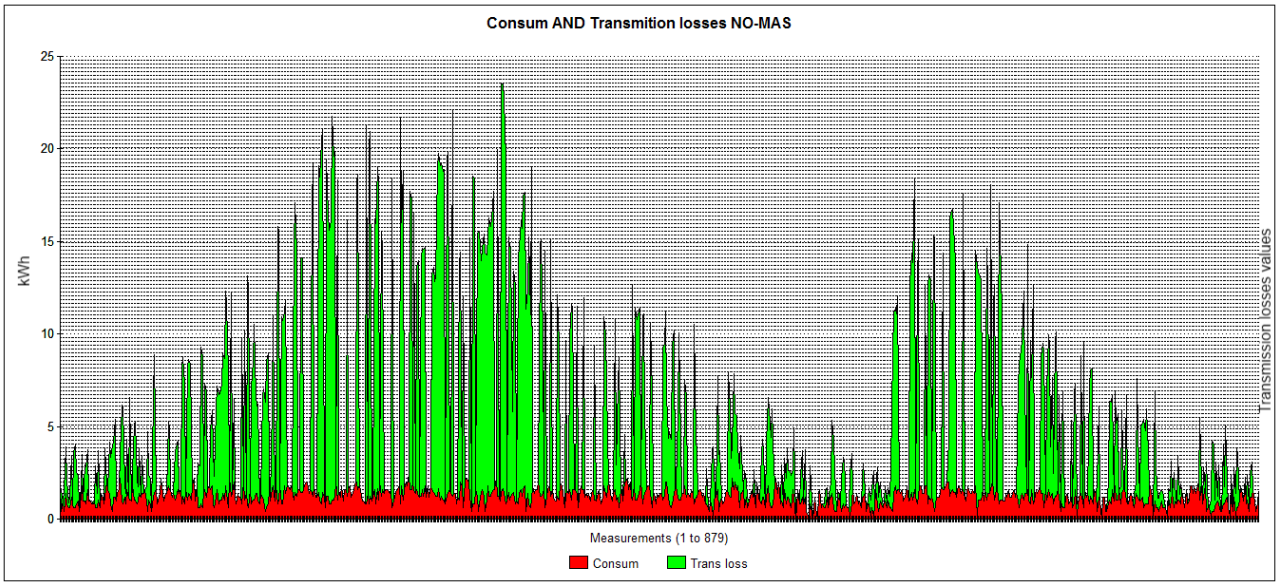
Illustrasjon 29: TEST 2b MAS med variabler for åpent eller lukket vindu og ventilasjons-tap

Illustrasjon 30: TEST 2b verdier for åpent vindu og ventilasjons-tap under 100 målinger.



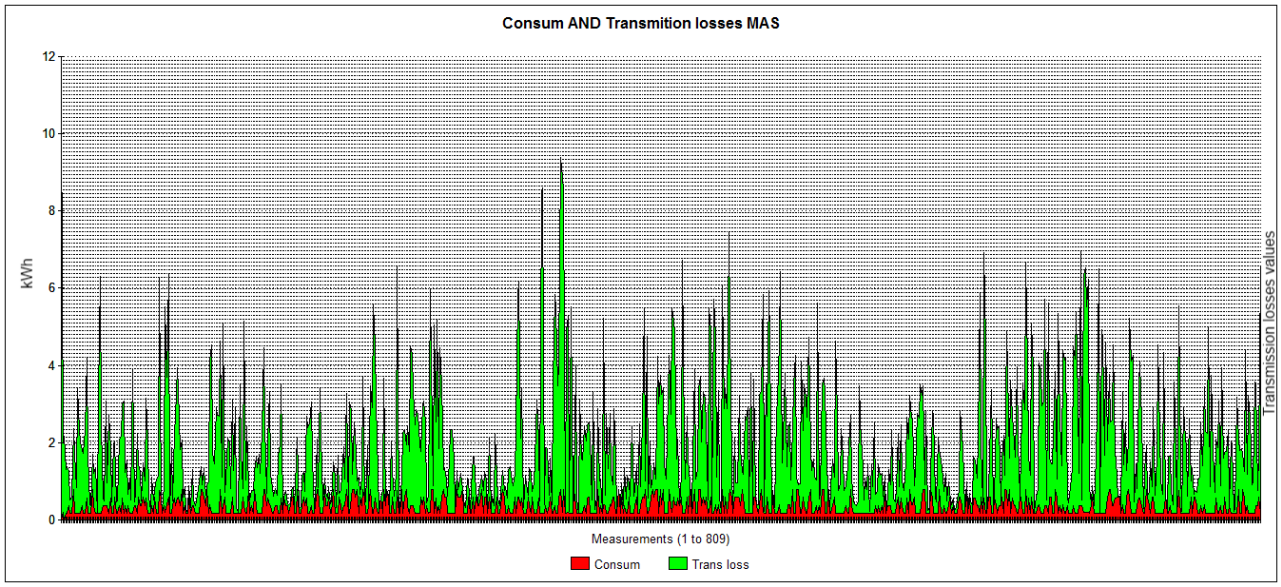
- Test 3a og 3b: Finnes det sammenheng mellom transmisjons-tapet og energiforbruk?

NO-MAS (879 målinger) Illustrasjon 31:



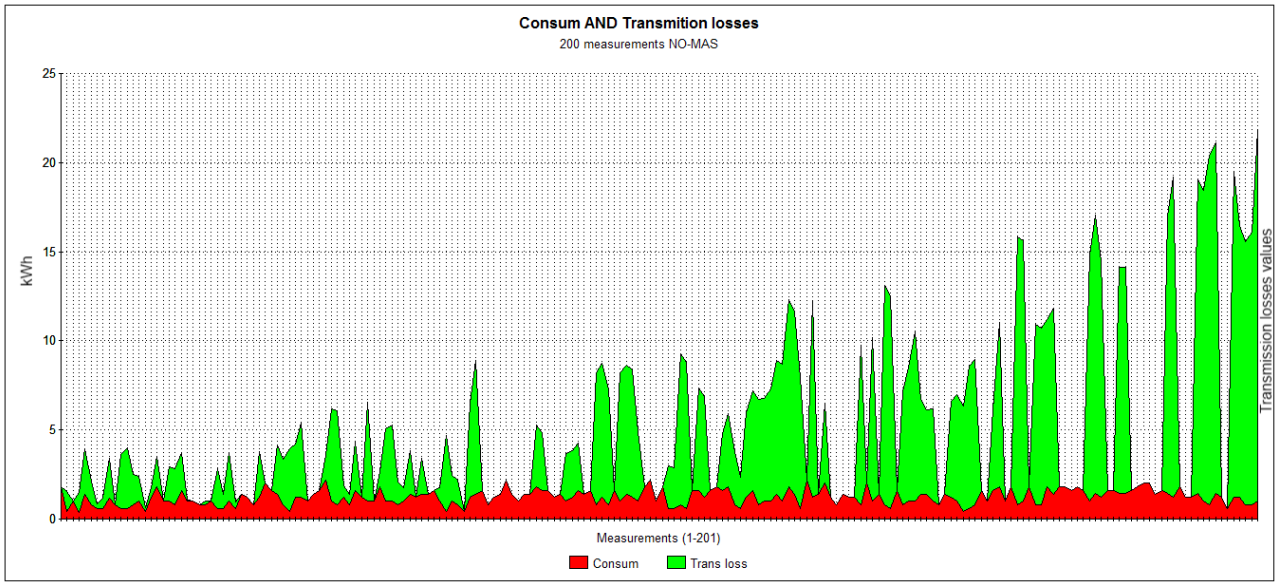
Illustrasjon 31: Test 3a NO-MAS målinger på forbruk og transmisjons-tap

MAS (808 målinger) Illustrasjon 32:



Illustrasjon 32: Test 3b MAS målinger på forbruk og transmisjons-tap

Fra begge målinger illustrasjon 31 og illustrasjon 32 er avstanden i verdiene på forbruk og transmisjons-tap svært store og gjør det vanskelig å finne målbare verdier. Det ser ut til at det finnes en viss sammenheng mellom energiforbruk og transmisjons-tap for NO-MAS modellen. Med en viss sammenheng menes her at ved å forstørre opp datasettene finnes det likhetstrekk i grafene som kan antyde at når transmisjons-tap øker, så virker dette inn på energiforbruk som sees på illustrasjon 33.



Illustrasjon 33:

Konklusjonen på dette scenarioet er at det finnes en viss, eller lite merkbar sammenheng mellom utendørstemperatur og varmegjennomgangskoeffisientene. Det var ikke mulig å finne noen sammenheng for NO-MAS modellen mellom energiforbruk og transmisjons-tap. For ventilasjons-

tapet var sammenhengen målbar ved åpent og lukket vindu for begge modeller og selv om det er små marginer på verdiene gir det forskjell i energiforbruket. For MAS modellen kan det konkluderes med at det finnes en viss sammenheng mellom energiforbruk og transmisjons-tap.

8.1.8 Konklusjon fra scenarioene

Sett fra mindre forbruk i energi gir kapitel 8.1.1 *Utgangspunktet for målinger, scenario 0,* et positivt bilde av husmodeller med et sofistikert agent system og illustrasjonen 9 viser en markant differanse i forbruket mellom MAS og NO-MAS. Tre scenarioer er laget for å se på gjennomsnitt forbruk, energiforbruk og oppvarming, varmegjennomgangskoeffisientene kontra forbruk og inneholder til sammen 9 tester som har sett på hvilke variabler som er knyttet til de forskjellige kontekstene. På slutten av hvert scenario finnes en konklusjon som sier noe om resultatet til testene og som skal danne verdiene for dette avsnittet.

Det første scenarioet (8.1.2 gjennomsnitt forbruk MAS og ikke MAS klokken 00:00 og 08:00) ser på sammenhengen mellom antall aktive individer totalt i husmodellen og det generelle energiforbruket fra komponenter i alle rom. Scenarioet konkluder med at det finnes sammenheng med øking av energi når et individ kommer inn eller befinner seg i et rom og er i aktiv tilstand.

Andre scenario (8.1.3 energiforbruk og oppvarming MAS og NO-MAS) er todelt hvor den ene delen ser på om det finnes en målbar sammenheng mellom bruk av komponenter som varmer opp rom og energiforbruk, mens den andre delen ser nærmere på sammenhengen mellom aktive individer og energiforbruk rommene enkeltvis. Konklusjonen er at det finnes i rom individuelt en sammenheng mellom verdiene for oppvarming, aktive individer og energiforbruk for begge modeller i simulatoren. For MAS er verdiene for spart energi opptil ca. 70% sammenlignet med NO-MAS, dette kan blant annet sees på kortere frekvens for oppvarmingelementet, heater, true modus i MAS modellen.

Tredje scenario (8.1.4 varmegjennomgangskoeffisientene og forbruk) ser på utendørstemperatur, energiforbruk og varmegjennomgangskoeffisientene for husmodellene generelt. Det konkluderes med å finne felles for begge modeller målbar variasjon for varmegjennomgangskoeffisientene. For variabelen transmisjons-tap (se 5.2 Transmisjons-tap) ble det i NO-MAS ikke funnet noen sammenheng for forbruk og transmisjons-tap, mens det i MAS ble konkludert med å finne en mindre målbar sammenheng mellom disse to. Ventilasjons-tap (se 5.3 Ventilasjons-tap) var verdiene for begge modellene i testen målbar ved å se på variabelen for åpent eller lukket vindu, men med svært små marginer og variabelen ble forkastet for videre testing i dette prosjektet..

Scenarioene kan konkluderes med følgende (sortert etter viktighet):

1. Det finnes sammenheng mellom aktive individer og energiforbruk i rom for begge modeller som kan si noe om differansen mellom disse målt i tid og forbruk. Med tid menes hvor lenge en komponent behøver å være aktivert for å oppnå tilfredsstillende resultat og med forbruk menes mengden av energi komponenten forbruker under aktiviteten. Denne konklusjonen kan brukes for å si hvilken modell som oppnår best mulig resultat i simuleringen og si noe om kontroll over energiforbruket.
2. Det finnes en sammenheng mellom oppvarming og energiforbruk for begge modeller som sier noe om differansen mellom disse målt som forbruk over en periode på 540 dager. Denne konklusjonen kan si noe om hvilken modell som er mest energibesparende.
3. Det finnes en viss målbar sammenheng mellom transmisjons-tap og forbruk i MAS.

9 Konklusjon

Resultater etter testing av forskjellige scenarioer viser at det i MAS-ENERGY gir gode resultater ved å bruke et system med lærende agenter som kommuniserer med hverandre og som gir bedre kontroll over energiforbruk. Modellene for MAS har vist seg rasjonelle ved å vise hvordan faktorer som utendørstemperatur og ventilasjon gir riktige beslutninger ovenfor innendørs klima og gir den beste brukertilfredsstillelse for hvert enkelt individ. I MAS brukes ikke et eget definert agentspråk, men maskinlære teknikker slik som beslutningstre, hvor logiske modeller brukes av agentene for å lære deduktivt (ved hjelp av eksempler) slik at den beste beslutningen finnes. Sammen med beslutnings-tre og bruk av observasjons mønster for kommunikasjon mellom agentene, har dette resultert i lavere energiforbruk en det mindre sofistikerte systemet NO-MAS.

Konstanter for NO-MAS modellene, slik som som maks- og minimum- verdier, blir brukt i kondisjonelle regler som gir mindre kjøretid på simuleringen sammenlignet med MAS modellene som må gjennom læreprosesser for hver iterasjon. Dette betyr i praksis at beslutninger kommer svært rask og i noen tilfeller med akseptable verdier for sparing av energiforbruk. Men hva om viten omkring tilstanden til objektet som er i konteksten? For objekter i NO-MAS hvor kommunikasjon er ikke eksisterende, har disse ingen kjennskap om eksistensen av andre objekter i verden, som vil si at for en agent som skal utføre justering av romtemperatur, kjenner denne ikke noe til om hvorfor det er forandring i temperaturen eller hvilken eventuelle bi-effekt handlingen vil ha. Dette løses i MAS fordi alle beslutninger som foretas gjøres av en hovedagent som kjenner til alle tilstander i rommet og dermed kan vise til **hvorfor** det er forandring i temperatur.

Konflikter mellom agenter er et større problem innen MAS og årsak til mye forskning innenfor AI feltet. For MAS-ENERGY er det ikke laget noen løsning som fører til at denne problemstillingen løses, men i motsetting til Boman et. al. (1999) simulator som bruker et mellomlag for forhandling mellom agenter med en «pronouncer» hvis det oppstår konflikt, er det i MAS-ENERGY en agent som benytter maskinlære teknikker for å gi tillatelser til agenter som skal utføre handlinger. Dette har ført til en konfliktfri simulering per se.

På bakgrunn av denne konklusjonen sies det dermed at hypotese 1 er korrekt:

> *H1 - er at bruk av intelligente agenter som kommuniserer og lærer kan senke totalt energiforbruk sammenlignet med mindre sofistikerte systemer.*

10 Videre utvikling

Formålet med denne oppgaven var å undersøke om MAS med lærende agenter kan være medvirkende til å senke energiforbruk i et smart-house. Det er mange aspekter som fremdeles behøver mer oppmerksomhet og kan føre til mer variasjon av resultatene en dagens simulator og her presenteres noen.

10.1 Individene

Individer (se 4.6 populasjon) har i denne versjonen ikke annen funksjon en å være aktiv eller ikke aktiv og selv om programvaren har tatt høyde for utvidet bruk av individer i simuleringen for å forbruke mer energi, er dette ikke inkludert. Årsaken til å ekskludere denne funksjonen var at det ikke er individer som er i fokus, men hvordan energi kan spares og det var ikke behov for flere parametre en oppvarming for å vise til resultater. I en fremtidig versjon kan det tenkes på hvordan individer som behøver assistanse på grunn av nedsatt funksjon kan bli hjulpet av intelligente systemer, enten ved å påminne dem om en tilstand til en komponent som å slukke lys, trekke ned gardiner eller som kan agerer på deres vegne. Slike systemer kan kanskje benytte seg av utility based agenter (se 6.2 utility based agenter) og bayesianske nettverk hvor man kan tenke seg at *health* (se 12.2 javadoc population.abstracts.isPerson setHealth()) variabelen står sentralt?

10.2 Plot og smart-grid

I en tenkt verden hvor flere instansieringer av MAS-ENERGY er koblet sammen med en simulator for smart-grid, kunne det være interessant å måle forbruk som kan spares ved at de forskjellige MAS kommuniserer sammen. Det vil kanskje være mulig å koble inn alternative energikilder, slik som sol- vind- eller bølge kraft når strømforsyning svikter på hovednettet? Kanskje finnes det muligheter for at smart-house kan avsette sine alternative energi kilder til andre hus i området hvis deres forsyning svikter? Eller at det kan oppdages lagringsmuligheter for energi som idag kun dekkes av batteri funksjoner?

10.3 AI relatert

AI teknikker brukt i MAS-ENERGY har vært regelsett (se 6.1.1 kondisjonelle baserte regler) , beslutningstre (se 6.3 beslutningstre og ID3) og informerte søk (se 6.4 informerte søk) som i stor grad har operert uavhengige av hverandre unntatt i, for eksempel enkelte logiske modeller for Controller, hvor det ble laget regler som senere ble brukt i eksempelsettene for beslutningstre. Det vil være interessant å kunne utvide disse teknikkene for å oppnå bedre målinger av energiforbruk

eller si mer om individer og dermed aktivitetene for hver modell. Med bedre måling menes som eksempel forskjellen av en plassering for et individ, nært eller fjernt en ventilasjons-taps kilde vil ha på brukertilfredsstillelsen eller om forskjellen mellom en type kilde til oppvarming vil være bedre en annen type.

Spesielt for beslutningstre er tanken om å utvides til mer rikere klasser som sier mer om tilstanden, et mål for fremtidige versjoner av MAS-ENERGY og kan oppnås ved å bruke C 4.5 algoritmen som nevnt i Giorgio Ingargiola (u.å.) i stedet for ID3. Dette kan gi fordeler som ved bruk av kontinuerlige data slik som temperatur, varmegjennomgangskoeffisienter i modeller og treningsett. Informerte søk kan da brukes i et søke rom av kontinuerlige data til C 4.5 og selv om dette kan være kostbart, ville det være mulig å sende beslutninger videre til en utility agent (se 6.2 Utility based agent) som tar best mulige beslutning i forhold til sin utility funksjon. Dette betyr kanskje i praksis at det må finnes en løsning på hvordan en læreprosess kan ivaretas etter hver iterasjon, slik at simuleringen ikke blir for kostbar i tid?

Refleks agenter i MAS-ENERGY kan utvides til å omfatte flere kondisjonelle regler og utvide sitt søk område til å gjelde flere rom eller hele husmodellen. Nåværende versjon tillater kun agenter til å handle *en* komponent og bruke data fra rommet komponenten befinner seg i. Dette vil i praksis kunne føre til et større MAS hvor en refleks-agent kan inneholde en liste av andre refleks-agenter, holde på deres oppdateringmetode og bruke et sett av kondisjonelle regler utfra persepsjonen for å utføre en handling.

10.4 Scenarioer

Denne versjonen av MAS-ENERGY har scenarioer som sier noe om energiforbruk for systemer som bruker sofistikerte agent systemer kontra mindre sofistikerte. Det er mange muligheter i fremtidige versjoner for å bruke flere parametre eller å utvide eksisterende til å omfatte flere attributter, kun fantasien setter grenser her. Et forslag kan være bedre bruk av variablene for varmegjennomgangskoeffisientene og spesielt transmisjons-tap (se 5.2 Transmisjons-tap), blant annet med å kalkulere korrekt mengde kilowatt per time i forhold til taps verdien. Bruk av dårlig materiale for nybygg eller slitasje er som oftest årsak til dårlige verdier for transmisjons-tap og for fremtidige versjoner kan det være mulig å bruke detektorer for å sjekke tilstanden og finne avvik fra normalen. Dette vil effektivisere og lettere gjøre vedlikehold av hus og føre til sparte kostnader både i tid og økonomi.

11 Referanser

11.1 Bøker:

Erich Gamma, Richard Helm, Ralph Johnson og John Vlissides (1995-2005), Design Patterns, «Elements of Reusable Object-Oriented Software», Addison-Wesley, ISBN- 0201633612

Stuart Russel og Peter Norvig (2003), Artificial Intelligence «A Modern Approach», Pearson education Inc.

11.2 Internett:

AbstractFactory (2011) Design mønster [Internett] tilgjengelig fra:
http://www.dofactory.com/Patterns/Patterns.aspx

AgentSpeak (2010), Jason -Java-based interpreter for an extended version of AgentSpeak, [Internett] Tilgjengelig fra:
http://jason.sourceforge.net/Jason/Jason.html

Cyryl Krzyska (2006),Smart House Simulation Tool [Internett] PDF tilgjengelig fra:
http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=4973

Davidsson, P., and Boman, M. (1998). Energy saving and value added services: Controlling intelligent buildings using a multi-agent systems approach. In DA/DSM Europe DistribuTECH, PennWell. [Internett] Tilgjengelig fra: http://www.enersearch.com/company/knowledgebase/publications/by_project/ISES/ises9/da-dsm98/da-dsm98paper.pdf

Eclipse EE (2011), IDE integrated development environment [Internett] tilgjengelig fra:
http://www.eclipse.org/

eKlima (2010), Metrologisk institutt, historiske værdata [Internett] Tilgjengelig fra:
http://sharki.oslo.dnmi.no

ERA-NET (2010), A smart European electrical grid, [Internett] Tilgjengelig fra:
http://ec.europa.eu/research/fp7/pdf/era-net/fact_sheets/fp7/smartgrids_en.pdf#view=fit&pagemode=none

ECHELON (2010), LonWorks 2.0, [Internett] Tilgjengelig fra:
http://www.echelon.com/products/lonworks_platform.htm

Frico (2011) Teknisk håndbok [Internett] PDF tilgjengelig fra:
http://www.frico.se/no/pdf/technical_handbook_fan_heaters_no.pdf

Giorgio Ingargiola (u.å.), Building Classification Models: ID3 and C4.5, [Internett] Tilgjengelig fra
http://www.cis.temple.edu/~giorgio/cis587/readings/id3-c45.html

H. Raiffa (1968), Data Interactive Design Analysis [Internett] Tilgjengelig fra:
http://books.google.es/books/about/Decision_analysis.html?id=vQ9PAAAAMAAJ&redir_esc=y

Java (2011) Java ServerSocket [Internett] tilgjengelig fra:
http://download.oracle.com/javase/1.4.2/docs/api/java/net/ServerSocket.html

Json (2011) JavaScript Object Notation [Internett] tilgjengelig fra:
http://www.json.org/

Java (2011) [Internett] tilgjengelig fra:
http://www.java.com/en/download/index.jsp

Java Observable(2011), [Internett] Tilgjengelig fra:
http://docs.oracle.com/javase/1.4.2/docs/api/java/util/Observable.html

JUnit (2011) Junit testing framework [Internett] tilgjengelig fra:
http://www.junit.org/

J. Ross Quinlan (1975), The ID3 Algorithm, [Internett] Tilgjengelig fra:
http://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-papers/2.htm

Kevin Bouchard et. al (2009) SIMACT [Internett] Tilgjengelig fra:
http://sourceforge.net/projects/simact/

https://springerlink3.metapress.com/content/j4g35l38913w2j0t/resource-secured/?target=fulltext.pdf&sid=5wqr4tjn2mpaycqlwwxjutan&sh=www.springerlink.com

Kevin Bouchard et. al (2009) SIMACT [Internett] Tilgjengelig fra:
http://sourceforge.net/projects/simact/

Kevin Murphy (2012) , A brief introduction to Bayes' Rule, [Internett] Tilgjengelig fra:
http://www.cs.ubc.ca/~murphyk/Bayes/bayesrule.html

Lovdata (2010), Forskrift om krav til byggverk og produkter til byggverk (TEK), [Internett] Tilgjengelig fra:
http://www.be.no/beweb/info/energikurs07/Energi-veilederkurs/Forskriftstekst%20og%20veiledning/TEK_8-2.pdf

Lawrence Turner (2012) , Boolean Expression Truth Table Generator, [Internett] Tilgjengelig fra
http://turner.faculty.swau.edu/mathematics/materialslibrary/truth/

M. Boman, P. Davidsson, and H.L. Younes. (1999) Artificial decision making under uncertainty in intelligent buildings. [Internett] Tilgjengelig fra:
http://www.tempastic.org/papers/uai1999.pdf

Netica (2011) , Norsys Software Corp. [Internett] Tilgjengelig fra:
http://www.norsys.com/netica.html

Observer Pattern (2011) Design mønster [Internett] tilgjengelig fra:
http://www.dofactory.com/Patterns/Patterns.aspx

The SmartHouse/SmartGrid Vision (2010) [Internett] Tilgjengelig fra:
http://www.smarthouse-smartgrid.eu/index.php?id=245

SIMACT (2009), 3D smart home simulator[Internett] Tilgjengelig fra:
http://sourceforge.net/projects/simact/

Stephen D. J. McArthur et. al. (2007) IEEE Transactions for power system, Vol. 22, No.4 [Internett] tilgjengelig fra:
http://www.supergen-amperes.org/Library/Multi-Agent%20Systems%20for%20Power%20Engineering%20Applications%20Part%201.pdf

SMILE (2011) , GeNIe. [Internett] Tilgjengelig fra:
http://genie.sis.pitt.edu/about.html

Subversion (2012), Universitet i Bergen, IT-avdelingen , [Internett] Tilgjengelig fra:
https://it.uib.no/Subversion

SEEDS (2011), Self Learning Energy Efficient builDings and open Spaces , [Internett] Tilgjengelig fra
http://www.seeds-fp7.com/summary.php

UMLet (2011) Free UML Tool for Fast UML Diagrams [Internett] tilgjengelig fra:
http://www.umlet.com/

12 Vedlegg

12.1 Kode

Fås ved hendvendelse til eirik.bremnes@gmail.com.

12.2 Illustrasjoner

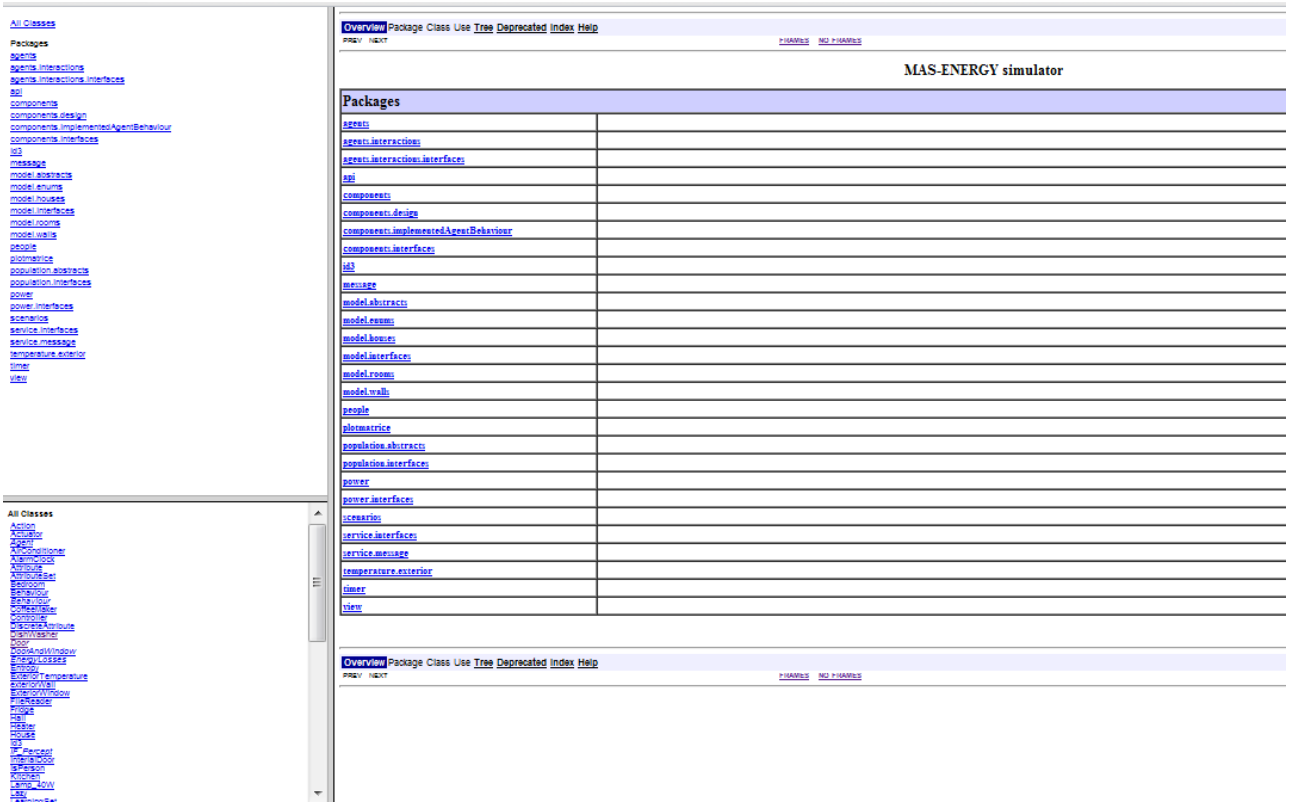
12.2.1 UML

Fås ved hendvendelse til eirik.bremnes@gmail.com.

12.2.2 Scenarioene

Fås ved hendvendelse til eirik.bremnes@gmail.com.

12.2 Java dokumentasjonen (javadoc)



Java dokumentasjon Fås ved hendvendelse til eirik.bremnes@gmail.com.