SEVENTH FRAMEWORK
PROGRAMME

THEME 8

Socio-economic sciences and Humanities

Collaborative Project, Small or medium scale focused research project

GILDED Work Package Five Report:

Agent-Based Modelling

Project acronym:	GILDED
Project full title:	Governance, Infrastructure, Lifestyle Dynamics and Energy Demand: European Post-Carbon Communities

Grant agreement no.: 225383

Date: 30 April 2012

1 Introduction

This report describes GILDED work package 5, Agent-Based Modelling. GILDED (Governance, Infrastructure, Lifestyle Dynamics and Energy Demand) is a European Commission Seventh Framework Programme project funded under the topic *Socio-Economic Factors and Actor Shaping the "Post-Carbon" Society*.

Agent-based modelling stresses the interaction between social actors (most often individuals or households) and their physical and social environment. It may be contrasted with conventional economic modelling, which generally assumes rational, self-interested individuals, making their decisions on the basis of complete information; even where restrictions are placed on rationality or the availability of information, they are conceptualised as deviations from this abstract ideal. Agent-based modelling, on the other hand, focus on the relatively simple heuristics human agents employ for most of their dealings with the external world, and on the influence that habit and social interaction have on how they behave. Agent-based modelling has been extensively used in the study of social dilemmas: situations in which each of a group of actors will be better off if all cooperate, for example to limit use of some resource, but each has an incentive to cheat (Gotts, Polhill & Law, 2003).

A classic study is of the use of rain-fed irrigation systems in Bali (Lansing & Kremer, 1993) which indicated that bottom-up organisation of planting and cropping schedules could generate near-optimal results without central planning. Agent-based modelling has been combined in sustainability studies with a variety of data-gathering techniques, including sample surveys (Schreinemachers & Berger, 2006) in a study that aimed to disentangle the combined effects of soil fertility decline, population growth, and market institutions on the dynamics of poverty and productivity in Uganda. Alcamo (2001) described a method for combining simulation modelling with scenario development, the "story and simulation" (SAS) approach, in which scenario storylines produced by experts or stakeholders are reiteratively refined and quantified using simulation modelling, large numbers of model runs are used to find a policy approach that will give acceptable results across a wide range of possible futures. While the work reported here has not used either approach as described by these authors, it contains elements of both.

In the context of GILDED, agent-based modelling constitutes work package 5. The function of the work package within GILDED is described as follows in the Description of Work:

> Agent-based modelling acts as a test field and means of scaling up findings from field research. Principles drawn from the lifestyles and structural analyses will be utilised designing the structural features ('rules of the game'), communication and interaction networks, and decision-rules guiding agent behaviours in the models, thus illuminating the interplay of these issues. The chief contribution of agent-based modelling to the domain of computer simulation is through the recognition of the importance of individual differences in social system (as opposed to the assumption made in neoclassical economics that such differences generally cancel out), and of the significance of interactions among individuals in generating behaviour observed

> at the social scale. As such, agent-based modelling draws on complex systems theory (Holland and Miller, 1991). *GILDED will develop agent-based models of energy use which demonstrate potential outcomes of specific energy-related policies and initiatives. It will form a key part of the project's foresight component, being used to explore clusters of related scenarios for the period to 2050. Each such cluster will investigate the potential of specific policy instruments in reducing carbon-intensive energy demand.*
>
> Use of agent-based modelling will also provide a real-world test-bed for a new approach to developing agent-based models, based on using "ontologies" (Staab and Studer 2004). An ontology in this sense is a formal classification scheme for the entities in a computer or conceptual model, in a narrative, or in the system a model is intended to represent. The appropriate use of ontologies will make the structure of agent-based models more explicit and open to discussion with non-specialists, through using ontologies as intermediate formal descriptions between natural language and computer programming languages.

This document describes and explains the work we have undertaken in pursuit of the objectives outlined above. Following this introduction, the report first describes (section 2) ABMED (Agent-Based Model of Energy Demand) the prototype model constructed in the early stages of the research. Section 3 then covers our work on eliciting ontological information from the Scottish case study Stakeholder Advisory Group, and from our non-modelling colleagues within GILDED. Section 4 is an overall description of CEDSS, the revised model, based on ABMED, the ontology elicitation exercises, and data collected both from the questionnaire on values and behaviours related to energy and climate change, and the accompanying carbon calculator, distributed to households in Aberdeen City and Aberdeenshire in work packages 3 and 4, and from a wide range of publicly available sources. The description follows an increasingly widely-used protocol for describing agent-based and related models, ODD (Grimm et al 2006, 2010), which has many features in common with the ontological approach. Section 5 describes how an ontology can be automatically extracted from models written in the Netlogo language (both ABMED and CEDSS are written in Netlogo).

The next three sections describe how empirical data has been used in implementing CEDSS, and the results we have obtained so far from running it. CEDSS requires a number of input data files in order to run; section 6 describes those that remained fixed, or partly so, for all the model runs we performed while calibrating the model, on the urban subsample of households who completed the GILDED questionnaire and carbon calculator in 2010. Section 7 describes the process of calibration, and of validating the calibrated model, CEDSS 2A:14, on the corresponding rural subsample. Section 8 describes the scenario runs of CEDSS 2A:14 to 2050, under a range of possible conditions and policy initiatives. We conclude that the tendencies of recent years for domestic energy for heating to decrease, while increases in energy demand for household appliances partially offset this trend, are likely to continue, and will be modified primarily by the trajectories of household incomes and fuel prices. However, policy measures directly aimed at reducing domestic energy demand, in particular the regulation of appliance efficiency, can exert a significant downward pressure on the increase in household appliance energy demand. Moreover, we see signs that individual communities may show anomalously low domestic energy demand, possibly as a result of social interactions leading to the spread of unusually pro-environmental values. Finally, section 9 draws conclusions, and indicates directions for future work with CEDSS.

2 The Prototype Model: ABMED

This work was presented to the Sixth Conference of the European Social Simulation Association 2009 (Gotts 2009).

2.1 Introduction

Household energy use and personal transport account for a considerable proportion of total energy use, and greenhouse gas emissions, in countries. In Europe, about 35% of all primary energy use and 40% of all greenhouse gas emissions come from private households—with regional differences (Mäenpää 2005, Weber and Perrels 2000). Home energy, personal travel, and food and beverages are the most important sets of activities. US studies find similar results (Bin and Dowlatabadi 2005). Given the vital importance of reducing greenhouse gas emissions from energy use, it is surprising how little attention has been given to the dynamics of household energy demand; and in particular to the interactions between technological change – which can act both to reduce energy use through greater efficiency, and to increase it by producing an unending supply of new household appliances – economic conditions, and socio- cultural forces. This section introduces and gives the first results from a prototype model of energy demand in a small community.

2.2 Method

ABMED was built as a prototype agent-based model of community energy demand, to act as a precursor to the main CEDSS model, presented later in this report. As such, it was intended to help in identifying and exploring socio-economic and psychological influences acting on, within and between households, which are both important in determining the level of demand. The need for rapid prototyping led to the decision to write ABMED in NetLogo (Wilensky 1999).

The agents in the simulations described here represent households; the individuals within these households are not represented. Each household is located in a house, in which it remains for the duration of the simulation (12 years). The houses are arranged on a grid of streets; in the simulations reported, this grid “wraps around” in both directions to produce a toroidal topology. A household has a net monthly income (this is taken to be net of costs other than those related to household energy consumption, and buying energy-using household appliances), which remains fixed for the simulation’s duration; net monthly. The net monthly incomes of households are drawn from an exponential distribution, with a mean that is a model parameter.

Households begin with just one appliance, a boiler, taken to provide space heating and hot water. Possession of a boiler is taken to be essential – it must be replaced if it breaks down (all appliances have an associated probability per month of breaking down irreparably, but ongoing maintenance costs are not represented). It may be assumed that households also possess other appliances generally taken to be essential in rich countries, such as a refrigerator and television, but these also are not currently represented in the model. During the course of the simulation, a household can acquire additional appliances.

An appliance has the following properties:

- "monetary-capital-cost": the amount a household must pay to acquire it, taken to include installation costs, insurance, etc.
- "energy-capital-cost": how much energy it took to manufacture and deliver it – unrealistically, households are assumed to know this.
- "energy-monthly-cost-list": a 12-element list of average monthly energy usage: many appliances, not only those providing heating, will have different patterns of use at different times of the year.
- "essential": either true or false; in the current simulations, only a boiler is regarded as essential.
- "breakdown-probability": per month.
- "can-replace-list": a list of the appliances the given appliance can replace if they break down.
- "first-month-available", "last-month-available": the model has a monthly time-step, with new appliances being bought at the end of the month.

Values for the "monetary-capital-cost" and "energy-monthly-cost-list" have been based where possible on information from commercial websites, and some run by UK non-governmental organizations such as the Energy Saving Trust[1]; but the concern has been to get reasonable rather than precise values (the latter, of course, vary considerably across appliances of the same type, and in the case of energy use, across households); the "energy-capital-cost" and "breakdown-probability" values were estimated for the purposes of making explorations with this initial prototype. Appliances also have an identifying description drawn from a three-level categorization: high-level function (currently, just "heating" or "other"), general type of appliance ("standard-boiler", "condensing-boiler", "TV", "freezer", etc.), and specific type of appliance. For "other" appliances, replacement is allowed of any appliance of the same general type (e.g. any TV can replace another). Among heating appliances, there are two general types: "standard" and "condensing" boilers. In line with UK government regulations, it is not permitted to install a standard boiler (in the real world, some exceptions are permitted). Thus when a standard boiler breaks down, it must be replaced with a (considerably more efficient) condensing boiler; the household can also decide to make the change even without a breakdown. In the runs reported here, a condensing boiler becomes available at a subsidized price at month 24 (the first month being month 0 – both month 0 and month 24 represent January).

In addition to their monthly net income, households have an initial capital reserve. They can also borrow up to a fixed multiple of their monthly net income (this multiple is a model parameter). In addition, they have three parameters representing the likelihood that they will adopt a particular "goal frame" (Lindenberg and Steg 2007) when making decisions. In the term used by these authors, the three types of goal-frame: "hedonic", "gain", and "normative", represent competing clusters of values: roughly, those of immediate enjoyment, longer-term material gain, and doing what is perceived as right. In the context of ABMED, they appear as three alternative decision-making procedures, applied when deciding whether to buy an energy-using household appliance. The "hedonic" procedure makes decisions based on the "hedonic-score" of appliances; the "gain" procedure tries to

[1] http://www.energysavingtrust.org.uk/

minimize expenditure (buying only when an essential appliance must be replaced, or when a net saving can be expected (from buying a more energy-efficient appliance) within a set time, and economising on the use of those already owned); the "normative" procedure is similar but aimed at saving energy rather than money. Which is applied on any one occasion is determined probabilistically, but each household has its own set of probabilities. In the simplest case, these are fixed for the duration of a simulation run, and the expected value of each of the three probabilities concerned is 1/3. However, these expected initial values can be varied; and values for specific households can also be allowed to change over time as a result of social contacts.

At the start of a run, pairs of households are socially linked. The initial links are assigned probabilistically, with spatially closer households being more likely to be linked. Subsequently, each household has the opportunity each month to initiate either losing or gaining a link, with equal probability. If a link is to be lost, it will be among those with the most dissimilar set of possessions (i.e. household appliances); and among those, one of the most distant spatially. If a link is to be gained, it will be drawn at random from the contacts of a selected contact – the selection of the intermediary depending primarily on similarity of possessions, secondarily on spatial proximity. Social contacts will also influence what new appliances a household buys: whenever a decision is to be made, the possessions of a randomly selected contact (who is thought of as being "visited" at home) will have an enhanced chance of being chosen. At the same time, the household may (depending on a model parameter) adjust its values in the direction of those of that contact.

2.3 Results

The approach taken in the initial stages of exploring the dynamics of this model was:

1) To find as a starting point a set of parameters, which, where they cannot be constrained by readily available information (as prices and monthly energy use of appliances can, for example) produce "reasonable" model behaviour:
    - Most households acquiring some new appliances but not buying everything on the market.
    - Overall levels of energy use not changing too radically – the secular movement remaining no greater than the variation between winter and summer.
    - Overall levels of wealth likewise neither increasing nor decreasing too wildly.
2) Keeping the implementation fixed, and varying parameters systematically around the starting point, trying single runs of all combinations of a small sets of alternative values for a subset of parameters, and looking at the effect on a small number of global output measures, in this case primarily the mean over time of the overall energy use; secondarily the mean capital reserve (summed over households); and the number of social contacts.

The starting point parameters give the three "goal-frames" equal treatment in the initial allocation of characteristics to households. *For each household* at the start of a run, two random floating point numbers between 0 and 1 are chosen, and used to divide the unit interval into three (in general unequal) parts and hence generating three probabilities that sum to 1. These are then assigned at random to the “hedonic”, “gain” and “normative” decision procedures. Seven further parameter sets are then generated by assigning minimum probabilities, applied across all households, to one, two, or all three of the decision procedures. (If all three are assigned the same non-zero minimum probability, the expected probability assigned to each for each household is 1/3, just as it is if no minima are assigned; but the variance of probabilities across households will be reduced, i.e. households will be more alike in their values.) The number of parameter sets was increased to 24 by varying the availability of credit: a household being able to borrow 0, 5 or 10 times its net monthly income.

Table 2.1 shows the mean values across time for the total energy use for each of the runs in this first set. The rows show results for each combination of assigning or not assigning minimum initial probabilities for the three decision procedures: those for which a minimum is assigned being in bold underlined font. Columns indicate availability of credit.

Table 2.1. Mean monthly total energy use across households for each of 0, 5 or 10 times net monthly income (n.m.i) as availability of credit, for different settings of the goal frame parameters.

			No Credit	**5 × n.m.i.**	**10 × n.m.i.**
Hedonic	Gain	Normative	109349	102217	99741
Hedonic	Gain	**Normative**	81826	80977	78949
Hedonic	**Gain**	Normative	84486	82348	81204
Hedonic	**Gain**	**Normative**	80577	77884	76068
Hedonic	Gain	Normative	120409	119466	124104
Hedonic	Gain	**Normative**	115223	105948	106618
Hedonic	**Gain**	Normative	112801	115750	110303
Hedonic	**Gain**	**Normative**	107763	105273	100892

Conditions in which households are more likely to use the “hedonic” decision procedure have greater energy use, as might be expected. Conversely, greater likelihood of using either the “gain” or the “normative” decision-procedure tends to reduce energy use. Somewhat less obviously, when the hedonic procedure is not likely to be used, or when the normative is likely to be used, greater availability of credit reduces energy use (it does so even when both hedonic and normative decision procedures are more likely to be used than the gain decision procedure. Thus, even at a very early stage of investigation using agent-based models, we saw indications of interesting interactions between external economic conditions, and the “motivations” or “values” of the household agents. It may be noted that the same range of parameter sets gave rise to much greater variation in total capital reserve across agents, with clear (and expected) outcomes that greater hedonism, and greater availability of credit, gave rise to lower – and in some cases strongly negative – capital balances, often with a clear declining trend throughout the 144 months of the simulation.

Table 2.2 shows the mean total energy use across a different slice of the parameter space. Here, no minimum is set for the initial probability assigned to any of the decision procedures, but the credit limit is varied, and two further parameters are each given two different values: the monthly net income (this is intended to represent the income available for spending on household energy and energy-using appliances), and a measure of the number of consumption decisions made per household per month. The italicised cells indicate parameter sets duplicated from table 1, top row.

Table 2.2. Mean monthly total energy use across households for various availabilities of credit – 0, 5, 10 and 20 times net monthly income (n.m.i.), with different values for monthly net income (m.n.i.) and number of consumption decisions made per household per month.

m.n.i.	Consumption decisions	No credit	5 × n.m.i.	10 × n.m.i.	20 × n.m.i.
100	Low	86263	86195	86544	84584
100	Standard	*1108521*	*108513*	*102025*	101197
200	Low	92785	90727	88923	89015
200	Standard	131657	125271	113936	117487

As can be seen more consumption decisions produce greater energy use. The indication from table 2.1 that greater credit availability decreases energy use is confirmed, but not at the new, highest level (rightmost column). On the other hand, higher monthly net income increases energy use. This may seem paradoxical, but a likely explanation is that much credit is used early on, by households considering long-term savings of money or energy, before even the richer households have had time to build up capital reserves, to buy condensing boilers.

2.4 Discussion and conclusion

Agent-based modelling work on the use of land and natural resources is common; work on demand for centrally distributed services such as power and water is not. Perhaps the work that is closest to that reported here is the work on water demand described by Edmonds and Barthelemy (2002), and in much greater detail by Barthelemy (2007). That work, like this, combined considerations of technological change with spatially-embedded socio-cultural dynamics. However, the decision- making procedures were very different: agents influenced each other by exchanging "endorsements": recommendations of specific rules of behaviour, which would be more or less influential depending on the relationship between the two. Households observed all their neighbours, while in ABMED, there must be a social link between any two households, neighbours or not, before there is behavioural influence between them.

While technological innovation can reduce the energy requirement for specific activities, their impact in reducing carbon-intensive energy use will depend critically on broad public and political commitment to such a reduction. Without such commitment, increases in energy efficiency may simply raise demand for energy- intensive products and services either in the same category of energy use.

3 Ontology Elicitation

This work was presented to the Third World Congress on Social Simulation 2010 (Polhill et al. 2010).

3.1 Introduction

Each case study area in GILDED has a Stakeholder Advisory Group (SAG) to ensure relevance of research design and act as a platform for communication between the research team and the organisations members of the group represent. Scottish SAG members include a Member of the Scottish Parliament, and representatives from Aberdeen City and Aberdeenshire regional councils, the Scottish Government and local NGOs. Although the GILDED project is chiefly aimed at reporting to the EC, results from the models also need to be relevant to the SAG members. Those collaborators on the GILDED research team who do not have experience in modelling are also in some sense stakeholders in the model, since they will have the responsibility of synthesising model outcomes with other findings from the project when preparing the reports.

Though the resulting model needs to produce output relevant to all its stakeholders, none of them will be end-users of the modelling software. Neither will they necessarily receive output directly from the model itself—in the case of the SAG, for example, they are most likely to receive policy briefs that summarise research findings of which output from any simulation model will form only a part. Hence participatory modelling approaches (Ramanath and Gilbert 2004) are not appropriate here, and in any case involve a level of engagement in the development process that SAG members could not commit to.

Grimm et al.'s (2006, 2010) 'Overview, Design Concepts and Details' (ODD) protocol is a document structure aimed at encouraging authors to write complete descriptions of their models (to enable replication) in logically ordered manner. The logical ordering of the information is a possible explanation for why Grimm has subsequently argued that ODD is a useful way to think about designing and building a model. Though ODD is not universally adopted as a norm for describing agent-based models in text, its use is growing. ODD divides the description of a model into the following sections and subsections:

- **Overview**
    - *Purpose*—what the model is being used for.
    - *Entities, State Variables and Scales*—what is, or could be, contained in a snapshot of the model at any one time; its ontology.
    - *Process Overview and Scheduling*—brief summary of the dynamics of the model: how its state changes from one time to another.
- **Design Concepts**—short summaries of properties the model has under various headings (most of which pertain to agents' behavioural capabilities), to facilitate comparison between models: *Emergence*, *Adaptation*, *Objectives*, *Learning*,

Prediction, *Sensing*, *Interaction*, *Stochasticity*, *Collectives*, *Observation*. (Headings not addressed by a model can be left out of this section.)

- **Details**
  - *Initialisation*—how the initial state of the model is created.
  - *Input Data*—exogenous time series data driving the behaviour of the model.
  - *Submodels*—detailed descriptions of the processes outlined in the Overview section, including algorithms, rules and formulae.

Modellers are used to thinking about the world in a particular way, which enables them to translate observations into statements in programming languages. We assume here that ODD is representative of that way of thinking. There are 'entities', 'attributes' (which ODD terms 'state-variables') and 'processes'. To these we may add 'relationships' (between entities—in the terminology of object-oriented programming and design this would be aggregations or looser associations), and 'drivers' (exogenous influence on model behaviour—which in ODD are captured in the Initialisation and Input sections). Though the main purpose of the ontology elicitation exercise was to ensure the relevance of the model(s) to collaborators and stakeholders, we were also interested in how non-modellers respond to the key categories modellers use, and in their reaction to the ordering of information presented in ODD. We see this as providing two related insights—first, the extent to which ODD is useful for structuring the knowledge exchange process with stakeholders and non-modelling collaborators; second, how intuitive (if at all) the way modellers think about the world is to non-modellers. We are chiefly concerned with the categories associated with the Overview part of ODD.

3.2 Method

ODD has a certain logic when writing up an agent-based model: How can one discuss what is in the model until its purpose is clear? How can one discuss the processes that operate in a model until one has discussed what is in it? How can one decide how to implement these processes until one has decided the design concepts on which the model will be based? Likewise, how can one discuss the submodel implementations until the input data and initialisation are known?

It is interesting to see whether the way ODD breaks down information about a model is helpful in (at least partially) co-constructing it with stakeholders and non-modelling collaborators. In particular the way information is structured maps reasonably cleanly onto different knowledge exchange activities: the *Entities*, *State Variables and Scales*, together with the *Process Overview and Scheduling*, describe the ontology of the model (in a broad sense of the term), whilst the Initialisation and Input subsections could be driven in part by data collection, but also by scenario construction. The *Submodels* are implementation details, and though of primary interest to the model developers rather than non-modelling collaborators or stakeholders, those not involved in modelling may well be interested in the provenance of the implemented algorithms. The *Design Concepts* are quite an eclectic mix of headings. Six of them pertain to agent behaviour (*Adaptation*, *Learning*, *Objectives*, *Prediction*, *Sensing* and *Interaction*), and as such may be of interest to those with a background in some of the controversies surrounding representation of human decision-making. *Collectives* is an ontological matter. *Emergence* and *Observation* are scenario or purpose-driven concepts, depending on what the model is being used to explore. *Stochasticity* is a modelling methodology question.

The first workshop (A), with the six non-modelling collaborators (none of whom are SAG members), was used to try discussing the purpose of the model strictly before its ontology. The hour and a half-long workshop combined this with the demonstration of a prototype model (ABMED; Gotts, 2009), which also drew on the ODD protocol to structure the presentation, as follows:

A.1. Presentation on the context of the model, and the parts of the project it will feed into.

A.2. Open recorded discussion on what would be relevant or useful to see in the synthesis workpackage of the project, with a view to eliciting model purpose.

A.3. Participants asked to imagine creating a simulation to address the purpose discussed in step A.2, and then, “What would you put in the virtual world you would be creating?” One post-it to be used for each item in the virtual world.

A.4. Structuring the post-its—group discussion.

A.5. Re-organising the post-its into ontological categories.

A.6. Presentation of the prototype model, structured by the ODD protocol.

A.7. Opportunity to reflect.

Based on our experiences with workshop A, we designed a workshop (B) for the SAG (of whom six were present at the workshop). The agenda for the associated meeting allowed 1 hour 20 minutes for the workshop. This removed any discussion of the purpose of the model (which is in any case pre-defined in the GILDED project) and concentrated on eliciting an ontology, with the secondary purpose of ascertaining how intuitive key ontological categories (entities, attributes, relationships and processes) are:

B.1. Using an icon representing a house as a guide, participants were asked to use post-it notes to identify “anything associated with household energy use or household energy activities.”

B.2. Participants then put the post-its on the picture.

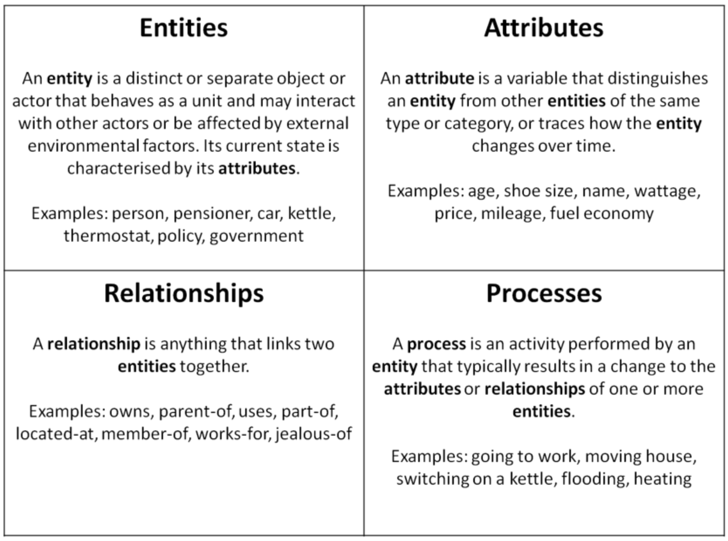
B.3. Some discussion of the post-its then took place, and participants were given the opportunity to add more post-its.

B.4. A demo of the prototype model was given.

B.5. The post-its were put onto cards on a table, and participants asked to sort them into categories that made sense to them.

B.6. The modelling categories were presented (figure 3.1) and participants were asked to re-sort the cards into these categories. While so doing, participants were asked how they responded to the modelling categories, and how difficult they were finding the task.

B.7. Participants were then given an opportunity to add more post-its.



Entities	Attributes
An **entity** is a distinct or separate object or actor that behaves as a unit and may interact with other actors or be affected by external environmental factors. Its current state is characterised by its **attributes**. Examples: person, pensioner, car, kettle, thermostat, policy, government	An **attribute** is a variable that distinguishes an **entity** from other **entities** of the same type or category, or traces how the **entity** changes over time. Examples: age, shoe size, name, wattage, price, mileage, fuel economy
Relationships	**Processes**
A **relationship** is anything that links two **entities** together. Examples: owns, parent-of, uses, part-of, located-at, member-of, works-for, jealous-of	A **process** is an activity performed by an **entity** that typically results in a change to the **attributes** or **relationships** of one or more **entities**. Examples: going to work, moving house, switching on a kettle, flooding, heating

Figure 3.1. Definitions of the modelling categories as presented to the SAG in step B.6. Those of Entities and Attributes are taken from Grimm et al. (2010).

A third workshop (C), was held with the GILDED project team at the consortium meeting in Budapest in June 2010. This followed broadly the same structure as workshop B, but omitted steps B.6 and B.7, and replaced steps B.1 and B.2 with a step in which participants individually wrote on index cards directly. Step B.5 was split into two steps, as participants were divided into two groups; they first sorted their own index cards, and then sorted the index cards of the other group.

C.1. Using an icon representing a house as a guide, participants were asked to write on index cards "anything associated with household energy use or household energy activities."

C.2. A demo of the prototype model was given.

C.3. Participants were divided into two groups, and told to take their index cards to their group's location, and arrange the cards belonging to all members of their group into categories that made sense to them. In so doing, they were given the opportunity to write new cards.

C.4. The cards' relative positions in each group were recorded with a camera, and the groups told to move to the location of the other group, and rearrange their cards.

3.3 Results

In workshop A, participants struggled with step A.2, finding the question too open-ended. The discussion focused on policies and types of intervention associated with household energy use. Step A.3 proceeded relatively smoothly, though some clarification was needed on the question. Participants were initially asked what they would put in a simulation aimed at addressing the policy areas and intervention types in step A.2, but needed to be given a specific example to get the idea. They chose to limit themselves to 3-5 post-its each, and produced about 30 of them. One participant suggested model purpose could be discussed as part of this exercise. In step A.4, restructuring the post-its yielded eight groups. The process, however, was led by the facilitator—although the participants engaged in the discussion, the groups were not entirely their own. In step A.5, the modelling categories proved too unintuitive for the participants to work with, and we agreed that step A.5 would be performed by a model developer outside the meeting.

Reaction to the prototype model presentation in step A.6 was also interesting. ABMED is a relatively simple model, but still required a lot of detail to explain. ODD stipulates a complete description of everything in the model; consequently there was a lot of information to take on board during the presentation of the Entities, and this proved difficult for some to grapple with. However, the presentation did enable the participants to comment on areas where the model's behaviour was counter-intuitive, or on missing phenomena.

Step A.7 was in the end not conducted.

The results of step A.3 were translated by a developer into ontological categories (see figure 3.2). Some post-its required relatively little interpretation, such as 'Consumers', 'Goods' (clearly entities), and 'available options' (clearly a relationship between Consumers and Consumer Goods—as a modeller, one would expect this relationship to be computed from attributes such as price, and disposable income). An ambiguous example was 'Insulation', which could be a process (i.e. the action of installing insulation), an entity (the physical presence of insulation), or an attribute (e.g. of a house, such as its U-value). Other post-its required more interpretation to create usable ontological categories, such as 'How does the policy help or hinder other policies', which besides the existence of Policies as entities and relationships 'help' and 'hinder' between them, also suggests goals or success criteria as attributes of Policy and processes by which they are compared for conflicts or measured.

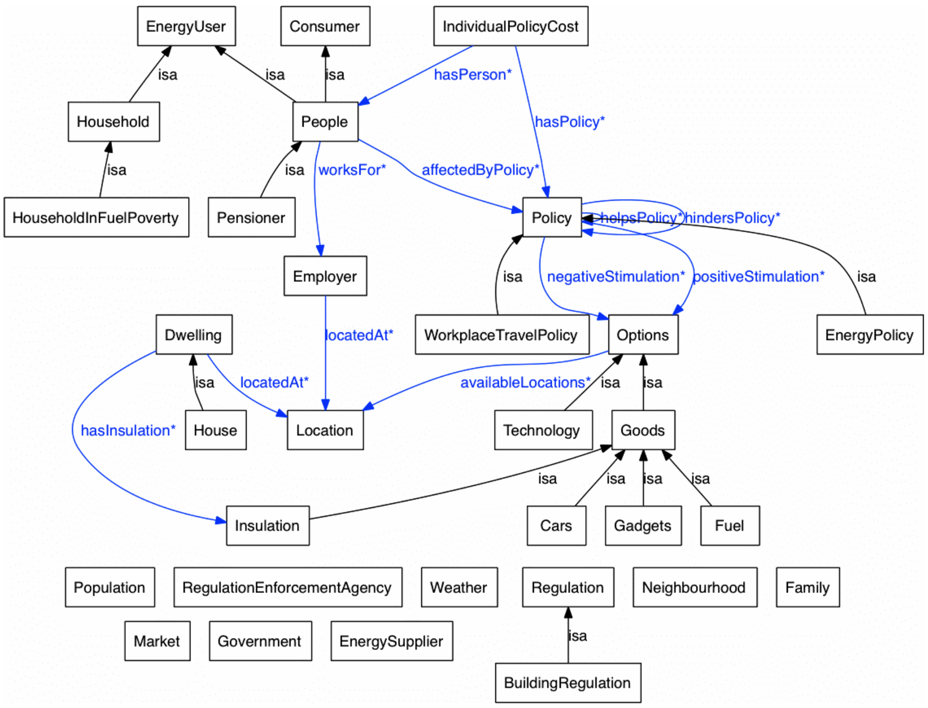


Figure 3.2. Ontology derived from post-its in Workshop A.

Workshop B, with the SAG, generated more post-its in step B.1 than A.3 in the previous workshop), though here participants were encouraged to think of as many things as they could, rather than limiting themselves to 3-5. The post-its were transcribed to index cards whilst they were given the presentation on ABMED. Once again, ABMED was presented using ODD to structure the information—the presentation having been modified to present the entities in the model in a way that did not immediately overwhelm them with all the details.

The participants spent about 15 minutes categorising the index cards on a table. Though not specifically asked to do so, they did this as a group. The spatial relationship of some of the categories was important to the participants, and it was noted that some index cards belonged in more than one category. Much as for workshop A, an ontology was create d from the index cards (figure 3.3); some cards again requiring interpretation to represent their contents. The categories the participants created sometimes suggested superclasses (e.g. 'Appliances' and 'Energy efficiency measures'), but not always. The category 'Household context', for example, is a somewhat abstract concept. The index cards they put in it, such as 'Owner occupier', 'Private tenant', 'Access to finance, time and knowledge' were not subclasses in the sense that any instance of 'Owner occupier' is not an instance of 'Household context'. The context of a household would instead be a property of it, inferred from properties of the occupiers.

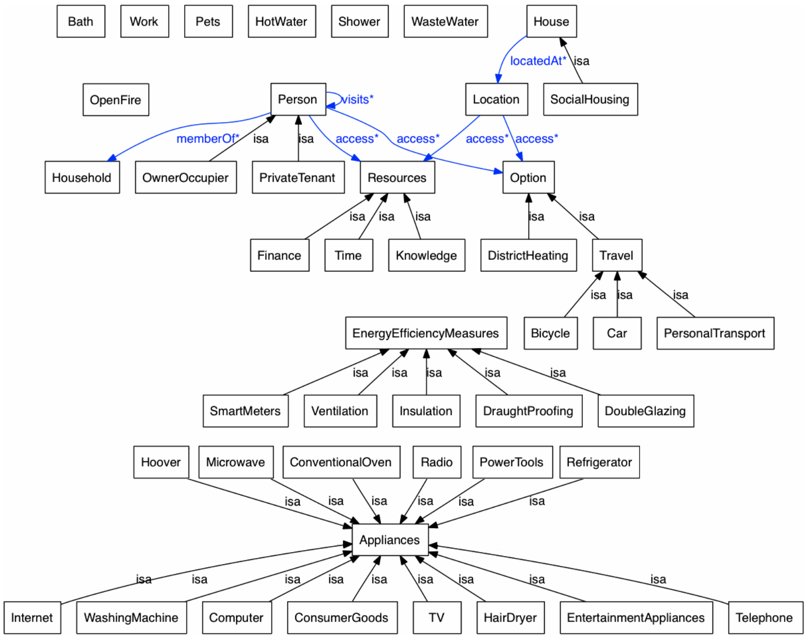


Figure 3.3. Ontology derived from Workshop B.

Whilst sorting the cards into their own categories proved relatively straightforward, when shown the modelling categories and asked to put the index cards into these, the participants were somewhat flummoxed initially. Clearly the categories were not intuitive, and some found their definitions circular. They discussed trying to put the categories they had created earlier into the ontological categories we had given them—perhaps hoping this would make the exercise easier, as then all cards in a category they had created would (in their view) belong in the same ontological category. When asked how they were finding this process, there was general agreement that it was difficult, and that the ontological categories were quite unlike those they had created. They also noted that which ontological category they might put a card into depended on its interpretation and how it was written on the card. One participant remarked that this forced them to think about things in more detail.

Observing that essentially all their cards were entities,[2] they decided to pick one example ('Car') and work it through. As they thought about this example, they wrote extra cards for the attributes, relationships and processes. Although what they wrote for the attributes ('age, mileage, fuel economy') drew on the examples they were given (figure 3.1), one participant described attributes as the reasons why you would buy the car, suggesting that to some extent (albeit specifically focused on the example they were working on) they understood the idea. Participants struggled more with relationships writing 'no. cars/household' and 'car sharing'. The former as worded could be an attribute of a household, or interpreted as the 'owns' relation. The latter could be translated into a relation by wording it as 'shares-car-with' (a relationship between two people), but this wouldn't capture which car was being shared. A modeller would probably create a 'Car Share' concept, with relations to a Car and the People sharing it. The participants interpreted processes as what the Car is used for, such as commuting ("for work"), accessing

[2] However, the ontology created from the index cards by the model developer has data and object properties interpreted from the cards. For example, the card 'Gas/off gas' was interpreted as a data property of a location.

public transport ("to nearest railway station"), leisure activities ("weekend only" and "leisure"), and "short journeys (e.g. school run)". Whilst not all these might be included in a model, their rewording suggests that processes were reasonably well understood by the participants. This was as far as they got in 15 minutes, though the workshop schedule allowed more time for this exercise. It is possible that, having found their own categories, they were reluctant to break them up, and/or were tired of categorising things by this point.

Workshop C was more successful than workshop B in eliciting ontological elements, though again, index cards mainly referred to entities rather than attributes relationships or processes. However, this may be more a feature of the instruction to 'categorise' the cards in steps C.3 and C.4. Step C.4 was an interesting addition to the workshop plan, enabled by the larger number of participants. Though given the option to accept the categorisation of their colleagues, each group nevertheless made their own categorisation. While this may suggest that categorisation is subjective, there were broad similarities in the way cards were categorised by each of the two groups. Many cards were given exactly the same category label (e.g. 'Transport' or 'Appliances'), whilst other categories used related vocabulary (e.g. 'Kitchen' and 'Cooking'). Sometimes cards were associated with more specific categories, for example the 'Blender' card was assigned the general category 'Appliances' by group 2, whilst group 1 assigned it the category 'Cooking' to associate it more specifically with the task for which the appliance is used. It would not be correct to say that a blender is not an appliance, nor to say that it is not used for cooking – the fact of a difference in categorisation thus suggests a difference of emphasis rather than a fundamental difference of type. The ontology derived from Workshop C is presented in Figure 3.4.

3.4 Discussion

Various researchers have noted the preference of stakeholders for simpler models. By contrast, agent-based modellers acknowledge the significance of individual heterogeneity and the interactions of those individuals. A consequence of the commitment to modelling at the micro scale is that models contain a lot more detail. That this detail is hard to communicate is a prime motivation for the creation of ODD.

It is clear from workshops A and B that the volume of representation in ABMs can be difficult for non-modellers to cope with. ODD's structuring of the knowledge in the presentation of the prototype model did enable participants to understand and ask pertinent questions about the model, even if some of them found all the details somewhat overwhelming. For co-constructing a model, however, a linear approach adhering to the ODD ordering did not prove helpful. Nevertheless, an exercise that effectively amounts to a quick ontology elicitation process was able to produce material from which a formal ontology could be constructed, albeit that interpretation of the material by the model developer was required to do so.

In workshop B, we tested the ontological categories, which had also proved difficult for participants to understand in workshop A. Interestingly, the participants' own categorisation expressed something of a concept hierarchy, though the card sorting exercise did not enable them to fully express themselves—in particular, where a card belonged in more than one of their identified categories. They found the definitions of the categories circular, but were

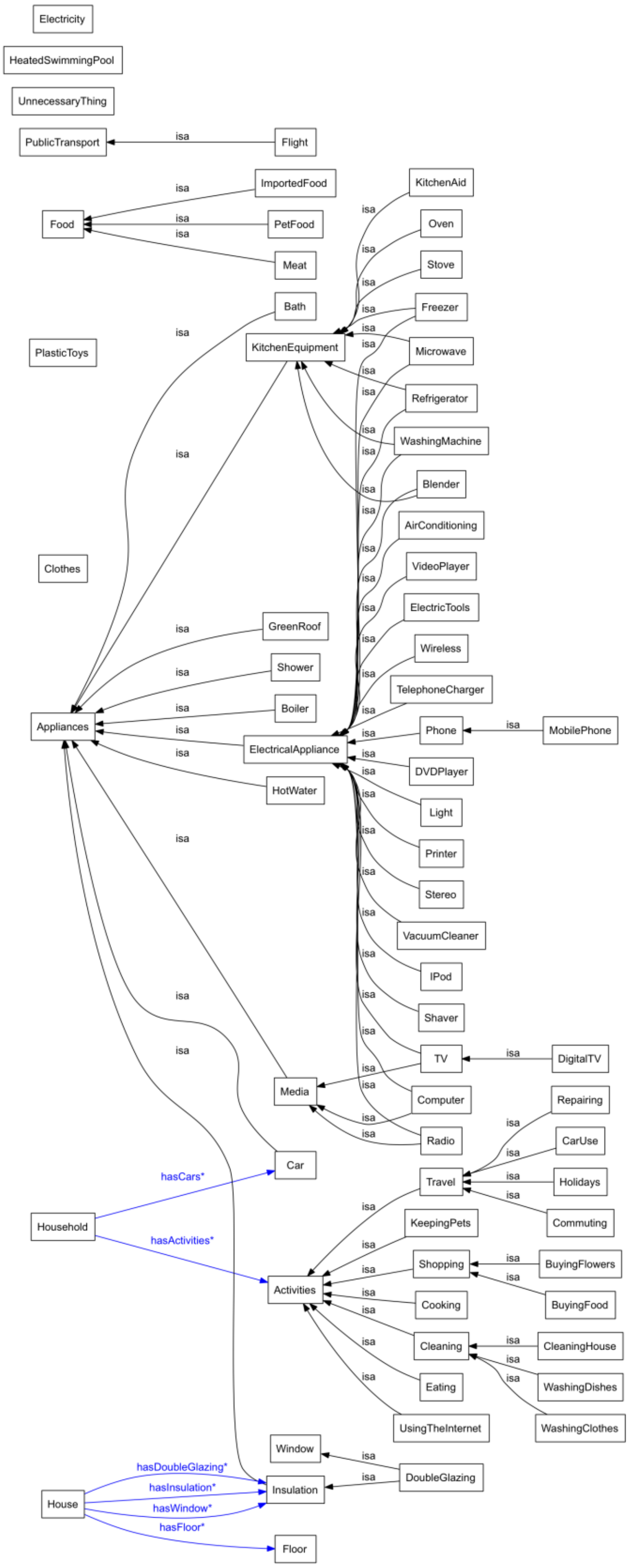


Figure 3.4. Ontology derived from Workshop C.

beginning to grasp them working through the ‘Car’ example. It is possibly significant that, realising the detail they had to produce, and seeing the large number of other cards that they identified as being mostly entities, they decided not to work through further examples, although there was time for them to do so. Agent-based modelling requires more in the way of details than other modelling approaches, and only the most dedicated and co-operative stakeholders will be willing to think through all those details.

That respondents did not respond intuitively to the ontological categories may be because they had first used their own categories, the definitions used (though for attributes and processes, the group successfully developed their own interpretation), and the time available).

However, since the modelling process will inevitably involve some translation and interpretation of the output from participants, it may be better not to involve them directly in creating utterances in formal languages—at least when the process does not allow time for educating participants in the implications. If participants are willing, part of the iterative process of involving them in model development can include checking the inferences that will be drawn from the formalisation of their post-it notes and categorisation.

By the time workshop C was undertaken, ODD had no bearing on the workshop design (though arguably the purpose of the workshop was to elicit concepts to include in the Entities, State Variables and Scales and the Process Overview and Scheduling sections of ODD), and no reference was made to ontological categories used by modellers.

3.5 Conclusion

The main objective of the exercise was successful, in that both workshops created a context in which the models can have meaning for the participants, as opposed to a more traditional situation in which the model developers do everything.

Though it is clear that modellers conceptualise the world in a way that is not intuitive to everyone, workshop B shows there is potential for non-modelling experts to understand it. However, such understanding may require an investment of time that cannot always be afforded, and it may in any case be better for modellers to do the formalisation; checking the assumptions with stakeholders afterwards.

An ontology covering all three workshops was created, and the concepts and relationships are shown in figure 3.5. Appendix 2 shows the documentation of the translation of index cards from all three workshops into ontological categories.

The ontology provides considerable support for the structure of CEDSS, and in particular for the focus on appliances and space/water heating.

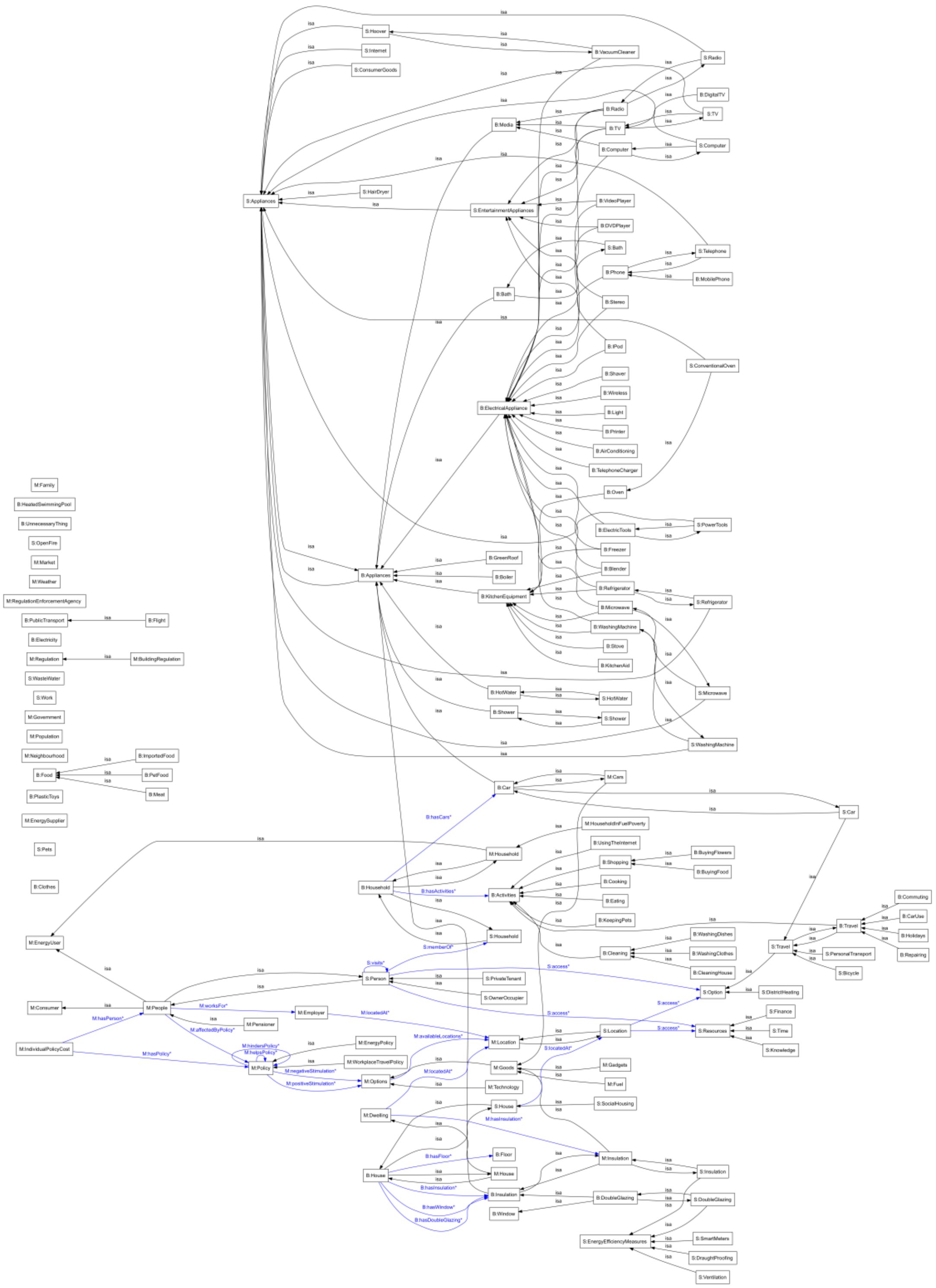


Figure 3.5. Ontology combining results from all workshops. Labels with a prefix 'M' (for Macaulay) are from Workshop A, those with a prefix 'S' (for Scottish SAG) from workshop B, and the prefix 'B' (for Budapest) denotes workshop C.

4 Final Version of CEDSS

We use the 'Overview, Design concepts and Details' (ODD) protocol of Grimm et al. (2006, 2010) to describe CEDSS.

4.1 Overview

4.1.1 Purpose

The purpose of CEDSS (Community Energy Demand Social Simulator) is to simulate the household energy demand of a small rural or urban community (e.g. a housing estate or village), with respect to energy used for space and water heating, and for appliances, over the period 2000-2050.

4.1.2 Entities, State Variables and Scales

The entities and relationships among them are summarised in figure 4.1. Reified relationships (those having data stored about them) are shown in blue boxes.

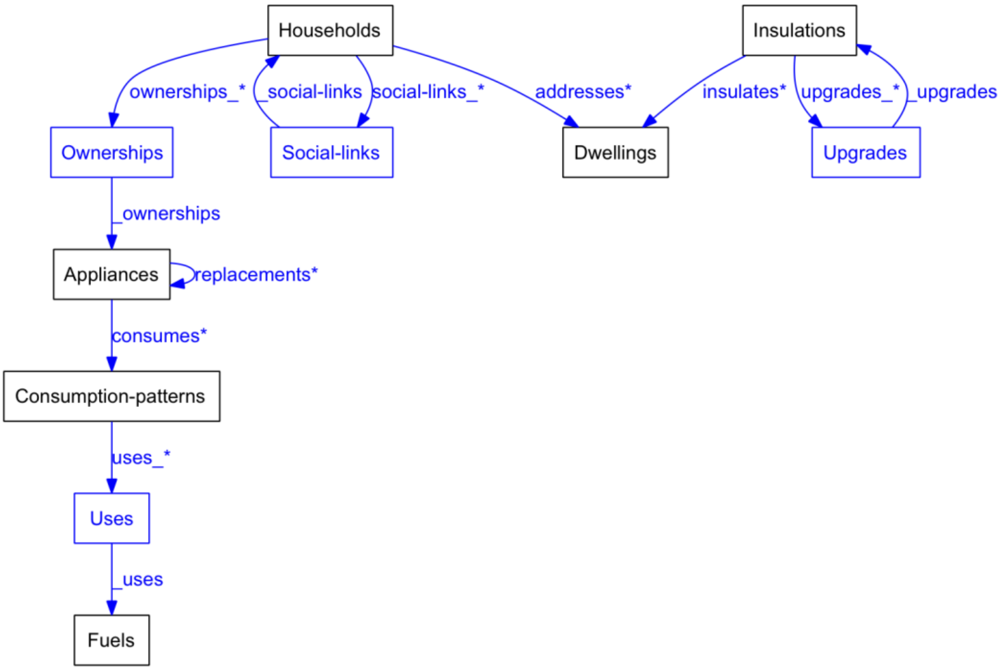


Figure 4.1. Entities and relationships in the CEDSS model.

The state variables of the entities and reified relationships are summarised in the following tables.

Table 4.1. Households

State variable	Type	Description
household-id	string	An identifier for the household
household-type	string	A descriptor for the type of the household (e.g. demographic type/class)
planning-horizon	integer	How far the household looks ahead when computing projected running and energy costs of appliances.
steply-net-income	double	How much income the household receives per step (one step could represent a month, or a quarter, depending on the model configuration).
capital-reserve	double	How much money the household has in, e.g. savings.
goal-frame	string	The goal frame the household is currently using to make decisions.
frame-adjustment	double	Adjustment parameter for values when the household is influenced by their peers.
gain-orientation	double	A representation of the household's egoistic value.
greenness	double	A representation of the household's biospheric value.
hedonism	double	A representation of the household's hedonic value.
steps-total-energy-use	double	How much energy the household has used in this step.
usage-mode	string	A descriptor for the way in which households are using appliances (e.g. economising).
breakdown-list	Appliances	A record of the broken appliances that the household has not yet replaced.
wish-list	Appliances	A list of appliances that the household desires.

Table 4.2. Dwellings

State variable	Type	Description
dwelling-id	string	An identifier for the dwelling.
dwelling-type	string	A descriptor for the type of dwelling (e.g. four-bedroom detached house, one-bedroom flat)
tenure	string	The type of tenure – e.g. owner-occupied, or rented.

Table 4.3. Insulations

State variable	Type	Description
insulation-state	string	A descriptor for the state of the insulation (e.g. 270mm glass-fibre loft insulation).
fuel-use-factor	double	How the space/water heating fuel use is adjusted by installing this insulation.
insulation-dwelling-type	string	The dwelling type the insulation can be applied to.

Table 4.4. Appliances

State variable	Type	Description
category	string	A high level category for the appliance (e.g. TV).
subcategory	string	A subcategory for the appliance (e.g. LCD TV, CRT TV, plasma TV).
name	string	The identifier for the appliance (make and model).
cost-list	double	List of purchase costs for the appliance in each step for which it is available.
breakdown-probability	double	Probability the appliance will break down in any step.
embodied-energy	double	Energy associated with the manufacture and delivery of the appliance.
energy-rating	string	Energy rating for the appliance.
energy-rating-provided?	Boolean	Whether the energy rating has been provided.
essential?	Boolean	Is the appliance essential for all households?
hedonic-score	double	A number reflecting the degree to which the appliance activates the hedonic value when it is bought.
last-step-available	integer	Last step in which the appliance is available, if this is bounded.
last-step-available-unbounded?	Boolean	True if the appliance does not have a last step in which it is available.

Table 4.5. Consumption-patterns

State variable	Type	Description
for-dwelling-type	string	Dwelling type with which the consumption pattern is associated
for-household-type	string	Household type with which the consumption pattern is associated
for-purpose	string	Purpose with which the consumption pattern is associated.
for-tenure-type	string	Tenure with which the consumption pattern is associated.
in-step	integer	Time of year with which the consumption pattern is associated.
in-usage-mode	string	Usage mode with which the consumption pattern is associated.

Table 4.6. Fuels

State variable	Type	Description
fuel-type	string	Type of fuel.
kwh-per-unit	double	kWh used per unit of fuel.
total-kwh	double	Total kWh of this fuel used by all households (for observation purposes).
unit	string	Label for the unit of the fuel.

Table 4.7. Upgrades

State variable	Type	Description
upgrade-cost	double	Cost of making the insulation upgrade.

Table 4.8. Social-links

State variable	Type	Description
n-visits	integer	Number of visits that have been made by the 'from' household to the 'to' household.

Table 4.9. Ownerships

State variable	Type	Description
age	integer	How many steps the household has owned the appliance.
broken?	Boolean	Whether the appliance is broken.

Table 4.10. Uses

State variable	Type	Description
units-per-use	double	How many units of the fuel the consumption pattern uses.

4.1.3 Process Overview and Scheduling

The model does the following in each time step:

1. Decide which owned appliances break down.
2. Transition household state – this simulates changes in demographics (e.g. having children or retiring) and in/out migration from the community.
3. Each household then does the following:
    a. Choose goal frame.
    b. Adjust goal frame.
    c. Choose usage mode.
    d. Compute the total energy use for this step from using appliances and space/water heating.
    e. Compute financial situation.
    f. Replace broken appliances.
    g. Update wish list (if goal frame is 'hedonic' only).
    h. Buy insulation (if goal frame is not 'hedonic').
    i. Buy new appliances.
    j. Visit social neighbours.
    k. Update social links.

4.2 Design Concepts

Basic Principles

The model is based on the psychological theory of 'goal frames' (Lindenberg and Steg 2007), in which households make decisions in one of three modes: 'hedonistic', 'egoistic' and 'biospheric'; which mode they choose depends on their 'values': stored parame ters representing 'hedonism', 'gain-orientation' and 'greenness'.

Emergence

Emergent properties of the model are the community-level consumption of energy from various sources, and the numbers of different appliances owned.

Objectives

Households must ensure they keep essential equipment running, otherwise, households' objectives depend on their dominant goal frame. Hedonists aim to buy as many of their desired appliances as they can afford, egoists aim to save as much money as they can, whilst biospherics aim to minimise their energy consumption.

Prediction

Households may compute the expected running costs and space heating costs when buying appliances and considering insulation options, depending on the mode in which they are making decisions.

Sensing

Households are aware of appliances owned by their friends when they visit them.

Interaction

Households visit each other, according to their social-links. They adapt their social-links according to how similar their profile of appliances is with that of the people they visit. The profile of appliances is used as a proxy for lifestyle characteristics in the model, and the assumption is made that people are more likely to be friends with those having similar lifestyles.

Stochasticity

Stochasticity is used in setting up and updating social-links, and may also be used in setting households' goal frame parameters. It is used in determining exactly when an appliance will break down, and in determining timings of household transitions, if a run of CEDSS makes use of these.

Observation

The model collects data on energy use (broken down by fuel), how much money households have, how many social links they have, how many of each category of appliance are owned by households in the community, how many of each category of appli ances have been thrown away by the community, goal frames used to make decisions, goal frame parameters, and numbers of visits made per social link.

4.3 Details

4.3.1 Initialisation

Initialisation of CEDSS is achieved by loading in a series of files. These are described in detail elsewhere in the report, but are covered briefly here in the order in which they are loaded during initialisation:

1. *Maximum in category file.* This specifies, for each category of appliance, the maximum number of appliances that a household can own.
2. *Insulation file*. This contains data on the insulation state, fuel use factor and insulation dwelling type for each insulation.
3. *Insulation upgrade file*. This contains cost data used to create the upgrades links.
4. *Patch legend file*. This specifies the colour to use for each type of patch.
5. *Patch layout file*. This specifies the type of each patch in the space.
6. *Dwellings file*. The dwellings file specifies the type of each dwelling, its tenure and its initial insulation state. After the dwellings file is loaded in, 'blocks' of dwellings are identified.
7. *Fuel file*. This creates the fuels, and initialises their type, unit and kWh per unit.
8. *Usage mode matrix file*. This specifies the conditions under which each goal frame has a particular usage mode.
9. *Appliances file*. Initialises the name, category, subcategory, whether the appliance is essential, hedonic score, cost list, energy rating, embodied energy, breakdown probability, first and last step available of each appliance.
10. *Appliances replacements file*. This is used to initialise the replacements link breeds.
11. *Appliances fuel use file*. Initialises the consumption patterns.
12. *Initial appliances file*. This is used to create initial appliances owned by particular households, types of households, or households living in particular dwelling types.
13. *Households file*. Initialises the identifier, type, income, capital reserve, goal frame parameters, goal frame adjustment parameters, planning horizon and dwelling of the household.
14. *Household transition matrix file*. If used, this specifies the probabilities of households changing type each time step. To 'not use it', the matrix file can simply consist of an identity matrix indicating a probability of 1 for transitioning from each household type to itself, and a 0 for all other transitions.
15. *Social link matrix file.* If used, this file specifies the probability of making a social link between households and dwellings of different types.
16. *Social link file.* If used, this file allows a social network to be set up among specific households.

4.3.2 Input

The model has a number of time-series data that are used to implement scenario storylines during the course of a model run. These data are loaded from files into global data structures during initialisation.

1. *Insulation update file*. This contains data used to update the set of insulation upgrades that are available during the course of the run.

2. *Suppliers file.* This contains data about the suppliers for each type of fuel, and its price per unit at each time step.
3. *Appliances file.* Much of the data in the appliance file, mentioned earlier under initialisation, is input rather than initialisation data. In particular, the cost list specifies the price of the appliance in each time step for which it is available, and the first and last time step available are also time series data.
4. *Households file.* The households file also contains time series data pertaining to income.
5. *In-migrant household file*. If used, this can be used to implement new household parameters for given household transitions, and simulate changing attitudes and values as part of the scenario.

4.3.3 Submodels

1. Decide which owned appliances break down.

All ownership links between households and appliances are checked against the appliances' breakdown probabilities. For each link, if the breakdown probability of the appliance is more than a random number in the range 0 to 1, then the broken? flag on the ownership link is set to true.

2. Transition household state.

Each household looks up the transition probabilities for its type in the household transition matrix. A random number between 0 and 1 is chosen, and the transition probabilities summed to form an ascending series. The position of the random number relative to the series determines the new state of the household. If the random number is more than th e last number in the series, then no transition takes place.

3a. Choose goal frame.

The goal frame is chosen by choosing a random number *R* in the range 0 to the sum of the goal frame parameters (hedonism, gain-orientation and greenness). The selection of goal frame is then made thus:

R < hedonism	hedonistic
R < hedonism + gain-orientation	egoistic
Otherwise	biospheric

3b. Adjust goal frame.

Adjustment of the goal frame is done to implement a 'habit' component – the more a goal frame is used, the more likely it is to be used in future. The habit-adjustment-factor parameter (*H*) is used to make this adjustment. If *T* is the sum of all goal frame parameters, then let *A* be the parameter corresponding to the selected goal frame, and *B* and *C* be the other two. *A* is increased by *H*, and *B* and *C* decreased by *H* / 2. If the result causes *A* to be more than *T*, then adjustments are made to ensure that $A + B + C = T$, and that *B* and $C \geq 0$. The algorithm is provided below:

```
$A = A + H$

If $A > T$ Then $A = T$; $B = C = 0$
Else

    $B = B – H/2$
    $C = C – H/2$

    If $B < 0$ Then $C = T – A$; $B = 0$
    Else If $C < 0$ Then $B = T – A$; $C = 0$
```

3c. Choose usage mode.

The usage mode of the household is chosen using the usage mode matrix file as a guide. This specifies conditions under which a usage mode can be selected. Currently the only condition implemented is if the household has a negative capital-reserve. This allows usage modes to be defined for when the capital-reserve is negative, and when it is not negative.

3d. Compute the total energy use for this step from using appliances and space/water heating.

For each appliance owned by the household, the step's fuel consumption and energy use is computed based on the consumption pattern associated with the household's type, the type of their dwelling, the usage mode, and the time of year.

3e. Compute financial situation.

The energy cost of running the appliances is deducted from the capital-reserve of the household, and the income for that step is added.

3f. Replace broken appliances.

If the appliance is essential and the household only had one of them before it broke, then a new item will be bought using a method relevant to the current goal frame of the household and the tenure of their dwelling thus:

Table 4.11. Essential appliance replacement methods for different contexts.

Context	**Essential appliance replacement method**
tenure is rented or goal frame hedonistic	Choose the cheapest replacement
goal frame is egoistic	Choose a random replacement
goal frame is biospheric	Choose a random replacement among those with the best energy rating (if this is supplied) or the lowest breakdown probability if the energy rating is not supplied

The justification for biospheric mode decision makers knowing the breakdown probability is that such households are assumed to do research when buying an appliance.

Otherwise, if the appliance is not essential, it is added to the breakdown-list of the household.

3g. Update wish list (if goal frame is 'hedonic' only).

The wish list of the household is updated to contain items chosen in all of the following three ways:

- Up to *N* appliances (not to do with heating) each belonging to a different new subcategory introduced in the last *N* steps.
- One random item not already owned from *V* visits.
- One random replacement for an item more than *T* steps old.

Here, *N* corresponds to the global new-subcategory-appliances-per-step parameter, *V* to the global visits-per-step parameter, and *T* to the old-product-steps parameter. These three parameters are set on the CEDSS interface.

3h. Buy insulation (if goal frame is not 'hedonic').

Insulation is bought using a method specific to the selected goal frame. If the goal frame is egoistic, then the household chooses an insulation state reachable from the current state that will save the most money and make a positive monetary saving over the planning horizon of the household. If the goal frame is biospheric, then the household chooses an insulation state reachable from the current state that will leave a positive capital reserve and save the most energy.

3i. Buy new appliances.

New appliances are bought in a different way depending on the goal frame:

Table 4.12. New appliance purchase methods for different household goal frames.

Goal frame	**New appliance purchase method**
hedonistic	Buy as many affordable appliances as possible from the union of the breakdown list and the wish list (in descending order of hedonic score), but not more than one from the same category. An affordable appliance is one the cost of which is less than the capital reserve plus the household's income this step multiplied by the credit-multiple-limit parameter.
egoistic	Choose the cheapest replacement for an item on the breakdown list.
biospheric	Choose the item with the best energy rating (if supplied) or lowest breakdown probability (if not) that is a replacement for an item on the breakdown list.

3j. Visit social neighbours.

Up to *V* (= visits-per-step) randomly chosen social links may be visited each step (it will be less than *V* only in the event that the household has no social links). For each visit, the household adjusts their goal frame parameters; if the reciprocal-adjustment parameter is set to true, then the visited household also adjusts their parameters. Each goal frame parameter

G is adjusted in the following way: let G_i be the goal frame parameter of the household making the adjustment; let G_j be that of the other household; then:

$$G_i = G_i + F(G_j - G_i)$$

where F is the frame-adjustment parameter, set on the CEDSS interface. After this, if G_i is less than zero, $G_i = 0$.

3k. Update social links.

With probability 0.5 either way, the household either loses or gains a social link. (If the choice is made to lose a link, then this only happens if there is a link to lose; if the choice is made to gain a link, and the household already has max-links social links, where max-links is a parameter set on the CEDSS interface, then no link will be gained.)

To lose a social-link, the household first determines the set of weak contacts – those with minimum appliance similarity. If this set has more than one member, then the set is reduced to those who have the maximum block distance from the household. If the set still has more than one member, then the link dropped will be randomly chosen from those the household has visited least.

To gain a social link, the household determines the set of strong contacts – those with maximum appliance similarity. One of these is randomly chosen, and an attempt made to find one of their social links that the household is not already connected to. If such a household exists, then it will be selected as a new social link; if more than one such household exists, then one of them will be chosen at random.

Appliance similarity is computed as the count of the number of appliances the two households have in common, minus the number of appliances that one has that the other does not.

Block distance is computed as the absolute difference in the X co-ordinates of the two dwellings, plus the absolute difference in the Y co-ordinates of the two dwellings.

5 Ontology Export from Netlogo

Semantic differences between classical object-oriented programming languages and description logics have been highlighted by Polhill and Gotts (2009), with reference to Lalonde and Pugh's (1991) distinctions among the three inheritance relations subtype, subclass and is-a. Of these three, the first is for compiler convenience in checking types involved in operations, the second for programmer convenience in facilitating code re-use, whilst the third reflects descriptive, logical meanings that are of primary interest from an ontological point of view. These differences mean that direct ontology learning from model source code is not possible in object-oriented programming languages.

Netlogo models do not have these issues because Netlogo is not an object-oriented programming language, and hence there is no inheritance among the language's main entity ('agent' in Netlogo terminology) types, which are known as 'breeds'. The facility for explicit representation of relationships (or 'links') among agents, rather than encapsulating associations in classes as in object-oriented languages, further contributes to enabling automatic ontology learning from Netlogo model code, though it imposes a programming style in which important relationships among agents are represented as links. Mapping from breeds to ontological classes also encourages model developers to make greater use of breeds for model entities than otherwise they might. An 'agent' might be normally thought of as something that undertakes an action, a conceptualisation that tends to oppose the representation of inanimate entities as agents as is required to automatically extract an ontology from Netlogo source code.

Netlogo's extension facility enables software to be written that provides functionality from within Netlogo to automatically extract an OWL ontology from the model. The mapping is shown in table 5.1.

Table 5.1. Mapping from Netlogo syntax to OWL ontology types.

Netlogo syntax	OWL type(s)	Comments
breed	Class	Breeds in Netlogo are not necessarily disjoint – agents can change breed using the set breed command.
directed-link-breed (no 'own')	ObjectProperty	For all link breeds, there is no syntax to constrain the breeds of agent that are at either end of the link, meaning that domains and ranges for links cannot be inferred from the code. If the link breed does not have any of its 'own' variables, it can be declared directly as an object property…
directed-link-breed (with 'own')	Class and ObjectProperty	…otherwise the link must be reified, and two object properties used to link the breeds that can be connected in this way. The object property going in to the class representing the property can be declared inverse functional; the object property going out can be declared functional. A property chain can be used to represent the link as a whole if the ontology is in

Netlogo syntax	OWL type(s)	Comments
		OWL 2.
undirected-link-breed	ObjectProperty or Class and ObjectProperty	In addition to the points applying to directed link breeds, object properties corresponding to undirected link breeds (in the case of reified links, this will apply to the chain) can be declared symmetric in OWL. If the ontology is in OWL 2, all links can be declared as irreflexive as Netlogo does not permit an agent to link to itself.
breeds-own	DataProperty	The domain of the data property can be declared to be the class corresponding to the breed. Note that different breeds can have the same variable.
link-breeds-own	DataProperty	The domain of the data property can be declared to be the class corresponding to the link breed. Note that different link breeds can have the same variable, but a variable cannot be shared between link breeds and breeds.
agent	Individual	Agents in Netlogo can be asserted as individuals, and members of the class corresponding to the breed of which they are currently a member.

The 'owl' extension to Netlogo was written for the GILDED project to implement the above. The extension is written for Netlogo 5.0, and consists of the file owl.jar. This, together with version 3.1.0 of the OWL-API (http://owlapi.sourceforge.net/), which is available under the GNU Library General Public Licence, should be placed in a subdirectory named 'owl' of the directory in which the .nlogo file using the owl extension is located. The file 'owl.zip' – the release version of the extension, contains owl.jar and the OWL-API, ready for use. To permanently install the extension, put it in the extensions folder of the Netlogo applications directory.

5.1 Usage

The extension makes available a number of commands to a Netlogo model containing the following command (typically, near the beginning of the code):

```
extensions [owl]
```

The commands are given in approximate order in which they must occur during execution of the model. Specifically, owl:domain, owl:range and owl:imports cannot be used after owl:model, and owl:structure and owl:state may only be used after owl:model.

The commands assume a distinction between the *structure* and the *state* of the model and corresponding ontologies. An ontology describing the structure of a model consists of terminology (T-box) axioms describing the kinds of entity that the model contains (OWL classes), properties they have (OWL data properties) and relationships they may have with other types of entity (OWL object properties). By contrast, an ontology describing the state of a model applies to a particular snapshot of an instance of it running at one time. The state ontology imports the structure ontology, and then adds assertion (A-box) axioms about the specific instances of the various classes of entity that exist in the model when the snapshot is taken.

5.1.1 owl:domain *link-breed breed*

The owl:domain command takes two arguments. The first is the (string) name of a link-breed, and the second is the (string) name of a breed. The command causes an assertion to be made in the ontology that the OWL class corresponding to the breed is in the domain of the OWL object property corresponding to the link-breed.

This command will not affect your model (that is to say, while the model runs, no check will be made for the link-breed that only agents of the specified breed are in the domain of the link-breed.

5.1.2 owl:range *link-breed breed*

The owl:range command takes two arguments. The first is the (string) name of a link-breed, and the second is the (string) name of a breed. The command causes an assertion to be made in the ontology that the OWL class corresponding to the breed is in the range of the OWL object property corresponding to the link-breed.

Just as for owl:domain, owl:range does not cause any checks to be made while the model runs that only agents of the specified breed are in the range of the link-breed.

5.1.3 owl:imports *IRI…*

The argument(s) to the owl:imports command are strings containing Internationalised Resource Identifiers for ontologies that the model structure ontology is to import. Again, this does not affect the model in any way, or cause any checks to be made, but allows the ontology to be created with the imports in place ready for use with ontology visualisation, reasoning and analysis tools.

5.1.4 owl:model *IRI*

Define the Internationalised Resource Identifier for the model structure ontology, which is contained in the string argument. All ontologies have an IRI, which constitutes an identifier for the ontology.

As stated above, all owl:domain, owl:range and owl:imports commands must execute before owl:model is executed, and no owl:structure or owl:state command may execute until owl:model has executed.

5.1.5 owl:structure *file-name*

Save a model structure ontology to the file name given as arguments.

5.1.6 owl:state *file-name time-step* [*extension*] (experimental)

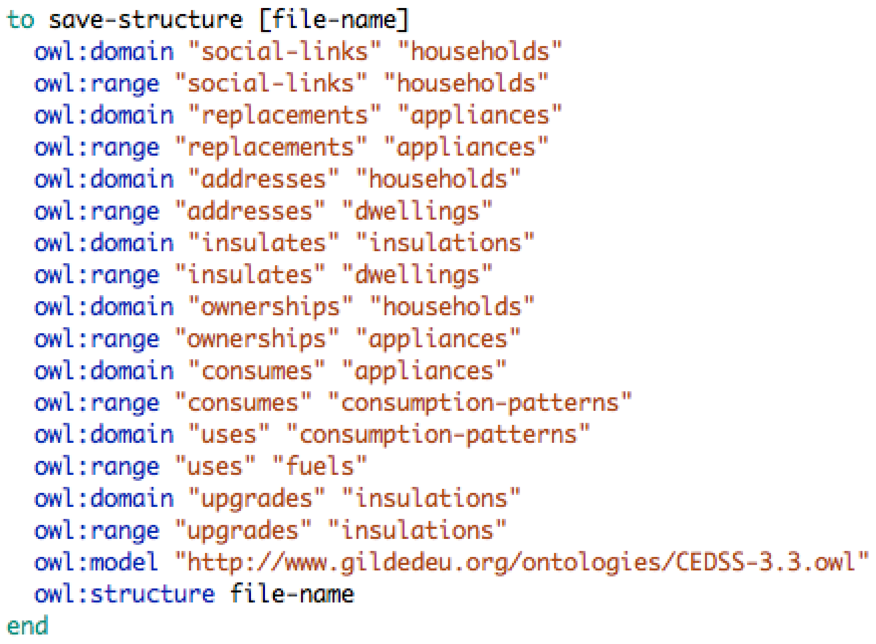
Save an ontology of the current state of the model to the file name. The logical IRI for the state ontology will be constructed from the model IRI specified by the earlier owl:model command, the time-step given as second argument to this command, and, if given the extension string. Specifically, any .owl suffix will be removed from the model IRI, and then a dash (-) appended, followed by the time-step, followed by the extension string if given, followed by the .owl suffix if the model IRI had it originally.

For example, the model IRI “http://www.gildedeu.org/ontologies/CEDSS-3.3.owl” given the arguments 125 for the time-step and “run3” for the extension, would produce a state ontology IRI “http://www.gildedeu.org/ontologies/CEDSS-3.3-125run3.owl”

This is an experimental command, as NetLogo documentation about accessing the state of the model from within an extension was not available at the time the extension was written.

5.2 Example

The following procedure demonstrates the use of some of the above ontology features to save a structure ontology from CEDSS:



```
to save-structure [file-name]
  owl:domain "social-links" "households"
  owl:range "social-links" "households"
  owl:domain "replacements" "appliances"
  owl:range "replacements" "appliances"
  owl:domain "addresses" "households"
  owl:range "addresses" "dwellings"
  owl:domain "insulates" "insulations"
  owl:range "insulates" "dwellings"
  owl:domain "ownerships" "households"
  owl:range "ownerships" "appliances"
  owl:domain "consumes" "appliances"
  owl:range "consumes" "consumption-patterns"
  owl:domain "uses" "consumption-patterns"
  owl:range "uses" "fuels"
  owl:domain "upgrades" "insulations"
  owl:range "upgrades" "insulations"
  owl:model "http://www.gildedeu.org/ontologies/CEDSS-3.3.owl"
  owl:structure file-name
end
```

Figure 5.1. Example procedure to extract a structure ontology from CEDSS.

Using the OntoViz plug-in for Protégé 3, a DOT graph can be created of the resulting ontology, which (with some minor editing to simplify the text and highlight reified link-breeds in blue) is shown in figure 5.2 on the following page.

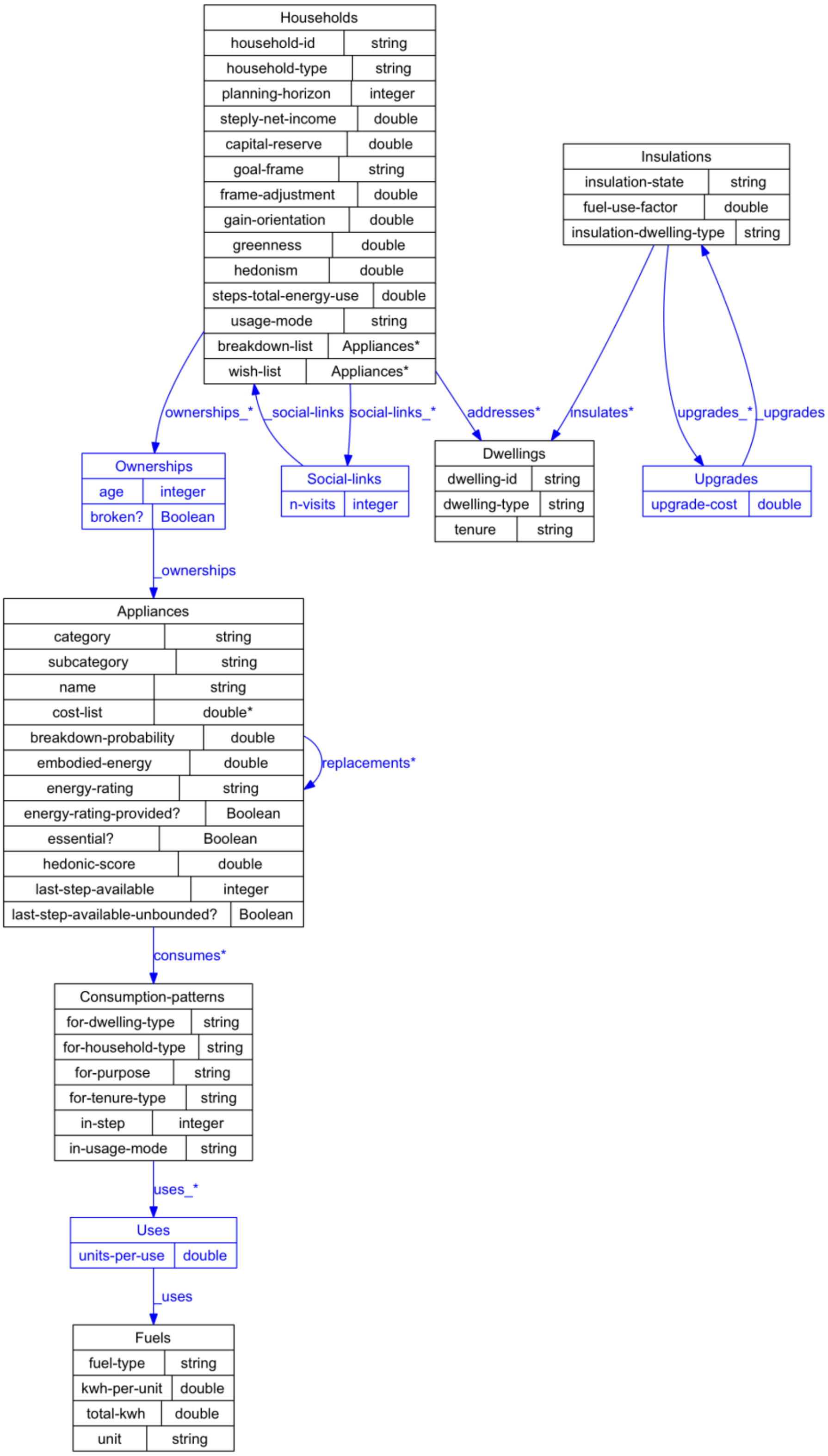


Figure 5.2. Ontology extracted from CEDSS visualised with OntoViz.

6 Data collection, and Implementation of the CEDSS Model

The process for updating ABMED to make a model that could use real-world data consisted chiefly in allowing the model to work with several input files, and to add functionality that enabled the exploration of certain aspects of scenarios, in particular home insulation, which is of considerable importance in determining total domestic energy demand.

It must be admitted that the effort required for data collection of the agent-based model for GILDED were considerably underestimated. So far as is known, no such model of household energy demand at community level has previously been designed and implemented, so we had no good precedents to draw on. It was intended that most of the necessary data would be derived from the questionnaire and carbon calculator surveys undertaken in workpackages 3 and 4, and these did indeed provide essential information. However, the need to keep these research tools to a length that subjects would be willing to complete, while satisfying the data requirements of those workpackages, limited the amount of data relevant to CEDSS that could be gathered. Moreover, the importance of issues relating to household appliances revealed by the knowledge elicitation exercises described above, necessitated gathering data about appliances that would not be available from the households themselves, and that turned out not to be readily available at all. In order to make the collection of adequate data feasible, it was decided that the model should focus on direct energy demand (for electricity, gas and other fuels) in the home; and it became clear during the process of model-building that it would not be feasible, within the resources available to GILDED, to extend the model to cover more than the Scottish case-study area.

Since the model was intended to simulate the process of change in household energy demand over time, in scenarios reaching to 2050, it was decided to test it by setting it up to represent the evolution of a village or urban neighbourhood over a period leading up to the point in the second quarter of 2010 when the first GILDED survey, including a questionnaire and carbon calculator, was distributed, completed, and collected. However, this presented considerable problems. The questionnaire and carbon calculator mainly asked questions about current energy-using equipment and energy-conserving insulation (some questions were asked about the age of equipment, but these were asked purely in order to calculate likely current energy requirements). No questions were asked about how purchasing decisions were made, but such decisions were central to the model. It was therefore necessary to combine the information from the questionnaire and carbon calculator with data from a range of publicly available sources about changes over time in the availability and ownership of relevant equipment. The survey's sample of households from Aberdeen city and Aberdeenshire was divided into "urban" (Aberdeen City – 197 households once a number with vital data missing were excluded) and "rural" (Aberdeenshire – 200 households) subsamples, both because possible differences between the populations of these areas was one of the topics GILDED was to investigate, and for the purposes of calibration and validation. This section focuses on the urban subsample, and the model parameters constructed using it as a basis – although many of the parameter files described were shared with the corresponding rural parameter set, and those that were not shared were constructed in a parallel fashion. The names of those parameter files that differ between the urban and rural models are followed in the subsection headings by "(urban)".

After a preliminary investigation of the available data sources, it was determined that it was feasible, although by no means easy, to "retrodict" aspects of the survey households' dwellings and energy-using appliances as they might plausibly have been at the start of 2000. If running the model forward to mid-2010 then gave energy demands reasonably close to the figures derived from the survey, we could have sufficient confidence in the model to extend such runs to 2050 with a reasonable expectation of getting meaningful and useful results. The current section describes those aspects of the model that could be based, to varying extents, on the 2010 survey data, and a range of publicly available data sources; and how those sources were used. The processes of calibrating those aspects of the model that could not be based on data in this way, and of validating that the calibrated model was satisfactory, is described in the next section; and its use in scenarios to 2050 in the next but one. It is worth noting that the model has capabilities which have not yet been used, mostly concerned with demographic processes; these capabilities rely on further input files, which are briefly described in section 9, on future work.

A wide range of such sources was consulted, but the most important of those used directly in constructing the CEDSS parameter files were as follows:

- UK Department of Energy and Climate Change (DECC) time series of prices for domestic electricity, gas and heating oil (taken from files qep413.xls, qep551.xls and qep591.xls).
- United Kingdom Department of Energy and Climate Change's *Great Britain's Housing Energy Fact File 2011*. (Palmer and Cooper 2011).
- UK Office for National Statistics (ONS) "Family Spending" series (officially known first as the Family Expenditure Survey, then as the Expenditure and Food Survey, then as the Living Costs and Food Survey). These were used for the estimates of spending on fuels, household appliances, and household maintenance and repair, by each gross income decile of the UK household income distribution, over the period 2000-2010; and the percentage of households with washing machines, tumble dryers and dishwashers in 2000 and 2010.
- The Institute for Financial Studies working Paper 02/21, *The Distribution of Financial Wealth in the UK: Evidence from 2000 BHPS Data* (Banks, Smith and Wakefield 2002). This was used for figures on the net wealth of the UK population by income quintile in 2000.
- Argos catalogues, 2000-2010. Argos is a UK retailer, selling a wide variety of household appliances. CEDSS requires a list of appliances offered for sale, with prices and (where they exist) energy efficiency ratings, for each quarter-year covered by a run. Such information was by no means easy to acquire: repeated enquiries to retailers elicited no assistance. Eventually, a source was found at the Victoria and Albert Museum in London.

The details of how the sources listed were used are given below, where the construction of each of the parameter files for CEDSS is described. Further sources of lesser importance are mentioned in relation to the files to which they were relevant.

The following subsections detail the structure of the input files constructed and then largely held constant (with exceptions described in section 7) during calibration, and how the versions of these files used in calibration were constructed.

6.1 Household file (urban)

This can be regarded as the key input file, in relation to which other files were constructed.

The household file allocates initial households and their parameters to each dwelling. It has a heading row indicating the columns, which may be in any order. The following columns are required: id, type, income, capital, hedonism, gain, norm, frame, planning, dwelling. Of these, capital, hedonism, gain, norm, frame, and planning are expected to be numeric; income is expected to be a list of one or more numbers; dwelling is expected to contain the identifier of an existing dwelling to allocate the household to (see the description of the dwellings file below) and type is provided to allow for households to be assigned demographic characteristics Income is intended to represent quarterly household income available for spending on domestic energy, energy-using appliances, and in the case of owner-occupiers, replacement heating systems, and insulation, while capital represents the net monetary wealth available to a household at the start of a model run. The remaining fields represent decision-making characteristics of the household. The first three, "hedonism", "gain" and "norm" represented the extent to which three different types of value described in the "Outline Description of CEDSS" subsection of the Introduction guide the household's decision-making: hedonistic values (focused on comfort and enjoyment), "egoistic" values (focused on gaining and keeping resources, and in this context, specifically saving money), and "biospheric" values (care for the environment). The "frame" field governs the extent to which these values are subject to change through social contacts, and the "planning" field the time horizon over which the household calculates savings of money or energy use when considering improving insulation or installing a new heating system.

Below are the first few lines of the urban household file used in the calibration runs:

```
id,type,dwelling,income,capital,hedonism,gain,norm,frame,planning
hhu-29,household,dwu-29,[322 322 322 322 295 295 295 295 295 295 295 295 312
312 312 312 280 280 280 280 296 296 296 296 322 322 322 322 341 341 341 341
350 350 350 350 429 429 429 429 429 429],341,3.790019416,1.714560112,
0.884983635,0.2,20
hhu-42,household,dwu-42,[363 363 363 363 332 332 332 332 332 332 332 332 302
302 302 302 359 359 359 359 360 360 360 360 341 341 341 341 428 428 428 428
439 439 439 439 441 441 441 441 441 441],8925,4.732196171,3.968977439,
0.264718996,0.2,20
hhu-46,household,dwu-46,[212 212 212 212 212 212 212 212 212 212 212 212 202
202 202 202 233 233 233 233 211 211 211 211 230 230 230 230 251 251 251 251
256 256 256 256 313 313 313 313 313 313],1,0.184920598,4.588994936,
0.03211146,0.2,20
hhu-51,household,dwu-51,[322 322 322 322 295 295 295 295 295 295 295 295 312
312 312 312 280 280 280 280 296 296 296 296 322 322 322 322 341 341 341 341
350 350 350 350 429 429 429 429 429 429],341,2.46654544,3.051400624,
0.925957717,0.2,20
...
```

The "id" field in each line identifies a specific (but anonymised) urban household (households are identified as urban or rural in the dataset) in the dataset from the GILDED 2010 questionnaire and carbon calculator. In the work reported here, different types of household were not used; instead, all households were treated as "average" households for the purposes of calculating energy demand for water-heating, the only point in the model - and in the document used to calculate emissions in the GILDED 2010 and 2011 surveys of households in Aberdeen City and Aberdeenshire (Department of Energy and Climate Change 2009) at which household size made a difference. For income figures over the entire period, we began with the questionnaire figures for "monthly household income after taxes", which divided the survey households into 11 bands. These bands corresponded reasonably well with the income deciles in table A6 of the UK Office for National Statistics (ONS) 2010 Living Costs and Food Survey (Horsfield 2011), once the eighth and ninth questionnaire bands were amalgamated: the top row in table 6.1 below. It must be recognised that the correspondence is not exact, particularly given the different basis for the figures, but given the sources of uncertainty in both figures, it was considered good enough.

Table 6.1: Lower bounds of Horsefield (2011) 4-weekly gross income deciles compared with those of monthly household income after taxes from GILDED 2010 survey

ONS decile	1	2	3	4	5	6	7	8	9	10
ONS lower bound	0	640	952	1260	1652	2088	2604	3204	4060	5472
GILDED lower bound	0	444	888	1332	1776	2220	2664	3109 3554	3998	4442

Tables in Horsfield (2011) and corresponding ONS reports for earlier years, back to 2000, provide the mean weekly household expenditure on "electricity, gas and other fuels", "household appliances", "TV, videos and computers" and "maintenance and repair of dwelling" for each gross income decile; multiplying these figures by 13 gave a mean quarterly spend under each of these headings. It was assumed that the quarterly total of the first three of these four categories gave a reasonable estimate of the money households in the corresponding survey bands would have had to spend on domestic energy and energy-related equipment per quarter for each calendar year of the period covered in the calibration runs (from the start of 2000 to midway through 2010) if renting their dwelling; and this amount plus the average quarterly spend on "maintenance and repair of dwelling" for owner-occupier households. Again, it must be recognised that this procedure gives only a rough approximation, but in the absence of data on the actual income available for these areas of spending for the survey households over the decade preceding the survey, is the best that could be done. As described in section 7, we experimented with varying these figures during the calibration process.

Figures for capital are perhaps even rougher. They were drawn from an Institute for Fiscal Studies report on the distribution of financial wealth in 2000 (Banks, Smith and Wakefield 2002), which in Table A4 gives figures for the distribution of net financial wealth in the UK for each quintile of the income distribution. Taking all the households in the urban subsample and grouping the deciles already defined into quintiles, 1/4 of them were assigned the Banks Smith and Wakefield (2002) 25th percentile of net wealth for the corresponding quintile, 1/2 were assigned the median figure, and 1/4 the 75th percentile.

For the remaining fields, we had in effect no real-world data: while we might have used some of the questions on values and attitudes to assign relative strengths to the "hedonism", "norm" and "gain" fields, the Scottish sample showed no statistically significant relationship between answers to these questions, and energy demand. We therefore tried a range of different possibilities for these, and for the "frame" and "planning" fields, as described below.

6.2 Dwellings file (urban)

The dwellings file gives properties of the dwelling of each household (empty dwellings can also exist, but are not used in the model runs reported here). The file should have a heading row, with the following headings specified in any order: “id”, “tenure”, “type”, “insulation”. There is then one row for each dwelling, where the entry in the “id” column corresponds to the dwelling-id in the patch file. The set of tenure types will be inferred from this file, and is expected to be consistent with all other files where tenure is mentioned. Entries in the insulation column should correspond to entries in the insulation file. Here are the first few lines of the file used in the calibration runs:

```
id,tenure,type,insulation
dwu-29,owned,house-semi-solid-3,minimum
dwu-42,owned,flat-non-top-solid-1,dg-loft100
dwu-46,owned,flat-top-solid-1,dg-loft100
dwu-51,owned,flat-non-top-cavity-2,loft100
...
```

The set of dwelling-types used was taken from those in the GILDED carbon calculator, and the types were assigned to individual households on the basis of their survey responses. The main types are as follows:

- Bungalow (detached)
- Bungalow (semi-detached)
- House (detached)
- House (semi-detached)
- House (end-terrace)
- House (mid-terrace)
- Flat (top-floor)
- Flat (non top-floor)

Each of these main types is subdivided according to the number of bedrooms (1, 2, 3, 4, 5, 6 or more), and by whether the external walls are solid, or have an internal cavity that is or could be filled to improve insulation. Tenure was either “owned” or “rented”, again taken from survey responses. The dwelling type and tenure were assumed to have remained the same since 2000.

The 12 possible insulation-states were constructed from three components: whether or not there was double-glazing, whether loft insulation was present and if so whether it was 100mm or 270mm thick, and whether there was wall insulation. There were some complications: flats other than top flats were assumed to be partially insulated from above by the flat above them, taken to be equivalent to 100mm of loft insulation; the cost and effect of wall insulation differed as between houses with solid and cavity walls, and tenants and those living in flats were assumed not to be in a position to improve their dwelling's insulation. The combination of dwelling type and insulation state is used in the model to calculate “useful energy demand” for space heating.

With regard to insulation-state, it was not reasonable to assume that there had been no change since 2000; we knew that there has been considerable change both locally (Aberdeen Council 2007) and nationally (Palmer and Cooper 2011) in home insulation. The method adopted was to compare national figures for changes in the penetration of the various types of insulation for 2000 and 2010 (taking the figures from Palmer and Cooper (2011)), and then assume that the same proportion of homes in the urban GILDED sample that had each type of insulation in 2010 would not have had it in 2000, as the national figures indicated would be the case for the whole population of UK households.

6.3 Patch file (urban) and Patch legend file

Netlogo divides space into square “patches”. The patch file states the patch type of a patch. It has no header row; data are given in order X, Y, patch-type, dwelling-id..., where dwelling-id is a comma-separated list of dwellings on that patch, and is only used if patch-type is “dwelling”. The patch legend file specifies the permitted types of patch in a model, and the colours they are assigned when the model is run using the Netlogo GUI. The patch legend file for CEDSS is as shown below:

```
dwelling,green
street,gray
park,52
junction,6
```

where the numbers “52” and “6” represent particular shades of green and grey respectively. Currently only one type of patch has any direct effect on CEDSS: “dwelling”. These are the patches on which dwellings may be located. Shown below are extracts from the patch file used in the calibration process:

```
2,1,dwelling,dwu-900
3,1,dwelling,dwu-911
4,1,dwelling,dwu-940
5,1,dwelling,dwu-1014
...
5,23,dwelling,dwu-1046
6,23,dwelling,dwu-408
```

```
0,0,junction,
0,11,junction,
...
1,1,park,
1,10,park,
1,12,park,
1,21,park,
...
1,0,street,
2,0,street,
3,0,street,
4,0,street,
...
```

It was decided not to use real-world street-networks as the basis of those used in CEDSS: it was considered that the additional time required to do so could be more usefully spent in improving the model and collecting other types of data, as any specific street-layout used would not have corresponded to the locations in which the GILDED household surveys were undertaken (survey respondents, on whose data characteristics of model households' dwellings and household appliances were based, were widely distributed geographically across the case-study area). The two patch files (for the urban and rural subsamples) for the runs reported here were constructed using the following procedure:

1. A procedure for generating a street-network able to accommodate any desired number of dwelling-patches was defined. The street-network consists of a set of "dwelling-rows", each dwelling-row consisting of eight patches on which a dwelling included in the model run could be placed. All dwelling-rows except the last include exactly eight such dwellings, one per patch. Four dwelling-rows make up a dwelling-square, and each successive dwelling-squares is completely filled, with 32 dwellings, before the next is begun. The order in which patches were assigned dwellings was to first fill the row on the "south" side, west to east; then the west side, south to north; the east side, south to north, and the north side, west to east. The first square filled was that in the south-west corner of the network; the second abutted it to the east; the third abutted the first to the north; the fourth abutted both the second and the third. The fifth abutted the second to the east, the sixth abutted the fourth to the east, and the seventh and last abutted the third to the north.
2. All dwellings of one type in the dwellings file were assigned consecutive patches in the ordering thus defined, to reflect the fact that similar dwellings are likely to be located near each other in real-world settlements.

6.4 Insulation file

The insulation file is used to specify the space heating energy demand for every permitted combination of insulation state and dwelling type, relative to an arbitrarily chosen standard combination, a two-bedroomed top floor flat with double glazing and 100mm of loft insulation, but no wall insulation. This factor is multiplied by a figure for the energy needed for space heating for a dwelling of that type and insulation state, for each type of heating system covered (see the "Appliances fuel file" below). The unique identifier for an insulation is given by its insulation state and dwelling type combined. The file used in the calibration runs begins:

```
insulation-state,fuel-use-factor,dwelling-type
minimum,2.487,bung-det-cavity-1
dg,2.211,bung-det-cavity-1
loft100,1.987,bung-det-cavity-1
loft270,1.693,bung-det-cavity-1
...
```

The factors used are taken from tables in the document used to calculate CO_2 emissions in GILDED's 2010 and 2011 surveys (Department of Energy and Climate Change 2009).

6.5 Insulation upgrade file

The insulation upgrade is used to specify the insulation upgrades that are available, taking a dwelling from one insulation-state to another. There is one line in this file for each possible upgrade for each dwelling type, giving the cost of the upgrade. The file used in the calibration runs begins

```
dwelling-type,from-state,to-state,cost
bung-det-cavity-1,minimum,dg,3700
bung-det-cavity-1,minimum,loft270,500
bung-det-cavity-1,minimum,wi,500
bung-det-cavity-1,loft100,dg-loft100,3700
...
```

The prices were taken from Cambridge Architectural Research et al. (2009), p.31.

6.6 Appliances file

The appliances file gives properties of each appliance available for purchase by households in the model. The model currently covers heating systems, cookers, refrigerators, freezers, washing machines, dryers, dishwashers and televisions. The file includes a header row, and each subsequent row gives the required details for each appliance: name, category, subcategory, essential, hedonic-score, cost, energy-rating, embodied-energy, breakdown-probability, first-step-available ,last-step-available. The name is a unique name for the appliance; category and subcategory are used in the process of determining what a household buys, while if essential is set to "true", households will replace the item immediately if it breaks down. Hedonic score is not currently used. Cost is a list of purchase prices, one for each quarter during which the item was available, energy-rating is information about the energy consumption of the appliance given to consumers, with a higher score meaning better energy consumption (this is not available for all categories of appliance – for those for which it is not available, all appliances in the category are given the same score). (The actual energy consumed by appliances for different purposes is given in the appliances fuel file.) The embodied-energy field is not currently used. The breakdown-probability is the probability, per time step, of the equipment breaking down. The first and last steps available are the numbers of the time steps in the model indicating when they are first and last available. The start of the file used in the calibration runs is as follows:

```
name,category,subcategory,essential,hedonic-score,cost,energy-
rating,embodied-energy,breakdown-probability,first-step-available,last-step-
available
gas-standard-boiler-D,heating,gas-boiler-sub,TRUE,0,[2000],6,0,0.013,18,27
gas-standard-boiler-F,heating,gas-boiler-sub,TRUE,0,[2000],8,0,0.013,2,17
```

```
gas-standard-boiler-G,heating,gas-boiler-sub,TRUE,0,[2000],9,0,0.013,0,1
gas-condensing-boiler-A,heating,gas-boiler-
sub,TRUE,0,[2500],3,0,0.013,36,Inf
gas-condensing-boiler-B,heating,gas-boiler-sub,TRUE,0,[2500],4,0,0.013,0,35
oil-standard-boiler-C,heating,oil-boiler-sub,TRUE,0,[2000],5,0,0.013,18,27
oil-standard-boiler-D,heating,oil-boiler-sub,TRUE,0,[2000],6,0,0.013,2,17
oil-standard-boiler-G,heating,oil-boiler-sub,TRUE,0,[2000],9,0,0.013,0,1
oil-condensing-boiler-A,heating,oil-boiler-sub,TRUE,0,[2500],3,0,0.013,0,Inf
electric-heaters,heating,electric-heating,TRUE,0,[2000],0,0,0.013,0,Inf
Teba TFR14,cooker,electric-cooker-sub,TRUE,0,[200],11,0,0.014,0,3
Creda Capri C150E,cooker,electric-cooker-sub,TRUE,0,[250 250 250 250 230
230],11,0,0.014,0,5
Beko DV 5631,cooker,electric-cooker-sub,TRUE,0,[400 400 400 400 380 380 380
380 380 380 380 380],11,0,0.014,0,11
...
```

For heating systems, figures for energy use were calculated from Department of Energy and Climate Change (2009) figures on energy efficiency, and the energy requirements of a two-bedroom top floor flat with double glazing and 100mm of loft insulation but no wall insulation (see above under "Insulation file" for information on adjustments to the figure for other types of dwelling and insulation states). Figures for prices were taken from the file consumerissues.jobsandmoney.htm retrieved from http://www.guardian.co.uk/money/2005/apr/02/consumerissues.jobsandmoney, and energy ratings corresponding to the boiler efficiencies from the website http://www.sedbuk.com/pages/bands.htm, which is linked to the UK government's Standard Assessment Procedure (SAP) for assessing energy efficiency of buildings. Standard boilers are no longer available after the second quarter 0f 2007, when it became compulsory to fit condensing boilers in new homes and when replacing an old heating system in Scotland.

The remaining appliances listed in the file used in the calibration process are all taken from Argos catalogues of household appliances and other products. Argos is a large UK retailer, which publishes a catalogue every six months. It was judged impractical to include in the model the full range of household appliances available during the period 2000-2010, and indeed, accessing information about even a fraction of them proved challenging. Repeated requests to a number of retailers brought no positive responses. Relevant pages of the catalogues from the second half of 2000 through 2007 were consulted at the Victoria and Albert Museum in London; catalogues. Catalogues for Spring/Summer 2008, Autumn/Winter 2009, and Spring/Summer 2010 were obtained from colleagues. The contents of the catalogues limited the range of appliances categories it was feasible to cover to cookers, refrigerators, freezers, washing machines, dryers, dishwashers and televisions. The most important omissions were probably lighting, where complications due to different numbers and types of light fittings made their inclusion in the model too difficult; and home computer equipment, of which Argos sells very little. The categories treated as essential are heating systems, cookers, refrigerators and washing machines; those treated as non-essential are freezers, dryers, dishwashers and televisions. Four categories were divided into subcategories:

- heating-systems were divided by fuel (gas, electricity or oil) and in the case of gas and oil, by whether the boiler was a standard or condensing boiler;
- cookers were divided into electrical with a standard hob, electrical with a ceramic induction hob, gas, and dual fuel (electric oven and gas hob);
- refrigerators were divided into those with and without a freezer compartment;
- televisions were divided into cathode ray tube (CRT), liquid crystal display (LCD) and plasma.

The following algorithm was used to select items from each subcategory (categories that were not subdivided were regarded as their own sole subcategory):

- For each catalogue, and each sub-category:
  - List the items for sale, ignoring small numbers of items outside the main catalogue section for the category. In the case of photographed catalogues, include items that are not legible enough to use, unless it is unclear what sub-category they belong to. Note which appear too illegible to use, note any duplicate photos, and any pages within the category that are not to be used due to illegibility.
  - Note the photograph number and catalogue item number of the first, middle and last item in catalogue order (these items are "selection points"). If there are an even number of items in the sub-category, list the first of the two items nearest the middle (e.g. if there are six items, list the first, third and sixth). If there are fewer than three items, list them all. (If there are two, they are counted as the first and last items, there being no middle item; if there is only one, it counts as the first item, there being no middle or last. This has implications for later catalogues explained below.
- For the Autumn/Winter 2000 catalogue (the first in which the categories of interest appear):
  - If a selection point item is legible, select it.
  - Otherwise, look for a substitute item. For the first selection point, use the legible item closest to the start; for the last selection point, that closest to the end; for the middle selection point, the next choice is that immediately after the selection point item, then that immediately before it, then that two after it, that two before it, and so on. A first item is to be selected before attempting to select a last item, and the latter before attempting to select a middle item.
- For Spring/Summer 2000, use the items selected for Autumn/Winter 2000, together with their prices and other characteristics.
- For each catalogue from Spring/Summer 2001 to Spring/Summer 2008:
  - Check which of the items from the preceding catalogue reappear. If the first item from the preceding catalogue is present, do not select a new first item starting from the current catalogues. If the last item from the preceding catalogue is present, do not select a new last item. If the middle item from the preceding catalogue is present, do not select a new middle item. Any items carried forward from the preceding catalogue thus retain the position (first, middle or last) they had in that preceding catalogue, and only the unfilled positions are filled by new selections.
  - If any new items are to be selected, this is done in the same way as for the earliest catalogue, except that any items carried over from the preceding

    catalogue cannot be re-selected (in this phase, they are treated the same as illegible items).
  - If there are fewer than three legible items (old or new) for a sub-category in the current catalogue, additional items are carried over from the preceding catalogue, if available, even if they do not occur in the new catalogue, in the order first, last, middle. Again, these items retain their former position as first, middle or last.
- For Autumn/Winter 2008 and Spring/Summer 2009 (for which no catalogue was available), carry forward all items from Spring/Summer 2009.
- For Autumn/winter 2009 and Spring/Summer 2010, revert to the procedure used from Spring/Summer 2001 to Spring/Summer 2008.
- Notes:
  - "Packages", e.g. of a washing machine and dryer, or television and video recorder or set-top box, have not been included.
  - Items on are generally numbered consecutively on each spread (pair of facing pages). Occasionally, a number is omitted, in the case of a full-page promotion. More frequently, one number will cover a number of variants of an item. In the cases of cookers, fridges, fridge-freezers, freezers, washing machines and dryers, the cheapest variant has been used and the others ignored. In the case of televisions, variants which concern screen size have been counted as different, and given identifying letters (a, b, …)
  - Note: "widescreen" and "flatscreen" televisions have been placed in the CRT category unless they are specified as LCD or plasma.

Trends that were noticed during the process of extracting data from the catalogue were :

- There was a considerable increase in the number of items in most categories and subcategories, with the exception of CRT televisions, which had disappeared by spring/Summer 2010, and plasma televisions, which first grew and then declined in number.
- There was some improvement in the energy-rating of appliances over the 10.5 year period.
- There was a frequent pattern whereby an item would appear at one price, and decline in price during its period of availability, while new items in the same subcategory entered the catalogue at successively higher prices. (Some items did increase in price, and the amount of decline varied between categories and subcategories, with plasma televisions showing by far the largest falls.)

Figures for breakdown probability of appliances (where "breakdown" means a failure so serious the appliance cannot be repaired or is not worth repairing) per year are not easy to find. In fact, breakdown probability varies over time, with older appliances and perhaps very new ones more prone to breakdown, but attempting to replicate such effects was judged not to be worth the amount of time and effort required. Survival curves for televisions and refrigerators are given by Gutierrez (20): 50% of televisions have been discarded after about 200 months (66.67 quarters) while for refrigerators half are gone in 150 months (50 quarters). A quarterly failure rate of 0.014 for refrigerators will reproduce the latter result, and in the absence of other information this figure was used for other "white goods" (all the appliances included in the model other than heating systems and televisions); a quarterly failure rate of 0.010 for a television reproduces the figure for median failure time. Owen

(2006) p. 30, says that turnover of heating systems is 5% per annum, corresponding to a probability of 0.013 per quarter.

6.7 Appliance replacement file

The appliance replacement file gives replacements for each appliance: the first item on a line can be replaced by any of the subsequent items on that line, but by nothing else. In the file used in calibration runs, any item can replace any other in the same subcategory. Additionally, condensing boilers can replace standard boilers of the same type, electric cookers with and without ceramic induction hobs can replace each other, gas and dual-fuel cookers can replace each other, LCD and plasma televisions can replace each other.

```
gas-standard-boiler-D,gas-condensing-boiler-A,gas-condensing-boiler-B
gas-standard-boiler-F,gas-standard-boiler-D,gas-condensing-boiler-A,gas-
condensing-boiler-B
gas-standard-boiler-G,gas-standard-boiler-F,gas-standard-boiler-D,gas-
condensing-boiler-A,gas-condensing-boiler-B
gas-condensing-boiler-A
gas-condensing-boiler-B,gas-condensing-boiler-A
oil-standard-boiler-C,oil-condensing-boiler-A
oil-standard-boiler-D,oil-standard-boiler-C,oil-condensing-boiler-A
oil-standard-boiler-G,oil-standard-boiler-D,oil-standard-boiler-C,oil-
condensing-boiler-A
oil-condensing-boiler-A
electric-heaters
Teba TFR14,Teba TFR14,Creda Capri C150E,Beko DV 5631,Tricity Bendix SIE
324,Jackson by Creda J15 1EW,Belling Forum 335,Zanussi ZCE611X,Zanussi
ZCE531X,Zanussi ZCE641,New World 50WLG,Belling Forum 317,Beko D532,Hotpoint
HL500 electric,Indesit KD3E1,Hotpoint HW170E,Creda Menu M350E,Tricity Bendix
SIE 514,Beko DVC61,Creda X155,Tricity Bendix SIE 554,Tricity Bendix
SE340,Belling 644,Beko DCC 4521,Leisure Roma 50,Beko DC543,Leisure Zenith
ZE6HV,Beko DVC563,Indesit K6 C320,Hotpoint EW74,Hotpoint Creda C367
```

6.8 Fuel file

The fuel file gives properties of each fuel used to supply an appliance. Most appliances will only use one type of fuel, but some (e.g. gas boiler or cooker) might use two (e.g. gas and electricity), or possibly more. This is a simple three-column file with columns type, unit and kWh. The last column gives the number of kWh of energy per unit of consumption of the fuel.

For the purposes of calibration, gas and electricity were subdivided according to whether their use was for space heating, water heating, or appliances; and oil according to whether its use was for space or water heating. All fuels were measured in kWhs (Department of Energy and Climate Change (2009), the source for usage figures, gives all figures in kWh). Here is the file used in calibration runs:

```
type,unit,kWh
space-heating-elect,sh-elect-kWh,1
water-heating-elect,wh-elect-kWh,1
appliance-elect,ap-elect-kWh,1
space-heating-gas,sh-gas-kWh,1
water-heating-gas,wh-gas-kWh,1
appliance-gas,ap-gas-kWh,1
space-heating-oil,sh-oil-kWh,1
water-heating-oil,wh-oil-kWh,1
```

6.9 Appliances fuel file (urban)

The appliances fuel file gives the fuel used by households for each use of an appliance in each context. It has a heading row with columns "appliance" giving the name of the appliance, "household" giving the household type, "dwelling" giving the dwelling type, "tenure" giving the tenure, "purpose" giving the purpose for which the appliance is used, "mode" giving the usage mode (not used in the runs described here, where the usage mode is always "normal"), "fuel" giving the type of fuel used, and "units 1" to "units" giving the number of units. Obviously the file has the potential to get very long. To mitigate this, a wildcard (*) may be used for the dwelling, tenure and mode columns (the household column cannot currently use wildcards, as household-types-list has not yet been defined when this file is read in). Here are extracts from the file used in the calibration runs:

```
appliance,category,subcategory,household,dwelling,tenure,purpose,mode,fuel,u
nits 1,units 2,units 3,units 4
gas-standard-boiler-D,heating,gas-boiler-sub,household,*,*,space-
heating,*,space-heating-gas,2355,785,785,2354
gas-standard-boiler-F,heating,gas-boiler-sub,household,*,*,space-
heating,*,space-heating-gas,2551,850,850,2551
...
gas-standard-boiler-D,heating,gas-boiler-sub,household,flat-non-top-cavity-
1,*,water-heating,*,water-heating-gas,1109,1109,1109,1109
gas-standard-boiler-D,heating,gas-boiler-sub,household,flat-non-top-cavity-
2,*,water-heating,*,water-heating-gas,1109,1109,1109,1109
gas-standard-boiler-D,heating,gas-boiler-sub,household,flat-non-top-cavity-
3,*,water-heating,*,water-heating-gas,1109,1109,1109,1109
...
Stoves Newhome 800 DF DOM,cooker,dual-fuel-cooker-
sub,household,*,*,hob,*,appliance-gas,83,83,83,83
Leisure Rangmaster 110,cooker,dual-fuel-cooker-
sub,household,*,*,hob,*,appliance-gas,83,83,83,83
...
Stoves Newhome 800 DF DOM,cooker,dual-fuel-cooker-
sub,household,*,*,hob,*,appliance-gas,83,83,83,83
Leisure Rangmaster 110,cooker,dual-fuel-cooker-
sub,household,*,*,hob,*,appliance-gas,83,83,83,83
Zanussi ZCM610X,cooker,dual-fuel-cooker-sub,household,*,*,hob,*,appliance-
gas,83,83,83,83
...
```

Data for this file was sourced from Department of Energy and Climate Change (2009). The urban and rural versions of the file differ only in the fuel requirements for water heating. The rural households in the GILDED 2010 survey had a slightly larger mean household size (2.41 as opposed to 2.34 for the urban subsample), and these figures were used in calculating those fuel requirements.

6.10 Suppliers file

The suppliers file gives energy prices offered by different suppliers for each fuel type each step. This replaces the energy-monthly-cost-list in CEDSS 2. There is no functionality at present to create a market for suppliers. Fuel prices are instead exogenous time series. There is little point in having more than one supplier for each fuel type. The file has suppliers in the first row, and fuel types in the next row. In subsequent rows, one for each time step, are the energy prices (per unit) offered by the supplier for the fuel type. If the model runs out of lines, the last price will continue to be used. The file will allow you to specify multiple suppliers for each fuel type if you wish; however the program will simply take the cheapest price in each time step as the price all agents use for that fuel. For the purposes of calibration, we distinguished electricity, gas and oil used for different purposes, as in the appliances fuel file. Data on prices was taken from the Department of Energy and Climate Change time series of prices for gas, electricity and oil, taken from files qep413.xls, qep551.xls and qep591.xls, all available online.

```
Supplier1,Supplier2,Supplier3,Supplier4,Supplier5,Supplier6,Supplier7,Suppli
er8
space-heating-elect,water-heating-elect,appliance-elect,space-heating-
gas,water-heating-gas,appliance-gas,space-heating-oil,water-heating-oil
0.0706,0.0706,0.0706,0.0166,0.0166,0.0166,0.0173,0.0173
0.0706,0.0706,0.0706,0.0166,0.0166,0.0166,0.0173,0.0173
0.0706,0.0706,0.0706,0.0166,0.0166,0.0166,0.0173,0.0173
0.0706,0.0706,0.0706,0.0166,0.0166,0.0166,0.0173,0.0173
0.0699,0.0699,0.0699,0.0171,0.0171,0.0171,0.0193,0.0193
0.0699,0.0699,0.0699,0.0171,0.0171,0.0171,0.0193,0.0193
0.0699,0.0699,0.0699,0.0171,0.0171,0.0171,0.0193,0.0193
0.0699,0.0699,0.0699,0.0171,0.0171,0.0171,0.0193,0.0193
…
```

6.11 Maximum in category file

The maximum in category files is used to place limits on how many appliances of each category each type of household may possess (if an item is about to be added in excess of this limit, the oldest item in the category will be sent to discarded first). There is one line for each type of household, with the first item on a line identifying the household type, and successive pairs of items identifying a category and setting the limit for that category. The limit should never be zero (a limit of zero or below will cause an error). If no limit is set in this file for a household type-category pair, no limit is enforced. The file used for the calibration runs is as follows:

```
household,heating,1,cooker,1,freezer,2,fridge,3,washer,1,dryer,1,dishwasher,
1,TV,5
```

The limits used were taken from the maximum number of appliances of a category owned by any household in the GILDED 2010 survey.

6.12 Household initial appliance file (urban)

The household initial appliance files assigns initial appliances to households. The first column is the name of the household, the remaining columns are appliance names. (It is possible to assign the same list of items to all households of a particular type and living in a particular type of dwelling, but that facility has not been used in the runs reported here.) The file used in the calibration runs begins as follows:

```
hhu-29,gas-standard-boiler-G,Indesit RG2190,Teba SBUZ01-08,Candy
CD242,Servis M3510,Teba TFR14,Sanyo 14M3 14 in
hhu-42,gas-standard-boiler-G,Indesit RG2190,Servis M3510,Stoves Newhome 800
DF DOM,Sanyo 14M3 14 in
```

```
hhu-46,gas-standard-boiler-G,Indesit RG2190,Servis M3510,Teba TFR14,Sanyo
14M3 14 in
hhu-51,gas-standard-boiler-G,Indesit RG2190,Servis M3510,Teba TFR14,Sanyo
14M3 14 in
hhu-93,gas-standard-boiler-G,Indesit RG2190,Teba SBUZ01-08,Candy
CD242,Servis M3510,Creda Menu M350E,Sanyo 14M3 14 in
hhu-98,gas-standard-boiler-G,Indesit RG2190,Servis M3510,Teba TFR14,Sanyo
14M3 14 in
hhu-104,gas-standard-boiler-G,LG GR-151SSF,Teba SBUZ01-08,Servis M3510,Teba
TFU 21,Sanyo 14M3 14 in
hhu-107,gas-standard-boiler-G,Indesit RG2190,Servis M3510,Hotpoint TDL30
Aquarius,Teba TFU 21,Sanyo 14M3 14 in
hhu-115,electric-heaters,LG GR-151SSF,Servis M3510,Teba TFR14,Sanyo 14M3 14
in
```

The heating system supplied to a household at the start of 2000 was determined by the “primary” fuel that household reported using for heating in 2010 (where more than one fuel was listed as primary, gas was preferred to electricity and oil, electricity to oil). In the case of gas and oil using households, was less efficient than heating systems available in 2010. For other appliances, all appliances assigned were the middle one of the three items in a subcategory drawn from the Autumn/winter 2000 Argos catalogue. All households were assumed to have a cooker of the same subcategory as they possessed in 2010, a refrigerator which would have a freezer compartment (that is, be in the “fridge-freezer-sub” subcategory) if they had one in 2010, a washing machine, and one CRT television. All households with one or more freezers in 2010 were assigned one freezer in 2000. For dryers and dishwashers, which were the categories of appliance in which the change in prevalence between 2000 and 2010 was greatest, according to the table A45 “Percentage of households with durable goods, 1970 to 2010 United Kingdom” of the 2010 Living Costs and Food Survey, the same approach as for insulation states was taken: it was assumed that the proportion of households possessing such an item in 2010 who would not have possessed one in 2000, was the same as for UK households as a whole, and each household with such an item in 2010 was assigned such an item in 2000 with the appropriate probability.

6.13 Summary

It will be seen that constructing the data files described above, to represent the situation of the households in the model in 2000, involved making a considerable number of assumptions. Nevertheless, there are few aspects of the files that do not have any empirical justification. It should also be noted that all households retain their identity throughout the runs – and indeed, throughout the scenario runs to 2050, with the movement of households in and out of dwellings not being modelled. Obviously this is not realistic, but on the other hand, there is no particular reason to expect incoming households to differ in any particular direction from outgoing ones – indeed, since households select their dwellings on the basis of their size and cost, the set of households occupying a given set of dwellings is likely to be similar over considerable periods of time, even though there will then be considerable turnover as individual households grow or shrink in size, and change their economic status.

7 Calibration and Validation of the CEDSS model

The input files described above were held constant throughout the calibration process, except for those aspects of the households file that were left unspecified in the preceding section, and the income figures in that file. Four rounds of calibration runs, covering the period from the start of 2000 to the middle of 2010, were undertaken to find values of these aspects of the households file, and of the remaining model parameters used in the work reported here, that gave the best match to the total usages of electricity, gas and oil for space heating, water heating, and non-heating appliances for the urban subsample in the GILDED 2010 survey. (Very few households reported using any other fuels, and these households were excluded from the subsample.) The resulting set of parameters is referred to as the "Urban 2000-2010 Model".

To validate CEDSS for the Scottish case study area, a second partial set of input files was created based on data from the rural (Aberdeenshire) subsample, and combined with the additional parameters selected in stage 2 to produce the "Rural 2000-2010 Model". It was then verified that this model gave reasonably accurate results for the total usages of electricity, gas and oil for space heating, water heating, and non-heating appliances for the rural subsample in the GILDED 2010 survey. Ideally, both calibration and validation procedures would have been more extensive than time allowed, but the results are judged satisfactory, particularly given the need to construct the 2000 starting point for the calibration and validation runs without data for the surveyed households from that time.

7.1 Calibration Stage 1A

Even keeping the parameter files described in section 6 above fixed, CEDSS still has a considerable number of "free" parameters (parameters unconstrained by real-world data) to be specified. Calibration requires an exploration of a model's parameter space, experimenting with a range of combinations of parameter values, but even if using only two values per parameter, the number of possible combinations to be tried was too great to be explored simultaneously. The largest component of domestic energy demand, by a considerable margin, is for space heating. It was therefore decided, after some preliminary experimentation, to begin by varying those parameters considered most likely to affect this demand, while holding the rest constant. The plan followed was first (calibration stage 1A) to perform single runs of the widest practicable range of possible combinations of these parameters, and to single out for further tests the 16 parameter combinations that gave the best *overall sum of absolute errors*. The overall sum of absolute errors was calculated as the sum of the absolute values of eight figures: the differences between the energy demand calculated from the urban subsample in the 2010 GILDED survey, and the corresponding model result, over the last four quarters of a run, for electricity, gas and oil used for space heating, for electricity, gas and oil used for water heating, and for electricity and gas used for non-heating household appliances (the only gas appliances were gas cookers).

Parameters varied in calibration stage 1A were as follows:

- *Household file*
    - *Income*. Because of the uncertainties inherent in the calculations of income levels, and the importance of this parameter, incomes uniformly 1.5 and 0.75 times those calculated as described in section 6.1 were substituted for those calculated.
    - *Value strength parameters* ('hedonism', 'gain' and 'norm', corresponding to hedonistic, egoistic and biospheric or pro-environmental values, and used in determining which values predominate during each time step). Because the GILDED 2010 Scottish survey indicated that expressed pro-environmental values did not have a significant effect on energy demand, we made the default assumption that these values were weak relative to hedonic and egoistic values, at least when the purchasing decisions CEDSS simulates are made. The default strengths assigned were hedonism 5, gain 5, norm 1. Alternative settings tried were 1:1:1, 5:5:5, 1:5:1.
    - *Planning horizon* parameter ('planning'), used in determining whether to adopt insulation measures, and how to replace a broken heating system. The default value was 20 time steps (5 years). An alternative value of 4 time steps (1 year) was tried.
- *habit-adjustment-factor*. This is one of the model's numerical parameters. It determines the maximum amount by which households adjust the strength of their value strength parameters in the direction of the value that has predominated in their decision-making in the current time step (the strengths have a floor of 0, and their sum never changes, so this maximum is not always reached). The default value was 0.1; alternative values of 0 and 0.5 were tried.
- *credit-multiple-limit*. Another numerical parameter. When buying non-essential items, a household will not buy if the result would place them in debt (make their capital negative) by more than their current income multiplies by this number. The default value was 5; alternative values of 0 and 20 were tried.
- *Social link matrix file*. If two households have a social link, either can "visit" the other. As a result, their value strength parameters will move closer together, and the visitor will usually add an appliance which the host has to their "wish-list" of appliances they will consider buying. This file specifies the probability that a social link will exist between a pair of households at the start of the model run. For each pair of household/dwelling type combinations, the probability of making initial links between agents belonging to these types under various circumstances. Tenure is ignored. These circumstances are:

  a) Between households on dwellings on the same patch
  b) Between households on dwellings on neighbouring patches within a specified distance
  c) Between households on dwellings separated by one or more contiguous patches of a given type.

Each probability is treated independently. Hence if you have two type (b) links, one with radius x, probability p_1, another with radius y, probability p_2, where $y > x$, then the probability of making a link within radius x is $p_1 + (1 - p_1)p_2$.

Social links can be both gained and lost: a new social link is always made via an existing link; links are most likely to be lost when the two households are distant in space, and have few of the same household appliances (used as a proxy for similarity of lifestyle).

The default social link matrix file allows the same probabilities for all pairs of households on the same square of 0.1, and of all pairs of households anywhere of 0.05. It should be noted that there would be some initial assortment of social links by dwelling-type, as dwellings of the same type were placed close together. The alternative tried at this stage was to have no social links at all.

These ranges of possibilities gave a total of 3.4.2.3.3.2 = 432 possible combinations. Each was run once. The 16 giving the smallest sum of absolute errors were as shown in table 7.1. For comparison, the total annual energy demand in kWh across all households and uses in the GLDED 2010 Scottish urban subsample was 4,775,091 kWh.

Table 7.1. The sixteen runs from calibration stage 1A giving smallest sum of absolute errors.

Identifier	Income multiple	Values parameters	Planning horizon	Habit adjustment factor	Credit multiple limit	Social links	Sum of absolute errors
1A:1	1	5:5:1	20	0.5	20	Yes	216562
1A:2	1	5:5:1	20	0.5	5	Yes	230931
1A:3	1	5:5:5	20	0.5	20	Yes	233209
1A:4	0.75	5:5:1	4	0	20	No	246743
1A:5	1	5:5:5	20	0.5	5	Yes	247642
1A:6	0.75	5:5:1	20	0.5	5	Yes	248262
1A:7	1	5:1:1	20	0	20	Yes	251139
1A:8	1.5	5:5:1	20	0.1	20	Yes	254533
1A:9	0.75	5:5:5	20	0.5	5	Yes	256755
1A:10	0.75	5:1:1	4	0	20	No	257008
1A:11	1.5	5:5:1	20	0.5	20	Yes	264376
1A:12	1.5	5:5:1	20	0.5	0	Yes	269621
1A:13	1.5	5:5:5	4	0.5	0	No	270567
1A:14	1.5	5:1:1	20	0	0	Yes	270585
1A:15	1	1:5:1	20	0	5	Yes	277378
1A:16	0.75	5:5:1	20	0.1	0	Yes	278119

7.2 Calibration Stage 1B

The 16 best versions of CEDSS (i.e. parameter combinations) from stage 1A, listed above, were next run 10 times each. The intention had been to take the version with the lowest mean sum of absolute errors as the basis for further parameter space exploration. However, a number of the model versions had a very similar mean sum of absolute errors, and a detailed examination of the different errors indicated that this might not be the best course of action. The relevant figures are given in table #. The GILDED survey urban subsample totals for appliance, space heating and water heating energy demand are 379,109kWh, 3,485,086kWh and 910,897kWh respectively, summing to 4,775,091 kWh as already noted (figures are rounded to the nearest kWh).

Table 7.2. Results of runs during calibration stage 1B.

Identifier	Mean sum of absolute errors	Mean net appliance error	Mean net space heating error	Mean net water heating error
1A:1	260767	-4473	-28080	84624
1A:2	261620	-9170	-66346	82253
1A:3	322227	-8244	-157105	83156
1A:4	246424	-34856	-74153	78717
1A:5	298095	-8940	-134896	79298
1A:6	252522	-13959	-90127	80296
1A:7	357378	10667	-138881	81988
1A:8	340656	1280	-158158	79084
1A:9	302505	-19828	-149066	82281
1A:10	258170	-39233	-76481	82287
1A:11	252477	-1755	-39626	87595
1A:12	260186	-8119	-88242	77187
1A:13	276043	-83031	-72535	83102
1A:14	362093	6619	-157534	77888
1A:15	332007	-37700	-182152	74779
1A:16	297824	-18088	-154268	79986

What leaps out of this table is that all the models overestimate water heating energy demand (and the figure from the GILDED 2010 survey for the Scottish urban subsample is 910,897kWh, so the overestimate is considerable). Model 1A:1 has the smallest magnitude mean net space heating error (the net space heating error for a run is calculated by adding the total space heating demand over all three fuels for the last four quarters of the run, then subtracting the corresponding value for the 2010 survey urban subpopulation, then the mean for this value is taken over all 10 runs), and the third smallest magnitude mean appliance error (calculated in the corresponding way); and it was decided that it would be the best basis for further exploration of parameter space. At the time of writing, the consistent positive errors in water heating energy demand remain unexplained, and further investigations are planned; we could easily get rid of them by reducing the mean size of household, but doing so without understanding their source would be unwise.

7.3 Calibration Stage 2A

Calibration stages 2A and 2B followed the same lines as stages 1A and 1B, but exploring changes in a different set of parameters. Because some of the best parameter sets in Stage 1B (1A:11 and 1A:12) had an increased income relative to the default, this parameter was once again varied, using values of 1.25 and 1.5 those of the model selected as the basis of further exploration (1A:1). Apart from this, all the parameters explored in stage 1 kept the values of 1A:1 for all runs. Instead, four parameters expected primarily to affect non-heating appliances were varied. These were as follows:

- *new-subcategory-appliances-per-step*. A number of appliances that are in a recently introduced subcategories can be added to each household's wish-list; this parameter specifies how many can be added. The default value (that used for 1A:1) is 2; alternative values of 1 and 4 were tried.
- *new-subcategory-steps*. This parameter specifies how long (for how many time steps) a subcategory is considered new. The default value is 4; an alternative of 8 was tried.
- *old-product-steps*. Even if appliances are not broken, the household may decide to replace them. This parameter specifies how many time step an item must have been owned for before this may happen. The default is 4; an alternative of 8 was tried.
- *visits-per-step*. As explained above, when one household "visits" another, it may add an appliance that household owns to its wish-list. This parameter specifies how many such visits a household may make per time step. The default value is 2; alternatives of 1 and 4 were tried.

These combinations of parameters produce 3× 3 × 2 × 2 × 3 = 108 possibilities, each of which was run once. As for stage 1A, we list the best 16 in table 7.3.

Identifier	**income multiple**	**new-subcategory-appliances-per-step**	**new-subcategory-steps**	**old-product-steps**	**visits-per-step**	**Sum of absolute errors**
2A:1	1.5	1	8	8	1	170277
2A:2	1.5	2	8	4	2	180083
2A:3	1	1	8	4	1	184452
2A:4	1	4	8	4	2	190336
2A:5	1.5	1	4	4	4	193651
2A:6	1.5	1	4	4	2	195663
2A:7	1.25	4	4	4	2	195839
2A:8	1.5	1	8	4	1	197511
2A:9	1	4	8	4	1	198347
2A:10	1.5	4	4	8	1	199275
2A:11	1.5	2	8	4	1	200810
2A:12	1	4	8	8	1	202532
2A:13	1	4	4	4	4	203547
2A:14	1	2	8	8	1	203632
2A:15	1.25	2	4	4	1	205597
2A:16	1.25	4	8	4	2	206112

7.4 Calibration Stage 2B

As in stage 1, 10 runs of each of the best 16 versions of the model were run, with the results shown in table 7.4.

Table 7.4. Results from stage 2B.

Identifier	Mean sum of absolute errors	Mean net appliance error	Mean net space heating error	Mean net water heating error
2A:1	238781	-17983	-59849	79658
2A:2	228647	-27075	-77018	87178
2A:3	238408	-37212	-46904	81201
2A:4	240802	-28492	-97324	77579
2A:5	264918	-22888	-125765	78889
2A:6	262452	-28500	-10592	82455
2A:7	244245	-25922	-93634	77262
2A:8	225661	-40441	-68200	75680
2A:9	228349	-38269	-57894	78805
2A:10	249586	-10592	-72245	79138
2A:11	238565	-37995	-62859	77503
2A:12	248063	-19289	-61198	80641
2A:13	229345	-20286	-88356	83328
2A:14	223354	-19453	-31127	84679
2A:15	230607	-34199	-46934	82812
2A:16	241632	-30091	-78228	86059

On this occasion, the version with the lowest sum of absolute errors, 2A:14, had the second lowest mean net appliance errors, and the fourth lowest mean net space heating errors; this was selected as the best version to validate, and if the validation was acceptable, to use for future scenario runs to 2050. Figures 7.1 and 7.2 show more detailed mean results in graphical form. The meaning of the labels on the x-axis, and the figures from the GILDED 2010 Scottish urban subsample in relation to which errors are calculated are as follows:

Appliance electricity (A/E):	328,629 kWh
Appliance gas (A/G):	50, 839 kWh
Space heating electricity (S/E):	159,740 kWh
Space heating gas (S/G):	3,307,148 kWh
Space heating oil (S/O):	18,198 kWh
Water heating electricity (W/E):	73,544 kWh
Water heating gas (W/G):	833,652 kWh
Water heating oil (W/O):	3701 kWh
Total:	**4,775,091 kWh**

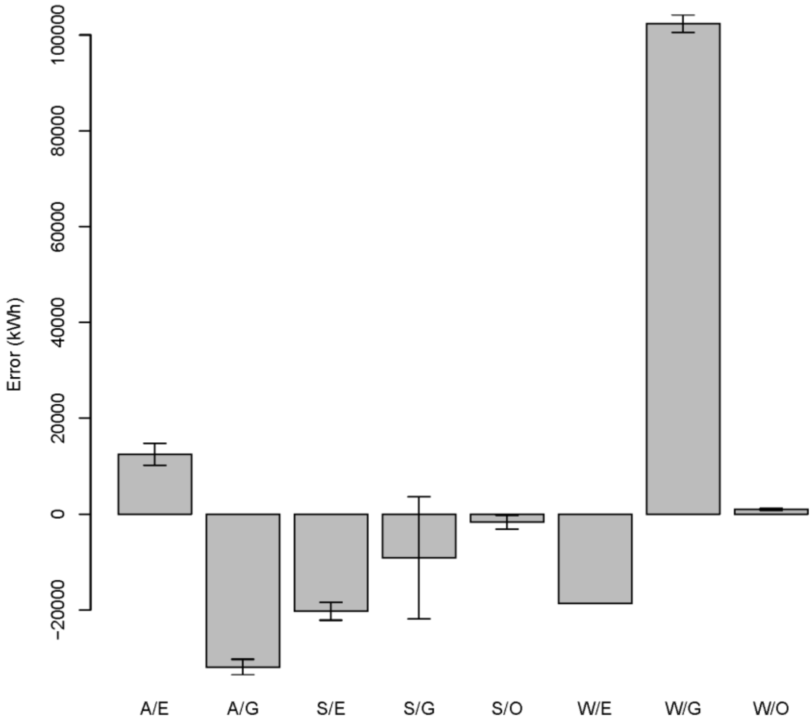


Figure 7.1. Mean net errors of CEDSS model 2A:14 on urban subsample

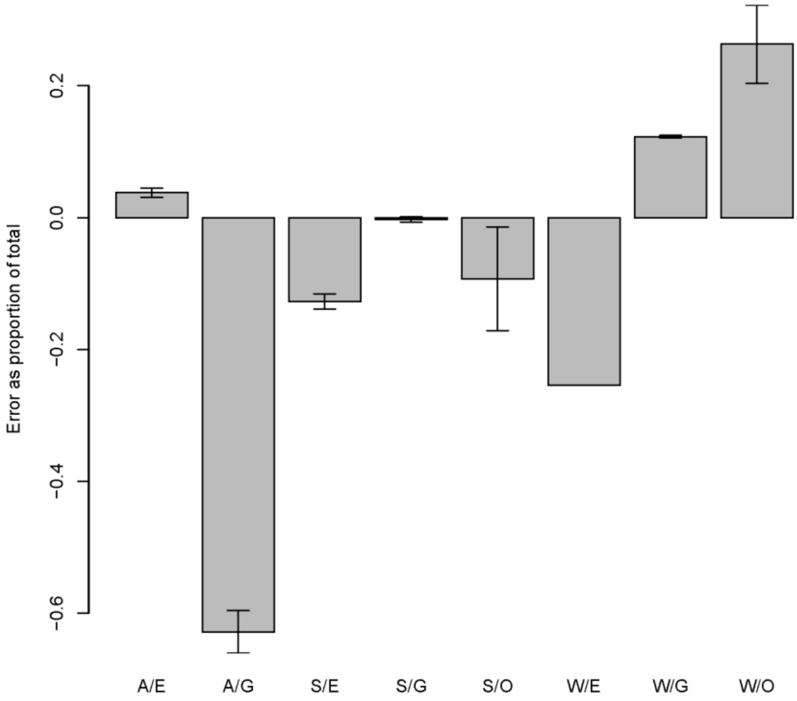


Figure 7.2. Mean proportional errors of CEDSS model 2A:14 on urban subsample

As can be seen, all the net mean errors are small in relation to the total energy demand from the survey. The largest proportional demand by far is for appliance gas, but this makes up only a little over 1% of total energy demand. The systematic overestimate of water heating gas has already been noted.

7.5 Validation on the Rural Subsample

Validation was carried out by running model 2A:14 on the rural subsample of the GILDED 2010 Scottish survey ten times. The mean sum of absolute errors over ten runs was 294,830 kWh, compared with a total energy demand of 6,272,414 kWh. Figures 7.3 and 7.4 show mean net error results in graphical form.

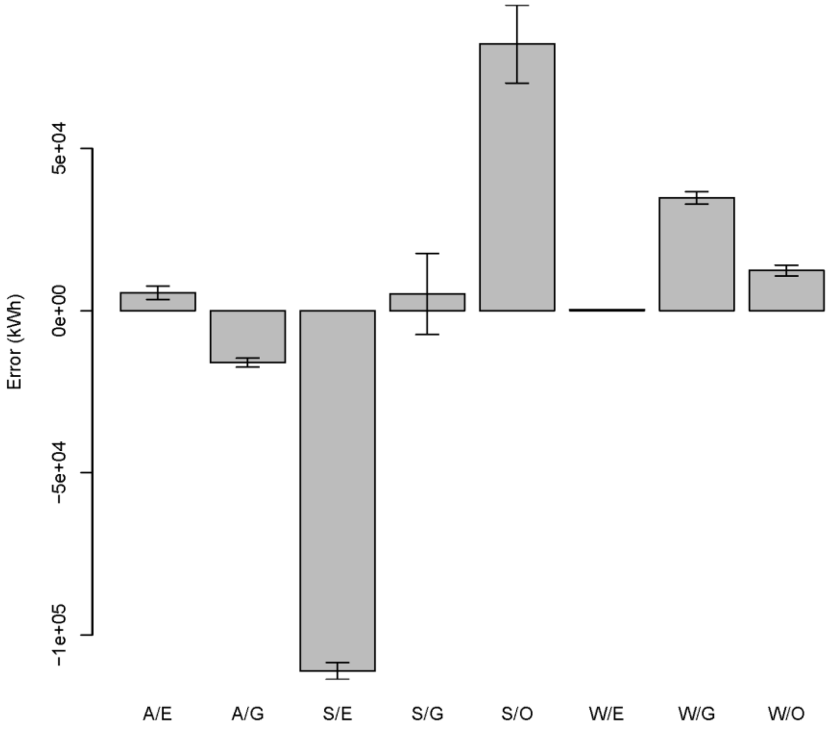


Figure 7.3. Mean net errors of CEDSS model 2A:14 on rural subsample

The figures from the GLDED 2010 Scottish rural subsample in relation to which errors are calculated are as follows:

Appliance electricity:	348,036 kWh
Appliance gas:	29116 kWh
Space heating electricity:	469,836 kWh
Space-heating gas:	1,780,935 kWh
Space-heating oil:	2,751,044 kWh
Water-heating electricity:	365, 088 kWh
Water-heating gas:	100,032 kWh
Water-heating oil:	428,407 kWh

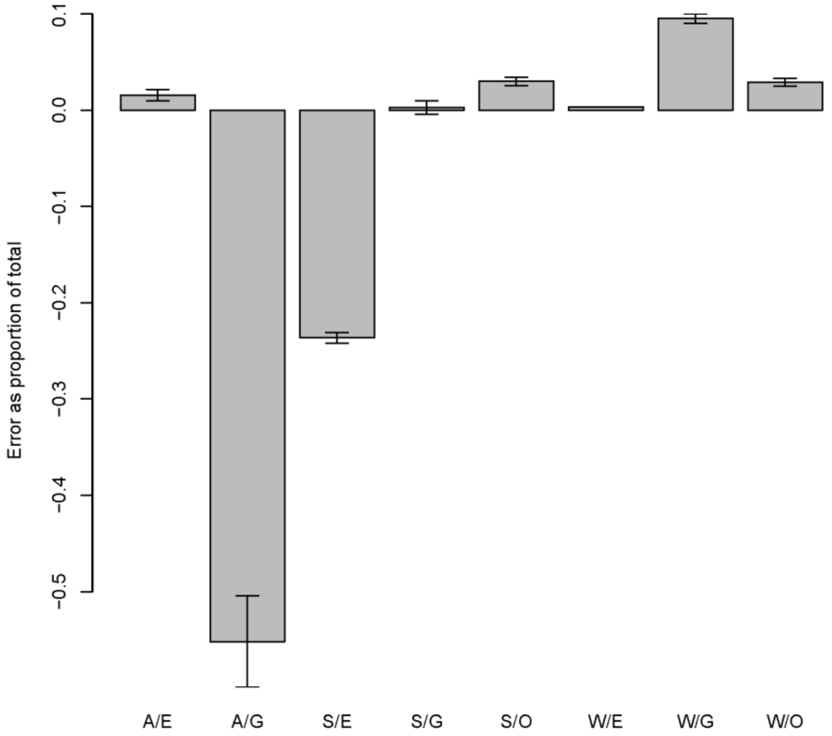


Figure 7.4. Mean proportional erros of CEDSS model 2A:14 on rural subsample

As can be seen there are two relatively large proportional mean net errors, on appliance gas, and to a lesser extent space heating electricity. However, the GILDED survey figure for appliance gas is less than 0.5% of the total, while the error on space heating electricity is almost balanced by that on space heating oil: it is possible that we have counted some households as suing electricity for heating when in fact they use oil, or a mix of both, since if both electricity and oil were noted in the survey as "primary" fuels, we assumed they used oil for space (and water) heating.

Overall, considering that the total energy use for the rural subsample is nearly 1/3 larger than that for the urban subsample, and the balance of fuels is very different, with much more oil used by the rural subsample (many rural households in the sample do not have mains gas), it was considered that the model, while certainly not perfect, was good enough to be used as the basis for future scenarios, considering the large uncertainties that any such scenario modelling must involve.

8 Scenarios to 2050

In applying CEDSS to future scenarios, the main problem is in selecting what aspects of possible futures to study, both with regard to policy decisions, and with regard to possible trends and occurrences over which policy makers have limited or no control. We have defined clusters of scenarios by specific policy measures, and scenarios within those clusters by aspects of the future over which policy-makers control is limited, although not necessarily absent. Before describing these clusters and scenarios, however, some aspects of the models as applied to all clusters and scenarios need to be briefly described.

As with model runs for the period 2000 to mid-2010, discussed in the preceding two sections, we have not as yet attempted to model demographic change, although CEDSS does have the facility to do this. Thus, we have assumed that all urban households have the average size found in the GILDED survey for the urban subsamples, and similarly for rural households in relation to the rural subsample.

With regard to energy efficiency, we have assumed that no significant improvements in boiler efficiency will occur (current condensing boilers are already rated at 90% efficiency or better), and that any improvements in the energy efficiency of televisions will be balanced by increases in size and functionality. For other appliances, we have assumed in all the runs reported here that there will be a gradual increase in efficiency and that this will not be offset by increases in size or functionality. Specifically, an improvement of one energy-rating grade every 15 years is assumed, but this takes place in 5-year steps. In the scenarios where no specific policy measures are taken to discourage the sale of less efficient appliances, households always have a choice between three items in each subcategory, at different prices (see below). For the remainder of 2010, no change occurs in what is available. From then on, a new set of items replaces the old every 5 years. If there are any differences in energy efficiency, the more expensive items are always the more efficient, but there is never more than one grade difference between the most and least expensive items. For 2011-2015, the energy efficiencies (and the ratings indicating them) are set on the basis of what is available in the Spring/Summer 2010 Argos catalogue. Thereafter, an improvement of one rating step occurs every five years at one of the three price levels. For example, in 2011-2015, the most expensive and mid-range freezers are energy-rated A, the cheapest B. In 2016-2020, all are rated A, in 2021-2025 the most expensive is rated A+, and in 2026-2030 both the most expensive and the mid-range freezer are A+, while only the cheapest is rated A. Energy demands corresponding to the different ratings are taken from Department of Energy and Climate Change (2009), as before.

Prices and incomes only have significance in relation to each other. We have chosen to keep appliance prices stable, and vary fuel prices, and household income available for spending on the goods and fuels included in the model in relation to these. Appliance prices other than for heating systems are set at the median and lower and upper quartiles of the prices in those subcategories for the last three years of the 2000 to mid-2010 runs. Prices of heating systems, and improvements to insulation, are stable unless altered as a result of a policy decision, as described below.

Thus far, our investigations have focused on four variable features of possible domestic energy futures:

1. Household incomes. We have examined scenarios where these are stable (in relation to appliances prices), and where they are rising at 2% per annum. Of course, household incomes – and particularly their distribution across income levels – are affected by policy decisions; but these are not in general aimed at altering energy demand, so we can reasonably regard them as exogenous to CEDSS's domain of interest.
2. Fuel prices. We have examined scenarios where these are stable (relative to appliance prices), or increase at either 2% or 4% per annum. As with incomes, fuel prices are of course affected by policy decisions, and in this case, policies on taxation and/or subsidy may be aimed at reducing (or increasing) energy demand; although of course these prices are also affected by factors beyond the control of policy makers. The different schedules for incomes and fuel prices combined define six families of "income-fuel-price scenarios".
3. Subsidisation of boiler replacement and insulation measures. By default, we have assumed that prices of boilers and insulation measures remain the same. Alternatively, we have assumed that these prices are subsidised from 2015, falling by 30% at that date. This required use of a parameter file not used in the 2000-mid-2010 runs: the insulation update file. There is one line in this file for each update. An update can be one of three options: removing an upgrade option, adding an upgrade option, or changing the cost of an upgrade option. The file used in the runs reported here begins:
   ```
   step,command,dwelling-type,from-state,to-state,cost
   60,change,bung-det-cavity-1,minimum,dg,2590
   60,change,bung-det-cavity-1,minimum,loft270,350
   60,change,bung-det-cavity-1,minimum,wi,350
   60,change,bung-det-cavity-1,loft100,dg-loft100,2590
   60,change,bung-det-cavity-1,loft100,loft270,350
   ...
   ```
4. Regulation of the energy efficiency of appliances. By default, we have assumed no such regulation. Alternatively, we have assumed that once sufficient choice is available of a particular type of appliance at or above a particular energy-rating, appliances with lower ratings are no longer allowed to be sold. Specifically, once the mid-priced item of the trios described above reach a given rating, the cheaper and lower-rated appliance is excluded from the market. Subsidisation and regulation together define four clusters of “policy scenarios”.

In total, we thus have 24 scenarios, grouped in four policy-defined clusters, which cut across six income-fuel-price defined families. We ran each of the 24 scenarios 16 times each for both urban and rural subsamples. Detailed statistical analysis of the results remains to be done, and will be reported in papers to be submitted in the next few months, but the following figures display some interesting outcomes in graphical form.

First, we show plots of the energy demand in 2050 in the urban scenarios, for appliances, and for space and water heating combined. Figure 8.1 colour-codes the clusters, and shows members of different families by different symbols, while figure 8.2 reverses this.

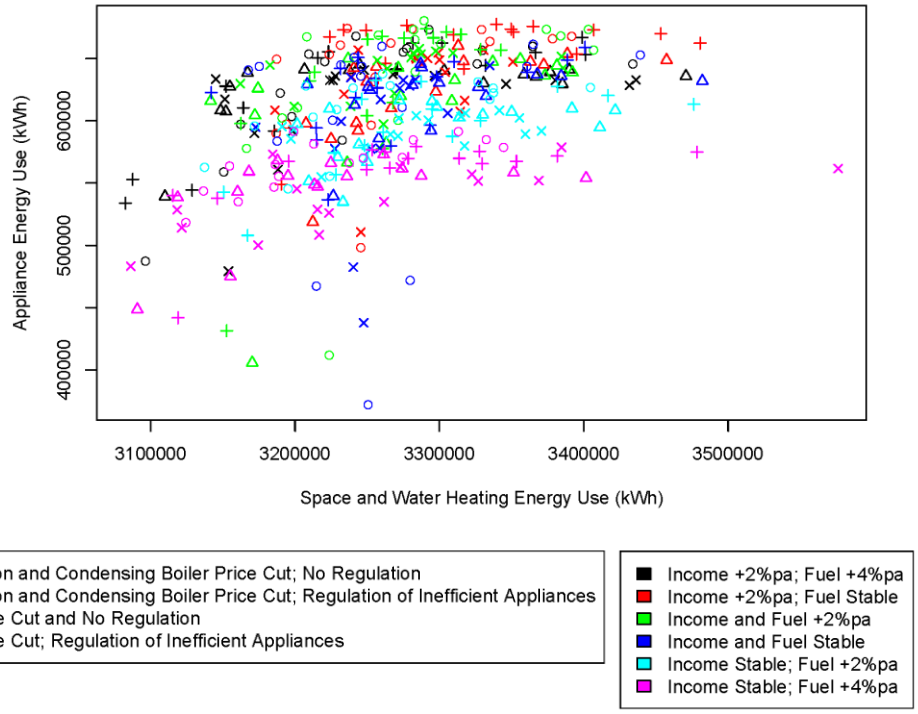


Figure 8.1. Urban scenarios to 2050, coloured by family

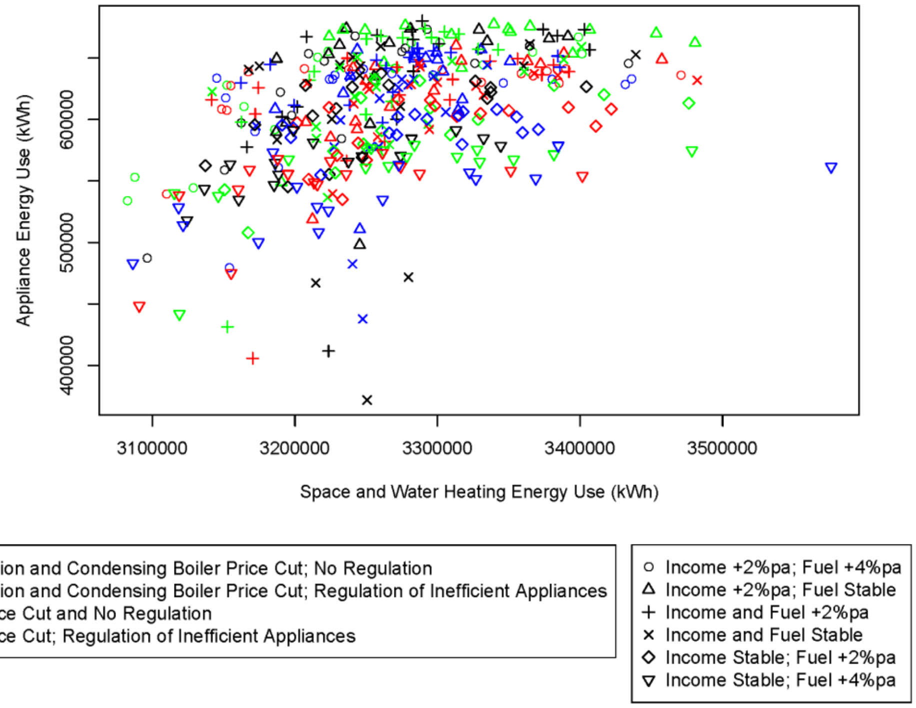


Figure 8.2. Urban scenarios to 2050: coloured by policy cluster

At a glance, it appears that, although there are six colours in figure 8.1, denoting income-fuel-price families, and only four in figure 8.2, denoting policy clusters, the colours are more spatially distinct in the former, implying that differences in income and fuel-price trajectories have made more difference to the outcome than the policy changes. From figure 8.1, it also appears that income and fuel price changes have made more difference to energy demand for appliances than for space and water heating: this might be expected, as the former depends heavily on what appliances have been bought, which in turn depends on the funds available to buy them, which will be greater not only if there is more income, but also if fuel prices are lower. Thus, as we might expect, the red symbols (fuel prices stable, incomes rising at 2% per annum) cluster towards the top of the plot (high applian ce energy demand), while the cyan (fuel prices rising by 4% per annum, incomes stable) cluster toward the bottom. Also worth noting is the multicoloured scatter of symbols to the lower left on both plots: runs which resulted in relatively low overall energy demand, which are a mix of both families and clusters of scenarios. These, we surmise, result from the chance choices of biospheric values by an unusual number of households at some point in the scenario, leading via the "habit adjustment" and social influence mechanisms in the model to a population of households with unusually biospheric values.

We now look at some individual clusters and families of urban scenarios. Figure 8.3 singles out scenario runs from the cluster with no policy initiatives: the effects of different income and fuel price trajectories is quite clear. The other three clusters are similar.

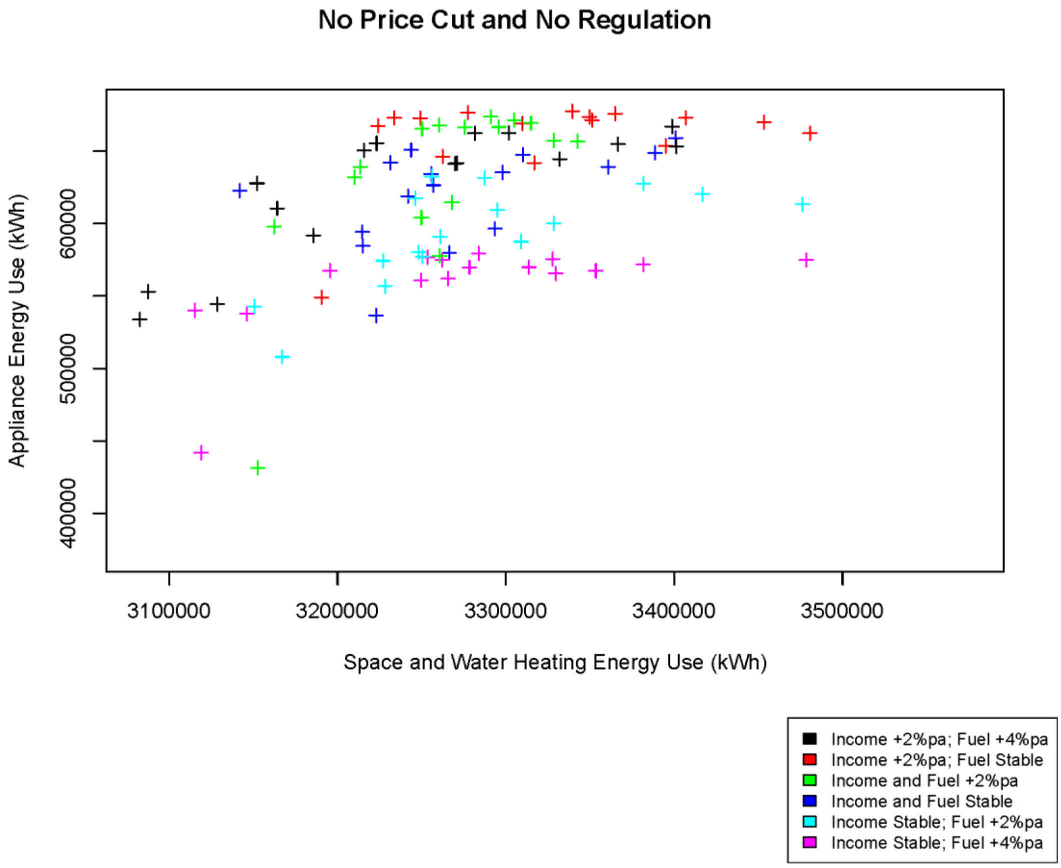


Figure 8.3. Scenario runs from the cluster with no policy initiatives.

By contrast, different income-and-fuel-price families of scenarios show rather different patterns from each other.

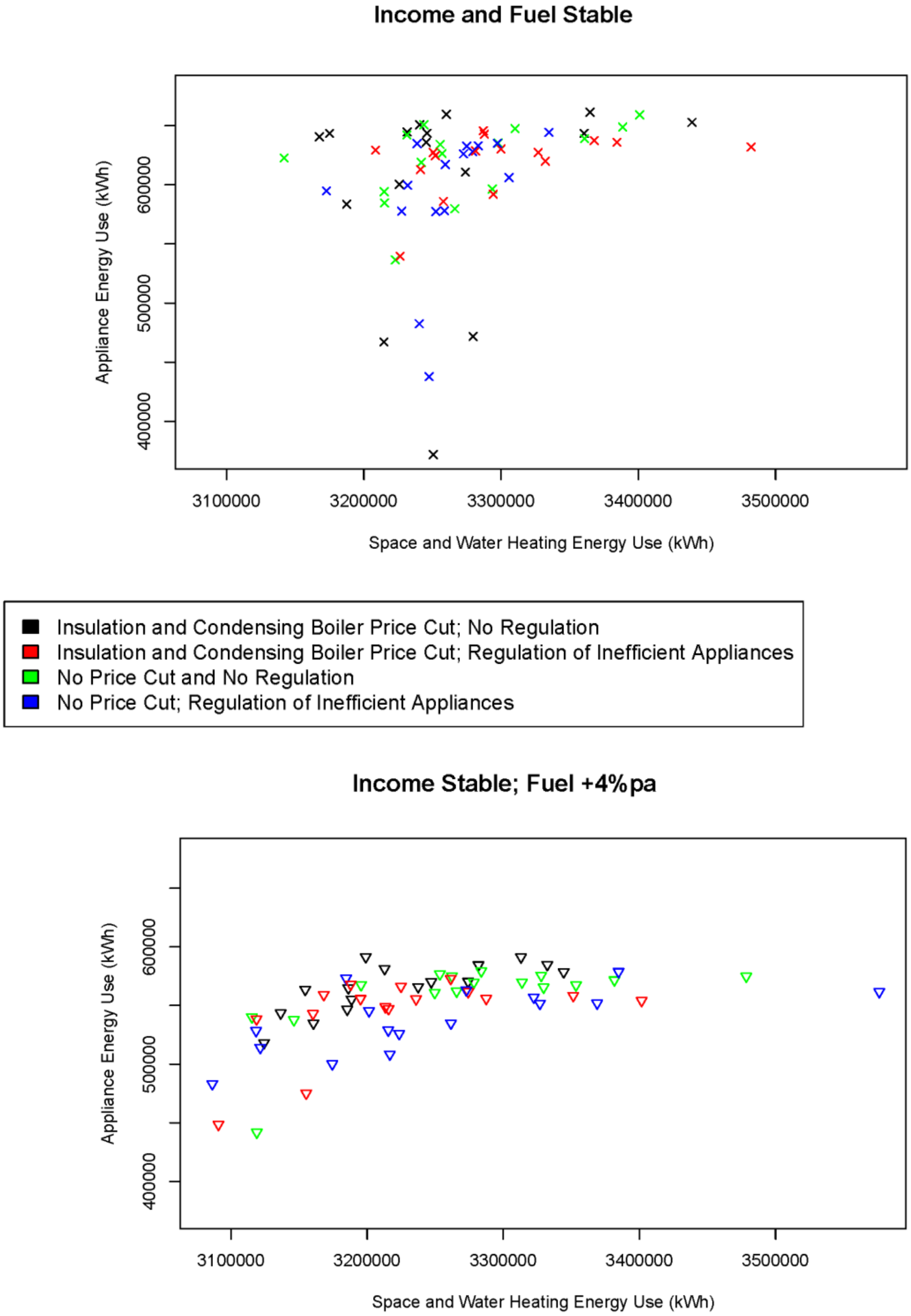


Figure 8.4. Top graph: results from runs with income and fuel prices stable; bottom graph: results from runs with income stable and fuel prices increasing by 4% per annum.

When income and fuel prices are stable, there is very little apparent separation between the different policy cluster runs, as shown at the top of the figure. By contrast, when income is stable but fuel rises in price at 4% per annum, as shown in the lower part of figure 8.4, separation of policy scenarios is apparent, particularly if red and blue (regulation of inefficient appliances) are considered together in contrast to the green and black (no such regulation). In none of the families does such a clear separation appear in the horizontal direction, where we might have expected black and red (subsidies for installing condensing boilers and insulation) to have clustered to the left (low demand for space and water heating) and green and blue (no such subsidies) to the right.

We can also look at the trajectories of energy demand over time. All show fairly similar patterns, with heating energy demand falling more or less continuously, while appliances energy first rises gradually, then more sharply, then falls.

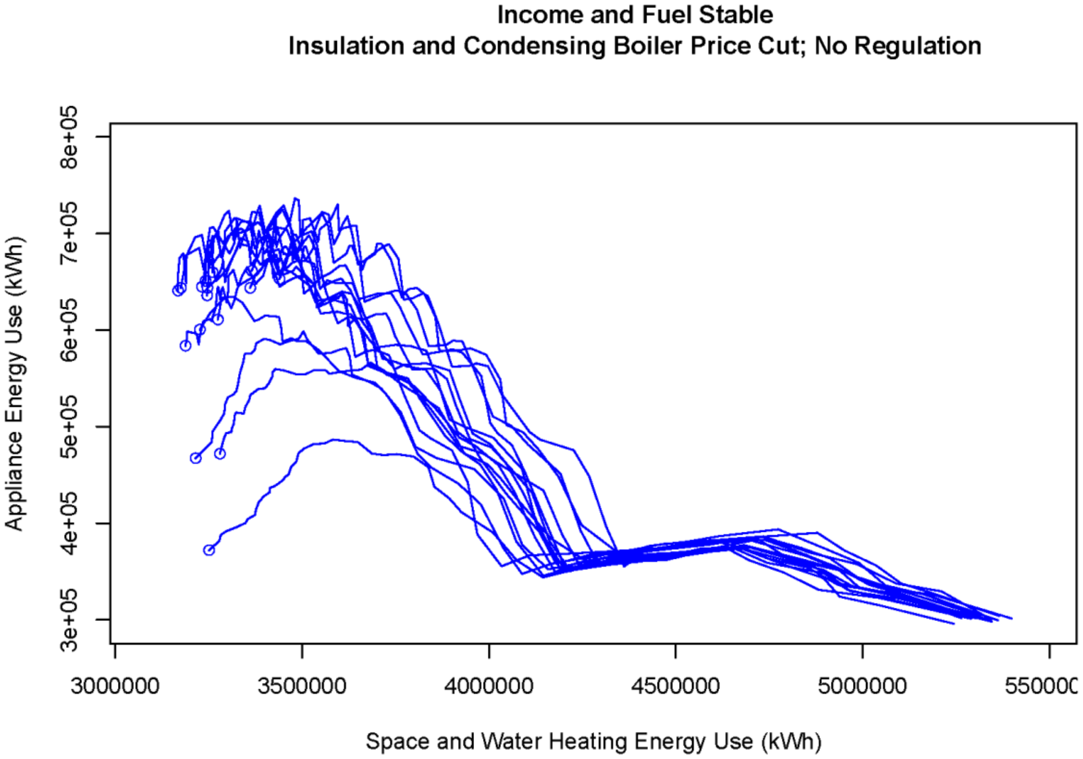


Figure 8.5. Time series trajectories of heating energy consumed against appliance energy consumed.

The 16 trajectories shown here all start at the lower right. This set was chosen because it shows three clear outliers, ending at significantly lower points than the main cluster: possible examples of a number of chance decisions to follow biospheric values at a particular point in time setting the entire community of households on a lower-demand path through habit and social interaction effects; it is worth noting, nevertheless, that in qualitative terms they show a similar pattern to the rest of the runs, with heating energy demand falling throughout, while appliance energy demand rises and then falls. The overall change from start to finish – of falling heating energy demand partially offset by rising appliance energy demand – reproduces that seen in UK statistics in recent years. The fall in appliance energy demand may result from households hitting the limits on the number of appliances they can possess in each category, together with the absence of new kinds of appliances in our scenarios.

In most respects, patterns in the rural scenarios were quite similar to those in the urban ones, as can be seen by comparing figure 8.6 with figure 8.1. However, the rural scenarios show noticeably more divergence in heating energy demand, as can be seen in figure 8.1: almost all the urban scenarios show final year heating energy demand between 3,100,000 kWh and 3,500,000 kWh, while the rural scenarios are spread somewhat more evenly, and range from 4,100,000 kWh to over 4,700,000 kWh. The reason for this difference is not known.

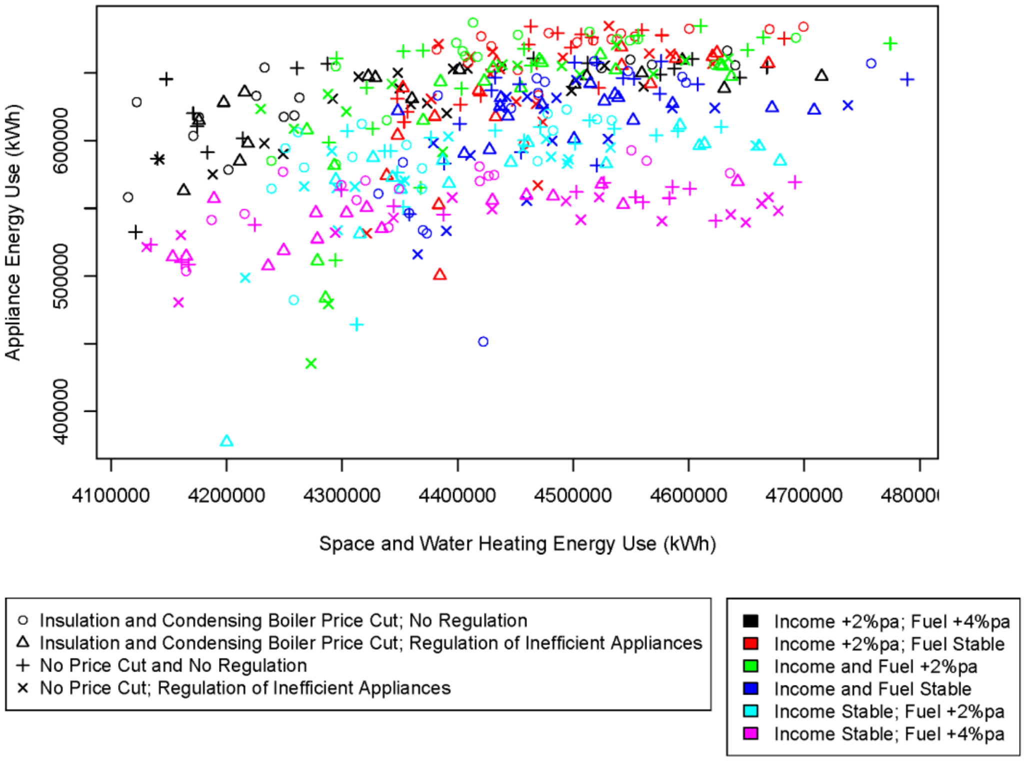


Figure 8.6. Rural scenarios to 2050, coloured by family.

9 Conclusions and Future Work

We consider that the agent-based modelling component of GILDED has demonstrated that the technique has a significant contribution to make to the investigation of behavioural issues in domestic energy demand, and in particular of possible policy approaches to reducing such demand by encouraging changes in decision-making about the purchase of energy-using and energy-saving equipment.

The CEDSS model has reproduced, without them being coded in in any explicit way, the major trends in domestic energy use in the UK over the past decade, notably the reduction in energy used to heat the home as a result of the installation of more efficient boilers and better insulation; and the partial offsetting of this improvement by the increase in the number of electrical appliances bought for and used in the home. The future scenarios we have run indicate that this trend is likely to continue, although the extent to which it does so will be influenced both by factors largely beyond the control of those policy-makers whose focus is on energy demand, such as household incomes and fuel prices, but also by policy decisions. The policy initiative we have investigated so far are relatively modest, but the progressively tighter regulation of inefficient household appliances appears able to make a noticeable difference, particularly in scenarios where fuel prices rise in relation to household incomes – which seems likely to be the case in the coming decades. It was not clear that subsidising the purchase of condensing boilers and insulation made a difference; this may be because most of our households had already taken the cheaper insulation measures, and did not – as indeed, most households in reality probably do not – replace their heating systems until they break down, when they must of necessity be replaced, and few adopt the more expensive insulation measures, such as wall insulation.

We are not aware of any previous model that has attempted to model household decision-making processes in this area. The process of designing and implementing such a model in itself has revealed many of the complexities of how people think about their domestic energy use, and energy using and energy saving equipment, as described in section 3 above. It has also perhaps clarified how agent-based modelling can complement more established approaches to social science, which in this area tend to focus on what people say about their energy use and its relationship to their values, as opposed to hard data about the decisions they make and their medium to long-term consequences. While the survey data gathered in other GILDED workpackages has been essential to our modelling work, in order to implement a model that could plausibly tell us something about the future, we needed to be able to model the recent past, and change over that period, as well as the present; and in order to do that, we needed a wide range of quantitative data about that recent past: energy prices, household incomes, heating systems, insulation measures, the prices and properties of household appliances. This information is not readily available in convenient forms and formats, and we were obliged to make more assumptions than we would have liked. Where we were unable to access any relevant data – as in the case of the influence of social contacts on purchase decisions – we have been obliged to experiment with different parameter settings to calibrate those aspects of our model. The fact that we were nevertheless able to construct a version of the model that produced outcomes in the present that were quite close to those indicated by the GILDED survey we regard as a vindication of the agent-based modelling approach.

Use of the CEDSS model is written in to work to be done at the James Hutton Institute for the Scottish Government, specifically in relation to pathways to a "low-carbon rural economy". In the immediate future, we will be continuing to explore and analyse the scenarios described in the preceding section, and writing these analyses up for publication in peer-reviewed journals. We will also be extending the range of scenarios explored, examining a wider variety of trajectories for household income and fuel prices, the possible introduction of new types of household appliance, and additional policy measures. The UK government is currently deciding whether or not to oblige householders to improve insulation at the point at which a new boiler is installed, or windows are replaced; and this kind of compulsion might have a more definite effect than we have found for the use of subsidies. We will also be attempting to identify the factors responsible for those aspects of the current version of CEDSS (defined by the 2A:14 set of parameters) that failed to match the survey results, notably the overestimate of energy demand for water heating relative to those results, and the underestimate demand for gas for household appliances (i.e., cookers).

The longer-term intention is to use CEDSS in "backcasting" mode: specifying a current state, and a desired future state (in this case in 2050, the date the Scottish Parliament has set for a highly ambitious 80% greenhouse gas emission reduction target (Climate Change (Scotland) Act, 2009)), and looking for feasible transitions from one to the other. We also plan to make use of the facilities CEDSS already has to model demographic change, although this will require the collection and integration of a new class of real-world data, drawing on the UK census and a range of local government statistics on the composition and activities of households in Aberdeen and Aberdeenshire. This work will be integrated with further empirical social science work on Scottish household energy-related values, attitudes and behaviours, and ways of influencing them.

10 References

Aberdeen City Council (2007) *Home Energy Conservation Act 1995: Fifth Progress Report*. Aberdeen City Council.

Alcamo, J. (2001) Scenarios as tools for International Environmental Assessments. (Environmental Issue Report No. 24, Experts' Corner Report: Prospects and Scenarios No. 5). European Environment Agency, Copenhagen.

Bankes, S. (2002) Tools and Techniques for Developing Policies for Complex and Uncertain Systems, *Proceedings of the National Academy of Sciences*, 99, pp. 7263-7266.

Banks, J., Smith, Z. and Wakefield, M. (2002) *The Distribution of Financial Wealth in the UK: Evidence from 2000 BHPS Data*. The Institute for Fiscal Studies.

Barthelemy, O. (2007) Untangling Scenario Components with Agent Based Modelling: an Example of Social Simulations of Water Demand Forecasts. *PhD thesis, Centre for Policy Modelling, Manchester Metropolitan University*.

Bin, S. and Dowlatabadi, H. (2005) Consumer lifestyle approach to US energy use and the related CO2 emissions. *Energy Policy* 33: 197–208.

Cambridge Architectural Research, Cambridge Econometrics, Roger Talbot & Associates Ltd, Alembic Research (2009). modelling Greenhouse Gas emissions for Scottish Housing: Final Report. Scottish Government Social Research.

Department of Energy and Climate Change (2009). *Act on CO_2 Calculator Version 2.0: Data, Methodology and Assumptions Paper*. United Kingdom Department of Energy and Climate Change.

Edmonds, B. and Barthelemy, O. (2002) Domestic water demand and social influences: an agent-based modelling approach. *CPM Report No: CPM-02-103, Centre for Policy Modelling, Manchester Metropolitan University*.

Gotts, N. (2009) ABMED: A prototype model of energy demand. *Sixth Conference of the European Social Simulation Association, University of Surrey, Guildford, Surrey, 14-18 September 2009*.

Gotts, N.M., Polhill, J.G. and Law, A.N.R. (2003). Agent-based simulation in the study of social dilemmas. *Artificial Intelligence Review,* 19, 3-92.

Grimm, V., Berger, U., Bastiansen, F., Eliassen, S., Ginot, V., Giske, J., Goss-Custard, J., Grand, T., Heinz, S. K., Huse, G., Huth, A., Jepsen, J. U., Jørgensen, C., Mooij, W. M., Müller, B., Pe'er, G., Piou, C., Railsback, S. F., Robbins, A. M., Robbins, M. M., Rossmanith, E., Rüger, N., Strand, E., Souissi, S., Stillman, R. A., Vabø, R., Visser, U. and DeAngelis, D. L. (2006) A standard protocol for describing individual-based and agent-based models. *Ecological Modelling* 198: 115–126.

Grimm, V., Berger U., DeAngelis, D. L., Polhill, J. G., Giske, J. and Railsback, S. F. (2010) The ODD protocol: A review and first update. *Ecological Modelling* 221: 2760-2768.

Gutiérrez, E., Adenso-Díaz, B., Lozano, S. and González-Torre, P. (2010). A competing risks approach for time estimation of household WEEE disposal. *Waste Management* 30: 1643-1652.

Horsfield, G. (2011) Family spending: A Report of the 2010 Living Costs and Food Survey. United Kingdom Office for National Statistics.

LaLonde, W. and Pugh, J. (1991) Subclassing ≠ subtyping ≠ is-a. *Journal of Object-Oriented Programming* 3(5): 57-62.

Lansing, J.S. and Kremer, J.N. (1993) Emergent properties of Balinese water temple networks: coadaptation on a rugged fitness landscape. *American Anthropologist* 95:97-114.

Lindenberg, S., and Steg, L. (2007) Normative, gain and hedonic goal-frames guiding environmental behavior. *Journal of Social Issues*. 63 (1): 117-137.

Mäenpää, I. (2005) Analysis of environmental impacts of consumption in Finland. In Hertwich, E. G., Briceno, T., Hofstetter, P. and Inaba, A. (eds.) *Norwegian University of Science and Technology, Industrial Ecology Program: Trondheim Vol.2005/1*. pp. 1-21.

Owen P. (2006) *The rise of the machines: a review of energy using products in the home from the 1970s to today*. Energy Saving Trust. Available online: URL http://www.energysavingtrust.org.uk/Publications2/Corporate/Research-and-insights/The-rise-of-the-machines-a-review-of-energy-using-products-in-the-home-from-the-1970s-to-today.

Palmer, J. and Cooper, I. (2011). *Great Britain's Housing Energy Fact File 2011*. United Kingdom Department of Energy and Climate Change.

Polhill, J. G. and Gotts, N. M. (2009). Ontologies for transparent integrated human-natural systems modelling. *Landscape Ecology* 24 (9): 1255-1267.

Polhill, G., Galan-Diaz, C., Gotts, N. M., Craig, T., Marshall, K., Sutherland, L.-A., Kriel, A. and Fischer, A. (2010) The ODDness of modelling: early experiences from a transdisciplinary modelling exercise. *Third World Congress on Social Simulation, University of Kassel, Kassel, Germany, 6-9 September 2010*.

Ramanath A. M. and Gilbert N. (2004) The design of participatory agent-based social simulations. *Journal of Artificial Societies and Social Simulation* 7 (4): 1.

Schreinemachers, P. and Berger, T. (2006) Land-use decisions in developing countries and their representation in multi-agent systems. *Journal of Land Use Science* 1(1): 29–44.

Weber, C. and Perrels, A. (2000) Modelling lifestyle effects on energy demand and related emissions. *Energy Policy* 28: 549-566.

Wilensky, U. (1999) *NetLogo*, http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Appendices

Appendix 1: Description of Workpackage 5 from GILDED Grant Agreement

Work package number	**5**	**Start date or starting event:**		**Month 2**	
Work package title	**Agent-Based Modelling**				
Activity Type	**RTD**				
Participant number	**1**	**2**	**3**	**4**	**5**
Personmonths per participant:	**13**	**1**	**1**	**1**	**1**

Objectives

- To integrate empirical findings from the case studies (WPs 2, 3 and 4), relevant theories and knowledge in a formal framework as an ontology.
- To use the ontology, together with scenarios of policy measures, technology methods and attitude changes from WP4, to develop agent-based models exploring trajectories of possible transitions to low-carbon economies in each case study, for the period to 2050.
- To provide results from the agent-based models for WP6 and for dissemination to WP1.
- To test the ontology-based modelling platform and associated model-development methodology using real-world case studies.

Description of work (possibly broken down into tasks), and role of participants

The work in WP5 essentially involves the incremental development of the agent-based model CEDSS with associated ontologies, and simulation experiments with CEDSS to analyse scenarios of interest to other workpackages (e.g. WP4 and WP6). CEDSS will be developed in consultation with the case study teams (WPs 2 & 3), initially using a prototype model developed early on in the project as a basis to facilitate discussion. Following that, the major development of CEDSS will take place, focused initially on the Scottish case study, hence the resulting model is called Scot-CEDSS, though it is expected to have wider applicability. Finally, minor alterations will be made to Scot-CEDSS in consultation with the other case study teams to produce full-CEDSS. In detail, the tasks are as follows:

Tasks

T 5.1 Develop a prototype for CEDSS (proto-CEDSS) using the ontology-based framework. (MLURI)

T 5.2 Present proto-CEDSS to the Scottish case-study team and WP3 leaders, with a view to exchanging ideas on information required from the case-studies to provide empirical information for CEDSS, and to gather information needed to define the Domain Ontology on which CEDSS will be based. (MLURI, RuG)

T 5.3 Ontology development (MLURI, WP4 & 6)

T 5.4 Develop and validate the first version of full CEDSS (Scot-CEDSS), with a view to implementing it on the Scottish case study. The approach will involve a process of incremental versioning, liaising closely with the Scottish case study team, both over the structure of the model, and emerging requirements for empirical data. (MLURI)

T 5.5 Design simulation experiments to use with Scot-CEDSS/full-CEDSS, create the associated scenario ontologies, and conduct the necessary simulations and gather results. (MLURI, WP4 & 6)

T 5.6 Demonstrate Scot-CEDSS to the other case study teams (at the second consortium meeting) and gather feedback for making the minor alterations needed for the final version of CEDSS (full-CEDSS), and any changes to the domain or framework ontologies. (MLURI, all)

T 5.7 Develop full-CEDSS and incrementally validate it, making appropriate minor alterations to Scot-CEDSS agreed in the meeting. (MLURI)

T 5.8 Dissemination of results of simulation experiments and ontology-based model development methodology. (MLURI)

Deliverables (brief description and month of delivery)

D11 Final report, covering experiments with full-CEDSS and summarising work done in this WP. (month 36).

Milestones

M 5.1 Meeting with RuG to exchange information on CEDSS (month 3)

M 5.2 Meeting (consortium) to demonstrate Scot-CEDSS and decide full-CEDSS (month 18)

M 5.3 Experiments with Scot-CEDSS (month 24)

M 5.4 Experiments with full-CEDSS. (month 32)

Appendix 2: Derivation of ontological concepts from each workshop

Workshop A, 19 April 2010

Post-it	Nominal grouping	Ontological type: Concept, Data Property (attribute), Object Property (relation), Process	Notes	What should/could be done with it if it doesn't directly fit such a type
Cold Winter	Context/Influence	-	-	Suggests 'Weather' as a concept, with 'Temperature' as an attribute. Also suggests a context for a scenario (i.e. an exogenous driver).
What will people be asked to do differently in their lives?	Context/Influence	-	-	Suggests a process, which will be determined by scenario (e.g. a particular intervention or set of interventions). The process will thus be an ontological change of some sort (e.g. to regulation) or some less substantial thing like a 'Communication'.
Influence of policy on behaviour	Context/Influence	-	-	Suggests a process, which translates an intervention into rule changes for individuals
Change in government	Context/Influence	-	-	This could be a driver of what people will be asked to do differently…
How much do people care about it [the policy?]	Between People and Context/Influence, and linked to Outcome	Relations between people and policy	This doesn't capture 'how much?'	
Aging society	Between People and Context/Influence	-	-	Suggests attribute 'Age', and individualised representations of population-level attributes 'birth-rate' and 'death-rate'. Also suggests that changes in these population-level attributes need to be representable as part of a scenario.
Different types of households, e.g. pensioners (fuel poverty)	People	Concept 'Household' and various subconcepts. (Including 'Household in Fuel Poverty')	Is a pensioner a subclass of household or of person?	The fuel poverty part suggests Income as an attribute of a Household.
Energy users	People	Concept 'Energy User'	Should person be a subclass of Energy User?	
Person → Family → Neighbourhood	People	Concepts 'Person', 'Family', 'Neighbourhood' and relationships between them e.g. Person 'memberOf' Family. Person 'locatedIn' Neighbourhood.	What is the link between Family and Household?	
Consumers	People	Concept 'Consumer',		

Post-it	**Nominal grouping**	**Ontological type: Concept, Data Property (attribute), Object Property (relation), Process**	**Notes**	**What should/could be done with it if it doesn't directly fit such a type**
		or perhaps process of 'consumption'.		
Energy Prices	Drivers and Outcome	Attribute of supplier		
Market	Drivers and Outcome	Process of choosing supplier and/or consumer goods	Depending on the scale modelled, consumer influence on energy prices in the market place may not be a feedback loop we can introduce.	
Rising fuel prices	Drivers and Outcome	-	-	Part of a scenario, a driving variable for energy prices.
Equitable fuel pricing	Intervention	-	-	Suggests some sort of process, which may be scenario driven.
Access to technology	Link between People and Options	Attribute 'cost' of consumer goods.	Another possibility is to geographically locate access. (e.g. broadband is only available in certain areas, restricting who can work from home; other goods/services may also be difficult to access in remote areas and/or may be made available in different areas at different times.)	-
Available options	Options	Relationship of Person to Consumer Goods and Energy Suppliers (also to Employer?)	This would probably be derived from other variables.	
Goods	Options	Concept		
New gadgets	Options	Concept	Also suggests process of innovation of new goods.	Could be scenario driven
Cars	Options	Concept		
Energy suppliers	Feed into Options	Concept		
Market stimulation (+ve and –ve)	Feed into Options	Process		Could be scenario driven—manner of implementation would be scenario-dependent.
Standards of living	Link between Outcome and Context and Influence	Attribute of household (or person?)	Should be derived from other variables—would this be a subjective or objective term—i.e. is standard of living something that is measured against standard criteria, or a question of individual perception based on their own criteria (and possibly relative to peers)?	
How many people does it affect? (the policy)	Outcomes	Relationship Person 'affectedBy' Policy	Would have to be derived from other variables.	
Cost of policy	Outcomes	Attribute of Policy	-	
Savings overall	Outcomes	Global attribute 'total energy use' (assuming that is what was being considered)		

Post-it	**Nominal grouping**	**Ontological type: Concept, Data Property (attribute), Object Property (relation), Process**	**Notes**	**What should/could be done with it if it doesn't directly fit such a type**
How much does it cost me?	Outcomes	Concept Individual Policy Cost with attribute Cost and relationships hasPerson and hasPolicy	If cost is not zero, then Person has been affectedBy Policy	
How does the energy policy help or hinder other policies?	Outcomes	Concept Policy, Subconcept Energy Policy, relationships 'helps' and 'hinders'	The relationships would have to be derived	Scenarios to determine which policies are included.
Where do people live and work?	Constraint and Infrastructure	Attributes of Person		Also suggests Concept 'Job' with attribute 'locatedIn' and relationship to Person. Not to mention processes of 'commute' and job and person selection. (N.B. some jobs are not necessarily locatedIn a specific geographical point, e.g. travelling salesman, cab-driver—though they may have a region.)
Working from home/policies regarding workplace travel	Infrastructure	-	-	Suggests various things. 'option to work from home' as attribute of Job, process of deciding to work from home (which would depend on availability of resources to do so). Policies regarding workplace travel—is this company policy or government? And are we talking about preferred modes of transport?
Houses/Dwellings	Infrastructure	Concepts		
Upgrading existing housing stock	Infrastructure	Process affecting attributes of Houses/Dwellings	Which ones?	Scenario driven.
Building regulation enforcement	Infrastructure	Process		Presumably this note is partly about the extent to which the enforcement takes place.
Insulation	Infrastructure	Attribute of Houses/Dwelllings; Process; Subconcept of Consumer Goods.		

Workshop B, 23 April 2010

Index card	Category	Ontological type: Concept, Data Property (Attribute), Object Property (Relation), Process	Notes	What should/could be done with it if it doesn't directly fit such a type
Owner occupier	Household context	Concept	Not a subclass of Household context—rather, an aspect of it	-
Private tenant		Concept		-
Social housing		Concept	Does this refer to the housing, or to the occupier?	
Access to e.g. District Heating		Relation of house to option	Links to 'options' from the previous meeting. Access in this sense is geographical	Suggests District Heating as an option, but there are other options…
Gas/off gas		Attribute of location		
Rural/urban		Attribute of location		
Access to finance, time and knowledge		Attribute of occupiers (I assume) or a relation of occupier to finance, time, and knowledge if explicitly represented		
Age of occupants and structure of household	Occupier(s)	Age is an attribute of a person		Not clear how to handle household structure—does this refer to who is living in the house and how they are related?
Gordon		Instance of person		
Pets		Concept		
Access to resources: knowledge, time, finance		Attributes or relations (see above)		Duplicate card? (Deliberate in order to put it under two categories, or a mistake of transcription from post-its?)
Reduce demand (behavioural)	Values	Process?	Population scale? Intervention?	
Re-using		Process		
Embodied energy	Between Values and Location	Attribute	What of?	
Location of house in relation to weather	Location	Relation of house to location, and location to weather (or attribute of location)		
Energy/CO2 embodied in house		Attribute of house		
Maintenance and repairs		Processes		
Materials, internal walls etc.	Construction Type	Attributes of house		
Location of home for travel needs	Travel	Relation of location to travel options	This might need to be broken down a bit	
Travel		Process		
Personal transport		Concept	This strikes me as an abstract concept	
Car		Concept	Subclass of personal transport?	
Bicycle		Concept	Subclass of personal transport?	
Visitors getting to you (transport)		-	-	Suggests processes of visiting and being visited, but also how they do so, and

Index card	Category	Ontological type: Concept, Data Property (Attribute), Object Property (Relation), Process	Notes	What should/could be done with it if it doesn't directly fit such a type
				constraints on that.
Water heating	Water/Water Heating	Process		
Hot water		Concept		
Shower		Concept	Could also be a process	
Bath		Concept	Could also be a process	
Waste water		Concept		Suggests processes that produce it
Water usage		Process or attribute of a process	How to represent attributes of processes?	
Insulation	Energy Efficiency Measures	Process, concept or attribute		
Double Glazing		Process, concept or attribute		
Draught proofing		Process, concept or attribute		
Smart meters		Process, concept or attribute		
Ventilation		Process, concept(?) or attribute		
Heating	Temperature control	Process		
Open fire (wood/coal)		Concept		
Cooling		Process		
Work	Access to finance/disposable income	Process	As contextualised by the category, work is a process providing money.	Several other aspects of work, e.g. location, but perhaps these are not meant by the stakeholders.
Communication telephone internet	Appliances	Concept		
Carbon footprint of items purchased e.g. TV, etc.		Attribute		
Consumer goods		Concept	Abstract	
Conventional oven		Concept		
Washing machine		Concept		
Cleaning		Process		
Lighting		Process or Concept		
Entertainment appliances TV radio computers		Concept(s)	Abstract, with suggested subconcepts.	Also suggests attribute of appliances—what they are used for (or relationship between them and processes)
Refrigeration		Process or Concept	Depends on interpretation	
Hoover		Process or Concept	As process it would come under Cleaning	
Microwave		Concept		
CO2 produced from use of internet		Attribute		
Power tools for gardening		Concepts		
Hairdrying		Process or Concept		
Computer		Concept		
Recreational activities	Lifestyle	Process		
Growing own food		Process		
Local shop		-	-	Concept of a Shop, and location relative to House—issue of

Index card	Category	Ontological type: Concept, Data Property (Attribute), Object Property (Relation), Process	Notes	What should/could be done with it if it doesn’t directly fit such a type
				defining ‘Local’. Could also suggest a process or a constraint.
Cooking food		Process		
Socialising/entertaining		Processes		
Waste/recycling	Waste disposal recycling/reuse	Concept/Process		
Building construction	(not in a category)	Process	Assuming what is meant is the process of creating new buildings.	

Workshop C, 29 June 2010

Note that in Workshop C, participants were divided into two groups, and each categorised the other’s cards.

Group 1’s cards

Group 1’s categorization constituted a tree-like structure—this is indicated using ‘>’.

Index card	Category assigned by Group 1	Category assigned by Group 2	Ontological type: Concept, Data Property (Attribute), Object Property (Relation), Process	Notes	What should/could be done with it if it doesn’t directly fit such a type
Refrigeration	Food > Activities	Electricity	Process	-	-
Buying food	Food > Activities	Food	Process	-	-
Washing dishes	Food > Activities	Electricity	Process	-	-
Refrigerator	Food > Appliances	Electricity	Concept	-	-
Oven	Food > Appliances	Electricity	Concept	-	-
Kitchen equipment (fridge…)	Food > Appliances	Electricity	Concept	Abstract concept	-
Freezer	Food > Appliances	Electricity	Concept	-	-
Kitchen aid	Food > Appliances	Electricity	Concept	Abstract concept	-
Stove	Food > Appliances	Electricity	Concept	-	-
Microwave	Food > Appliances	Electricity	Concept	-	-
Holidays (flight, etc.)	Transport > Activities	Transport	Process	-	-
Travel	Transport > Activities	Transport	Process	Abstract process	-
Commuting	Transport > Activities	Transport	Process	-	-
Car use	Transport > Activities	Transport	Process	Abstract process	-
Car	Transport > Appliances	Transport	Concept	-	-
Double glazing	Housing > Heating > Appliances	Heating	Concept	-	-

Index card	Category assigned by Group 1	Category assigned by Group 2	Ontological type: Concept, Data Property (Attribute), Object Property (Relation), Process	Notes	What should/could be done with it if it doesn't directly fit such a type
Insulation	Housing > Heating > Appliances	Heating	Concept	-	-
Green roof	Housing > Heating > Appliances	Heating	Concept	-	-
Boiler	Housing > Heating > Appliances	Heating	Concept	-	-
Hot water	Housing > Heating > Appliances	Heating	Concept	-	-
Bath-shower	Housing > Heating > Appliances	Heating	Concept(s)	-	-
Stereo	Housing > Electricity > Appliances	Electricity	Concept	-	-
Media (TV, Radio, Computer	Housing > Electricity > Appliances	Electricity	Concept(s)	…with abstract concept	-
(Digital) TV	Housing > Electricity > Appliances	Electricity	Concept(s)	Possible abstract concept TV	-
Wireless	Housing > Electricity > Appliances	Electricity	Concept	I assume they mean WiFi rather than a radio…	-
Stand-by	Housing > Electricity > Appliances	Electricity	Data property?	It's a state of an electronic good, or perhaps an option it has (only things with the option can have the state…)	
Phone	Housing > Electricity > Appliances	Electricity	Concept	-	-
iPod	Housing > Electricity > Appliances	Electricity	Concept	-	-
Computer	Housing > Electricity > Appliances	Electricity	Concept	-	-
Washing machine	Housing > Electricity > Appliances	Electricity	Concept	-	-
Vacuum cleaner	Housing > Electricity > Appliances	Electricity	Concept	Could also be a process, I suppose…	-
Light	Housing > Electricity > Appliances	Electricity	Concept	N.B. This card is obscured in the Group 1/1 photo, but this is what I deduce it to be.	-
Air condition[ing]	Housing > Electricity > Appliances	Electricity	Concept or Process	-	-
Repairs (electric tools)	Housing > Electricity > Appliances	Electricity	Concept or Process	It could be a Process (i.e. of DIY), or an abstract concept of the tools used to do it.	-
Shaving thing	Housing > Electricity > Activities	Electricity	Process or Concept	(It could be the concept of a shaver)	-
Garden sewing	Housing > Electricity >	Consumption	Process or Concept	Not clear what is meant by this card—the activity of	-

Index card	Category assigned by Group 1	Category assigned by Group 2	Ontological type: Concept, Data Property (Attribute), Object Property (Relation), Process	Notes	What should/could be done with it if it doesn't directly fit such a type
	Activities			sewing seeds in the garden isn't a particularly energy-consuming one? Do they mean mowing the lawn? Concept could be tools for the activity.	
Washing clothes	Housing > Electricity > Activities	Electricity	Process	-	-
Cleaning of the house	Behaviour	Electricity	Process	-	-
Unnecessary things (bought but never used)	Indirect energy use	Consumption	Concept	Abstract concept	-
Money spent	Indirect energy use	Consumption	Data property	Could also be a process	-
Electricity and heating at the workplace	Indirect energy use	Electricity	-	-	Suggests concepts of work, activities associated with that, equipment used, and energy consumption resulting.
Carbon emissions	Indirect energy use	Can't find it…	Data property	-	-
Clothes	Indirect energy use	Consumption	Concept	Could also be a data property describing the embodied energy (as for other categorised under 'Indirect energy use'	-
Flowers (buy)	Indirect energy use	Consumption	Process	-	-
Pets (food)	Indirect energy use	Food	Concept	-	-
Imported food	Indirect energy use	Food	Concept	-	-
Meat	Indirect energy use	Food	Concept	-	-

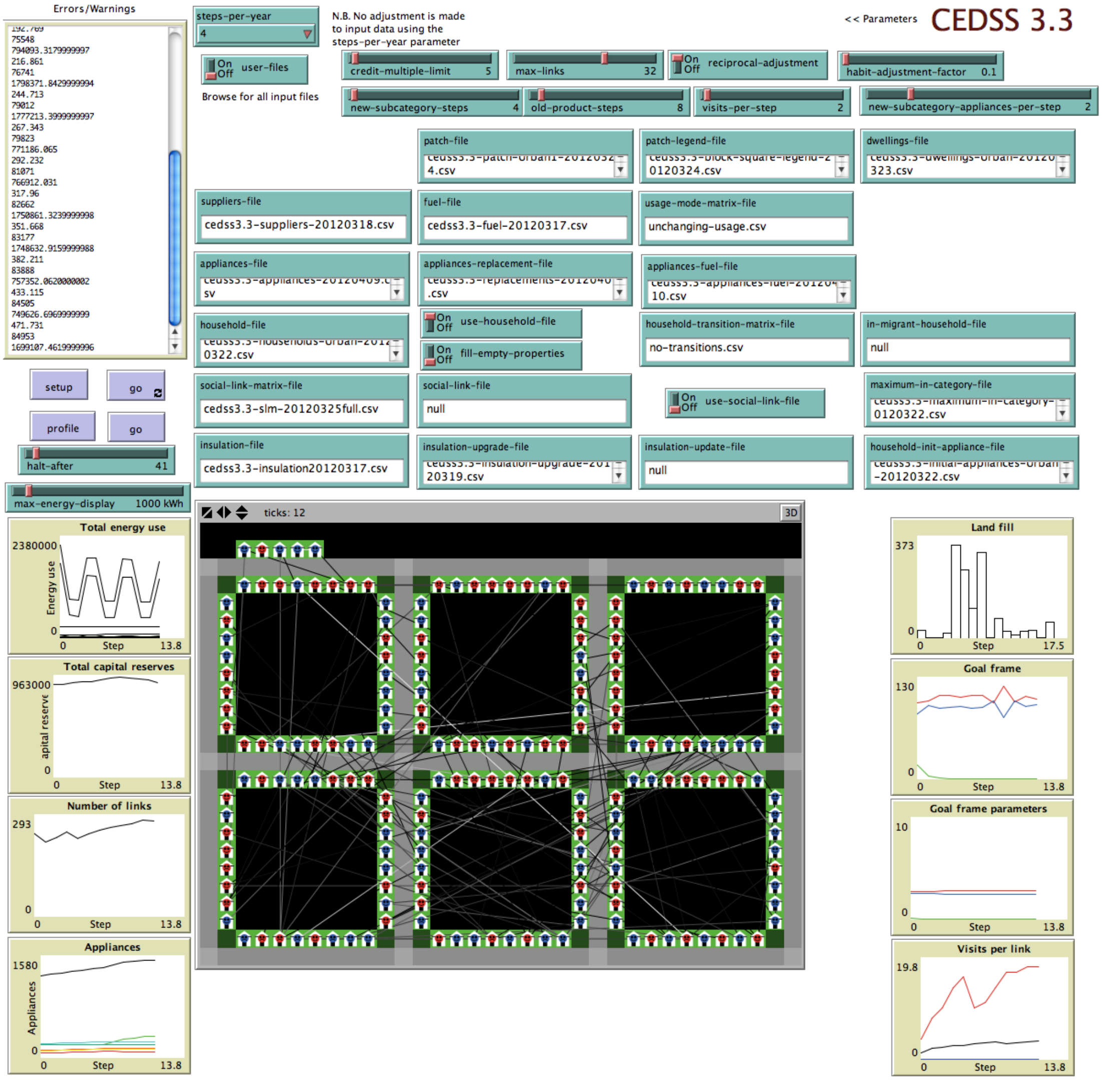
Group 2's cards

Index card	Category assigned by Group 2	Category assigned by Group 1	Ontological type: Concept, Data Property (Attribute), Object Property (Relation), Process	Notes	What should/could be done with it if it doesn't directly fit such a type
Size of house	[None clearly assigned]	Heating	Data property	-	-
# of people living there [in the house]	[None clearly assigned]	Heating	Data property (or relation if people are explicitly represented)	-	-
Washing machine	Washing	Appliances	Concept	-	-
Buying and washing clothes	Washing	Cleaning	Process	-	-
Washing [x2]	Washing	[Not used]	Process	-	-
Bath shower	Washing	Cleaning	Process or Concept	-	-
Heated swimming pool	Heating	Conspicious [?sp: conspicuous]	Concept	-	-

Index card	Category assigned by Group 2	Category assigned by Group 1	Ontological type: Concept, Data Property (Attribute), Object Property (Relation), Process	Notes	What should/could be done with it if it doesn't directly fit such a type
		(decadent) activities			
Warm water heating	Heating	Cleaning	Process	-	-
Showering	Heating	Cleaning	Process	-	-
Hot water	Heating	[Not used]	Concept	-	-
Insulation [x2]	Heating	Heating	Concept	-	-
Heating [x5]	Heating	[Made this a category]	Process	-	-
How warm the house is kept in winter	Heating	Heating	Data property	-	-
Insulation and double glazing	Heating	[Not used]	Concept (or data property of house, depending on how it is represented)	-	-
Windows	Heating	[Not used]	Concept	-	-
Isolation of the floor	Heating	[Not used]	Data property	Not clear what is meant by this… Did they mean 'insulation'? Or are they referring to the floor not being in contact with the ground?	-
Using the internet	Appliances	Conspicious [?sp: conspicuous] (decadent) activities	Process	-	-
Telephone chargers	Appliances	Appliances	Concept	-	-
Digital TV	Appliances	Appliances	Concept	-	-
Computer [x3]	Appliances	Appliances	Concept	-	-
TV [x2]	Appliances	Appliances	Concept	-	-
PC, TV, Radio	Appliances	Appliances	Concept	-	-
Household level telecommunications (phones, TV, videoplayer, mobile phones)	Appliances	Appliances	Concept	-	-
DVD player	Appliances	Appliances	Concept	-	-
Air conditioning [x2]	Appliances	Appliances	Concept or Process	With all appliances, there is the item itself, and the use thereof	-
Household electric machines	Appliances	Appliances	Concept	-	-
Blender	Appliances	Cooking	Concept	-	-
Electric appliances	Appliances	[Not used]	Concept	-	-
Printer	Appliances	Appliances	Concept	-	-
Keeping the house (repairing … grass mowing)	Appliances	Heating	Process	-	-
Lighting [x3]	Lighting	Lighting	Concept	-	-
Electricity	Lighting	[Not used]	Concept	-	-
Using an aquarium and keeping pets (food for pets)	Consumption	Conspicious [?sp: conspicuous] (decadent) activities	Process	-	-
Plastic toys	Consumption	Conspicious [?sp: conspicuous]	Concept	-	-

Index card	**Category assigned by Group 2**	**Category assigned by Group 1**	**Ontological type: Concept, Data Property (Attribute), Object Property (Relation), Process**	**Notes**	**What should/could be done with it if it doesn't directly fit such a type**
		(decadent) activities			
Shopping goods into the house	Consumption	[They declared this a category in its own right]	Process	-	-
Food consumption, freezers	Kitchen	Cooking	Process, Concept	-	-
Food	Kitchen	Cooking	Concept	-	-
Refrigerator [x2]	Kitchen	Cooking	Concept	-	-
Microwave	Kitchen	Cooking	Concept	-	-
Cooking [x2]	Kitchen	Cooking	Process	-	-
Buying and cooking food	Kitchen	Cooking	Process	-	-
Cars, public transport	Transport	[Not used]	Concept	-	-
Cars [x2]	Transport	Transport	Concept	-	-
Car use	Transport	[Not used]	Process	-	-
Vacation [x2]	Transport	Transport	Process	-	-
Public transport	Transport	Transport	Concept	-	-
Flights	Transport	Transport	Concept	-	-
Travel	Transport	Transport	Process	-	-
How many cars	Transport	Transport	Data property of household, or relation if cars are explicitly represented	-	-
Commuting	Transport	Transport	Process	-	-

Appendix 3: Screenshot of CEDSS Model



Errors/Warnings

```
192.769
75548
794093.3179999997
216.861
76741
1798371.8429999994
244.713
79012
1777213.3999999997
267.343
79823
771186.065
292.232
81071
766912.031
317.96
82662
1750861.3239999998
351.668
83177
1748632.9159999988
382.211
83888
757352.0620000002
433.115
84505
749626.6969999999
471.731
84953
1699107.4619999996
```

steps-per-year 4

N.B. No adjustment is made to input data using the steps-per-year parameter

<< Parameters CEDSS 3.3

On Off user-files

Browse for all input files

credit-multiple-limit 5

max-links 32

On Off reciprocal-adjustment

habit-adjustment-factor 0.1

new-subcategory-steps 4

old-product-steps 8

visits-per-step 2

new-subcategory-appliances-per-step 2

patch-file: cedss3.3-patch-urban1-2012032 4.csv

patch-legend-file: cedss3.3-block-square-legend-2 0120324.csv

dwellings-file: cedss3.3-dwellings-urban-20120 323.csv

suppliers-file: cedss3.3-suppliers-20120318.csv

fuel-file: cedss3.3-fuel-20120317.csv

usage-mode-matrix-file: unchanging-usage.csv

appliances-file: cedss3.3-appliances-20120409.c sv

appliances-replacement-file: cedss3.3-replacements-2012040 .csv

appliances-fuel-file: cedss3.3-appliances-fuel-201204 10.csv

household-file: cedss3.3-households-urban-2012 0322.csv

On Off use-household-file

On Off fill-empty-properties

household-transition-matrix-file: no-transitions.csv

in-migrant-household-file: null

social-link-matrix-file: cedss3.3-slm-20120325full.csv

social-link-file: null

On Off use-social-link-file

maximum-in-category-file: cedss3.3-maximum-in-category- 0120322.csv

insulation-file: cedss3.3-insulation20120317.csv

insulation-upgrade-file: cedss3.3-insulation-upgrade-201 20319.csv

insulation-update-file: null

household-init-appliance-file: cedss3.3-initial-appliances-urban -20120322.csv

halt-after 41

max-energy-display 1000 kWh

ticks: 12

Appendix 4: CEDSS Code

```
;;     CEDSS version 3.1, an agent-based model of household energy demand
;;     Copyright (C) 2011  Macaulay Land Use Research Institute
;;
;;     This program is free software: you can redistribute it and/or modify
;;     it under the terms of the GNU General Public License as published by
;;     the Free Software Foundation, either version 3 of the License, or
;;     (at your option) any later version.
;;
;;     This program is distributed in the hope that it will be useful,
;;     but WITHOUT ANY WARRANTY; without even the implied warranty of
;;     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;;     GNU General Public License for more details.
;;
;;     You should have received a copy of the GNU General Public License
;;     along with this program.  If not, see <http://www.gnu.org/licenses/>.


;; Enable the profiler, arrays and "tables" (property-value lists) to be used.
extensions [array table profiler]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;; Globals                                                                    ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

globals [
  patch-legend
  energy-price ;; the current table of fuel-type to price
  energy-price-list ;; a list of tables of fuel-type to price
  equipment-descriptor-scores
  patchset-data
  use-social-links
  steps-all-household-total-energy-use
  steps-all-household-appliance-energy-use
  steps-all-household-heating-energy-use
;; energy-use
;; steps-all-household-total-electricity-use
;; steps-all-household-total-gas-use
;; steps-all-household-total-coal-use
;; steps-all-household-total-oil-use
;; steps-all-household-total-LPG-use
  all-household-capital-reserves
  total-links
  household-transition-matrix-list
  current-household-transition-matrix
  named-in-migrants ;; a table of household type and dwelling type to a list of hh data
  in-migrant-types ;; a table of household type and dwelling type to hh dists
  in-migrant-links ;; a table of id to list of ids
  next-id
  patch-links ;; table of hh type and dwelling type to probability of link
  link-radii-list
  radius-links ;; table of hh type and dwelling type to probability of link in radii
  link-patch-types-list
  patch-type-links ;; table of hh type and dwelling type to probability of patch link
  patch-blocks ;; list of block-ids
  next-block-id
  usage-mode-matrix ;; table of goal frame to table of usage mode conditions
  household-types-list
  dwelling-types-list
  usage-modes-list
  steps-list
  dwelling-temp-colours
  new-subcategories
  land-fill
  tenure-types-list
  insulation-updates
  maximum-in-category-table
  initial-hh-appliances
  initial-hh-dw-type-appliances
  initial-hh-address-appliances
  ;; Remaining lines are for dummy variables used in debugging
  test-list
  test-item
  current-appliances
]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;; Breeds                                                                     ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
patches-own [
  patch-type
  block-id
]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; dwellings
;;
;; Dwellings are locations where households live. Each dwelling belongs to one
;; household, and is located on one patch in the space. Each patch may, however
;; contain more than one dwelling

breed [dwellings dwelling]

dwellings-own [
  dwelling-id
  dwelling-type
  tenure ;;; 'owned' or 'rented' (or something ending in 'rented')
]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; households
;;
;; Households are the main 'agents' (not in the NetLogo sense) of the model.
;; They are responsible for buying appliances and using energy

breed [households household]

households-own [
  household-id
  household-type
  steply-net-income
;; The above variable name remains the dame, but it becomes a list.
;; The variable first-step-available is added
  first-step-available
  capital-reserve
  hedonism
  gain-orientation
  greenness
  goal-frame
  usage-mode
  planning-horizon
  frame-adjustment
  breakdown-list
  wish-list
  steps-total-energy-use
]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; appliances
;;
;; Appliances are energy consuming devices used by Households

breed [appliances appliance]

appliances-own [
  category
  subcategory
  name
  essential?
  hedonic-score
  cost-list
  embodied-energy
  energy-rating ;; lower numbers are better
  energy-rating-provided?
  breakdown-probability
  first-step-available
  last-step-available
  last-step-available-unbounded?
]


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; consumption-atterns
;;
;; Patterns of fuel consumption for appliances

breed [consumption-patterns consumption-pattern]

consumption-patterns-own [
  for-household-type
  for-dwelling-type
  for-tenure-type
  for-purpose
  in-usage-mode
  in-step
]
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; fuel
;;
;; Fuel is used by appliances

breed [fuels fuel]

fuels-own [
  fuel-type
  unit
  kWh-per-unit
  total-kWh ;; for observation
  fuel-plot-colour ;; for observation
]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; insulation
;;
;; Insulation saves fuel

breed [insulations insulation]

insulations-own [
  insulation-state ;; type of insulation
  insulation-dwelling-type ;; dwelling type to which the fuel use factor applies
  fuel-use-factor ;; It is assumed the same factor applies to all fuel types
]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; supplier
;;
;; Suppliers sell fuel

; breed [suppliers supplier]

; suppliers-own [
;  supplier-id
; ]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; links
;;
;; Links between various kinds of object/agent

;; Households own appliances
directed-link-breed [ownerships ownership]
ownerships-own [
  broken?
  age
]

;; Using appliances for a particular purpose has a consumption pattern that
;; consumes fuel
directed-link-breed [consumes consume]
directed-link-breed [uses use]
uses-own [
  units-per-use
]

;; Insulations insulate dwellings
directed-link-breed [insulates insulate]

;; Insulations have upgrade costs for each dwelling type
directed-link-breed [upgrades upgrade]
upgrades-own [
  upgrade-cost ;; table of dwelling-type to cost
]


;; Suppliers supply fuel
; directed-link-breed [supplies supply]
; supplies-own [
;  cost-per-unit
;  CO2e-per-unit
;  renewably?
;]

;; Households live in dwellings
directed-link-breed [addresses address]

;; Appliances can replace each other
directed-link-breed [replacements replacement]

;; Appliances are similar to each other
;; Commented out as currently unused
;;undirected-link-breed [similarities similarity]
```

```
;;similarities-own [
;;  score
;;]

;; Households have social links with each other
undirected-link-breed [social-links social-link]
social-links-own [
  n-visits
]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;; Button procedures                                                          ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; profile
;;
;; uses the profiler to get timings for running the model

to profile
  profiler:start
  setup
  repeat halt-after [
    go
  ]
  profiler:stop
  print profiler:report
  print profile-setup
  print profile-go
  profiler:reset
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; setup
;;
;; set up the model for a run

to setup
  file-close-all ;; Added by GP as if NetLogo stops with an error whilst reading a file, it
doesn't close it
  show-licence-message

  ;; (for this model to work with NetLogo's new plotting features,
  ;; __clear-all-and-reset-ticks should be replaced with clear-all at
  ;; the beginning of your setup procedure and reset-ticks at the end
  ;; of the procedure.)
  __clear-all-and-reset-ticks ;; built-in procedure, sets all global variables to zero.
  setup-files
  output-print "setup-files"
  ifelse social-link-matrix-file = false or social-link-matrix-file = "null" or length social-
link-matrix-file = 0 [
    set use-social-links false
  ]
  [
    set use-social-links true
  ]
  setup-globals
  output-print "setup-globals"
  setup-insulation
  output-print "setup-insulation"
  setup-patches
  output-print "setup-patches"
  setup-energy
  ;; Order of text two calls reversed 20111002 to allow initial household ownership of appliances
  ;; to be specified in the households file.
  ;; Nick
  output-print "setup-energy"
  setup-appliances
  output-print "setup-appliances"
  setup-households

  output-print "setup-households"
  show-changes

  let colour-array array:from-list [red orange brown yellow green lime turquoise
    cyan sky blue violet magenta pink]
  let i 0
  set-current-plot "Appliances"
  foreach remove-duplicates [category] of appliances [
    create-temporary-plot-pen ?
    set-plot-pen-color array:item colour-array
      ((i mod (array:length colour-array)) + int(i / (array:length colour-array)))
    set i i + 1
  ]

  output-print "removed-duplicates"
  if count fuels > 1 [
    set-current-plot "Total energy use"
    ask fuels [
      create-temporary-plot-pen fuel-type
      set-current-plot-pen fuel-type
      set-plot-pen-color fuel-plot-colour
    ]
  ]
  reset-ticks
  output-print "reset-ticks"
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; go
;;
;; perform one time step of the model

to go
  set current-appliances appliances with [first-step-available <= ticks and (last-step-available-
unbounded? or last-step-available >= ticks)]

  ask fuels [
    set total-kWh 0
  ]
  ask ownerships [
    set age age + 1
  ]

  calculate-breakdowns
  update-globals

  ask households [
    step
  ]

  tick
  show-changes
  my-update-plots
  output-print timer
  set steps-all-household-appliance-energy-use calculate-appliance-energy-use
  set steps-all-household-heating-energy-use calculate-heating-energy-use
  output-print steps-all-household-appliance-energy-use
  output-print steps-all-household-heating-energy-use

  if (ticks = halt-after) [
    stop
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; calculate-appliance-energy-use
;;
;; Calculate appliance energy use for a step, to be output at the end of each step
;;

to-report calculate-appliance-energy-use
  let appliance-energy-use 0
  ask fuels with [fuel-type = "appliance-gas" or fuel-type = "appliance-elect"] [
    set appliance-energy-use appliance-energy-use + total-kWh
  ]
  report appliance-energy-use
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; calculate-heating-energy-use
;;
;; Calculate heating energy use for a step, to be output at the end of each step
;;

to-report calculate-heating-energy-use
  let heating-energy-use 0
  ask fuels with [fuel-type != "appliance-gas" and fuel-type != "appliance-elect"] [
    set heating-energy-use heating-energy-use + total-kWh
  ]
  report heating-energy-use
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; my-update-plots
;;
;; Update all the plots

to my-update-plots
  set-current-plot "Total energy use"
  set-current-plot-pen "default"
  plot steps-all-household-total-energy-use
  if count fuels > 1 [
    ask fuels [
      set-current-plot-pen fuel-type
      plot total-kWh
    ]
  ]

  set-current-plot "Total capital reserves"
  plot all-household-capital-reserves

;; This and similar conditionals condition are included to make the use of social links optional.
;; Nick
  if use-social-links [
    set-current-plot "Number of links"
    let link-count sum [count social-link-neighbors] of households
    plot link-count
    set total-links total-links + link-count
  ]

  set-current-plot "Appliances"
  set-current-plot-pen "default"
  plot count ownerships
  foreach remove-duplicates [category] of appliances [
    set-current-plot-pen ?
    let osum 0
    ask households [
      set osum osum + count (out-ownership-neighbors with [category = ?])
    ]
    plot osum
  ]

  set-current-plot "Land fill"
  let i 0
  plot-pen-reset
 ;; output-print sort remove-duplicates [subcategory] of appliances
  foreach sort remove-duplicates [subcategory] of appliances [
    set i i + 1
    let subcat ?
 ;; output-print subcat
 ;; output-print land-fill
    plot length (filter [[subcategory] of ? = subcat] land-fill)
    if ticks = 1 [
      print (word "Land fill plot " i " is subcategory " subcat)
    ]
  ]

  set-current-plot "Goal frame"
  set-current-plot-pen "hedonistic"
  plot count households with [goal-frame = "hedonistic"]
  set-current-plot-pen "gain"
  plot count households with [goal-frame = "gain"]
  set-current-plot-pen "norm"
  plot count households with [goal-frame = "norm"]

  set-current-plot "Goal frame parameters"
  set-current-plot-pen "hedonism"
  plot mean [hedonism] of households
  set-current-plot-pen "gain"
  plot mean [gain-orientation] of households
  set-current-plot-pen "norm"
  plot mean [greenness] of households

  set-current-plot "Visits per link"
  set-current-plot-pen "mean"
  if use-social-links [
    plot mean [n-visits] of social-links
    set-current-plot-pen "min"
    plot min [n-visits] of social-links
    set-current-plot-pen "max"
    plot max [n-visits] of social-links
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;; Procedures for setting up/creating the model                               ;;
;;                                                                            ;;
;; Note that these procedures do not include those for reading/writing to a   ;;
;; file. There is a separate section in the code for those.                   ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; setup-files
;;
;; Set up the file names to use.

to setup-files
  if user-files [
    user-message "Choose patch legend file (1)"
    set patch-legend-file user-file
    user-message "Choose patch file (2)"
    set patch-file user-file
    user-message "Choose dwellings file (3)"
    set dwellings-file user-file

    user-message "Choose energy suppliers file (14)"
    set suppliers-file user-file
    user-message "Choose fuel file (12)"
    set fuel-file user-file
    user-message "Choose usage mode matrix file (9)"
    set usage-mode-matrix-file user-file

    user-message "Choose appliances file (10)"
    set appliances-file user-file
    user-message "Choose appliances replacement file (11)"
    set appliances-replacement-file user-file

 ;;   user-message "Choose appliances similarity file (15)"
 ;;   set appliances-similarity-file user-file
    user-message "Choose appliances fuel file (13)"
    set appliances-fuel-file user-file
    user-message "Choose maximum in category file (19)"
    set maximum-in-category-file user-file
    user-message "Choose household initial appliances file (20)"
    set household-init-appliance-file user-file

    ifelse use-household-file [
      user-message "Choose household file (4)"
      set household-file user-file
    ]
    [
      set household-file false
    ]
    user-message "Choose household transition matrix file (5)"
    set household-transition-matrix-file user-file
    user-message "In-migrant household file (6)"
    set in-migrant-household-file user-file
    user-message "Choose social link matrix file (7) (click cancel if you do not want social
links)"
    set social-link-matrix-file user-file
    ifelse use-social-link-file [
      user-message "Choose social link file (8)"
      set social-link-file user-file
    ]
    [
      set social-link-file false
    ]

    user-message "Choose insulation file (16)"
    set insulation-file user-file
    user-message "Choose insulation upgrade file (17)"
    set insulation-upgrade-file user-file
    user-message "Choose insulation update file (18)"
    set insulation-update-file user-file

    set user-files false
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; setup-globals
;;
;; Set up the global variables. Read in the energy price file and equipment data

to setup-globals
  set total-links 0
  set next-id 0
  set next-block-id 0

  set steps-list n-values steps-per-year [? + 1]
  ;; nvalues steps-per-year [? + 1] produces the list [1 2 3... <steps-per-year>

  set dwelling-temp-colours array:from-list [102 blue cyan turquoise green
    yellow orange red pink 138]

  set land-fill []
  set in-migrant-links table:make
  ;; Added 20111015 to allow limits to be set on the number of appliances
  ;; each type of household can own of each category of appliance
  set maximum-in-category-table read-table2 maximum-in-category-file
  set initial-hh-appliances false
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; setup-insulation
;;
```

```
;; Set up the insulation

to setup-insulation
  read-insulation-file insulation-file
  read-insulation-upgrade-file insulation-upgrade-file
  read-insulation-update-file insulation-update-file
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; setup-patches
;;
;; Read in the patch layout and social link files

to setup-patches
  set patch-legend read-table patch-legend-file
  foreach table:keys patch-legend [
    if(is-string? (table:get patch-legend ?)) [
      table:put patch-legend ? (read-from-string (table:get patch-legend ?));
    ]
  ]
  read-patch-layout patch-file
  read-dwellings-file dwellings-file

  set dwelling-types-list remove-duplicates [dwelling-type] of dwellings
  set tenure-types-list remove-duplicates [tenure] of dwellings

  determine-patch-type-blocks
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; setup-appliances
;;
;; Set up the appliances

to setup-appliances
  read-appliances appliances-file
  output-print "read-appliances"
  ask households [
 ;; The following functionality moved into setup-households
 ;;   create-ownerships-to appliances with [first-step-available < 0] [
 ;;     set hidden? true
 ;;     set broken? false
 ;;     set age 0
 ;;   ]
 ;;   set breakdown-list []
  ]
  read-replacements appliances-replacement-file
  output-print "read-replacements"

 ;; read-appliance-similarity appliances-similarity-file
  read-appliances-fuel-use appliances-fuel-file
  output-print "read-appliances-fuel-use"
  if(household-init-appliance-file != false and household-init-appliance-file != "null") [
    read-initial-appliances-file household-init-appliance-file
  output-print "read-initial-appliances-file"
  ]

  ask appliances with [not last-step-available-unbounded?] [
    if count my-out-replacements = 0 [
      output-print (word "*** Warning: There are no replacements for appliance \"" name "\"")
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; setup-energy
;;
;; Set up energy/fuel and suppliers

to setup-energy
  read-fuel fuel-file
  read-energy-suppliers suppliers-file
  set usage-mode-matrix read-matrix usage-mode-matrix-file

  set usage-modes-list []
  foreach table:keys usage-mode-matrix [
    let umodes table:get usage-mode-matrix ?
    foreach table:keys umodes [
      set usage-modes-list fput ? usage-modes-list
    ]
  ]
  set usage-modes-list remove-duplicates usage-modes-list
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; setup-households
;;
;; Create and intialise the households and related global variables
```

```
to setup-households
  ifelse use-household-file [
    read-households-file household-file
    set household-types-list remove-duplicates [household-type] of households
  ;; output-print (word "household-types-list: " household-types-list)
    ask households [
      allocate-initial-appliances
    ]
  ]

  [
    set household-types-list []
  ]

  set household-transition-matrix-list read-numeric-ts-matrix
  household-transition-matrix-file ["in-migrant"]
  ;; output-print (word "household-transition-matrix-list: " household-transition-matrix-list)
  foreach table:keys (first household-transition-matrix-list) [
    if not member? ? household-types-list [
      set household-types-list fput ? household-types-list
    ]
  ]

  read-in-migrant-file in-migrant-household-file
  if use-social-links [
    read-social-link-matrix-file social-link-matrix-file
  ]
  ;; Allocate people from in-migrant file to empty dwellings
  ;; Next line changed to make it optional to fill empty properties.
  if fill-empty-properties and (count dwellings with [count in-address-neighbors = 0]) > 0 [
    foreach (sort dwellings with [count in-address-neighbors = 0]) [

      let this-dwelling ?

      let dwt-type (word ([tenure] of this-dwelling) ":" ([dwelling-type] of this-dwelling))

      let hh-types []
      foreach table:keys in-migrant-types [
        if table:has-key? (table:get in-migrant-types ?) dwt-type [
          set hh-types fput ? hh-types
        ]
      ]

      ifelse length hh-types > 0 [
        create-households 1 [
          create-address-to this-dwelling [
            set hidden? true
          ]
          set-household-nlogo-params
          set household-type one-of hh-types
          set household-id "new" ;; It will be set to a unique value in resample-parameters
          resample-parameters
        ]
      ]
      [
        output-print (word "*** Warning: Cannot create household for dwelling "
          [dwelling-id] of this-dwelling
          ": no household types associated with dwelling/tenure type "
          dwt-type " in the in-migrant household file")
      ]
    ]
  ]
  ask households [
    if use-social-links [
      make-random-social-links
    ]

    set breakdown-list []

    set wish-list []

    set goal-frame choose-goal-frame
  ]
  if use-social-link-file [
    read-social-link-file social-link-file
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; allocate-initial-appliances
;;
;; Allocate the initial appliances of a household

to allocate-initial-appliances

  ;; Default initial appliances defined by first/last step available in the
  ;; appliance file
  create-ownerships-to appliances with [first-step-available < 0 and last-step-available <= ticks]
[
    set hidden? true
    set broken? false
    set age 0
  ]

  ;; Has initial appliance list been specifically defined for this household?

  if initial-hh-appliances != false [
    ifelse table:has-key? initial-hh-appliances household-id [
      foreach (table:get initial-hh-appliances household-id) [
        create-ownerships-to appliances with [name = ?] [
          set hidden? true
          set broken? false
          set age 0
        ]
      ]
      table:remove initial-hh-appliances household-id
    ]
    [
      let my-dw one-of out-address-neighbors

      let hh-address (word household-type ":" ([dwelling-id] of my-dw))

      ;; Has initial appliance list been defined for this household type at this
      ;; specific address?
      ifelse table:has-key? initial-hh-address-appliances hh-address [
        foreach (table:get initial-hh-address-appliances hh-address) [
          create-ownerships-to appliances with [name = ?] [
            set hidden? true
            set broken? false
            set age 0
          ]
        ]
      ]
      [
        let hh-dw-type (word household-type ":" ([tenure] of my-dw) ":" ([dwelling-type] of my-
dw))

        ;; Has initial appliance list been defined for this household type at this
        ;; dwelling type and tenure combination?

        if table:has-key? initial-hh-dw-type-appliances hh-dw-type [
          foreach (table:get initial-hh-dw-type-appliances hh-dw-type) [
            create-ownerships-to appliances with [name = ?] [
              set hidden? true
              set broken? false
              set age 0
            ]
          ]
        ]
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; set-household-nlogo-params
;;
;; Set the netlogo parameters of a household

to set-household-nlogo-params
  set shape "face happy"
  set size 0.5
  set xcor [xcor] of [other-end] of one-of my-out-addresses
  set ycor [ycor] of [other-end] of one-of my-out-addresses
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; determine-patch-type-blocks
;;
;; Determine sets of patches in blocks of common patch-types. This uses two
;; approaches, an efficient one that works if the world is not wrapped, and
;; a less efficient one that can be used otherwise.

to determine-patch-type-blocks
  ask patches [
    set block-id 0
  ]

  ifelse (count [neighbors4] of patch min-pxcor min-pycor) > 2 [
    ;; the world wraps in one or more dimensions
    ask patches [
      set-block-id ;; This approach is a bit inefficient
    ]
  ]
  [
    ;; the world does not wrap -- use more efficient procedure
    let px min-pxcor

    while [px <= max-pxcor] [
      let py min-pycor

      while [py <= max-pycor] [
        ask patch px py [
          let my-patch-type patch-type
          let nbrs neighbors4 with [patch-type = my-patch-type and (not (block-id = 0))]
          ifelse count nbrs > 0 [
            ;; We can get the patch ID from the Von-Neumann neighbours
            let idlist [block-id] of nbrs
            set block-id first idlist

            set idlist but-first idlist
            foreach idlist [
              ;; Deal with neighours with different block-ids
              if ? != block-id [
                ask patches with [block-id = ?] [
                  set block-id ?
                ]
              ]
            ]
          ]
          [
            ;; All the neighbours have block-id 0 or different patch-type
            set next-block-id next-block-id + 1
            set block-id next-block-id
          ]
        ]
        set py py + 1
      ]
      set px px + 1
    ]
  ]

  ;; Get the list of unique block-ids of patches
  set patch-blocks (remove-duplicates ([block-id] of patches))
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; set-block-id
;;
;; Set the block-id of a patch (recursively checking all neighbouring patches)

to set-block-id
  let my-patch-type patch-type
  ifelse block-id = 0 [
    let same-nbrs neighbors4 with [patch-type = my-patch-type]
    ifelse count same-nbrs > 0 [
      let same-nbrs-block same-nbrs with [block-id != 0]
      ifelse count same-nbrs-block > 0 [
        ;; Some neighbours with the same patch-type have a block-id we can use
        let block-ids [block-id] of same-nbrs-block
        set block-id first block-ids

        set block-ids but-first block-id
        let my-block-id block-id
        foreach block-ids [
          ;; Recursively ensure all neighbours with the same patch-type have
          ;; the same block-id
          if ? != block-id [
            ask patches with [block-id = ?] [
              set-block-id-to my-block-id
            ]
          ]
        ]

        ;; Recursively ensure all neighbours with the same patch-type that
        ;; have not had their block-id set yet have this block-id
        ask same-nbrs with [block-id = 0] [
          set-block-id-to my-block-id
        ]
      ]
      [
        ;; There are no neighbours from which to get a block-id: Set this
        ;; block-id to the next one, and recursively set the block-id of
        ;; neighbours with the same patch-type
        set next-block-id next-block-id + 1
        set block-id next-block-id
        ask same-nbrs [
          set-block-id-to next-block-id
        ]
      ]
    ]
    [
      ;; no neighbours with the same patch-type -- just set this patch's
      ;; unique block id.
      set next-block-id next-block-id + 1
      set block-id next-block-id
    ]
  ]
  [
    ;; block-id has already been set -- recursively ensure neighbours with
    ;; the same patch-type have the same block-id
    let my-block-id block-id
    ask neighbors4 with [patch-type = my-patch-type and block-id != my-block-id] [
      set-block-id-to my-block-id
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; set-block-id-to [a-block-id]
;;
;; Recursively set the block-id of patches with the same patch-type

to set-block-id-to [a-block-id]
  set block-id a-block-id
  let my-patch-type patch-type
  ask neighbors4 with [patch-type = my-patch-type and block-id != a-block-id] [
    set-block-id-to a-block-id
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; make-random-social-links
;;
;; Use the rules defined in the social-link-matrix-file to create random social
;; links among households

to make-random-social-links
  let dw-type [dwelling-type] of one-of out-address-neighbors
  ;; the above assumes one address per household

  let my-type (word household-type ":" dw-type)
  let my-anydtype (word household-type ":*")
  let my-anyhtype (word "*:" dw-type)

  ifelse table:has-key? patch-links my-type [
    make-random-social-links-type my-type
  ]
  [
    ifelse table:has-key? patch-links my-anydtype [
      make-random-social-links-type my-anydtype
    ]
    [
      ifelse table:has-key? patch-links my-anyhtype [
        make-random-social-links-type my-anyhtype
      ]
      [
        if table:has-key? patch-links "*:*" [
          make-random-social-links-type "*:*"
        ]
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; make-random-social-links-type
;;
;; Make random social links for the specified household:dwelling type

to make-random-social-links-type [key]
  make-patch-social-links (table:get patch-links key)

  if length link-radii-list > 0 [
    make-radius-social-links (table:get radius-links key)
  ]

  if length link-patch-types-list > 0 [
    make-patch-type-social-links (table:get patch-type-links key)
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; make-patch-social-links
;;
;; Make social links with households on the same patch

to make-patch-social-links [link-table]
  let hh self
    ;; Nick. "with [self != hh]" added 20111027
  ask households-here  with [self != hh] [
    let dw-type [dwelling-type] of one-of out-address-neighbors
    let this-type (word household-type ":" dw-type)
    let this-anydtype (word household-type ":*")
    let this-anyhtype (word "*:" dw-type)

    ifelse table:has-key? link-table this-type [
      make-patch-social-links-type link-table this-type hh
    ]
    [
      ifelse table:has-key? link-table this-anydtype [
        make-patch-social-links-type link-table this-anydtype hh
      ]
      [
        ifelse table:has-key? link-table this-anyhtype [
          make-patch-social-links-type link-table this-anyhtype hh
        ]
        [
          if table:has-key? link-table "*:*" [
            make-patch-social-links-type link-table "*:*" hh
          ]
        ]
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; make-patch-social-links-type
;;
;; Construct a social link with households of the specified hh:dw type

to make-patch-social-links-type [link-table key hh]
  let link-p read-from-string table:get link-table key
  if random-float 1 < link-p and not social-link-neighbor? hh [
    create-social-link-with hh
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; make-radius-social-links
;;
;; Make social links with households within specified distances

to make-radius-social-links [link-table]
  let hh self
  let hh-patch [patch-here] of hh
  ask households [
    let dw-type [dwelling-type] of one-of out-address-neighbors
    let this-type (word household-type ":" dw-type)
    let this-anydtype (word household-type ":*")
    let this-anyhtype (word "*:" dw-type)

    ifelse table:has-key? link-table this-type [
      make-radius-social-links-type link-table this-type hh hh-patch
    ]
    [
      ifelse table:has-key? link-table this-anydtype [
        make-radius-social-links-type link-table this-anydtype hh hh-patch
      ]
      [
        ifelse table:has-key? link-table this-anyhtype [
          make-radius-social-links-type link-table this-anyhtype hh hh-patch
        ]
        [
          if table:has-key? link-table "*:*" [
            make-radius-social-links-type link-table "*:*" hh hh-patch
          ]
        ]
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; make-radius-social-links-type
;;
;; Make social links within specified distances using the given hh:dw type

to make-radius-social-links-type [link-table key hh hh-patch]
  (foreach link-radii-list (table:get link-table key) [
    if [distance hh-patch] of patch-here <= ?1 [
      if random-float 1 < read-from-string ?2 and not social-link-neighbor? hh [
        create-social-link-with hh
      ]
    ]
  ])
end
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; make-patch-type-social-links
;;
;; Make social links with households in or bordering shared patch blocks

to make-patch-type-social-links [link-table]
  let hh self
  let hh-patch [patch-here] of hh
  let hh-nbr-block-ids [block-id] of ([neighbors] of hh-patch)

  ask households [
    let dw-type [dwelling-type] of one-of out-address-neighbors
    let this-type (word household-type ":" dw-type)
    let this-anydtype (word household-type ":*")
    let this-anyhtype (word "*:" dw-type)

    ifelse table:has-key? link-table this-type [
      make-patch-type-social-links-type link-table this-type hh hh-patch hh-nbr-block-ids
    ]
    [
      ifelse table:has-key? link-table this-anydtype [
        make-patch-type-social-links-type link-table this-anydtype hh hh-patch hh-nbr-block-ids
      ]
      [
        ifelse table:has-key? link-table this-anyhtype [
          make-patch-type-social-links-type link-table this-anyhtype hh hh-patch hh-nbr-block-ids
        ]
        [
          if table:has-key? link-table "*:*" [
            make-patch-type-social-links-type link-table "*" hh hh-patch hh-nbr-block-ids
          ]
        ]
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; make-patch-type-social-links-type
;;
;; Make social links with households of specified hh:dw type in or bordering
;; shared patch blocks

to make-patch-type-social-links-type [link-table key hh hh-patch hh-nbr-block-ids]
  (foreach link-patch-types-list (table:get link-table key) [
    ifelse ?1 = "dwelling" [
      ;; dwelling patch type -- both households' dwellings must be in the
      ;; same block...
      if [block-id] of hh-patch = [block-id] of patch-here [
        if random-float 1 < read-from-string ?2 and not social-link-neighbor? hh [
          create-social-link-with hh
        ]
      ]
    ]
    [
      ;; ... for all other patch types, both households' dwellings must be
      ;; neighbours of the same block of the required type
      let my-nbr-block-ids [block-id] of (([neighbors] of patch-here) with [patch-type = ?1])
      if length intersection hh-nbr-block-ids my-nbr-block-ids > 0 [
        if random-float 1 < read-from-string ?2 and not social-link-neighbor? hh and hh != self [
          create-social-link-with hh
        ]
      ]
    ]
  ])
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;; Procedures for running a timestep with the model                           ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; update-globals
;;
;; Update globals at the start of a step

to update-globals
  if (ticks < length energy-price-list) [
    set energy-price first energy-price-list
    set energy-price-list but-first energy-price-list
  ]
  set steps-all-household-total-energy-use 0
  ;;  set steps-all-household-electricity-use 0
  ;;  set steps-all-household-gas-use 0
  ;;  set steps-all-household-coal-use 0
  ;;  set steps-all-household-oil-use 0
  ;;  set steps-all-household-LPG-use 0
  set all-household-capital-reserves 0

  if(length household-transition-matrix-list > 0) [
    set current-household-transition-matrix (first household-transition-matrix-list)
    set household-transition-matrix-list (but-first household-transition-matrix-list)
  ]

  ;; Get the list of subcategories of appliances introduced in the last
  ;; new-subcategory-steps

  set new-subcategories []
  let new-appliance-subcategories [subcategory] of appliances with
    [first-step-available >= (ticks - new-subcategory-steps)]
  foreach remove-duplicates new-appliance-subcategories [
    if (count (appliances with [(subcategory = ?) and
      (first-step-available < (ticks - new-subcategory-steps))]) = 0) [
      set new-subcategories (fput ? new-subcategories)
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; step
;;
;; Perform one step of the model (for households)

to step
  transition-household-state

  set goal-frame choose-goal-frame ;; Just do this once per step in CEDSS 3
  set usage-mode get-usage-mode

  calculate-moeu ; steply overall energy use
  calculate-finance

  replace-broken-appliances
  ifelse goal-frame = "hedonistic" [
    let hedonic-visited-list update-wish-list
    buy-new-appliances
    foreach hedonic-visited-list [
      visit ?
    ]
  ]
  [
    if [tenure] of one-of out-address-neighbors = "owned" [
      buy-insulation
    ]
    buy-new-appliances
    if count social-link-neighbors > 0 [
      repeat visits-per-step [
        visit one-of social-link-neighbors
      ]
    ]
  ]
  if use-social-links [
    update-links
  ]
  update-insulation-upgrades

  set all-household-capital-reserves all-household-capital-reserves + capital-reserve
  set steps-all-household-total-energy-use (steps-all-household-total-energy-use
    + steps-total-energy-use)
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; replace-broken-appliances
;;
;; households replace their broken appliances using the current goal frame

to replace-broken-appliances
  let broken-appliances []

  ask my-out-ownerships with [broken? = true] [
    set broken-appliances fput other-end broken-appliances
    set land-fill fput other-end land-fill
    die
  ]

  let my-dwelling one-of out-address-neighbors

  foreach broken-appliances [
    let broken-appliance ?
    ;; Nick 2011-10-18 Amended from this point on to make all essential appliances, not just
heating, landlord-supplied.
    ;; ifelse [category] of broken-appliance = "heating" [
    ifelse [essential?] of broken-appliance and one-of (my-out-ownerships with [[category] of
other-end = [category] of broken-appliance]) = nobody [
      let newitem false

      let this-tenure [tenure] of my-dwelling

      ifelse length this-tenure > 6 and substring (reverse this-tenure) 0 6 = "detner" [
        ;; The tenure ends with "rented"
        set newitem hedonistic-ess-replace broken-appliance
        ifelse newitem != false and is-agent? newitem and newitem != nobody [
          ; print (word household-id " replacing broken essential appliance " broken-appliance "
with " newitem)
          add-item-cost-free newitem 0
        ]
        [
          output-print (word "*** Warning: renting household \"" household-id
            "\" could not replace essential appliance \"" [name] of broken-appliance "\"")
          set breakdown-list (fput broken-appliance breakdown-list)
        ]
      ]
      [
        ifelse goal-frame = "hedonistic" [
          set newitem hedonistic-ess-replace broken-appliance
        ]
        [
          ifelse goal-frame = "gain" [
            ;; Next three lines added
            ifelse  [category] of broken-appliance = "heating" [
              set newitem heating-system-cost-advice self broken-appliance
            ]
            [
              set newitem gain-ess-replace broken-appliance
            ]
          ]
          [
            set newitem norm-ess-replace broken-appliance
          ]
        ]
        ifelse newitem != false and is-agent? newitem and newitem != nobody [
          ; print (word household-id " replacing broken essential appliance " broken-appliance "
with " newitem)
          add-item newitem 0
        ]
        [
          output-print (word "*** Warning: household \"" household-id
            "\" could not replace essential appliance \"" [name] of broken-appliance "\"")
          set breakdown-list (fput broken-appliance breakdown-list)
        ]

      ]
    ]
    [
      ;; Nick 2011-1018 Most of remainder of procedure edited out - already covered above.
      ;;ifelse [essential?] of broken-appliance [
      ;;  let newitem false

      ;;ifelse (goal-frame = "hedonistic") [
      ;;  set newitem hedonistic-ess-replace broken-appliance
      ;;]
      ;;[
      ;;  ifelse (goal-frame = "gain") [
      ;;    set newitem gain-ess-replace broken-appliance
      ;;  ]
      ;;  [
      ;;    set newitem norm-ess-replace broken-appliance
      ;;  ]
      ;;]
      ;;ifelse newitem != false and is-agent? newitem and newitem != nobody [
      ;;  ; print (word household-id " replacing broken essential " broken-appliance " with "
newitem)
      ;;  add-item newitem 0
      ;;  ]
      ;;  [
      ;;    output-print (word "*** Warning: household \"" household-id
      ;;      "\" could not replace essential broken equipment \"" [name] of broken-appliance
"\"")
      ;;    set breakdown-list (fput broken-appliance breakdown-list)
      ;;  ]
      ;;]
      ;;[
      ;;  ;; ...otherwise, just bung it on the list of broken equipment
      ;;  ; print (word household-id " saving broken non-essential " broken-appliance " for
later")
      set breakdown-list (fput broken-appliance breakdown-list)
    ]
  ]
end
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; buy-new-appliances
;;
;; Purchase new appliances

to buy-new-appliances
  ifelse (goal-frame = "hedonistic") [
    hedonistic-equip
  ]
  [
    ifelse (goal-frame = "gain") [
      gain-equip
    ]
    [
      norm-equip
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; buy-insulation
;;
;; Buy insulation (norm and gain mode only)

to buy-insulation
  ifelse goal-frame = "gain" [
    gain-insulation
  ]
  [
    norm-insulation
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; update-wish-list
;;
;; Update the wish list for appliances and return the list of neighbours visited

to-report update-wish-list
  ;; choose one random item with a subcategory introduced in the last
  ;; new-subcategory-steps

;;  Next 3 lines replaced to allow number of new-subcategory items added to wish-list to be set
;;  by a global variable given as an input parameter, new-subcategory-appliances-per-step.

;;  let new-appliances appliances with [member? subcategory new-subcategories and not (category =
"heating")]
;;  if count new-appliances > 0 [
;;  set wish-list fput (one-of new-appliances) wish-list
;;  ]

  let new-subcategory-appliances-to-choose new-subcategory-appliances-per-step
  let new-appliance-list sort appliances with [member? subcategory new-subcategories and not
(category = "heating")]
  while [length new-appliance-list > 0 and new-subcategory-appliances-to-choose > 0] [
    set wish-list fput (first new-appliance-list) wish-list
    set new-subcategory-appliances-to-choose new-subcategory-appliances-to-choose - 1
    set new-appliance-list but-first new-appliance-list
  ]

  ;; choose one random item from visits-per-step social links

  let hedonic-visited-list []
  if count social-link-neighbors > 0 [
    let link-items []
    repeat visits-per-step [
      let visited one-of social-link-neighbors
      let unowned-appliances ((appliances-i-dont-have-owned-by visited) with [not (category =
"heating")])
      if count unowned-appliances > 0 [
        set link-items (sentence link-items (sort unowned-appliances))
        ;; Doesn't matter if an appliance occurs more than once in the list
        ;; (Note that 'sort' is used to convert an agent-list into a list)
      ]
      set hedonic-visited-list fput visited hedonic-visited-list
    ]
    if length link-items > 0 [
      set wish-list fput (one-of link-items) wish-list
    ]
  ]

  ;; choose one random replacement for an appliance already owned that is more
  ;; than old-product-steps old

  ;; Nick 2011-10-18 Changed next line so that no essential item will be selected for replacement.
This is necessary
  ;; for consistent treatment of essential items when the tenure is "rented".
  ;; let an-old-ownership one-of (my-out-ownerships with [age >= old-product-steps and not
([category] of other-end = "heating")])
  let an-old-ownership one-of (my-out-ownerships with [age >= old-product-steps and not
([essential?] of other-end)])
  if(an-old-ownership != nobody) [
    set wish-list fput (one-of (current-replacements-for [other-end] of
      an-old-ownership))
    wish-list
  ]

  report hedonic-visited-list
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; update-insulation-upgrades
;;
;; use the information loaded from the insulation upgrade file to update the
;; insulation upgrades available

to update-insulation-upgrades
  let next-update false

  if length insulation-updates > 0 [
    set next-update first insulation-updates
  ]

  while [ next-update != false ] [
     ifelse table:get next-update "step" = ticks + 1 [
       update-insulation next-update

       set insulation-updates but-first insulation-updates
       set next-update first insulation-updates
     ]
     [
       set next-update false
     ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; update-insulation
;;
;; Implement an insulation update command

to update-insulation [cmd]
  let command table:get cmd "command"
  let dw-type table:get cmd "dwelling-type"
  let from-state insulations with [insulation-state = table:get cmd "from-state" and insulation-
dwelling-type = dw-type]
  let to-state insulations with [insulation-state = table:get cmd "to-state" and insulation-
dwelling-type = dw-type]
  let the-cost read-from-string table:get cmd "cost"

  if count from-state != 1 [
    output-print (word "*** Error in insulation update file " insulation-update-file
      ": not 1 insulation for state " (table:get cmd "from-state")
       " and dwelling type " dw-type)
  ]

  if count to-state != 1 [
    output-print (word "*** Error in insulation update file " insulation-update-file
      ": not 1 insulation for state " (table:get cmd "to-state")
       " and dwelling type " dw-type)
  ]


  ifelse command = "remove" [
    ask from-state [
      ask out-upgrade-to one-of to-state [
        die
      ]
    ]
  ]
  [
    ifelse command = "add" [
      ask from-state [
        create-upgrades-to to-state [
          set upgrade-cost the-cost
          set hidden? true
        ]
      ]
    ]
    [
      ifelse command = "change" [
        ask from-state [
          ask out-upgrade-to one-of to-state [
            set upgrade-cost the-cost
          ]
        ]
      ]
      [
        output-print (word "*** Warning: Ignoring unrecognised insulation "
          "upgrade update command \"" command "\"")
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; get-usage-mode
;;
;; Return the usage mode for a household

to-report get-usage-mode
  let mode []
  let mode-boolean table:make

  let frame-table table:get usage-mode-matrix goal-frame

  foreach (table:keys frame-table) [
    let condition-str table:get frame-table ?
    let condition-not false

    let condition-words split " " condition-str

    if first condition-words = "not" [
      set condition-not true
      set condition-words but-first condition-words
    ]

    let condition first condition-words

    if condition = "negative-capital-reserve" [
      ifelse negative-capital-reserve [
        add-mode mode-boolean ? condition-not
      ]
      [
        add-mode mode-boolean ? (not condition-not)
      ]
    ]
    if condition = "true" [
      add-mode mode-boolean ? condition-not
    ]
    if condition = "false" [
      add-mode mode-boolean ? (not condition-not)
    ]

    ;; Add new rules here
  ]

  foreach (table:keys mode-boolean) [
    if table:get mode-boolean ? = true [
      set mode fput ? mode
    ]
  ]

  if length mode > 1 [
    output-print (word "*** Error: ambiguous usage mode for agent with goal-frame \""
      goal-frame "\": " mode)
  ]
  if length mode = 0 [
    output-print (word "*** Error: no usage mode for agent with goal-frame \"" goal-frame "\"")
  ]

  report one-of mode
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; add-mode
;;
;; Add a mode to the mode boolean table

to add-mode [mode-table mode boolean]
  if table:has-key? mode-table mode [
    if not (table:get mode-table mode = boolean) [
      output-print (word "*** Error: conflict for usage-mode \"" mode "\" in agent with "
        "goal frame \"" goal-frame "\"")
    ]
  ]
  table:put mode-table mode boolean
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; transition-household-state
;;
;; Use the current household transition matrix to determine the change in state
;; of the household

to transition-household-state
  let transition-probabilities (table:get
    current-household-transition-matrix household-type)
  let p-sum 0
  let p-value random-float 1
  let changed false
  ;; not really "changed" if it happens to select the same new state

  foreach (shuffle table:keys transition-probabilities) [
    let p table:get transition-probabilities ?
    let migrant false
    set p-sum p-sum + p
    if (not changed) and (p-value < p-sum) [
      ifelse ? = "in-migrant" [
        set household-type one-of household-types-list
        resample-parameters
      ]
      [
        set household-type ?
      ]
      set changed true
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; resample-parameters
;;
;; A household resamples its parameters to simulate another household migrating
;; in

to resample-parameters
  let my-dwelling one-of out-address-neighbors
  let hh-type [household-type] of self
  let dw-type [dwelling-type] of my-dwelling
  let t-type [tenure] of my-dwelling
  let dwt-type (word t-type ":" dw-type)

  let resampled? false
  if household-id != "new" and table:has-key? named-in-migrants household-id [
    let hh-table table:get named-in-migrants household-id
    if table:has-key? hh-table dwt-type [
      let hh-list table:get hh-table dwt-type
      if length hh-list > 0 [
        let param-table first hh-list
        set hh-list but-first hh-list
        set household-id table:get param-table "id"
        set steply-net-income read-from-string (table:get param-table "income")
        set first-step-available ticks
        set capital-reserve read-from-string (table:get param-table "capital")
        set hedonism read-from-string (table:get param-table "hedonism")
        set gain-orientation read-from-string (table:get param-table "gain")
        set greenness read-from-string (table:get param-table "norm")
        set frame-adjustment read-from-string (table:get param-table "frame")
        set planning-horizon read-from-string (table:get param-table "planning")
        set resampled? true
      ]
      ifelse length hh-list > 0 [
        table:put hh-table dwt-type hh-list
      ]
      [
        table:remove hh-table dwt-type
      ]
    ]
    if table:length hh-table = 0 [
      table:remove named-in-migrants hh-type
    ]
  ]

  if not resampled? [
    if table:has-key? in-migrant-types hh-type [
      let hh-table table:get in-migrant-types hh-type
      if table:has-key? hh-table dwt-type [
        let param-table table:get hh-table dwt-type
        set next-id next-id + 1
        set household-id (word "household-" next-id)
;; Next line changed to read a list of values, not a distribution.
        set steply-net-income read-from-string (table:get param-table "income")
        set first-step-available ticks
        set capital-reserve sample (table:get param-table "capital")
;; The line below has two occurrences of "sample".
;; It should not have made any difference. Removed.
;;      set hedonism sample sample (table:get param-table "hedonism")
        set hedonism sample (table:get param-table "hedonism")
        set gain-orientation sample (table:get param-table "gain")
        set greenness sample (table:get param-table "norm")
        set frame-adjustment sample (table:get param-table "frame")
        set planning-horizon sample (table:get param-table "planning")
        set resampled? true
      ]
    ]
  ]

  ifelse resampled? [
    if use-social-links [
      ask my-social-links [
        die
      ]
      ;; need to resample social links...
      ifelse table:has-key? in-migrant-links household-id [
        make-social-links (table:get in-migrant-links name)
        table:remove in-migrant-links name
      ]
      [
        make-random-social-links
      ]
    ]

    ;; Reallocate appliances
    ask my-out-ownerships [
      die
    ]
    allocate-initial-appliances

    set wish-list []
  ]
  [
    output-print (word "*** Error: unable to resample parameters for household type "
      hh-type " and dwelling/tenure type " dwt-type)
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; calculate-moeu
;;
;; Calculate the steply overall energy use of a household. This procedure now
;; also deducts the price of that energy. Amended 2011-09-28 to deal with insulation.

to calculate-moeu
  let moeu 0 ;; this is now in kWh
  let ecost 0
  let hh-type household-type
  let dw-type [dwelling-type] of one-of out-address-neighbors
  let t-type [tenure] of one-of out-address-neighbors
  let umode usage-mode
  let my-insulation [insulation-factor] of one-of out-address-neighbors

  if count out-ownership-neighbors = 0 [
    output-print (word "*** Error: household \"" household-id "\" has no appliances ")
  ]

  ask out-ownership-neighbors [
    let consumptions out-consume-neighbors with [for-household-type = hh-type
      and for-dwelling-type = dw-type
      and for-tenure-type = t-type
      and in-usage-mode = umode
      and in-step = (ticks mod steps-per-year) + 1]
    if count consumptions = 0 [
      output-print (word "*** Error: appliance \"" name
        "\" doesn't use any consumption pattern for household type \"" hh-type
        "\", dwelling type \"" dw-type "\", tenure type \"" t-type
        "\", usage mode \"" umode "\" and step " ((ticks mod steps-per-year) + 1))
    ]

    ask consumptions [
      ;; for each fuel used by the appliance for any purpose this step
      let cons-ins-factor 1
      if for-purpose = "space-heating" [
        set cons-ins-factor my-insulation
      ]
      ask my-out-uses [
        let the-fuel other-end
        let energy-use (units-per-use * cons-ins-factor * [kWh-per-unit] of the-fuel)

        ask the-fuel [
          set total-kWh total-kWh + energy-use
        ]

        set moeu moeu + energy-use
        set ecost ecost + (units-per-use * table:get energy-price [fuel-type] of the-fuel)
      ]
    ]

  ]
  set steps-total-energy-use moeu
  set capital-reserve capital-reserve - ecost
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; calculate-current-running-cost
;;
;; An appliance reports its running cost (for fuel use) based on current prices

to-report calculate-current-running-cost [hh a-step]
  let hh-type [household-type] of hh
  let hh-dwelling one-of ([out-address-neighbors] of hh)
  let dw-type [dwelling-type] of hh-dwelling
  let t-type [tenure] of hh-dwelling
  let umode [usage-mode] of hh
  let dw-ins-factor [insulation-factor] of hh-dwelling

  let consumptions out-consume-neighbors with [for-household-type = hh-type
    and for-dwelling-type = dw-type
    and for-tenure-type = t-type
    and in-usage-mode = umode
    and in-step = (a-step mod steps-per-year) + 1]

  if count consumptions = 0 [
    output-print (word "*** Error: appliance \"" name
      "\" doesn't use any consumption pattern for household type \"" hh-type
      "\", dwelling type \"" dw-type "\", tenure type \"" t-type
      "\", usage mode \"" umode "\" and step " ((a-step mod steps-per-year) + 1))
  ]

  let total-cost 0

  ask consumptions [
    ;; for each fuel used by the appliance for any purpose this step
    let cons-ins-factor 1
    if for-purpose = "space-heating" [
      set cons-ins-factor dw-ins-factor
    ]
    ask my-out-uses [
      let the-fuel other-end

      set total-cost total-cost + (units-per-use * cons-ins-factor * table:get energy-price [fuel-
type] of the-fuel)
    ]
  ]

  report total-cost
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; calculate-projected-running-cost
;;
;; Calculate a running cost over a household's planning horizon, assuming
;; current energy prices

to-report calculate-projected-running-cost [hh]
  let hh-horizon [planning-horizon] of hh

  let total-cost 0

  let a-step ticks

  repeat hh-horizon [
    set a-step a-step + 1
    set total-cost total-cost + calculate-current-running-cost hh a-step
  ]

  report total-cost
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; calculate-finance
;;
;; Calculate the household's finance

to calculate-finance
;; Amended to allow for use of income time-series.
  set capital-reserve (capital-reserve + income-this-step)
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; calculate-breakdowns
;;
;; Calculate the breakdowns of pieces of equipment

to calculate-breakdowns
  ask households [
    ask my-out-ownerships [
      if random-float 1 < [breakdown-probability] of other-end [
        set broken? true
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; choose-goal-frame
;;
;; Choose a goal frame

to-report choose-goal-frame
  let goal-frame-parameter-total hedonism + gain-orientation + greenness
  let selection random-float goal-frame-parameter-total
  ifelse (selection < hedonism) [
    set hedonism hedonism + habit-adjustment-factor
    ifelse hedonism > goal-frame-parameter-total [
      set hedonism goal-frame-parameter-total
      set gain-orientation 0
      set greenness 0
    ]
    [
      set gain-orientation gain-orientation - (habit-adjustment-factor / 2)
      set greenness greenness - (habit-adjustment-factor / 2)
      ifelse gain-orientation < 0 [
        set greenness goal-frame-parameter-total - hedonism
        set gain-orientation 0
      ]
      [
        if greenness < 0 [
          set gain-orientation goal-frame-parameter-total - hedonism
          set greenness 0
        ]
      ]
    ]
    report "hedonistic"
  ]
  [
    ifelse (selection < hedonism + gain-orientation) [
      set gain-orientation gain-orientation + habit-adjustment-factor
      ifelse gain-orientation > goal-frame-parameter-total [
        set gain-orientation goal-frame-parameter-total
        set hedonism 0
        set greenness 0
      ]
      [
        set hedonism hedonism - (habit-adjustment-factor / 2)
        set greenness greenness - (habit-adjustment-factor / 2)
        ifelse hedonism < 0 [
          set greenness goal-frame-parameter-total - gain-orientation
          set hedonism 0
        ]
        [
          if greenness < 0 [
            set hedonism goal-frame-parameter-total - gain-orientation
            set greenness 0
          ]
        ]
      ]
      report "gain"
    ]
    [
      set greenness greenness + habit-adjustment-factor
      ifelse greenness > goal-frame-parameter-total [
        set greenness goal-frame-parameter-total
        set hedonism 0
        set gain-orientation 0
      ]
      [
        set hedonism hedonism - (habit-adjustment-factor / 2)
        set gain-orientation gain-orientation - (habit-adjustment-factor / 2)
        ifelse hedonism < 0 [
          set gain-orientation goal-frame-parameter-total - greenness
          set hedonism 0
        ]
        [
          if gain-orientation < 0 [
            set hedonism goal-frame-parameter-total - greenness
            set gain-orientation 0
          ]
        ]
      ]
      report "norm"
    ]
  ]
end
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; insulation-factor
;;
;; Return the insulation factor of a dwelling

to-report insulation-factor
  let factor 1.0

  ask in-insulate-neighbors [
    set factor factor * fuel-use-factor
  ]

  report factor
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; current-replacements-for
;;
;; Report an agent-set of the appliances that can replace the argument

to-report current-replacements-for [an-appliance]
  report current-appliances with [self = an-appliance
    or my-member? self ([out-replacement-neighbors] of an-appliance)]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; appliances-i-can-afford
;;
;; Report an agent-set of the appliances that a household can afford to buy.
;; Note that this includes appliances that may no longer or not yet be
;; available.

;;to-report appliances-i-can-afford
;;  let budget capital-reserve + steply-net-income * credit-multiple-limit
;;  report appliances with [cost <= budget]
;;end

to-report appliances-i-can-afford
;; Amended to allow use of income time-series
  let budget capital-reserve + income-this-step * credit-multiple-limit
  report appliances with [cost-this-step <= budget]
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; appliances-household-can-afford
;;
;; Report an agent-set of the appliances the household-agent can afford to buy.
;; Note that this includes appliances that may no longer or not yet be
;; available.

to-report appliances-household-can-afford [household-agent]
  report [appliances-i-can-afford] of household-agent
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; cost-this-step
;;

to-report cost-this-step
  let indexa ticks - first-step-available
  if (first-step-available = -1) [
    set indexa (indexa - 1)
  ]
  ;; Note that if the item is not yet available or no longer available, or if the cost-list has
fewer items
  ;; than the number of ticks for which it is available, the last element of the cost-list will be
returned.
  if (indexa < 0) or (length cost-list - 1 < indexa) [
    set indexa (length cost-list - 1)
  ]
  report item indexa cost-list
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; income-this-step
;;

to-report income-this-step
  let indexa ticks - first-step-available
  if (indexa < 0) or (length steply-net-income - 1 < indexa) [
    set indexa (length steply-net-income - 1)
  ]
  report item indexa steply-net-income
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;; current-appliances-i-can-afford
;;
;; Report an agent-set of the currently available appliances that a household
;; can afford to buy.

to-report current-appliances-i-can-afford
  report (appliances-household-can-afford self)
  with [first-step-available <= ticks and (last-step-available-unbounded? or last-step-available
>= ticks)]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; appliances-to-be-replaced
;;
;; Report an agent-set of appliances that are in the household's breakdown list

to-report appliances-to-be-replaced
  report appliances with [my-member? self ([breakdown-list] of myself)]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; current-replacements-for-appliances-to-be-replaced
;;
;; Use the methods above to return an agent-set of appliances that the household
;; passed as argument has to replace

to-report current-replacements-for-appliances-to-be-replaced [household-agent]
  let breakdown-set appliances with [my-member? self ([breakdown-list]
    of household-agent)]
  report current-appliances with
    [my-member? self ([out-replacement-neighbors] of breakdown-set)
      or my-member? self breakdown-set]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; current-replacements-for-my-appliances
;;
;; Report an agent set of current replacements for the appliances of a household

to-report current-replacements-for-my-appliances
  report (current-appliances with
    [my-member? self ([out-replacement-neighbors] of out-ownership-neighbors)
      or my-member? self out-ownership-neighbors])
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; appliances-i-dont-have-owned-by
;;
;; Report an agent set of appliances not owned by the household that another
;; household they visited does have

to-report appliances-i-dont-have-owned-by [some-one-i-visited]
  report ([out-ownership-neighbors] of some-one-i-visited) with
    [not my-member? self ([out-ownership-neighbors] of myself)]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; current-replacements-for-appliances-i-dont-have-owned-by
;;
;; Report an agent set of current replacements for appliances not owned by
;; the household that another household they visited does have

to-report current-replacements-for-appliances-i-dont-have-owned-by
  [some-one-i-visited]
  let joneses-appliances appliances-i-dont-have-owned-by some-one-i-visited
  report current-appliances with
    [my-member? self ([out-replacement-neighbors] of joneses-appliances)
      or my-member? self joneses-appliances]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; visit
;;
;; Visit another household; this essentially involves goal-frame adjustment

to visit [some-one]
  ask social-link-with some-one [
    set n-visits n-visits + 1
  ]
  ;; adjust values in the direction of those of the contact
  value-adjust some-one

  if (reciprocal-adjustment = true) [
    ;; adjust contact's values reciprocally
    ask some-one [value-adjust self]
  ]
end
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; hedonistic-ess-replace
;;
;; Replace a piece of essential equipment in hedonistic mode. If acting
;; hedonistically, household minimises the capital cost of replacing an
;; essential item (so as to maximise what is available for hedonistic spending)

;;to-report hedonistic-ess-replace [an-appliance]
;;  report one-of ((current-replacements-for an-appliance) with-min [cost])
;;end

to-report hedonistic-ess-replace [an-appliance]
  report one-of ((current-replacements-for an-appliance) with-min [cost-this-step])
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; heating-system-cost-advice
;;
;; Procedure representing the provision of advice to a household as to the
;; lowest financial cost option for replacing an appliance, based on its
;; expected energy use over the household's planning horizon.

to-report heating-system-cost-advice [hh an-appliance]
  let hh-horizon [planning-horizon] of hh
  let app-replacements current-replacements-for an-appliance

  if count app-replacements = 0 [
    report false
  ]

  let best-cost -1
  let best-app false


  ask app-replacements [
 ;;   let this-cost cost + calculate-projected-running-cost hh
    let this-cost cost-this-step + calculate-projected-running-cost hh
    ifelse best-cost = -1 [
      set best-app self
      set best-cost this-cost
    ]
    [
      if this-cost < best-cost [
        set best-app self
        set best-cost this-cost
      ]
    ]
  ]

  report best-app
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; gain-ess-replace
;;
;; Replace a piece of essential equipment in gain mode. This is based on cost
;; and running cost.

to-report gain-ess-replace [an-appliance]
  report one-of (current-replacements-for an-appliance)
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; norm-ess-replace
;;
;; Replace a piece of essential equipment in norm mode. This is based on
;; embodied and running energy cost.

to-report norm-ess-replace [an-appliance]
  report one-of ((current-replacements-for an-appliance) with-min [norm-cost])
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; gain-cost
;;
;; Report the gain cost of an appliance. This is just its purchase price

;;to-report gain-cost
;;  report cost
;;end

to-report gain-cost
  report cost-this-step
end
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; norm-cost
;;
;; Report the norm cost of an appliance. This is inversely proportional to
;; its energy rating

to-report norm-cost
  ifelse energy-rating-provided? [
  ;; Changed to correspond with carbon calculator used in survey
  ;; report 1 / energy-rating
    report energy-rating
  ]
  [
    report breakdown-probability
  ]
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; hedonistic-equip
;;
;; Purchase new appliances in hedonistic mode

to hedonistic-equip
  let choice-list []

  foreach (sentence wish-list breakdown-list) [
    set choice-list (sentence choice-list (sort current-replacements-for ?))
  ]

  set choice-list remove-duplicates choice-list


  ;; sort in *ascending* order of hedonic score...
  set choice-list sort-by [
    ([hedonic-score] of ?1) < ([hedonic-score] of ?2)
  ] choice-list

  let affordable-choice-list []
  foreach choice-list [
 ;; Next line amended to take account of both costs and incomes being time-series..
 ;;   if [cost] of ? < capital-reserve + (steply-net-income * credit-multiple-limit) [
      if [cost-this-step] of ? < capital-reserve + (income-this-step * credit-multiple-limit) [
      ;; ... because this will reverse the order
      set affordable-choice-list fput ? affordable-choice-list
    ]
  ]

  ;; Buy as many affordable things as possible
  ;; but not more than one from a category.
  while [(length affordable-choice-list > 0) and
 ;;   (capital-reserve - ([cost] of first choice-list)
      (capital-reserve - ([cost-this-step] of first choice-list)
 ;; Next line amended to allow for time series of incomes.
      > (income-this-step * (- credit-multiple-limit)))] [
    let new-item first choice-list
    add-item new-item 0

    ;; update the list of affordable things
    let new-choice-list []
    foreach but-first affordable-choice-list [
      if [category] of ? != [category] of new-item [
        set new-choice-list fput ? new-choice-list
      ]
    ]
    set affordable-choice-list new-choice-list
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; gain-equip
;;
;; Purchase new equipment in gain mode

to gain-equip
  let choice-set current-replacements-for-appliances-to-be-replaced self
  add-item (one-of choice-set with-min [gain-cost]) 0
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; norm-equip
;;
;; Purchase a new piece of equipment in norm mode

to norm-equip
  let choice-set current-replacements-for-appliances-to-be-replaced self
  add-item (one-of choice-set with-min [norm-cost]) 0
end
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; gain-insulation
;;
;; Add insulation in gain mode
;;

to gain-insulation
  ;; Choose the insulation state reachable from the current state that will save
  ;; the most money and make a positive monetary saving over the planning horizon.
  ;; Nick
  let dw one-of out-address-neighbors
  let current-insulations false
  ask dw [
    set current-insulations in-insulate-neighbors
  ]
  let old-insulation nobody
  let new-insulation nobody
  let saving 0
  let candidate-state false
  let candidate-cost 0
  let candidate-saving 0
  let current-fuel-use-factor [insulation-factor] of dw
  let candidate-fuel-use-factor 1
  let cost-of-upgrade 0
  let current-projected-space-heating-cost calculate-projected-space-heating-cost-over-planning-
horizon self
  ask current-insulations [
    let replaced-fuel-use-factor fuel-use-factor
    set old-insulation self
    ask my-out-upgrades [
      set candidate-cost upgrade-cost
      ask other-end [
        set candidate-fuel-use-factor current-fuel-use-factor * fuel-use-factor / replaced-fuel-
use-factor
        ;; The above assumes that fuel-use-factors act as a product (check for consistency with
insulation-factor)
        set candidate-state insulation-state
        set candidate-saving (current-projected-space-heating-cost * (1 - candidate-fuel-use-
factor / current-fuel-use-factor) - candidate-cost)
        if (candidate-saving > saving) [
          set saving candidate-saving
          set new-insulation self
          set cost-of-upgrade candidate-cost
        ]
      ]
    ]
  ]
  add-insulation dw old-insulation new-insulation cost-of-upgrade
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; norm-insulation
;;
;; Add insulation in norm mode

to norm-insulation
  ;; Choose the insulation state reachable from the current state that will
  ;; leave a positive capital-reserve and save the most energy
  ;; Nick
  let dw one-of out-address-neighbors
  let c-r capital-reserve
  let current-insulations false
  ask dw [
    set current-insulations in-insulate-neighbors
  ]
  let old-insulation nobody
  let new-insulation nobody
  let candidate-insulation false
  let saving 0
  let candidate-state false
  let candidate-cost 0
  let cost-of-upgrade 0
  let energy-saving-ratio 1
  let candidate-energy-saving-ratio 1
  ask current-insulations [
    let replaced-fuel-use-factor fuel-use-factor
    set old-insulation self
    ask my-out-upgrades with [upgrade-cost <= c-r] [
      set candidate-cost upgrade-cost
      ask other-end [
        set candidate-energy-saving-ratio fuel-use-factor / replaced-fuel-use-factor
        if (candidate-energy-saving-ratio < energy-saving-ratio) [
          set energy-saving-ratio candidate-energy-saving-ratio
          set new-insulation self
          set cost-of-upgrade candidate-cost
        ]
      ]
    ]
  ]
  add-insulation dw old-insulation new-insulation cost-of-upgrade
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; calculate-projected-space-heating-cost-over-planning-horizon
;;
;; Calculates the projected cost of space heating for a dwelling
;; over the houseg=hold's planning horizon
;; Nick

to-report calculate-projected-space-heating-cost-over-planning-horizon [hh]
  let hh-horizon [planning-horizon] of hh
  let total-cost 0
  let a-step ticks
  repeat hh-horizon [
    set a-step a-step + 1
    set total-cost total-cost + calculate-current-space-heating-cost hh a-step
  ]
  report total-cost * [insulation-factor] of (one-of [out-address-neighbors] of hh)
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; calculate-current-space-heating-cost
;;
;; Calculates the actual or projected cost of space heating of a dwelling
;; for a specific month, assuming current heating costs.
;; Nick

to-report calculate-current-space-heating-cost [hh a-step]
  let hh-type [household-type] of hh
  let hh-dwelling one-of ([out-address-neighbors] of hh)
  let dw-type [dwelling-type] of hh-dwelling
  let t-type [tenure] of hh-dwelling
  let umode [usage-mode] of hh
  let hs one-of out-ownership-neighbors with [category = "heating"]
  let current-cost 0
  ask hs [
    let consumption one-of out-consume-neighbors with [for-household-type = hh-type
    and for-dwelling-type = dw-type
    and for-tenure-type = t-type
    and for-purpose = "space-heating"
    and in-usage-mode = umode
    and in-step = (a-step mod steps-per-year) + 1]

    if consumption = nobody [
      output-print (word "*** Error: appliance \"" name
        "\" doesn't use any consumption pattern for household type \"" hh-type
        "\", dwelling type \"" dw-type "\", tenure type \"" t-type
        "\", usage mode \"" umode "\" and step " ((a-step mod steps-per-year) + 1))
    ]

    ask consumption [
      ask my-out-uses [
        let the-fuel other-end
        set current-cost units-per-use * table:get energy-price [fuel-type] of the-fuel

      ]
    ]
  ]
  report current-cost
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; add-insulation
;;
;; Add insulation to a dwelling

to add-insulation [dw old-insulation new-insulation cost-of-upgrade]
  if new-insulation != nobody [
    add-insulation-cost-free dw old-insulation new-insulation
    set capital-reserve (capital-reserve - cost-of-upgrade)
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; add-insulation-cost-free
;;
;; Add insulation to a dwelling cost-free
;; Currently, landlords do not insulate, so this will always be called from add-insulation

to add-insulation-cost-free [dw old-insulation new-insulation]
  if (new-insulation != nobody) [
    ask dw [
      ask my-in-insulates with [other-end = old-insulation] [
        die
      ]
      create-insulate-from new-insulation [
        set hidden? true
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; add-item
;;
;; Add a piece of equipment to a household

to add-item [new-item vintage]
  if (new-item != nobody) [
    add-item-cost-free new-item vintage

;;    set capital-reserve (capital-reserve - [cost] of new-item)
      set capital-reserve (capital-reserve - [cost-this-step] of new-item)
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; add-item-cost-free
;;
;; Add a piece of equipment to a household, without a charge

to add-item-cost-free [new-item vintage]
  if(new-item != nobody) [
;; Next line replaced 20111014 to allow households to retain multiple items with the same
function.
;; Numbers of items held by a household in any category  is limited by the maximum-in-category-
table.

;;  ask (my-out-ownerships with [my-member? other-end [in-replacement-neighbors] of new-item]) [

    let category1 [category] of new-item
    let hh-type [household-type] of self
    if (table:has-key? maximum-in-category-table hh-type) [
      let hh-type-table table:get maximum-in-category-table hh-type
      if (table:has-key? hh-type-table category1) [
        let category1-ownership-list []
        ask (my-out-ownerships with [[category] of other-end = category1]) [
          set category1-ownership-list fput self category1-ownership-list
        ]
        ;; Use of read-from-string below is a kludge: read-table2 reads the category limits in as
        ;; strings instead of integers.
        ;; if (length category1-ownership-list + 1 > read-from-string (table:get hh-type-table
category1)) [
        ;; Since maximum-in-category-table is the only one read in using read-table2, and list-to-
table
        ;; is only used by read-table2, the read-from-string has been put there. (GP)
        if (length category1-ownership-list + 1 > table:get hh-type-table category1) [
       ;; output-print (word "category1-ownership list: " category1-ownership-list)
          let sorted-list sort-by [[age] of ?1 > [age] of ?2] category1-ownership-list
          ;; Could alternatively remove the excess number of items - in practice, this
          ;; should make no difference, I think.
          ask first sorted-list [
            set land-fill fput other-end land-fill
            die
          ]
        ]
      ]
    ]

    create-ownership-to new-item [
      set hidden? true
      set broken? false
      set age vintage
    ]

    let updated-breakdown-list []
    foreach breakdown-list [
      if(? != new-item and not (my-member? ? [in-replacement-neighbors] of new-item)) [
        set updated-breakdown-list fput ? updated-breakdown-list
      ]
    ]
    set breakdown-list updated-breakdown-list

    let updated-wish-list []
    foreach wish-list [
      if(? != new-item and not (my-member? ? [in-replacement-neighbors] of new-item)) [
        set updated-wish-list fput ? updated-wish-list
      ]
    ]
    set wish-list updated-wish-list

    set steps-total-energy-use (steps-total-energy-use + [embodied-energy] of new-item)
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; update-links
;;
;; Update the social links of a household

to update-links
  ifelse (random 2 = 1) [
    lose-link
  ]
  [
    if (count social-link-neighbors <= max-links) [
      gain-link
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; lose-link
;;
;; A household drops a social link

to lose-link
  if (count social-link-neighbors > 0) [
    let weak-contacts (social-link-neighbors with-min
      [appliance-similarity out-ownership-neighbors [out-ownership-neighbors] of myself])

    set weak-contacts (weak-contacts with-max [block-distance [pxcor] of patch-here
      [pycor] of patch-here])

    let lose-contact one-of (weak-contacts with-min [[n-visits] of social-link-with myself])
    let link-to-lose (social-link-with lose-contact)
    ask link-to-lose [die]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; block-distance
;;
;; Report the block distance from the household to the *patch* co-ordinates
;; supplied as argument (bearing in mind that now multiple households can occupy
;; a patch, with randomly assigned coordinates.

to-report block-distance [x y]
  report (abs ([pxcor] of patch-here - x) + abs ([pycor] of patch-here - y))
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; gain-link
;;
;; Let a household gain a social link

to gain-link
  let strong-contacts (social-link-neighbors with-max
    [appliance-similarity out-ownership-neighbors [out-ownership-neighbors]
      of myself])

  ifelse (count strong-contacts > 0) [
    let intermediate (one-of strong-contacts)

    let poss-new-contacts shuffle (sort (other ([link-neighbors]
      of intermediate)))

    let poss-new-contacts2 []
    foreach poss-new-contacts [
      if (member? ? social-link-neighbors = false) [
        set poss-new-contacts2 (fput ? poss-new-contacts2)
      ]
    ]
    if (poss-new-contacts2 != []) [
      create-social-link-with (one-of poss-new-contacts2) [
        set n-visits 0
      ]
    ]
  ]
  [
    create-social-link-with (one-of (other households)) [
      set n-visits 0
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;; value-adjust
;;
;; Adjust the goal frames of the household

to value-adjust [contact]
  set hedonism hedonism + (frame-adjustment * ([hedonism] of contact - hedonism))

  set gain-orientation gain-orientation + (frame-adjustment
    * ([gain-orientation] of contact - gain-orientation))

  set greenness (frame-adjustment * ([greenness] of contact - greenness))

  set hedonism ifelse-value (hedonism < 0) [0] [hedonism]
  set gain-orientation ifelse-value (gain-orientation < 0) [0] [gain-orientation]
  set greenness ifelse-value (greenness < 0) [0] [greenness]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; appliance-similarity
;;
;; Report a measure of the similarity of two lists of equipment cea and ceb

to-report appliance-similarity [cea ceb]
  let shared appliances with [my-member? self cea and my-member? self ceb]

  let unshared appliances with [(my-member? self cea or my-member? self ceb)
    and not my-member? self shared]

;;  report (similarity-sum shared) - (similarity-sum unshared)
  report count (shared) - count (unshared)
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; similarity-sum
;;
;; This procedure uses the equipment similarity list to compute the sum of
;; similarities over a list of 3-element lists, each of which is an equipment
;; description such as: ["heating" "condensing-boiler" "condensing-boiler-type1"]

;; This procedure seems to be calculating a measure of the similarity
;; of members of a set of appliances to each other - which is not what I wanted. I have
;; substituted a simple count of number of shared minus number of unshared appliances in
;; appliance-similarity.

;;to-report similarity-sum [appliance-set]
;;  let simsum 0

;;  ask appliance-set [
;;    let app-A self
;;    ask appliance-set [
;;      if similarity-neighbor? app-A [
;;        set simsum simsum + [score] of similarity-with app-A
;;      ]
;;    ]
;;  ]

;;  report simsum
;;end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; intersection
;;
;; Report the intersection of two lists

to-report intersection [list1 list2]
  let outlist []
  foreach list1 [
    if (member? ? list2) [
      set outlist (fput ? outlist)
    ]
  ]
  report outlist
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; profile-setup
;;
;; Report the time used to set up the simulation

to-report profile-setup
  report profiler:inclusive-time "setup"
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; profile-go
;;
;; Report the time used to run the simulation (other than setting up)
```

```
to-report profile-go
  report profiler:inclusive-time "go"
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; mean-links
;;
;; Report the mean number of social links each time step over the course of the
;; simulation

to-report mean-links
  report total-links / ticks
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; n-household
;;
;; Report the number of households

to-report n-households
  report count households
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; show-changes
;;
;; Provide some graphical visualisation of the status of a household

to show-changes
  if max-energy-display > 0 [
    ask households [
      ifelse capital-reserve < 0 [
        set shape "face sad"
      ]
      [
        set shape "face happy"
      ]
      ifelse goal-frame = "hedonistic" [
        set color red
      ]
      [
        ifelse goal-frame = "gain" [
          set color blue
        ]
        [
          ifelse goal-frame = "norm" [
            set color green
          ]
          [
            set color yellow
          ]
        ]
      ]
    ]
    ;; Following lines altered to allow empty properties to exist.
    ask dwellings [
      ifelse (one-of in-address-neighbors != nobody) [
        let kwh [steps-total-energy-use] of one-of in-address-neighbors
        ifelse kwh >= max-energy-display [
          set color white
        ]
        [
          let ncol array:length dwelling-temp-colours
          let index int ((kwh * ncol) / max-energy-display)
          set color array:item dwelling-temp-colours index
        ]
      ]
      [set color gray
      ]
    ]

    ask social-links [
      set color ifelse-value (n-visits > 19) [ 9.9 ] [ n-visits / 2 ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                                ;;
;; CEDSS File I/O                                                                 ;;
;;                                                                                ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-table
;;
;; Read a table from a CSV file in the format key,value, with one pair per line
;; value may be anything netlogo can build from a string.

to-report read-table [table-file]
  let table table:make
  file-open table-file
  while [not file-at-end?] [
    let line file-read-line
    let data (split "," line)
    let key (first data)
    set data (but-first data)
    let value (first data)
    table:put table key value
  ]
  file-close
  report table
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-csv
;;
;; Read a CSV file, returning a list of rows, each row as a table of heading row
;; to cell entry

to-report read-csv [csv-file]
  let rows []
  file-open csv-file
  let headings (split "," file-read-line)

  let row 1
  while [not file-at-end?] [
    let line split "," file-read-line

    if length line != length headings [
      output-print (word "*** Error in file " csv-file ": there are "
        length headings " headings, and " length line " entries in row " row)
    ]

    let row-table table:make
    (foreach headings line [
      table:put row-table ?1 ?2
    ])
    set rows lput row-table rows
    set row row + 1
  ]
  file-close

  report rows
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-matrix
;;
;; Read a simple matrix with row and column headings and string entries. The
;; result returned is a table of tables. Use the row heading to access the
;; table from which the column heading name is used to access the entry. For
;; example:
;;
;; table:get (table:get result-of-this-procedure "row-heading") "column-heading"

to-report read-matrix [csv-file]
  let matrix-table table:make

  file-open csv-file
  let headings array:from-list (split "," file-read-line)

  while [not file-at-end?] [
    let line array:from-list (split "," file-read-line)
    let row-table table:make
    let i 1
    while [i < array:length line] [
      table:put row-table (array:item headings i) (array:item line i)
      set i i + 1
    ]
    table:put matrix-table (array:item line 0) row-table
  ]
  file-close

  report matrix-table
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-numeric-ts-matrix
;;
;; Read in a time series of matrices, each of which has a matching set of row
;; and column headings, with a list of extra columns. See the Information tab
;; (household transition matrix file) for more information. The result is a
;; list of tables of tables.
```

```
to-report read-numeric-ts-matrix [csv-file extra-cols]
  let table-list []

  file-open csv-file

  let last-step 0
  while [not file-at-end?] [
    let matrix-table table:make

    let headings array:from-list (split "," file-read-line)

    let tstep read-from-string (array:item headings 0)

    if tstep != last-step + 1 [
      output-print (word "*** Error in transition matrix time-series file " csv-file
        ": matrix for step " tstep " is not the next one after " last-step)
    ]

    set last-step tstep

    let row 1
    while [not file-at-end? and row < array:length headings - length extra-cols] [
      let line array:from-list (split "," file-read-line)
      let row-table table:make
      if (array:item headings row) != (array:item line 0) [
        output-print (word "*** Error in transition matrix time-series file " csv-file
          ": row heading \"" (array:item line 0) "\" does not match column heading \""
          (array:item headings row) "\"")
      ]
      let i 1
      while [i < array:length line] [
        table:put row-table (array:item headings i) (read-from-string (array:item line i))
        set i i + 1
      ]
      table:put matrix-table (array:item headings row) row-table
    ]

    set table-list fput matrix-table table-list
  ]
  file-close
  ;; output-print (word "table-list: " table-list)
  report table-list
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-dwellings-file
;;
;; Read in a dwellings file. The patch file and insulation file should have been
;; read first.

to read-dwellings-file [filename]
  if filename != false and filename != "null" and length filename > 0 [
    foreach (read-csv filename) [
      ask dwellings with [dwelling-id = (table:get ? "id")] [
        set dwelling-type table:get ? "type"
        set tenure table:get ? "tenure"
        if table:has-key? ? "shape" [
          set shape table:get ? "shape"
        ]

        let my-insulation-state insulations with [insulation-state = (table:get ? "insulation")
and insulation-dwelling-type = [dwelling-type] of myself]
        ifelse count my-insulation-state = 1 [
          create-insulates-from my-insulation-state [
            set hidden? true
          ]
        ]
        [
          output-print (word "*** Error in dwellings file " filename
            ": no insulations with insulation state \"" (table:get ? "insulation") "\" for
dwelling type \"" ([dwelling-type] of self) "\"")
        ]
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-insulation-file
;;
;; Read in the set of insulation states available for this run

to read-insulation-file [filename]
  if filename != false and filename != "null" and length filename > 0 [
    foreach (read-csv filename) [
      create-insulations 1 [
        set insulation-state table:get ? "insulation-state"
        set fuel-use-factor read-from-string table:get ? "fuel-use-factor"
        set insulation-dwelling-type table:get ? "dwelling-type"
        set hidden? true
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-insulation-upgrade-file
;;
;; Read in the insulation upgrade file. This lists the upgrade options available
;; for each dwelling type.


to read-insulation-upgrade-file [filename]
  if filename != false and filename != "null" and length filename > 0 [
    foreach (read-csv filename) [
      let dw-type table:get ? "dwelling-type"

      let from-state insulations with [insulation-state = (table:get ? "from-state")
        and insulation-dwelling-type = dw-type]
      let to-state insulations with [insulation-state = (table:get ? "to-state")
        and insulation-dwelling-type = dw-type]

      if count from-state != 1 [
        output-print (word "*** Error in insulation file " filename
          ": not 1 insulation for state " (table:get ? "from-state")
          " and dwelling type " dw-type)
      ]

      if count to-state != 1 [
        output-print (word "*** Error in insulation file " filename
          ": not 1 insulation for state " (table:get ? "to-state")
          " and dwelling type " dw-type)
      ]

      ask from-state [
        create-upgrades-to to-state [
          set upgrade-cost read-from-string table:get ? "cost"
          set hidden? true
        ]
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-insulation-update-file
;;
;; Read in a list of insulation updates

to read-insulation-update-file [filename]
  set insulation-updates []

  if filename != false and filename != "null" and length filename > 0 [
    foreach (read-csv filename) [
      set insulation-updates fput ? insulation-updates
    ]
    set insulation-updates reverse insulation-updates
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-households-file
;;
;; Read in the households file. The dwellings file should have been read first.

to read-households-file [filename]
  if filename != false and filename != "null" and length filename > 0 [
    foreach (read-csv filename) [
      create-households 1 [
        set household-id table:get ? "id"
        set household-type table:get ? "type"
        set steply-net-income read-from-string (table:get ? "income")
;; Next line added to allow income time series.
        set first-step-available 0
        set capital-reserve read-from-string (table:get ? "capital")
        set hedonism read-from-string (table:get ? "hedonism")
        set gain-orientation read-from-string (table:get ? "gain")
        set greenness read-from-string (table:get ? "norm")
        set frame-adjustment read-from-string (table:get ? "frame")
        set planning-horizon read-from-string (table:get ? "planning")
        let dwelling dwellings with [dwelling-id = (table:get ? "dwelling")]
        ifelse count dwelling = 1 [
          create-address-to one-of dwelling [
            set hidden? true
          ]
        ]
        [
          output-print (word "*** Error in household file " filename
            ": Unexpected number of dwellings with id "
            (table:get ? "dwelling") " (expected 1)")
        ]
        set-household-nlogo-params
        ;; The following added to allow
        let line-table ?
        let num 1
        while [table:has-key? line-table (word "initial-item " num)] [
          add-item-cost-free (table:get line-table (word "initial-item " num))
                  (table:get line-table (word "initial-age " num))
          set num num + 1
        ]
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-in-migrant-file
;;
;; Read in the in-migrant households file.

to read-in-migrant-file [filename]
  if filename != false and filename != "null" and length filename > 0 [
    set in-migrant-types table:make
    set named-in-migrants table:make

    foreach (read-csv filename) [
      ifelse table:get ? "id" = "*" [
        if not (table:has-key? in-migrant-types (table:get ? "type")) [
          table:put in-migrant-types (table:get ? "type") table:make
        ]
        if not member? (table:get ? "type") household-types-list [
          set household-types-list fput (table:get ? "type") household-types-list
        ]
        table:put (table:get in-migrant-types (table:get ? "type"))
          (table:get ? "dwelling-type") ?
        ;; So, in-migrant-types is HH type -> dwelling type -> parameter table...
      ]
      [ ;; it's a named household with specific parameters
        if not (table:has-key? named-in-migrants (table:get ? "type")) [
          table:put named-in-migrants (table:get ? "type") table:make
        ]

        let hh-table (table:get named-in-migrants (table:get ? "type"))

        if not (table:has-key? hh-table (table:get ? "dwelling")) [
          table:put hh-table (table:get ? "dwelling") []
        ]

        let hh-list (table:get hh-table (table:get ? "dwelling"))

        table:put hh-table (table:get ? "dwelling-type") (lput ? hh-list)
        ;; ...while named-in-migrants is HH type -> dwelling type -> list of
        ;; parameter tables
      ]
      if not member? (table:get ? "dwelling-type") dwelling-types-list [
        set dwelling-types-list fput (table:get ? "dwelling-type") dwelling-types-list
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-social-link-file
;;
;; Read in the social links. This assumes that the household file has already
;; been read

to read-social-link-file [filename]
  if filename != false and filename != "null" and length filename > 0 [
    file-open filename

    while [not file-at-end?] [
      let line (split "," file-read-line)
      let hh households with [name = first line]
      ifelse count hh = 0 [
        table:put in-migrant-links (first line) (but-first line)
      ]
      [
        ask hh [
          make-social-links but-first line
        ]
      ]
    ]
    file-close
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-social-link-matrix-file
;;
;; Read in the social link matrix file. This contains information on how to
;; create links randomly

to read-social-link-matrix-file [filename]
  if filename != false and filename != "null" and length filename > 0 [
    set patch-links table:make
    set link-radii-list []
    set radius-links table:make
    set link-patch-types-list []
    set patch-type-links table:make

    let first-row? true

    foreach (read-csv filename) [
      if first-row? [
        foreach table:keys ? [
          let parse-heading array:from-list (split " " ?)
          if array:item parse-heading 0 = "radius" [
            set link-radii-list (lput
              read-from-string (array:item parse-heading 1) link-radii-list)
          ]

          if array:item parse-heading 0 = "type" [
            set link-patch-types-list (lput
              (array:item parse-heading 1) link-patch-types-list)
          ]
        ]

        set first-row? false
      ]

      ;; GP The following two if statements (without the wildcard check) used to
      ;; be in the first-row? statement block above. I don't see why the first (non-heading)
      ;; row of the input file should be allowed to add dwelling types, and not the other
      ;; rows.
      if (not ((table:get ? "A-dwelling") = "*") and (not member? (table:get ? "A-dwelling")
dwelling-types-list)) [
        set dwelling-types-list fput (table:get ? "A-dwelling") dwelling-types-list
      ]
      if (not ((table:get ? "B-dwelling") = "*") and (not member? (table:get ? "B-dwelling")
dwelling-types-list)) [
        set dwelling-types-list fput (table:get ? "B-dwelling") dwelling-types-list
      ]

      if (not ((table:get ? "A-type") = "*") and (not member? (table:get ? "A-type") household-
types-list)) [
        set household-types-list fput (table:get ? "A-type") household-types-list
      ]
      if (not ((table:get ? "B-type") = "*") and (not member? (table:get ? "B-type") household-
types-list)) [
        set household-types-list fput (table:get ? "B-type") household-types-list
      ]

      let A-type (word (table:get ? "A-type") ":" (table:get ? "A-dwelling"))
      let B-type (word (table:get ? "B-type") ":" (table:get ? "B-dwelling"))

      if not table:has-key? patch-links A-type [
        table:put patch-links A-type table:make
        if length link-radii-list > 0 [
          table:put radius-links A-type table:make
        ]
        if length link-patch-types-list > 0 [
          table:put patch-type-links A-type table:make
        ]
      ]

      let A-patch-table table:get patch-links A-type
      let A-radius-table false
      if length link-radii-list > 0 [
        set A-radius-table table:get radius-links A-type
        table:put A-radius-table B-type []
      ]
      let A-type-table false
      if length link-patch-types-list > 0 [
        set A-type-table table:get patch-type-links A-type
        table:put A-type-table B-type []
      ]

      ifelse table:has-key? A-patch-table B-type [
        output-print (word "*** Error in social link matrix file " filename ": parameters linking
"
```

```
          "household:dwelling " A-type " to " B-type " have already been specified")
      ]
      [
        table:put A-patch-table B-type (table:get ? "p-patch")

        let row ?

        foreach link-radii-list [
          let key (word "radius " ?)
          table:put A-radius-table B-type
            (lput (table:get row key) (table:get A-radius-table B-type))
        ]

        foreach link-patch-types-list [
          let key (word "type " ?)
          table:put A-type-table B-type
            (lput (table:get row key) (table:get A-type-table B-type))
        ]
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-appliances [filename]
;;
;; read in the appliances from the filename

to read-appliances [filename]
  if filename != false and filename != "null" and length filename > 0 [
    foreach (read-csv filename) [
  ;;    output-print (word "processing line with name" table:get ? "name")
      ifelse count appliances with [name = table:get ? "name"] > 0 [
        output-print (word "*** Error: more than one appliance with name \""
          (table:get ? "name") "\" in appliances file " filename)
      ]
      [
        create-appliances 1 [
          set hidden? true
          set name table:get ? "name"
          set category table:get ? "category"
          set subcategory table:get ? "subcategory"
          set essential? read-from-string (table:get ? "essential")
          set hedonic-score read-from-string (table:get ? "hedonic-score")
 ;;         set cost read-from-string (table:get ? "cost")
          set cost-list read-from-string (table:get ? "cost-list")

          let energy-rating-str (table:get ? "energy-rating")
          ifelse energy-rating-str = "NA" [
            set energy-rating-provided? false
          ]
          [
            set energy-rating-provided? true
            set energy-rating read-from-string energy-rating-str
          ]

          set embodied-energy read-from-string (table:get ? "embodied-energy")
          set breakdown-probability read-from-string
          (table:get ? "breakdown-probability")
          set first-step-available read-from-string
          (table:get ? "first-step-available")

          let last-step-available-str (table:get ? "last-step-available")

          ifelse last-step-available-str = "Inf" [
            set last-step-available-unbounded? true
          ]
          [
            set last-step-available-unbounded? false
            set last-step-available read-from-string last-step-available-str
          ]
        ]
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-replacements [filename]
;;
;; Read the equipment replacements from the file. This is a CSV file without
;; headings in which the first column is the name of an appliance, and the
;; remaining columns for that row are the names of all the appliances that can
;; replace that appliance. The replacements file must be read after the
;; appliances file

to read-replacements [filename]
  if filename != false and filename != "null" and length filename > 0 [
    file-open filename
    while [not file-at-end?] [
      let line split "," file-read-line
      ask appliances with [name = first line] [
;;        output-print (word "processing line with name " first line)
        foreach but-first line [
          let other-appliance one-of appliances with [name = ?]
          if other-appliance != self [
;; Preceding line added, following lines commented out for convenience in constructing input
files. Since a check is
;; always made that a possible replacement is available at the current step, it does not matter
;; for the functioning of the program if an item recorded as a possible replacement is not
available
;; any time the item might need replacing; while if self is found in the list of replaciements, it
is now to be ignored.
;;          ifelse other-appliance != self [
;;            ifelse not ([last-step-available-unbounded?] of other-appliance)
;;            and [last-step-available] of other-appliance < first-step-available [
;;              output-print (word "*** Error in replacements file " filename ": appliance \""
;;                ? "\" cannot act as a replacement for appliance \"" name
;;                "\" because its last-step-available (" ([last-step-available] of
;;                  other-appliance) ") is before " name "'s first-step-available ("
;;                first-step-available ")")
;;            ]
;;            [
            create-replacement-to other-appliance [
              set hidden? true
;;              ]
            ]
          ]
;;          [
;;            output-print (word "*** Error in replacements file " filename ": appliance \""
;;              ? "\" is listed as a replacement for itself")
;;          ]
        ]
      ]
    ]
    file-close
  ]
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-energy-suppliers [filename]
;;
;; read in the list of energy suppliers for each fuel type. This will read a
;; list of fuel-type to price tables into the energy-price-list. If there are
;; multiple suppliers for one fuel, the cheapest will be selected. The fuels
;; file must have been read in first.

to read-energy-suppliers [filename]
  set energy-price-list []
  if filename != false and filename != "null" and length filename > 0 [
    file-open filename
    if file-at-end? [
      output-print (word "*** Error: Suppliers file " filename " is empty!")
    ]
    let dummy file-read-line ;; ignore suppliers

    if file-at-end? [
      output-print (word "*** Error: Suppliers file " filename " has no fuels!")
    ]
    let fuels-list split "," file-read-line

    ;; Before reading in the data, check the fuels have been defined

    foreach fuels-list [
      if count fuels with [fuel-type = ?] = 0 [
        output-print (word "*** Error: No fuel defined with type \"" ? "\"")
      ]
    ]

    ;; Now read in the data

    let fuels-arr array:from-list fuels-list

    while [ not file-at-end? ] [
      let prices array:from-list split "," file-read-line
      let cheapest-prices table:make
      let i 0
      while [ i < array:length fuels-arr ] [
        let this-fuel array:item fuels-arr i
        let this-price read-from-string array:item prices i

        ifelse table:has-key? cheapest-prices this-fuel [
          table:put cheapest-prices this-fuel ifelse-value
            (this-price < table:get cheapest-prices this-fuel)
            [this-price] [table:get cheapest-prices this-fuel]
        ]
        [
          table:put cheapest-prices this-fuel this-price
        ]
        set i i + 1;
      ]
      set energy-price-list lput cheapest-prices energy-price-list
    ]

    file-close
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-appliance-similarity [filename]
;;
;; Read the equipment similarity file. This file defines how "similar" different
;; items are, for use in gaining and losing social links. The file can be per
;; category/subcategory or per appliance.

;; Not currently in use

;;to read-appliance-similarity [filename]
;;  if filename != false and filename != "null" and length filename > 0 [
;;    foreach (read-csv filename) [
;;      ifelse table:has-key? ? "category A" [
;;       let cat-A table:get ? "category A"
;;        let cat-B table:get ? "category B"
;;        let sub-A table:get ? "subcategory A"
;;        let sub-B table:get ? "subcategory B"

;;        let apps-A appliances with [category = cat-A and subcategory = sub-A]
;;        let apps-B appliances with [category = cat-B and subcategory = sub-B]

;;        ifelse (count apps-A > 0 and count apps-B > 0) [
;;          ask apps-A [
;;            create-similarities-with apps-B [
;;              set hidden? true
;;              set score read-from-string table:get ? "similarity"
;;            ]
;;          ]
;;        ]
;;        [
;;          if count apps-A = 0 [
;;            output-print (word "*** Warning reading " filename
;;              ": no appliances with category \"" cat-A "\" and subcategory \""
;;              sub-A "\"")
;;          ]
;;          if count apps-B = 0 [
;;            output-print (word "*** Warning reading " filename
;;              ": no appliances with category \"" cat-B "\" and subcategory \""
;;              sub-B "\"")
;;          ]
;;        ]
;;      ]
;;      [
        ;; The file lists pairs of appliances
;;        let name-A table:get ? "appliance A"
;;        let name-B table:get ? "appliance B"

;;        let app-A one-of appliances with [name = name-A]
;;        let app-B one-of appliances with [name = name-B]

;;        ifelse (app-A != nobody and app-B != nobody and app-A != app-B) [
;;          ask app-A [
;;            ifelse similarity-neighbor? app-B [
;;              if [score] of similarity-with app-B != table:get ? "similarity" [
;;                output-print (word "*** Error reading " filename
;;                  ": inconsistent similarities between appliances named \""
;;                  name-A "\" and \"" name-B "\"")
;;              ]
;;            ]
;;            [
;;              create-similarity-with app-B [
;;                set hidden? true
;;                set score read-from-string table:get ? "similarity"
;;              ]
;;            ]
;;          ]
;;        ]
;;        [
;;          if app-A = nobody [
;;            output-print (word "*** Error reading " filename
;;             ": cannot find appliance named \"" name-A "\"")
;;          ]
;;          if app-B = nobody [
;;            output-print (word "*** Error reading " filename
;;              ": cannot find appliance named \"" name-B "\"")
;;          ]
;;        ]
;;      ]
;;    ]
;;  ]
;;end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-fuel [filename]
;;
;; Read fuel types from the file. This is a CSV file with two heading columns:
;; type and kWh. One fuel agent will be created for each row. Fuel type name
;; must be unique

to read-fuel [filename]
  if filename != false and filename != "null" and length filename > 0 [
    foreach (read-csv filename) [
      ifelse count fuels with [fuel-type = (table:get ? "type")] > 0 [
        output-print (word "*** Error in fuel file " filename
          ": More than one fuel with type \"" (table:get ? "type") "\"")
      ]
      [
        create-fuels 1 [
          set hidden? true
          set fuel-type table:get ? "type"
          set unit table:get ? "unit"
          set kWh-per-unit read-from-string (table:get ? "kWh")
          set fuel-plot-colour ifelse-value table:has-key? ? "colour" [
            table:get ? "colour"
          ]
          [
            black
          ]
        ]
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-appliances-fuel-use [filename]
;;
;; Read the fuel use of each appliance. This populates the consumes links. The
;; file format is CSV, where each row corresponds to a usage of an appliance in
;; a particular context. The fuel file must have been read first. The CSV file
;; must have a headings row with at least the following headings: appliance,
;; fuel, household, dwelling, purpose, step, units

to read-appliances-fuel-use [filename]
  if filename != false and filename != "null" and length filename > 0 [
    foreach (read-csv filename) [
      let this-appliance appliances with [name = table:get ? "appliance"]

      if count this-appliance = 0 [
        output-print (word "*** Error reading appliances fuel file \"" filename
          "\": no appliances with name \"" (table:get ? "appliance") "\"")
      ]
      if count this-appliance > 1 [
        output-print (word "*** Error reading appliances fuel file \"" filename
          "\": there are two or more appliances with name \""
          (table:get ? "appliance") "\"")
      ]

;; The next two lines did not work if a wild card was used, because household-types-list is
;; defined in setup-households, which is now read after setup-households. This needs a
;; more permanent bug-fix, but for now refrain from using the wild card here.
;;      let hhtlist ifelse-value (table:get ? "household" = "*")
;;        [household-types-list] [ (list (table:get ? "household")) ]
      let hhtlist (list (table:get ? "household"))
;; output-print (word "hhtlist: " hhtlist)
      let dwtlist ifelse-value (table:get ? "dwelling" = "*")
        [dwelling-types-list] [ (list (table:get ? "dwelling")) ]
;; output-print (word "dwtlist: " dwtlist)
      let ttlist ifelse-value (table:get ? "tenure" = "*")
        [tenure-types-list] [ (list (table:get ? "tenure")) ]
;; output-print (word "ttlist: " ttlist)
      let uselist ifelse-value (table:get ? "mode" = "*")
        [usage-modes-list] [ (list (table:get ? "mode")) ]
;; output-print (word "uselist: " uselist)
      let line-table ?

      ;;; Now do a foreach loop through all the nested loops

      foreach hhtlist [
        let hht ?

        foreach dwtlist [
          let dwt ?
          foreach ttlist [
            let tt ?

            foreach uselist [
              let use ?

              foreach steps-list [
                let mth ?

                let the-fuel fuels with [fuel-type = table:get line-table "fuel"]

                if count the-fuel = 0 [
                  output-print (word "*** Error reading appliances fuel file \""
                    filename "\": There are no fuels with type \""
                    table:get line-table "fuel" "\"")
                ]
                if count the-fuel > 1 [
                  output-print (word "*** Error reading appliances fuel file \""
                    filename "\": There are two or more fuels with type \""
                    table:get line-table "fuel" "\"")
                ]

                create-consumption-patterns 1 [
                  set hidden? true
                  set for-household-type hht
                  set for-dwelling-type dwt
                  set for-tenure-type tt
                  set for-purpose table:get line-table "purpose"
                  set in-usage-mode use
                  set in-step mth

                  create-use-to one-of the-fuel [
                    set hidden? true

                    if not table:has-key? line-table (word "units " mth) [
                      output-print (word "*** Error reading appliances fuel file \""
                        filename "\": No column data for step " mth
                        ". Check steps-per-year -- currently " steps-per-year)
                    ]
                    set units-per-use read-from-string table:get line-table (word "units " mth)
                  ]
                  create-consume-from one-of this-appliance [
                    set hidden? true
                  ]
                ]
              ]
            ]
          ]
        ]
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-initial-appliances-file [filename]
;;
;; Read the household-init-appliance-file

to read-initial-appliances-file [filename]
  if filename != false and filename != "null" and length filename > 0 [
    set initial-hh-appliances table:make
    set initial-hh-address-appliances table:make
    set initial-hh-dw-type-appliances table:make

    file-open filename
    while [not file-at-end?] [
      let data (split-no-null "," file-read-line)

      let hh-dw-id (split-no-null ":" (first data))
      ifelse length hh-dw-id = 1 [
        ;; it's a household name -- add it to initial-hh-appliances

        table:put initial-hh-appliances (first data) (but-first data)
      ]
      [
        ifelse length hh-dw-id = 2 [
          ;; it's a household-type:dwelling-name -- add it to initial-hh-address-appliances

          table:put initial-hh-address-appliances (first data) (but-first data)
        ]
        [
          ifelse length hh-dw-id = 3 [
            ;; it's a household-type:tenure:dwelling-type -- add it to initial-hh-dw-type-
appliances

            table:put initial-hh-dw-type-appliances (first data) (but-first data)
          ]
          [
            output-print (word "*** Warning: invalid household/dwelling identifier \""
              (first data) "\" in household-init-appliance-file " filename " -- "
              "ignoring this line")
          ]
        ]
      ]
    ]
    file-close
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-patch-layout [filename]
;;
;; The patch layout file is now a CSV file, which specifies the type of each
;; patch. If the type is 'dwelling', then a dwelling agent is sprouted at that
;; patch, and the patch may have several dwellings on it.

to read-patch-layout [filename]
  if filename != false and filename != "null" and length filename > 0 [
    file-open filename
    while [not file-at-end?] [
      let data (array:from-list (split-no-null "," file-read-line))

      let x read-from-string array:item data 0
      let y read-from-string array:item data 1
      let ptype array:item data 2

      ask patch x y [
        set pcolor table:get patch-legend ptype
        set patch-type ptype
      ]

      if ptype = "dwelling" [
        let i 3
        while [i < array:length data] [
          ask patch x y [
            sprout-dwellings 1 [
              set dwelling-id array:item data i
              set shape "house"
              ;; Give the dwellings a random perturbation so we can see them if
              ;; there are two or more dwellings on a patch
              set xcor x
              set ycor y
            ]
            if i > 3 and pcolor mod 9 > 0 [
              set pcolor pcolor + 1
            ]
          ]
          set i i + 1
        ]
      ]
    ]
    file-close
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-table2
;;
;; Creates a two-level table from a file in which each line encodes a table as
;; alternating keys and values. The first item in a line is a key for the main
;; table, the remaining items are alternating keys and values for a subtable.

to-report read-table2 [table-file]
  let table table:make
  file-open table-file
  while [not file-at-end?] [
    let line file-read-line
    let data (split "," line)
    let key (first data)
    let value (list-to-table but-first data)
    table:put table key value
  ]
  report table
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;; Utilities                                                                  ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; make-social-links [other-hh]
```

```
;;
;; Make all the social links from this household to households in the list

to make-social-links [other-hh]
  while [length other-hh > 0] [
    if not name = first other-hh [
      let hh one-of households with [name = first other-hh]
      if not hh = nobody and not social-link-neighbor? hh [
        create-social-link-with hh
        set other-hh but-first other-hh
      ]
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; sample [string]
;;
;; String is a sampled formatted string containing a distribution and parameters
;; for it in order

to-report sample [string]
  if is-number? string [
    report string
  ]
  let distribution array:from-list split " " string
  let i 1
  while [i < array:length distribution] [
    array:set distribution i (read-from-string (array:item distribution i))
    set i i + 1
  ]
  if array:item distribution 0 = "uniform" [
    let minimum array:item distribution 1
    let maximum array:item distribution 2
    report minimum + random-float (maximum - minimum)
  ]
  if array:item distribution 0 = "uniform-integer" [
    let minimum array:item distribution 1
    let maximum array:item distribution 2
    report minimum + random (1 + minimum - maximum)
  ]
  if array:item distribution 0 = "normal" [
    report random-normal (array:item distribution 1) (array:item distribution 2)
  ]
  if array:item distribution 0 = "poisson" [
    report random-poisson (array:item distribution 1)
  ]
  if array:item distribution 0 = "exponential" [
    report random-exponential (array:item distribution 1)
  ]
  if array:item distribution 0 = "gamma" [
    report random-gamma (array:item distribution 1) (array:item distribution 2)
  ]
  report read-from-string array:item distribution 0
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; assign-replacements [table]
;;
;; assign replacements to appliances using the hash table, with key the name
;; of the appliance, and value a list of names of appliances it replaces

;; This procedure is not used.
;;to assign-replacements [table]
;;  ask appliances [
;;    if table:has-key? table name [
;;      let replacement-list table:get table name
;;      foreach replacement-list [
;;        let other-appliance appliances with [name = ?]
;;        if count other-appliance = 1 and other-appliance != self [
;;          create-replacement-to one-of other-appliance [
;;            set hidden? true
;;          ]
;;        ]
;;      ]
;;    ]
;;  ]
;;end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; read-text [text]
;;
;; read the text given as the argument from the file, and print an error message
;; if that text is not read

to read-text [text]
  let text-read file-read-characters length text
  if text-read != text [
    output-print (word "*** Error in currently open file: expecting \"" text "\", found \""
      text-read "\"")
    stop
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; my-member?
;;
;; Return the whether or not an item is a member of a collection, allowing for
;; the possibility that either the item or the collection may be nobody
to-report my-member? [an-item a-collection]
  report ifelse-value (an-item = nobody) [
    false
  ]
  [
    ifelse-value (a-collection = nobody) [
      false
    ]
    [
      member? an-item a-collection
    ]
  ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; split
;;
;; Separate some text into a list of strings delimited by the separator. Why
;; NetLogo doesn't have a string function like this in its dictionary is a
;; mystery. Maybe it does and I couldn't find it :-).
;;
;; This procedure is copied from the LOCAWv1.nlogo model

to-report split [separator text]
  let cells []
  let mytext text
  while [position separator mytext != false] [
    set cells fput (substring mytext 0 (position separator mytext)) cells
    set mytext substring mytext ((position separator mytext)
      + length separator) length mytext
  ]
  set cells fput mytext cells
  report reverse cells
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; split-no-null
;;
;; Split and remove null entries

to-report split-no-null [separator text]
  let cells split separator text
  let no-nulls []

  foreach cells [
    if length ? > 0 [
      set no-nulls fput ? no-nulls
    ]
  ]

  report reverse no-nulls
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; list-to-table
;;
;; Returns a table made by interpreting a list as alternating keys and values

to-report list-to-table [list1]
  let table table:make
  while [length list1 > 0] [
    table:put table (first list1) read-from-string (first but-first list1)
    set list1 but-first (but-first list1)
  ]
  report table
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; show-licence-message
;;
;; show the GNU GPL message when the model runs

to show-licence-message
  print "CEDSS 3.0  Copyright (C) 2010  Macaulay Land Use Research Institute"
  print "This program comes with ABSOLUTELY NO WARRANTY. This is free software,"
  print "and you are welcome to redistribute it under certain conditions; for"
  print "more information on this, and the (lack of) warranty, see the LICENCE"
  print "section in the Information tab."
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; my-export-all-plots
;;
;; Export all plots to a file that is guaranteed not to exist

to my-export-all-plots [filename]
  ifelse file-exists? filename [
    let stem substring filename 0 (length filename - 4)
    let x 0
    set filename (word stem "-" x ".csv")

    while [file-exists? filename] [
      set x x + 1
      set filename (word stem "-" x ".csv")
    ]
    export-all-plots filename
  ]
  [
    export-all-plots filename
  ]
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;; Condition rules for the usage matrix                                       ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; negative-capital-reserve

to-report negative-capital-reserve
  ifelse capital-reserve < 0 [
    report true
  ]
  [
    report false
  ]
end
```