European Commission

Information Society and Media Directorate – General

Model of local energy resource cluster

SmartCoDe

Project No.:	ICT-2009-247473
Deliverable No.:	D-1.1.2
Deliverable Title:	Model of local energy resource cluster
Due Date:	Dec 31th, 2012

Nature:	Report
Dissemination Level:	public
Author:	Markus Damm, Roland Kopetz-ky, Peter Neumann
Lead Beneficiary No.:	3
Lead Beneficiary:	TUV

SmartCoDe has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013]) under grant agreement n° 247473

Table of Contents

1 Introduction

This document sums up the developments regarding the foundations of the SmartCoDe work package 1 since the submission of Deliverable D-1.1.1. The idea of D-1.1.1 was to define the basic components in the home/small businesses demand side management scenario as a base for all further considerations regarding energy management. D-1.1.1 contained

- a simple classification of Energy using Products (EuPs) into seven classes. It bundled all the EuPs together which could be treated in a similar fashion regarding interfaces and energy management approach with respect to the SmartCoDe application scenario.
- a similar classification of Local Energy Providers (LEPs) into 4 classes, with initial models for wind- and solar power prediction.
- first thoughts on the general approach of the SmartCoDe energy management by contrasting a classical centralised approach with a more innovative decentralized approach.

The last point probably saw the most consideration in the following months, and in D-1.1.1 the decentralized approach was already favoured. The final outcome was called a *semi-decentralised* approach, since it became obvious that even when shifting a lot of the decision-making from the Energy Management Unit (EMU) to the SmartCoDe nodes, the EMU still has a lot of opportunity (and responsibility) to influence the nodes' local decisions.

The rest of this document is organised as follows: In Section 2 the final outcome regarding the global energy management approach is shown. Section 3 discusses the EuP classification, with a focus of the final local energy-management algorithms. Section 4 gives a short overview of the results regarding the LEPs, before concluding in Section 5.

Note that parts of this document have been presented already in previous Deliverables and Reports. The intention of this document is to collect all the relevant information together in order to draw a coherent and concise picture of the fundamental research done in work package 1. Also, there are some adaptions made due to practical experience which haven't been documented yet which will be presented here.

2 Energy management

One of the first decisions to be made in work package 1 regarding the energy management was how much of the computing power of the wireless nodes (the SmartCoDe nodes) to use for the energy management process. Since there was formidable computing power available at the node level, which was needed anyway for running the ZigBee stack alone, the idea of using some kind of distributed energy management approach evolved, where the EMU would broadcast a message indicating at which times it would be best to use more or less energy, respectively.

The alternative was to take the simple path of using the SmartCoDe nodes merely as receivers of on-off signals (or more than two power levels if available), with the EMU making all the decisions by micro-managing every EuP. However, in the course of the EuP classification, which was done in parallel (see Section 3), it became apparent that this approach would create different problems.

For example, if a fridge would be controlled by the EMU with simple on-off signals, the fridges temperature is a major basis for the decision making, and therefore has to be transmitted through the whole network to the EMU constantly. Although this would not be a problem in the SmartCoDe demonstrator, in networks of bigger size this might cause bandwidth problems at some point. Also, if there is an EMU crash or a network breakdown, there has to be a fall-back solution on the SmartCoDe node for the broken control loop.

Table 1 shows on overview of the pros and cons of both approaches as they presented themselves at the time when the decision was made. It is easy to see that there are more pros for the decentralised approach and more cons for the centralised one. Also, even in a decentralised approach certain (classes of) EuPs can be still managed in a centralised manner if necessary or favoured (see section 3.3 for an example). This consequently led to the decision to develop a decentralised approach.

Most of the points in Table 1 are still valid. What probably isn't really an issue is the number of manageable EuPs, since also in a decentralised approach each node is known to the EMU for simple networking reasons. Also, the EMU is the part of the network which can in principle be upgraded as needed regarding computational power and/or memory to handle any reasonable workload in a centralised approach without increasing the overall system costs drastically.

The load balancing problem for the decentralised approach turned out to be easier to solve as expected, as will be shown later. On the other hand, the complexity of the software on the node level for the decentralised approach proved indeed to be considerably complex for some EuP classes, as will be seen in Section 3.

Table 1 Centralised vs. decentralised energy management in SmartCoDe

	centralised approach	decentralised approach
communication overhead	**very high:** • sensor data has to be transmitted • continuous control commands • Possible bandwidth problems	**low:** • No sensor data has to be transmitted • one EMU transmission every ~10 min • Some configuration parameters
number of manageable EuPs	**limited:** *Every* EuP is known to EMU and micro-managed	**unlimited** (in principle): Not every EuP must be known by the EMU
EMU crash / absence / communication problems	• SmartNodes "headless" • What happens to control loops?	SmartCoDe nodes should operate reasonably well for approx. 1 hour autonomously
SmartCoDe node design	**simple**	**challenging** (esp. software)
load balancing between EuPs	**easier** to achieve since EMU has complete control	**harder** to achieve due to autonomy of SmartCoDe nodes
EMU workload	**heavy:** • A lot of data has to be stored • Every single EuP is micro-managed	***probably* lighter:** • Not much EuP data has to be stored • Only the general directives have to be computed
SmartCoDe node workload	**light**	**rel. heavy** (optimization problems!)
Micro-managing	is the principle here	**still possible** for selected EuPs / EuP classes (e.g. SCDSVC)

2.1 The semi-decentralised approach

Figure 1 shows an overview of the SmartCoDe energy management approach. In general we assume that the EMU tries to achieve a certain overall target load profile for the whole cluster. This load profile might have been computed by the EMU itself as a cost-optimal solution to the current tariff and other circumstances. It might even be the result of automated negotiations between utility and EMU; but in fact, this is not important here since this is not the scope of SmartCoDe. The scope of SmartCoDe is to provide an infrastructure and a methodology to achieve any given target load profile as best as possible.

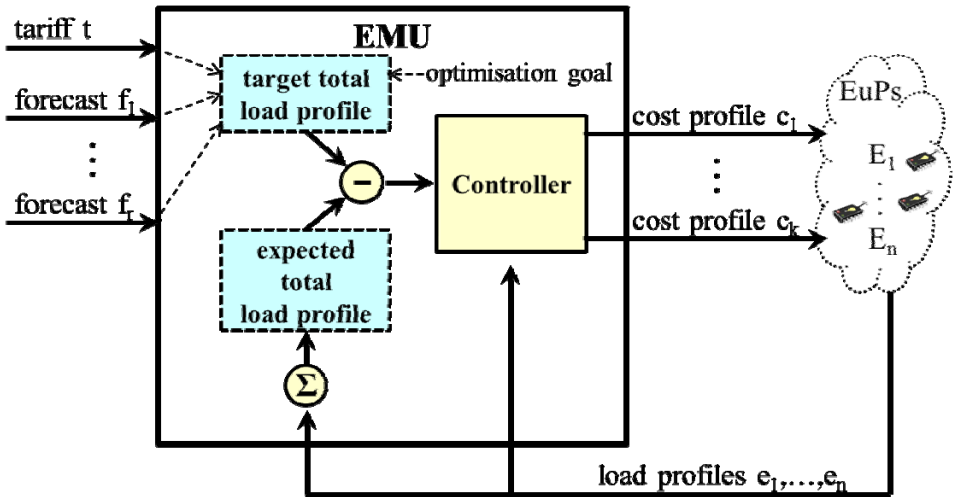

Figure 1 Schematic view of the SmartCoDe energy management approach

The general concept shown in Figure 1 is a big control loop: The EMU collects the load profiles from the single EuPs and compares the sum of these load profiles to the target load profile. The controller then tries to issue the next round of cost profiles to the SmartCoDe nodes such that the difference to the target load profile is minimised.

Figure 1 still leaves open several possibilities:

- The load profiles sent by the SmartCoDe nodes could be *planned* loads, *expected* loads or *past* loads.
- The cost profiles issued could target one EuP each, certain groups of EuPs, certain EuP classes, or it could be just one cost profile (k=1).
- The exact controller action is not specified. However, it is obvious that just using the difference to the target load profile is not enough input here, hence the additional input of the single load profiles to the controller.

In Section 2.3 the implementation as it was done at the SmartCoDe demonstrator is described. In Section 2.4 a possibility is described which was not implemented.

Note that the reason why this energy management approach is called *semi*-decentralised is twofold:

- With the option to send a dedicated cost-profile to a single EuP, the EMU still can (within bounds) control an EuP very tightly. For example, it could essentially *force* an EuP to switch on at a certain time for a certain amount of time by making all other options maximally costly; but of course this works only if this “forced” option is a possibility at all.
- A *true* decentralised energy management approach should not have any central component involved in the energy management; all the computation should be spread over the whole network.

2.2 Cost- and load-profile

In D-1.2 the general idea of using cost profiles (called *cost functions* at that time) was already presented. One inspiration was the load control event from the ZigBee Smart Energy profile. This is essentially a message which tells an EuP, or a group of EuPs, to behave in a certain way for a given time frame specified by a start time and a duration.

The message contains several fields (not all of them have to be used) to control different kinds of behaviour, e.g. temperature setpoints. One field, the *Criticality Level* (see Figure 2), indicates in abstract terms if the EuP should use more or less energy. The levels range from 1 (energy can be used freely), to 6 (save as much as possible). There are also 3 additonal levels to indicate emergencies, planned outages and service disconnects. The rest of the 8-bit value range is reserved.

Octets	**4**	**2**	**1**	**4**	**2**	**1**	**1**
Data Type	Unsigned 32-bit integer	16-bit BitMap	Unsigned 8-bit integer	UTC Time	Unsigned 16-bit integer	Unsigned 8-bit integer	Unsigned 8-bit integer
Field Name	Issuer Event ID (M)	Device Class (M)	Utility Enrolment Group (M)	Start Time (M)	Duration In Minutes (M)	Criticality Level (M)	Cooling Temperature Offset (O)

Octets	**1**	**2**	**2**	**1**	**1**	**1**
Data Type	Unsigned 8-bit integer	Signed 16-bit integer	Signed 16-bit integer	Signed 8-bit integer	Unsigned 8-bit integer	8-bit BitMap
Field Name	Heating Temperature Offset (O)	Cooling Temperature Set Point (O)	Heating Temperature Set Point (O)	Average Load Adjustment Percentage (O)	Duty Cycle (O)	Event Control (M)

Figure 2 Format of the ZigBee Smart Energy load control event

The idea of the SmartCoDe cost profile is to bundle a series of such (successive) load control events, but specialised only to the Criticality Level. That is, it contains a vector of duration - Criticality Level pairs describing the varied abstract cost of energy use for a certain time span beginning at the start time. For maximum flexibility, the time granularity can be chosen from 0.1 seconds to about 1.5 hours in the time resolution field.

Table 2 SmartCoDe cost profile

Data Type	Unsigned 8-bit int	Unsigned 8-bit int	Unsigned 16-Bit integer	UTC Time	Unsigned 8-bit int	Unsigned 8-bit int	Unsigned 8-bit int	(repeat) …	Unsigned 8-bit int	Unsigned 8-bit int
Field Name	appliance class	utility enrolement group	Time Resolution	Start Time	cost profile length n	Criticality Level 1	Duration 1 (in time resolution units)	…	Criticality Level n	Duration n (in time resolution units)

In the course of the project, especially when the overall energy management approach was refined due to practical experience at the demonstrator site, it turned out that the restriction of the Criticality level to the values 1-6 proved impractical regarding the implementation for the energy management approach which is presented in Section 2.3.

In this implementation the EMU stores a master cost profile which adds up the load plans received so far. This master cost profile is then added to the forecast of the turbine output and broadcasted. Obviously it is more practical in this setting if the criticality level values are directly proportional to the power usage of the load plans and the power forecast of the turbine. However, this can't be done with only with six available values. Therefore, the full 8 Bits of the Criticality Level are used in the final implementation.

This implicitly means that the Criticality Levels are now carrying information of available wattages. However for the nodes cost-dependent algorithms this is not important. The nodes just look for a solution for their load plans which minimises the costs, the unit of the cost does not matter. For the nodes, the values are still abstract.

In an implementation as it is described in Section 2.4, where each node receives a custom cost profile it would be possible to use the value range of 1-6 again, since each cost profile could be scaled accordingly before sending.

The load profile message used for the load plans is shown in **Fehler! Verweisquelle konnte nicht gefunden werden.**; it is very similar to the cost profile, the main difference is that the unit of the values is not abstract any more, but now gives the power to be used in watts. The resolution of the power fields can be set from 0.1 watts to about 6.5 kW.

Table 3 SmartCoDe load profile

Data Type	Unsigned 8-bit int	Unsigned 8-bit int	Unsigned 16-Bit integer	Unsigned 16-Bit integer	UTC Time	Unsigned 8-bit int	Unsigned 8-bit int	Unsigned 8-bit int	(repeat) …	Unsigned 8-bit int	Unsigned 8-bit int
Field Name	appliance class	utility enrolement group	Power Resolution	Time Resolution	Start Time	load profile length n	Load Level 1 (in power resolution units)	Duration 1 (in time resolution units)	…	Load Level n (in power resolution units)	Duration n (in time resolution units)

Note that the high resolution options available for cost- and load profile starting from 0.1 seconds and 0.1 watts are not really needed in your application scenario. However, since 16 Bits are available anyway it made more sense to offer higher resolution instead of allowing the resolution units to go up to 18 hours and 65 kW, respectively.

2.3 Energy management with committed load plans

This Section presents the global energy management algorithm as it was implemented in simulation, in the lab and the Buchberg demonstrator. It has been essentially presented in D-2.5 and is given here again for completeness.

The core idea of this algorithm is *load balancing*. We assume that we are given an initial cost-profile which could for example simply represent the predicted power output of the local wind turbine. In fact, this is exactly the setting used at the Buchberg demonstrator.

If the EMU broadcasts this cost-profile alone, all the EuPs would try to plan their consumption peaks in the cheap time frames. However, if all the *EuPs* do that, the final result will usually not be what is desired; the overall power consumption might exceed the turbines production such that still a lot of power

from the grid has to be drawn, while a while later turbine power might still be available (yet in less excessive amounts), but is not needed.

An important premise of the following algorithm is therefore also that the EuPs do not plan their consumption on arrival of the cost-profile, since this would also lead to a similar situation as described above. However, as will be seen in Section 3, this is not the case for the local energy management algorithms used.

However, the *core premise* of this algorithm is that the load profiles of the EuPs received are *committed load plans*. That is, these load plans will be executed as is and can be taken as a given barring unforeseeable circumstances, e.g. by user interference or override. Therefore, after a load-profile has been issued, all subsequent cost-profiles can be ignored until the current load plan has been executed. How this is done on the node level is also described in Section 3.

The assumption of load-plan commitment has a key advantage: We can use a *single* cost-profile for all EuPs. If a load plan is received by the EMU, it incorporates it into the new cost profile such that the cost in times of high load according to the received load plan rises. This new cost profile is then broadcasted to all EuPs.

Since the appliance which just sent the last load-plan ignores the new cost profile until the current plan is executed, it will not be blocked by its own recently sent load plan. And when the next load-plan is computed, the cost-profile values affected by the last load plan are already outdated and are not broadcasted anymore. These considerations now give rise to the following protocol for a group of load-plan committing appliance:

1. The EMU broadcasts a cost-profile to all appliances in the group. The base of this initial cost profile is of no concern here. For example it might reflect the power consumption of all other appliances in the network. It could also be based on a forecast of the local wind turbine power output or a tariff more volatile (e.g. hourly-changing) then commonly used today. If the only goal is load-balancing, the initial cost-profile would be constant 1 in our implementation.
2. At some point an appliance issues a load-plan to the EMU, e.g. when a freezer switches off the first time after finishing its learning phase and now computes a plan as described in the previous Section.
3. The EMU incorporates the load-plan into the current cost-profile and broadcasts it. For this, it holds an internal copy of the cost profile which is constantly updated with load-plans and forecasts.
4. Eventually, other appliances will compute and send load plans, based on the cost-function they have received at that point, giving rise to subsequent cost-profile updates.

Note that it is still possible for the EMU to send a custom cost-profile *only to the specific node* (i.e. point-to-point and not via broadcast) if it wants to persuade it to change its plans. Since in ZigBee a node can distinguish between a broadcasted message and one which was sent to it specifically, it would then know that the EMU wants it to explicitly break the previous load commitment. This might become necessary if conditions outside of the appliance group change (for example the wind-forecast or grid stability issues). Of course changing the load plan might not be possible from a certain point on, but it still is for example if a dishwasher has still to start the program, or a fridge has not switched its compressor yet.

2.4 An option with non-committed load plans

This Section describes how the load balancing would work without committed load plans in the SmartCoDe setting, for example with load plans which the EuPs are only partially committed to, e.g. in case of the extended load plan described at the end of Section 3.2. While this has not been implemented, it could easily be done by only changing the code on the application level of the EMU and the SmartCoDe node; there wouldn't be the need to change the SmartCoDe profile itself.

The protocol would be as follows:

- Instead of incorporating the received load plans to the internal cost-profile, the EMU has to store all load plans received and keep them in the memory until they are outdated.
- If a new load plan is received, the EMU computes for each individual appliance a custom cost-profile using all other relevant load-plans, together with the "base" cost-profile derived from forecasts and/or tariffs (i.e. on the Buchberg demonstrator the wind turbine output forecast). These custom cost-profiles are then successively sent to the respective appliances directly.

This obviously results in a considerably higher effort on the EMU-side with respect to computational effort and memory requirements; but since the EMU can have a considerably higher computational power than the sensor/actor nodes, this is not really an issue. Also, more cost-profiles messages have to be sent. However, a point-to-point message consumes less network-bandwidth than a broadcast. Also, a broadcast is less reliable than a point-to-point message. Because of that, in the implementation as described in Section 2.3, cost-profile updates are broadcasted in regular intervals in addition to the ones triggered by incoming load-plans. Therefore the apparent bandwidth-disadvantage here might actually not be so big.

3 Energy using Products

3.1 Classification

One of the first results of the SmartCoDe work package 1 was the classification of energy using products. This was done because it became apparent that originally proposed approach from the Description of Work (DoW) seemed not to be as constructive as it looked at first. The original approach was to base the abstract EuP models on discrete Markov state models which describe the different power states of the EuP and the respective transitions between them. With respect to the very Deliverable at hand, refinements of these Markov models gained from the practical experience in the course of the project could be given.

However, this Markov model approach does not take into account the specific service offered by the EuP as well as the user interaction. The power states of many EuPs (and hence of the corresponding Markov model) would often be simply on and off. That is, very different appliances would fall into the same category if we just look at the power states, e.g. a lamp would look very similar to an electric kettle.

Furthermore, similar devices may fall into different classes, e.g. fans might have different numbers of power consumption levels and therefore would look different although they all do the exact same thing and are also used the exact same way.

And when looking at the transitions characteristics, we may find that these are often user-specific. The Markov model of a washing machine in a one-person household will have different transition probabilities than one in a family household, since it is used less frequently.

Figure 3 Overview of the SmartCoDe classification of EuPs

Table 4 SmartCoDe classification of EuPs

Class	Description	Parameters			Energy Management		Examples
		Configuration	Sensor input	Online input	Strategy	cost	
VARSVC	**Variable Service:** The appliance provides a user-variable service which is balanced with sensor input.	tolerance bounds	current state of the service, e.g. illuminance	user demand, e.g. setpoint for illuminance	Minimise consumption by balancing the service with user demand, tolerance bounds and sensor measurement.	No	lighting controlled by illuminance level, dimmable lighting, blinds
THMSVC	**Thermal service:** The appliance provides a inert, thermal service which can serve as a virtual storage.	temperature bounds / hysteresis	temperature	user demand, e.g. setpoint for temperature	Adjust temperature to user demand while exploiting the virtual storage property to minimise costs.	Yes	Fridge, Freezer, Heating, A/C, Water-boiler
SCDSVC	**Schedulable Service:** The appliance provides a service which can be scheduled within a certain time-frame.	runtimes and power profiles of the different programs	none	deadline	Schedule program such that deadline is met and the program's load profile produces minimal costs.	Yes	washing machine, dryer, dishwasher, baking machine
ETOSVC	**Event-Timeout Service:** The appliance is control-led by sensor events and time-outs.	time span	sensor event, e.g. presence detection	none (indirectly through sensor input)	Control appliance according to sensor events and time-outs.	No	lighting controlled by presence detector
CHACON	**Charge Control:** The appliance charges a possibly removable device.	charging policy	current charge status, device presence	device removal re-insertion	Charge device such that costs are minimised, while obeying charging policy.	Yes	battery chargers, hand-held vacuum, emergency
COMCON	**Complete Control:** Like CHACON, but the usage of the charged power can also be con-trolled.	charging policy, duty cycles, time slots	current charge status	none	Like CHACON, but also control the usage of the appliance cost-effectively while obeying to the given time-slots	Yes	robot vacuum, robot lawn-mower
CUSCON	**Custom Control:** device does not fit into other classes or has too high user inetraction to be controllable.	none	none	user demand	Automatic Energy Management probably not tolerable by user; custom schemes can be defined which are implemented by the	No	HiFi, PC, Oven

Figure 3 shows an overview of the EuP classification, and Table 4 gives a more detailed look. Since deliverable D-1.1.1, there have been some minor changes and additions:

- In the row “Energy Management Strategy”, the sub-row “cost” indicates if the cost profile is of interest for energy management of the specific class. For example, this is the case for the VARSVC class which covers lighting applications, since automated switching of lights due to energy availability will probably not be accepted by the general user in the home/office sector.
- The class previously called “VSTSVC” (virtual storages) is now called “THMSVC” (thermal services). While the virtual storage property for this class is still valid, it turned out that in the home/office sector there are no appliances of this class where the electrical energy is not transformed to thermal energy. And since in the course of the project temperature forecasting played an important role for this class, it was appropriate to reflect this also in the naming
- The renaming of the “SKDSVC” class to “SCDSVC” is merely a correction of a typo caused by use of American English pronunciation by a non-native speaker (German) who transformed this pronunciation back to a wrong abbreviation.

3.2 Local energy management of the THMSVC class

The first ideas on how to control a THMSVC EuP were presented in D-1.2 (Section 5.1), and for simplicity we take the *fridge* as a representative of this class for the remainder of the discussion. The approach presented in D-1.2 was essentially a manipulation of the thresholds for the usual bang-bang control. for example, when the cost is high, the upper bound was raised by a certain amount (proportional to the cost) such that the fridge could stayed of a little longer during costly times.

With simulation it could be shown that this simple algorithm was already sufficient for load balancing: 4 simulated fridges were sent 4 different cost profiles that were phase-shifted by 90 Degrees, successively. Essentially, the fridges synced to their respective cost-profile such that the combined power consumption was more balanced.

To get a load forecast for a fridge using this algorithm on a specific cost-profile, the plan was to employ *temperature forecasting*. The idea is simple: The algorithm which runs on the real temperature measurements and the cost-profile is run beforehand on the simulated temperatures for a certain time period. That way, the load profile of *any* control algorithm (e.g. usual cost-independent bang-bang control) can be forecasted up to a certain degree.

It was possible to develop a very sophisticated curve-fitting algorithm using least-square-error approximation for the temperature forecast. This algorithm even used constant time and memory, and could be run on the SmartCoDe nodes without using considerable resources. This algorithm has been described extensively in D-1.4.

However, when moving from the simulation to the lab environment it became apparent that the temperature forecasting algorithm can't be used at the points where the temperature changes direction. That is, there is no seamless temperature forecast available which can be used over several off-on cycles, which was also explained in D-1.4.

The solution to this problem was found shortly after D-1.4 was submitted: an algorithm which provides a load plan for the next off-on cycle shortly after the compressor was switched off. This load plan is not a forecast but a committed plan which will be executed as-is unless unusual circumstances cause a failsave bang-bang control to kick in. While this algorithm is not dependent of the temperature forecast any more, it can employ it for better stability and performance. The algorithm was presented first in D-2.5. We describe the algorithm here again for completeness of this document.

The core idea is to replace the usual bang-bang controller with a PI-controller that computes a load-plan for the next Off-On cycle of the compressor. This load plan is then tweaked according to the cost profile, which in general will cause the maximal/minimal temperatures observed during this cycle to exceed or fall short of the temperature boundaries set. The PI-controller then counteracts these violations in the subsequent cycles.

In an *initial learning phase*, usual bang-bang control is used. Apart from initial values for the parameters used for temperature forecasting (see D-1.2), we also learn the normal duty cycle (P^n_{Off}, P^n_{On}). This duty cycle is used to parameterise a PI controller in the sense that the error values observed will manipulate this normal duty cycle. Also, the real extremal temperatures (b_u, b_l) attained are determined, since they usually differ from the temperature bounds used for the bang-bang control (e.g. when the compressor is switched off, the temperature usually goes down still for some time, sometimes up to 1°C).

After that, the node goes into *planning mode*, which works in the following way:

1. During the on-going Off-On cycle, the temperature maximum and minimum (t_{max}, t_{min}) is determined.
2. After switching off, the node waits until the temperature rises strong enough (w.r.t. the slope) to predict the number of steps when the upper bound is hit.
3. The PI controller now determines an initial schedule by looking at the difference between the observed extremal temperatures to the usual ones when using bang-bang control, i.e. $e_l = t_{min} - b_l$, $e_u = b_u - t_{max}$. Also, the respective errors sums $\Box e_l$ and $\Box e_u$ are updated and we compute: $y_l := 1+K_p \cdot e_l + K_I \cdot \Box e_l$ $y_u := 1+K_p \cdot e_u + K_I \cdot \Box e_u$ and $(P'_{Off}, P'_{On}) := (y_u \cdot P^n_{Off}, y_l \cdot P^n_{On})$ for certain factors K_p and K_I (see below). For example, when the minimum temperature tends to be too low, the On-phase is shortened, and when the maximum temperature is too low, the Off-phase is prolonged.
4. This initial schedule is now manipulated according to the forecast of the number of steps it takes to reach the usual bang-bang maximum by taking the median of the off-phase length

and the step forecast as new off-Phase, and scaling the On-phase linearly with it, resulting in a new schedule (P_{Off}, P_{On}).

5. The schedule is now multiplied by a factor f with 0.5 <f<1.5 such that the result f·(P_{Off}, P_{On}) is maximally cost-effective with respect to the cost-profile. This is done by a straightforward search algorithm using different values for f, e.g. 0.5, 0.55, …, 1.5.

The SmartCoDe node then sends back this load-plan and is *committed to it* until the next switch-off. If the temperature bounds are breached by an unacceptable amount due to drastic events like putting hot food in the fridge, a fail save bang-bang controller still can switch the appliance earlier if a bound is exceeded by more than 2°C.

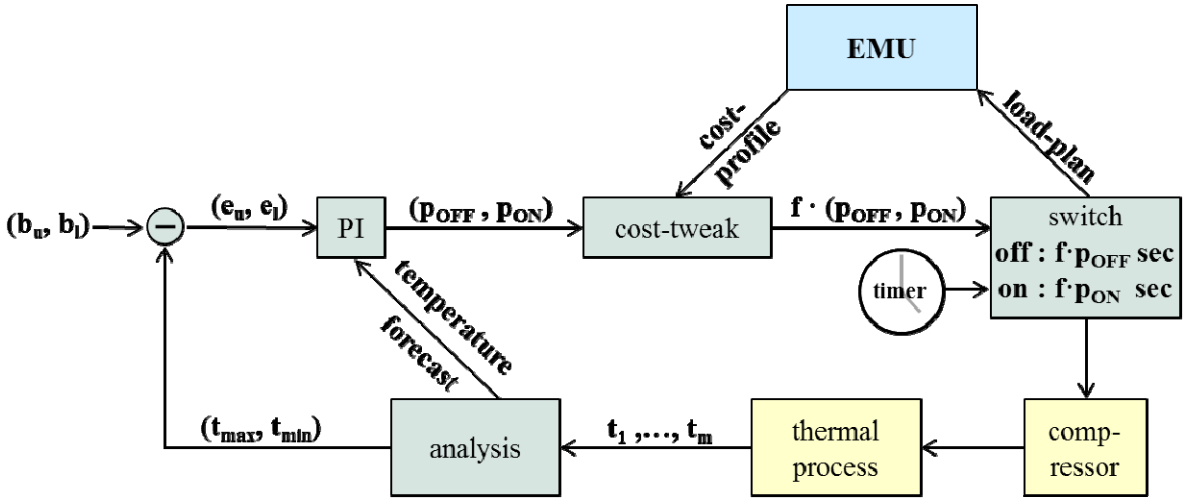

Figure 4 PI-based load-planning algorithm for THMSVC EuPs

Figure 4 shows an overview of this algorithm. The choice of K_P and K_I was initially done by analysing the possible errors and initially set to K_p=0.3 and K_I=0.15. K_P = 0.3 means that a difference of 1°C to the particular temperature bound would alter the respective cycle by 30% as a direct response, and since exceeding a bound by more than 2°C is intercepted by the fail save bang-bang safeguard, the maximal alteration of the P-part is 60%. Since the accumulated error-sum can grow larger, the lower value K_I=0.15 was chosen for the integral part.

After experimenting with simulation, the lab-setting and on the Buchberg demonstrator, the parameters used now are K_p=0.1 and K_I=0.05, since this leads to a more stable behaviour even if the cost-profile is constant over the time frame in question and therefore there is no cost-tweak. Also, the algorithm was slightly adapted by using the temperature forecast to ensure that the computed load-plan never violates by itself the failsave bounds.

Note that the choice of these parameters is not obvious. The problem is that the goals we have here are not the usual goals like stability when dimensioning a PI-controller. This PI controller computes initial load plans and inherently *organizes deposits and withdrawals into and from the thermal capacitance of the EuP*. Since this type of application of a PI controller is new to the best of our knowledge, it is not clear in how far classical control theory is applicable here.

The load plan as described above can also be extended by an *estimation for the next Off-On cycle* as follows: With the temperature forecast, the temperature after the first Off-phase according to the final load plan can be determined. This yields an estimation for the next error terms regarding the upper boundary temperature which, in turn, gives an estimation of the next P_{Off}. By using the same ratio of Off- to on-phase as for the current Off-On cycle, an estimation for the next (P_{Off}, P_{On}) can be given, which then also can be cost-tweaked if there are already cost-profile values for this time frame available.

With this extension, the EMU could be provided with a load-profile covering two off-on cycles, where the fridge is only committed to the first cycle; the second cycle is an estimation and can still be changed in the next round. However, these kinds of load plans would be appropriate for the variant of the global energy management approach described in Section 2.4.

3.3 Local energy management of the SCDSVC class

The *local control* for a schedulable service (SCDSVC) EuP with a given deadline and cost profile is straightforward if the load profile for the chosen program is known: The SmartCoDe node simply tries to find a start-time within in the given time frame such that the product of load- and cost-profile is minimised. This is a simple optimisation problem which was easily be implemented on the SmartCoDe nodes. For the Buchberg demonstrator, however, the start-time was actually computed by the EMU (namely the ennovatis SmartBox), by sending it the deadline and the load profile, which then returned a start-time. This was done for easier implementation since this mechanism was already in place for the ennovatis SmartSwitches. It also demonstrates that the semi-decentralised energy management approach can be still combined with some centralised micro-management.

However, in general computing the load profile is not as simple, since it might depend on several conditions like the current water temperature which influences the length of the initial heating cycle. In fact, in D-1.5 an impressive example is given how much electrical energy can be saved when pre-heated water is available. Another important factor in the case of a washing machine is the weight of the load. It should be possible for the manufacturer, who understands the process of the EuP best, to find a satisfactory estimate for the load profile. Also, inserting *interruptions* of the program when possible could be an option.

In fact, in the course of the project it became clear that for the SCDSVC class, there has to be higher integration of general EuP control and energy management. With a retrofitting approach as it was used for demonstration purposes there is not much that can be done. For fridges and freezers, in contrast, with the algorithm presented above basically *any* such EuP can be retrofitted easily, since the algorithm learns everything there is to know about the EuP by itself, and the only thing to do is switching a compressor.

When retrofitting SCDSVC EuPs like washing machines, the possibilities of fine-tuning the energy management algorithm suffer from limitations imposed by the hardware itself. Most of these appliances currently present in the market don't have an interface through which an external controller could obtain any complex information from the device. For example, the washing machine that was used for testing the SmartCoDe energy management algorithm is interfaced with the SmartCoDe node by using two digital I/O lines:

- The first line is used for transmitting the pulse that starts the machine operation. After that pulse is transmitted, there is no way for the SmartCoDe node to stop the machine. Thus, there is no possibility to apply a more complex algorithm which would allow interrupts of the operation.
- The second line provides the feedback information from the machine to the SmartCoDe node about the operation status: when operation is in progress, the logical `1` appears on the line, otherwise is `0`.

Especially it is not known which program the user was choosing when starting the machine, such that there was no choice but to use a worst-case load profile, derived from previous measurements, as a basis for the energy management. Of course, with such a worst-case profile it is illusionary to achieve something like freezers planning their consumption such that they avoid being switched on when the washing machine load profile peaks.

3.4 Local cost-dependent energy management of other classes

Regarding cost-dependent energy management, there are only two classes left regarding the classification

- The CHACON class represents appliances which are chargeable and (randomly) removed by the user. The most interesting representative here is the electric car. While devices like cell phones essentially also fall into this class, the energy consumption involved is too low to be of interest here.
- The COMCON class is similar, but the user does not use the appliance directly, but it is deployed automatically, such that this deployment in principle could also be controlled. Examples are robotic lawn mowers or vacuums. The use of these devices is still very uncommon.

Since there was no representative of these classes available for development, and since it was never the intention in SmartCoDe to cover every possible appliance, no specific algorithms have been developed for these classes. However, some conclusions can still be drawn.

The CHACON class can essentially be treated similar to the SCDSVC class, since charging such a device is very similar to running a certain program, since in both cases the result is a certain load profile (or plan) which can be scheduled. The difference here is the user interaction: When a user starts a

washing machine he already selects a program and therefore it's not too much to ask for the additional input of a deadline; some washing machines offer something similar even today for simple convenience purposes.

When a user plugs in a chargeable device, however, there is no further user interaction involved. The user would have to input a time when he expects the device to be charged again, which means that also a new user interface is needed. However, if this would is available, the further proceedings can indeed be like in the SCDSVC case.

The COMCON class was defined with the premise that the energy management would also encompass the *consumption* of the charged energy (therefore providing "complete control"). That means that such appliance could also be used as a virtual storage, like the THMSVC class, but with much better properties: The charged energy does basically not deteriorate like in the case of the THMSVC class. And depending on the specific service, the usage can be postponed much longer and more flexibly than in the case of a freezer.

The use of a robot vacuum, for example, could easily be delayed for half a day or more without significant consequences. And if favourable conditions for charging arise in the near future, the operation could be scheduled much earlier to discharge the vacuum until then.

4 Local Energy Providers

Regarding the Local Energy Providers (LEPs), a similar classification has been done, with no major changes since D-1.1.1. This classification arguably proved to be not as useful as the EuP classification, even though it is valid. While the EuP-classification allows to treat different appliances in the same manner, e.g. with the same software, this is not really possible here. While wind turbines and solar panels are both volatile energy providers according to the classification, they have very different characteristics, even if the interfaces for providing energy generation forecasts can be the same. Also two energy grids can be very different, e.g. regarding tariff models. Energy storages and local generators were not available in the project.

Altogether the SmartCoDe project wasn't able to develop a unified view on LEPs in the way that it did for EuPs. However, this was never in the core of the project. Models for the forecasting of photovoltaic generators and the windturbine on the demonstration site have been developed and are reported in D-1.1.1, D-1.3 and D-1.6. The LEP classification is repeated here in Figure 5 and Table 5 for completeness.

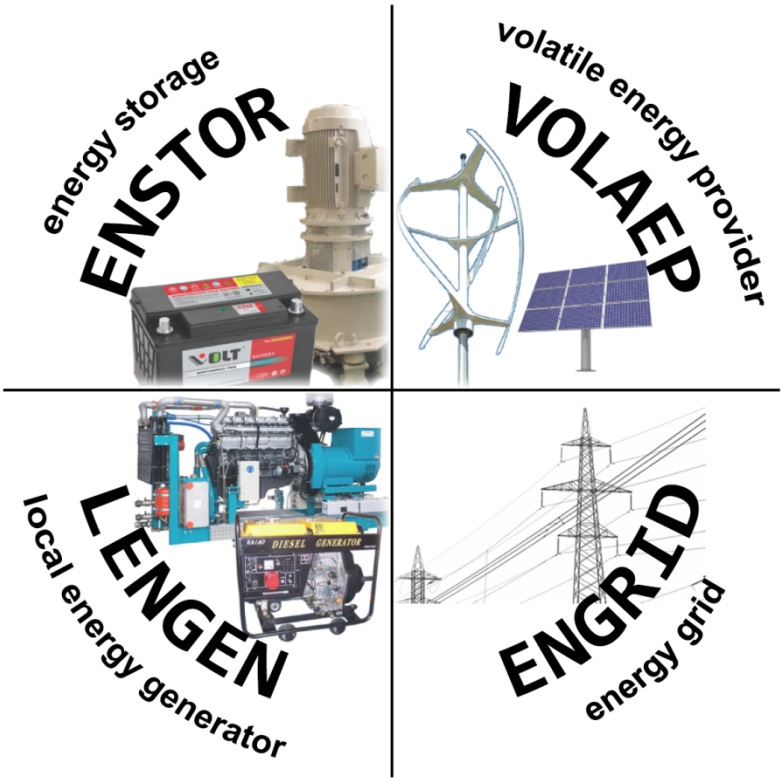

Figure 5 Overview of the SmartCoDe classification of LEPs

Table 5 SmartCoDe classification of LEPs

Class	Abbrev. from	Description	Parameters			Examples
			installation	from LEP	to LEP	
ENGRID	energy grid	conventional energy provider	feed-in possible (true/false)	pricing forecast supply forecast	feed-in to grid	local electrical power provider long-distance heating
VOLAEP	volatile energy provider	energy source which depends on weather, daytime etc.	switchable (true/false)	supply forecast	on/off if switchable, weather forecast?	wind turbine, water turbine, solar, geothermal
ENSTOR	energy storage	energy source which has to be charged	storage capacity	charging level	charge / provide	batteries, concrete heat storage
LENGEN	local energy generator	energy source which transforms some kind of fuel to energy	fuel capacity	fill level	on / off	block power generator, diesel generator

5 Conclusion

This document described the developments regarding the foundations of the SmartCoDe work package 1 during the project runtime. It showed how the global semi-decentralised approach was motivated by networking and controlling considerations. The final protocols and message formats used are presented, with a discussion of possible options.

On the EuP side, the basis for the local cost-profile dependent energy management was the EuP classification. By simulation and practical experiments, the local energy management could be refined and improved. Especially the PI-control based energy management algorithm is an original and unconventional result. Possible improvements are discussed regarding the SCSVC class, whose energy management could not be tackled to the detail it was intended.