Situation-Aware Energy Control by Combining Simple Sensors and Complex Event Processing

Leonard Renners and **Ralf Bruns** and **Jürgen Dunkel** [1]

Abstract. In recent years, multiple efforts for reducing energy usage have been proposed. Especially buildings offer high potentials for energy savings. In this paper, we present a novel approach for intelligent energy control that combines a simple infrastructure using low cost sensors with the reasoning capabilities of Complex Event Processing. The key issues of the approach are a sophisticated semantic domain model and a multi-staged event processing architecture leading to an intelligent, situation-aware energy management system.

1 Introduction

In recent years, global warming, greenhouse effect, as well as escalating energy prices have lead to enhanced efforts for reducing energy usage and carbon emission. Especially, buildings offer high potentials in reducing energy consumption: electric lighting, heating and air conditioning are highly energy-intensive and not well-adjusted to actual usage necessities.

In many buildings, there are already first approaches of more effective control mechanisms to improve energy consumption. Typically, light in public areas like corridors is controlled by using motion sensors instead of classic switches, preventing unnecessarily turned on lamps. On the contrary, heating is mostly controlled on base of fixed heating plans that determine the heating period by a predefined timetable. Often the heating schedule is not related to single rooms but to the entire building or larger parts of it (e.g. total floors). Overall, this leads to the situation that many rooms are heated although they are not in use, thus causing a huge waste of energy. In summary, we can conclude that in general energy control of buildings is not situation-aware.

In the following we present an approach for intelligent energy control that combines a simple infrastructure using cheap sensors with event stream processing of the plain sensor data. Our approach should reach the following goals:

(a) *Individual energy control* for every single room (instead of considering the entire building).
(b) *Low-cost solution* by utilizing the existing infrastructure as well as cheap and simple sensors (as opposed to new, sophisticated and expensive sensors).
(c) *Situation awareness:* the energy consumption should be controlled according to actual usage (in contrast to predefined and fixed schedules).
(d) *Proactive control*: the control mechanisms should exploit the knowledge about normal room occupancy, for instance based on room schedules and normal user behavior patterns.

[1] Hannover University of Applied Sciences and Arts, Germany, email: forename.surname@fh-hannover.de

To achieve these goals, our approach uses two different models:

1. A *Semantic Domain Model* describes the expert knowledge about the domain, i.e. in our case the structure of the building and its expected usage.
2. An *Event Model* defines all those events occurring in the building that are relevant for energy control. Different event sources can be distinguished: low cost sensors yield information about the current incidents in a building. Furthermore, domain-specific events are created by various knowledge sources, like heating plans or lectures schedules.

To react on relevant situations in real-time we apply *Complex Event Processing* (CEP) that has been proposed as a new architectural paradigm for in-memory processing continuous streams of events. CEP is based on declarative rules to describe event patterns, which are applied to the event stream to identify relevant situations in a certain domain.

The integration of the Semantic Domain Model with the Event Model allows the enrichment of event processing rules with domain knowledge. It is a key issue of our approach and provides an intelligent sensor data processing, leading to a reasonable, situation-aware heating and energy management.

The remainder of the paper is organized as follows. The next section 2 describes an application scenario for our approach. In section 3 the related work is discussed. In the subsequent section 4 we introduce our approach using CEP techniques to implement an intelligent energy management. Section 5 shows an example set-up and yields an evaluation. The final section 6 contains some concluding remarks and provides an outlook to future directions of research.

2 Scenario: Energy Control in Buildings

We selected the energy management of buildings as our application scenario. In particular, the scenario helps to discuss in some more detail the benefits that intelligent energy control provides. As already mentioned in the introduction, buildings have high potentials in improving energy efficiency, since they have many energy consumer units that are often unnecessarily turned on. In the following, we will use an university campus as a concrete example of our approach. Universities exhibit the following features that are common to most public buildings and which will influence energy management significantly:

- *Various room types* with different energy consumption profiles can be distinguished: For instance, server rooms have to be air conditioned below 20 degrees Celsius. Instead, offices and lecture rooms must be heated to achieve a temperature above 22 degrees,

but normal storage room must be neither cooled nor heated. In the following, we will focus on office and lecture rooms.

- *Non-uniform usage profiles*: Each room exhibits its individual occupancy depending on the room type and the specific behavior of different user groups. For instance, lecture rooms are occupied according to a prefixed schedule that might be changed only exceptionally. Instead, offices or cube farms are used according to the personal behavior of their occupants including absences due to vacation times or illness.
- *Spontaneous occupancies*: Furthermore, rooms are spontaneously and individually used, e.g. due to ad-hoc meetings, rescheduled lectures or unplanned project work. Note that these individual occupancies are inherently unpredictable.

To deal with non-uniform and spontaneous usage of rooms, which deviates significantly from predicted average occupancies, an intelligent and situation-aware energy control mechanism is required.

On the one hand, expert knowledge about the building and the behavior of its users should be exploited for adjusting the energy control to realistic usage patterns.

On the other hand, actual behavior must be monitored to react adequately on spontaneous and individual usage actions. Especially, if unexpected occupancies or periods of absence are detected in a certain room, the corresponding energy consumption units (like heating and lighting) can be switched on or off, respectively.

To provide an individual control, we have to observe the incidents and states in every single room. To achieve this goal we will incorporate already installed sensors in the building as well as we will equip the rooms with simple and cheap sensors: *motion sensors* for detecting movements, *temperature sensors* to measure the heating, and *contact switches* that register when a door or window is opened or closed, respectively.

3 Related Work

Our approach is based on the exploitation of fine-grained sensor data emitted by networks of simple sensors in buildings. Sensor networks possess intrinsic problem properties that are perfectly addressed by complex event processing. Several published approaches prove the suitability of event processing for sensor networks (e.g. [6, 9]).

Determining the current status of usage is an important topic in smart homes and intelligent facility management systems. Several approaches have shown how occupancy detection can be used to implement more effective and powerful behaviour [1, 2, 7].

There is also increasing interest using CEP technologies for energy management. Holland-Moritz and Vandenhouten examined different solutions for an intelligent management system (in general) and identified CEP as one very suitable and suggestive concept [5].

Xu et al. introduced a CEP approach with ontology and semantics supporting occupancy detection for an intelligent light management system [10, 12].

Another approach in a similar direction was made by Wen et al. within their industrial experience report about using CEP for energy and operation management, while focussing on predictive elements and adaptable behavior [11].

In summary, on the one hand some approaches address energy control adapted to the current occupancy/usage, on the other hand first approaches report the employment of event processing technology in energy management.

However, none of them is presenting a comprehensive approach of real-time situational awareness for the whole energy management, based on the integration of an extensive semantic model of the application domain in the event processing reasoning process. In particular, the investigated control of the heating process requires more sophisticated semantic models and more advanced event processing rules.

4 Intelligent Energy Control Using CEP

In this section we present our approach for a situation-aware energy control by applying intelligent sensor data processing on simple buildings. After giving a short overview of *Complex Event Processing* we present our general software architecture and a *Domain Event Model* that integrates knowledge about domain specific concepts and events. Finally, we illustrate the intelligent reasoning part of our approach by showing some event processing rules.

4.1 CEP Overview

Complex Event Processing (CEP) is a software architectural approach for processing continuous streams of high volumes of events in real-time [8]. Everything that happens can be considered as an *event*. A corresponding *event object* carries general metadata (event ID, timestamp) and event-specific information, e.g. a sensor ID and the measured temperature. Note that single events have no special meaning, but must be correlated. CEP analyses continuous streams of incoming events in order to identify the presence of complex sequences of events, so called *event patterns*.

A *pattern match* signifies a meaningful state of the environment and causes either creating a new *complex event* or triggering an appropriate action.

Fundamental concepts of CEP are an *event processing language* (EPL), to express *event processing rules* consisting of *event patterns* and *actions*, as well as an *event processing engine* that continuously analyses the event stream and executes the matching rules.[2] Complex event processing and event-driven systems generally have the following basic characteristics:

- *Continuous in-memory processing*: CEP is designed to handle a consecutive input stream of events and in-memory processing enables real-time operations.
- *Correlating Data*: It enables the combination of different events from distinct sources including additional domain knowledge. Event processing rules transform fine-grained simple events into complex (business) events that represent a significant meaning for the application domain.
- *Temporal Operators*: Within event stream processing, timer functionalities as well as sliding time windows can be used to define event patterns representing temporal relationships.
- *Distributed Event Processing*: Event processing can be distributed on several rule engines (physically or logically). Thereby scalability and the separation of different functionalities can be realized.

4.2 Event Processing Architecture

Luckham introduced the concept of *event processing agents* (EPA) [8]. An EPA is a software component specialized on event stream processing with its own rule engine and rule base. An *event processing network* (EPN) connects several EPAs to constitute a software architecture for event processing. Event processing agents communicate with each other by exchanging events.

[2] Sample open source CEP engines are Esper and Drools Fusion.

EPAs provide an approach for modularizing and structuring rules: Light-weighted agents with few rules fulfill a coherent domain-specific task and improve comprehensibility and maintainability. Furthermore, distributing the EPAs on different computing nodes enhances system performance and scalability [3].

Thus, the event-driven architecture of our energy control system is based on a multi-staged EPN for structuring and organizing the event processing rules. Figure 1 depicts the different EPAs and illustrates the flow of events:

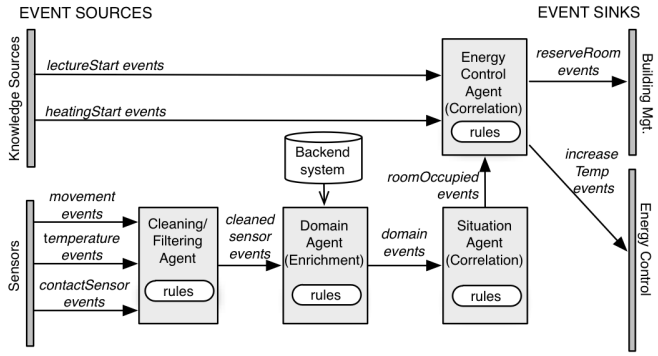

Figure 1. Event Processing Network (EPN) for Energy Management

Event Sources: We can distinguish different types of *event sources* that correspond to the information that is used by our energy control system (as already mentioned in section 2).

- *General knowledge sources*: There are general knowledge sources that can emit application-specific events relevant for the buildings energy management. For instance, a calendar containing the lecture schedules might create *lectureStart* events that signal the scheduled starting time of a teaching session.
- *Sensors*: Low cost sensors as described in section 2 are used to monitor the incidents in the building. For instance, motion sensors and temperature sensors emit *movement* events as well as *temperature* events. The contact switches produce *contactSensor* events that signal if doors or windows are opened or closed, respectively.

Event Processing Network (EPN): Event processing is nothing else than event transformation: the simple events emitted by the event sources are transformed into more abstract application-specific events for inferring appropriate control steps. The event transformations are processed by the EPAs depicted in figure 1.

- *Cleaning/Filtering Agent*: Due to technical problems, sensor data is often inconsistent: e.g. duplicated readings or outliers must be compensated. Therefore, in a cleaning step all sensor events have to be pre-processed to overcome inconsistencies [3].
  Furthermore, not all events are required in subsequent processing stages. For instance, motion sensors may emit many *movement* events within a small time interval, which are related to the same incident. Therefore irrelevant events are filtered out to reduce the total number of events. Using various processing rules, the Cleaning/Filtering Agent forwards *cleaned sensor events* events to the Domain Agent.
- *Domain Agent*: The *cleaned sensor* events contain only low-level technical information, e.g. sensor IDs that have no specific meaning in the application domain, and are often incomplete for further processing. Therefore, they should be transformed to domain events by mapping plain sensor event data to domain concepts. For instance, a measured temperature should be related to a certain room and to the desired temperature of the corresponding room type. The information necessary for this content enrichment step is retrieved from the backend systems. The Domain Agent transforms *cleaned sensor* events into enriched *domain* events and forwards them to the Situation Agent.
- *Situation Agent*: In a diagnosis step various domain events are synthesized to a new (complex) situation event that characterizes a particular state of the building. For example, *contactSensor* events and *movement* events are correlated to a new *roomOccupied* event signalizing that somebody is staying in a certain room. In summary, the Situation Agent processes a correlation step to create new types of complex events that are propagated to the Energy Control Agent.
- *Energy Control Agent*: Finally, the situation diagnosed from the stream of sensor events must be correlated with the information received from the general knowledge sources. The Energy Control Agent emits an *action event* to trigger a certain control action that reacts appropriately on the actual state of the building.
  For instance, *lectureStart* events emitted by a calendar are combined with *roomOccupied* events generated by the the Situation Agent to trigger an appropriate control actions by creating an action event of type *increaseTemperature*.

Event Sinks: The backend systems of the building management serve as event sinks of the events produced by the Energy Control Agent. Figure 1 shows two examples: an *increaseTemperature* event could be sent to the energy control system to change directly the heating of a certain room. As another example, we can consider a *reserveRoom* event that could be forwarded to the building management system for generating automatically an entry into the occupancy plan of the corresponding room.

4.3 Domain Event Model

A main contribution of our approach is integrating general *domain knowledge* with *sensor events* in a *Domain Event Model*. Figure 2 shows the general structure of the Domain Event Model that distinguishes two dimensions, and thus yielding four different quadrants:

	WORLD MODEL	EVENT MODEL
DOMAIN MODEL	Domain Concepts	Domain Events
SENSOR MODEL	Sensors	Sensor Events

Figure 2. Structure of the Domain Event Model

- The *World Model* describes the structural or static concepts regarded in the system: First, it defines the *domain concepts* like buildings, rooms or class schedules. Secondly, it defines the *sensor* infrastructure the building is equipped with. For instance, what different kind of sensors are used and where they are installed.
- The *Event Model* defines the dynamic aspects of the system, i.e. all types of events that are considered in the system [4]. First, all *sensor events* emitted by the different sensor types are described. Secondly, it considers all *domain events* in the system: On the one hand, these are the application events that are generated by CEP

rules and have a certain meaning in the application domain. For instance, that a room is occupied for a certain time. Furthermore, it defines context events, that are produced by general knowledge sources, like calendar applications containing room schedules.

Of course, there are interrelations between the concepts of the different model parts. For instance, the sensor concept 'contact sensor' is related to the domain concept 'room' specifying the concrete position of a certain sensor.

Figure 3 shows an excerpt of the Domain Event Model for our university building scenario. Note that for simplicity and clarity, no attributes are depicted in the diagram.

The **Model of the Domain Concepts** (upper left quadrant) describes the hierarchy of different types of rooms in an university, such as course rooms, offices, and server rooms. Course rooms can be further refined into lecture rooms, labs, etc. Other concepts modeled in figure 3 are heating plans that are related to each room and class schedules for each course room. Neighbouring rooms are specified by the *adjacent* relationship. Furthermore, the model defines room equipment as windows and doors.[3]

The **Model of the Sensors** (lower left quadrant) specifies the different types of sensors installed in the building. Note that in figure 3 sensor characteristics like measured variables, and quality of measures, like availability or accuracy, are omitted for clarity. However, the model represents the location of the sensors by relating them to a physical item of the domain model.

The **Model of the Sensor Events** (lower right quadrant) defines the types of sensor events. Depending on the specific type of a sensor different data can be produced. For instance, a contact sensor might produce data for signalizing that a door has been opened. Furthermore, each sensor event is related by the *sensed-by* relationship to a corresponding sensor, and thereby to a certain position. Note that this relation shows the connection between the world and the event model.

The **Model of the Domain Events** (upper right quadrant) presents all *application events* considered in the system. On the one hand, there are the *context events* that are produced directly by software components. Figure 3 defines *heatingStart* events and *lectureStart* events as specific examples of context events, which might be produced by a heating plan and a class schedule, respectively. On the other hand, *situation events* signal that a certain situation has occurred in the building, e.g. a room has been occupied or freed.

Furthermore, action events are considered like *increaseTemperature* or *reserveRoom* events. These events trigger some actions in the backend system, e.g. turning up the heating. They are produced by CEP rules that might correlate situation events and context events.

4.4 Event Processing Rules

In the following, we present some exemplary rules in a pseudo language to provide a better understanding of the intelligent reasoning capabilities our approach. An event processing rule contains of two parts: a *condition* part describing the requirements for the rule to fire and an *action* part to be performed if the requirements are matched. The *condition* is defined by an event pattern using several operators and further constraints.

[3] Note that this is not a very sophisticated model: many aspects are not shown, e.g. different user types and their behavior defined by working times and the usually used rooms.

Operators

`AND`	Combination of events or constraints
`NOT`	Negation of a constraint
`->`	Followed-by operator. Sequence of conditions.
`Timer`	*Timer(time)* defines a time to wait *Timer.at(daytime)* is a specific (optionally periodic) point of time.
`.within`	defines a time window for an event in which the event has to happen to be considered.

The following two rules are part of the rule base of the Situation Agent (see Figure 1) and produce a *situation event*. Note that these rules detect a certain situation in the building that can be exploited in different application domains. Here, we show how the Energy Control Agent can use identified situations to derive energy control actions. But also other kind or agents, for instance Security Agents can make use of *situation events* for detecting security risk or incidents.

The first rule produces a *situation event* of type *RoomOccupiedEvent* indicating that a certain room is currently occupied.

```
rule: "room occupied"
CONDITION DoorOpenEvent AS d ->
          Timer(5 minutes) ->
          MovementEvent AS m
          AND (d.room = m.room)
ACTION    new RoomOccupiedEvent(d.room)
```

A room is assumed to be occupied if the door is opened and five minutes later a movement is still observed. The delay will prevent false positives, like only cleaning the room or just quickly picking up some things. Note that the rule correlates two sensor events to derive a new complex event of type *situation event* with a new application-specific meaning.

The next rule considers the opposite situation: a room is not occupied if the door is closed and there is no movement within the following 10 minutes.

```
rule: "room not occupied"
CONDITION DoorCloseEvent AS d AND
          NOT MovementEvent.
              within(10 minutes) AS m
          AND (d.room = m.room)
ACTION    new RoomNotOccupiedEvent(d.room)
```

The following rules reside in the rule base of the Energy Control Agent (see Figure 1) and correlate *situation events* to derive some *action events* triggering some reactions in the backend system.

The first occupancy per day of an office is of special importance, since from then on the office is in use and needs to reach its operating temperature. Before that, room temperature could be a bit lower yielding a reduction of the heating costs.

```
rule: "first usage"
CONDITION Timer.at(06:00 AM) ->
          NOT RoomOccupiedEvent AS n ->
          RoomOccupiedEvent AS r
          AND (r.room.type = office)
          AND (n.room = r.room)
ACTION    IncreaseTemp(r.room)
```

The rule considers the situation in a certain room after 6:00 AM. If then a *RoomOccupied* event *r* occurs and there was no other *RoomOccupied* event *n* (between 6:00 and the occurrence of event

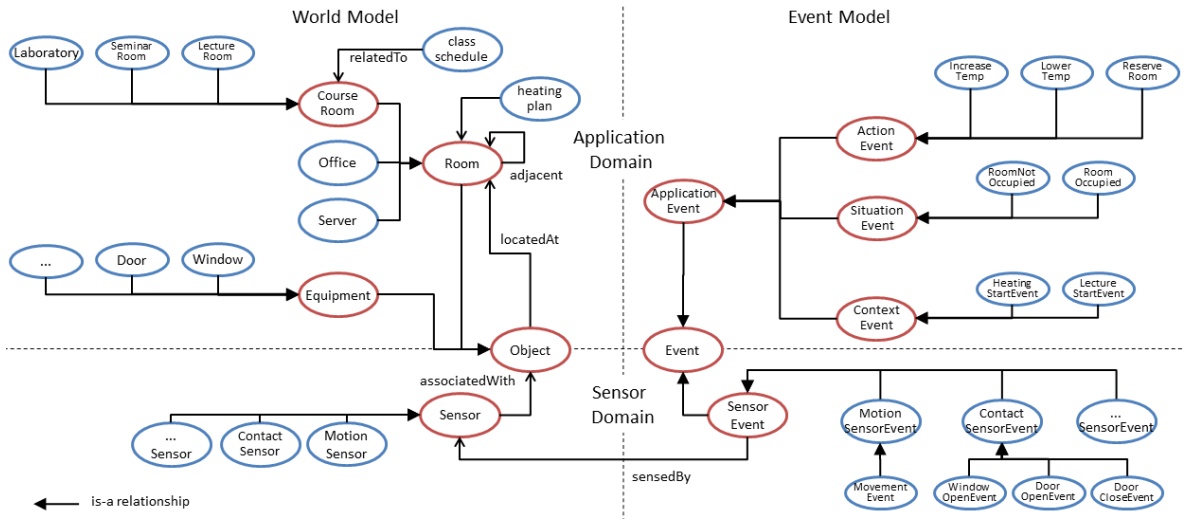

Figure 3. Domain Event Model

r) then the room is used for the first time that day and the temperature should be increased.

Another situation of interest is the 'final' absence of an employee. A shorter break, e.g. having lunch or a meeting, should not have the effect of cooling down the employees office. But after the typical end of the workday the probability that the room will be in use again is very low. Therefore, the heating can now be lowered to reduce the energy consumption.

```
rule: "after hour"
CONDITION Timer.at(06:00 PM) ->
          RoomNotOccupiedEvent AS r
          AND (r.room.type = office)
ACTION    LowerTemp(r.room)
```

If after 6:00 PM a *RoomNotOccupied* event is captured in an office room the temperature will be reduced.

The next rule illustrates how *context events* from general knowledge sources and *sensor events* are correlated to derive an *action event*. In particular, the rule describes the situation that though a lecture is scheduled, the lecture room is not occupied.

```
rule: "planned, but not used"
CONDITION LectureStartEvent AS l ->
          NOT RoomOccupiedEvent.
              within(15 minutes) AS r
          AND (l.room = r.room)
ACTION    lowerTemp(r.room)
```

The rule will fire if a *LectureStart* event is captured for a certain room, but within the following 15 minutes no *RoomOccupied* event occurs. This leads to the assumption that the lecture will not take place and accordingly the room will not be in use and the temperature can be lowered to the idle state.

Finally, we present a simple rule that exploits further domain-specific knowledge in the event reasoning. Several semantic relationships are represented in the Domain Event Model, which can be exploited to enhance the reasoning capabilities of event processing. For example, lecture rooms, seminar rooms and laboratories are all of type course room as specified by a 'is-a' relationship in the Domain Event Model. The semantical meaning of the 'is-a' relationship can be used in event processing: A rule for course rooms is implicitly valid for all subtypes as well.

```
rule: "course room not used for more
      than 1 hour"
CONDITION NOT RoomOccupiedEvent.
                within(60 minutes) as r
          AND (r.room.type = course)
ACTION    lowerTemp(r.room)
```

If a course room is not used for at least 60 minutes, then the temperature of the room can be decreased. This rule will match for a lecture room, but not for other rooms as offices.

Note that we will investigate the modeling of much more semantic relationships using an appropriate formalism in further researches. For instance with OWL, relationships between concepts can be described much more precisely. For OWL object properties ranges and domains can be specified as well as further property characteristics (as transitivity, symmetry or reflexivity). This information can be exploited by more sophisticated reasoning, for instance by using SPARQL query language.

5 Case Study / Evaluation

We have equipped one room of our university building with a sample setup of different physical sensors and implemented a prototype of our event-driven energy control system. As sensor hardware we used a *Phidget*[4] *Interface-Kit* and corresponding motion sensor and contact switches.

The event processing is implemented with the open source CEP engine *Esper* [5]. Esper provides the essential features of typical CEP systems like time windows, external method calls, and event pattern operators. The event processing rules are defined in a SQL-like rule language , the so-called *Continuous Query Language* (CQL). In contrast to SQL the CQL queries are not executed on a Database, but directly in-memory on the continuously arriving event stream.

Our experimental evaluation has proven the capabilities of CEP to correlate the sensor data and achieve a real-time analysis and

[4] http://www.phidgets.com
[5] http://esper.codehaus.org

reaction based on the sensor data stream and event patterns.

Since we could only realize an example installation for one distinct room, the usefulness of our approach is evaluated on the basis of considerations about the real usage. We assume a typical heating behavior in a static manner starting heating at 6 AM until 9 PM. The typical workday of an university lecturer (as an example of a non-uniform user type) may be structured as followed: *start of work* at 8 AM, *lecture* between 10 AM and 11:30 AM, *lunch break* between 11:30 AM and 12 PM, *exercise lesson* between 12 PM and 1:30 PM, and *end of work* at 5:30 PM.

Figure 4 visualizes the different heating behaviors by example of a lecturer's office room: (a) dynamic heating with our approach based on situational awareness compared to (b) the static solution with a fixed heating plan. The *Human Presence* depicts the occupancy of

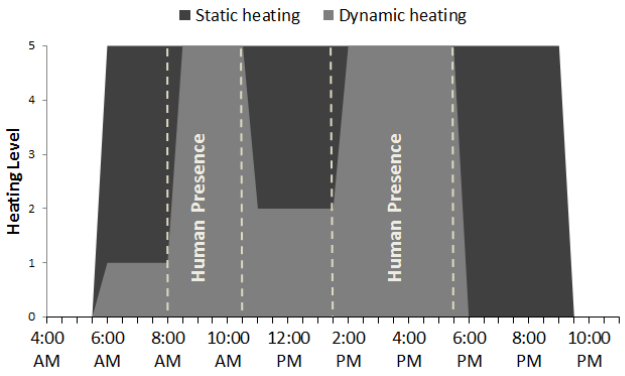

Figure 4. Static versus dynamic heating

the room according to the typical workday defined above. The two curves describe the heating level on the y-axes with respect to the time.

As can be seen, the biggest differences, and therefore energy savings, appear during the time before and after the workday. Notice that the usual hours of work may differ from lecturer to lecturer and thus the static schedule can not be fitted to the typical behavior of one lecturer.

In contrast, our approach provides a room-specific and situation-aware control mechanism enabling a precise energy management that additionally enables the heater to turn lower during temporary absence. In order to keep the room in a comfortable state, if the user returns, the heaters level is only decreased and not completely turned off for temporarily unoccupied rooms.

Based on these assumptions a calculation comparing the two different heating behaviors (static versus dynamic) results in a heating reduction up to 30% and, accordingly, lower energy consumption and carbon emission.

6 Conclusion

In this paper, a novel approach for intelligent energy management by means of complex event processing and simple sensors has been presented. The approach is different from other approaches in that is based on a sophisticated representation of domain as well as sensor knowledge and a multi-staged event processing architecture. By the integration of domain knowledge and semantic information into to the reasoning process we achieve an intelligent, situation-aware behavior.

The approach allows an individualized, situation-aware energy management of buildings according to the current occupancy status of the separate rooms. By means of complex event processing an existing infrastructure with everyday sensors can be expanded into an intelligent environment.

Directions of future research are, among others, the further enhancement of the semantic Domain Event Model as well as the development of advanced concepts for the incorporation of the semantic knowledge in event processing languages.

7 Acknowledgement

This work was supported in part by the European Community (Europäischer Fonds für regionale Entwicklung – EFRE) under Research Grant EFRE Nr.W2-80115112.

REFERENCES

[1] Y. Agarwal, B. Balaji, R. Gupta, J. Lyles, M. Wei, and T. Weng, 'Occupancy-driven energy management for smart building automation', in *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building (BuildSys)*. ACM, (2010).

[2] J. C. Augusto and C. D. Nugent, 'The use of temporal reasoning and management of complex events in smart homes', in *Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI)*, pp. 778–782. IOS Press, (2004).

[3] R. Bruns and J. Dunkel, *Event-Driven Architecture: Softwarearchitektur für ereignisgesteuerte Geschäftsprozesse*, Springer-Verlag, Berlin Heidelberg, 2010.

[4] J. Dunkel, A. Fernández, R. Ortiz, and S. Ossowski, 'Injecting semantics into event-driven architectures', in *Proceedings of the 11th International Conference on Enterprise Information Systems (ICEIS)*, pp. 70–75. Springer, (2009).

[5] R. Holland-Moritz and R. Vandenhouten, 'A flexible architecture for intelligent management systems', in *Proceedings of the 3rd International Symposium on Logistics and Industrial Informatics (LINDI)*, pp. 83–86. IEEE Computer Society, (2011).

[6] S. R. Jeffery, G. Alonso, M. J. Franklin, and J. Widom W. Hong, 'A pipelined framework for online cleaning of sensor data streams', in *Proceedings of the International Conference on Data Engineering (ICDE)*, p. 140, (2006).

[7] J. Landay, Y. Shi, D. J. Patterson, Y. Rogers, X. Xie, J. Scott, A.J. Bernheim Brush, J. Krumm, B. Meyers, M. Hazas, S. Hodges, and N. Villar, 'Preheat: Controlling home heating using occupancy prediction', in *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp)*, pp. 281–290. ACM, (2011).

[8] D. C. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*, Addison-Wesley, Boston, 2002.

[9] S. Selvakennedy, U. Rohm, and B. Scholz, 'Event processing middleware for wireless sensor networks', in *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW)*, p. 65, (2007).

[10] N. Stojanovic, D. Milenovic, Y. Xu, L. Stojanovic, D. Anicic, and R. Studer, 'An intelligent event-driven approach for efficient energy consumption in commercial buildings: Smart office use case', in *Proceedings of the International Conference on Distributed Event-based System*, pp. 303–312. ACM, (2011).

[11] J. Y. C. Wen, G. Y. Lin, T. Sung, M. Liang, G. Tsai, and M. W. Feng, 'A complex event processing architecture for energy and operation management', in *Proceedings of the 5th International Conference on Distributed Event-Based Systems (DEBS)*, pp. 313–316. ACM, (2011).

[12] Y. Xu, N. Stojanovic, L. Stojanovic, D. Anicic, and R. Studer, 'An approach for more efficient energy consumption based on real-time situational awareness', in *The Semanic Web: Research and Applications*, pp. 270–284. Springer, (2011).