Toolchain for User-Centered Intelligent Floor Heating Control

Mads Kronborg Agesen, Kim Guldstrand Larsen, Marius Mikučionis,
Marco Muñiz, Petur Olsen, Thomas Pedersen, Jiří Srba, Arne Skou
Department of Computer Science
Aalborg University, Denmark
{kronborg,kgl,marius,muniz,petur,tp,srba,ask}@cs.aau.dk

***Abstract*—Floor heating systems are important components of nowadays home-automation setups. The control of a floor heating system is a nontrivial task and the present solutions essentially implement variants of a simple bang-bang controller that opens for a hot water circulation in a room if its current temperature is below the user defined target temperature, otherwise it closes for the heating in the room. The disadvantage is that the heat exchange among the rooms, outside weather conditions, weather forecast and other factors are not considered. We propose a novel model-driven approach for intelligent floor heating control based on a chain of tools that allow us to gather the sensor readings from the actual hardware and use the state-of-the-art controller synthesis tool UPPAAL Stratego in order to synthesise abstract control strategies that are then executed on the real hardware platform provided by the company Seluxit. We have built a scaled demonstrator of the system and the experimental results document a 38% to 52 % increase in user satisfaction, moreover with additional energy savings between 2% to 12%.**

***Keywords*—Home automation, energy efficiency, hybrid systems, strategy synthesis.**

I. Introduction

The number and the variety of use of control systems in the context of home automation is continuously increasing. Usual applications include heating systems, lightning systems, fire alarm systems, etc. Common goals of these systems are to optimize the energy consumption and to improve the user comfort, which gives rise to new challenging research problems in the areas of control and optimization. In addition, these systems are usually produced by different companies, and most companies tend to use their own proprietary protocols. As a consequence there is a need for a middleware to interface heterogeneous home automation networks.

Floor heating systems are widely used for thermal comfort. We focus here on a floor heating controller offered by the Danish company Seluxit[1]. The system involves a large number of components e.g. temperature sensors, communication gateways, relay boxes and valves. The system is influenced by various factors such as outside temperature, adjacent rooms, status of the doors (open/closed) and the present weather conditions. The system implements a proprietary communication protocol which provides a control interface. Currently the system uses a "Bang-Bang" controller which only considers the current room temperatures and disregards other important factors, hence resulting in poor controlling performance. In order to provide better controlling, there is a need to integrate relevant information (weather forecast, user temperature profiles, etc.) with tools for optimal controlling of the floor heating system.

In the context of the floor heating, we have previously demonstrated by model-based performance analysis [1] that the control strategies obtained by the new tool Uppaal Stratego [2] significantly improves upon the currently applied "bang-bang" control strategies. The focus of this paper is on the complete toolchain architecture which allows for these strategies to be executed in order to control the heating in *real* houses through communication with *real* devices used in the house. Besides Uppaal Stratego the main component of the toolchain is that of the HomePort [3] middleware implementing a common interface to heterogeneous home automation networks. The main novelty of the present work is the automatic integration with web-services providing weather forecast, user-defined temperature profiles, the actual construction of a working system in our demonstrator (using the hardware components present in the real house) and the integration of abstract control strategies synthesize by the tool Uppaal Stratego into a real executable code.

Related Work

A key feature of the toolchain is that of improved controlling. For this we use a novel approach presented in [1]. In [4] a formal approach for automatic synthesis of an off-line controller for floor heating is proposed. The authors consider the same real house as we consider. The time cost of computing such an off-line controller was approx. 20 hours. Model predictive controllers (see e.g. [5]) are suitable for systems with enormous state space. However, they usually do not allow to model stochastic systems and are computationally expensive. Since the system requires a control input every 15 minutes, the previous approaches are not feasible. We use a very fast and computationally inexpensive approach. In [6] a synthesis tool PESSOA controlling cyber-physical systems. Similarly in [7] a subclass of hybrid systems is presented with a focus on controlle synthesis via abstraction techniques. In [8] a combination of statistical model checking with ANOVA has been suggested for synthesis of various control strategies. The

[1] www.seluxit.com

main novelty of our work is that we deal with an industrial-size case study with huge state-space complexity where the other studied methods and approaches do not scale.

II. Core Technologies

A key element in the toolchain is the ability to directly translate commands described in abstract models into sensor/actuator commands on physical devices. In this section, we provide a short description of the applied modelling notation (timed automata in UPPAAL), the platform for device management (HomePort), and the method for linking the two.

A. UPPAAL *and* UPPAAL STRATEGO

UPPAAL [9], [2] is a toolbox for modelling, simulation, synthesis and optimization of real-time systems. It is composed of three parts: a description language, a simulator, and analysis engines including a model checker, a statistical model checker, a timed game synthesizer and an optimizer. The description language uses communicating state machines with timers. The simulator enables exploration of possible executions of a model. The model checker can check safety, liveness and reachability properties on a model. The synthesizer can synthesize winning strategies of timed game models. The statistical model checker can estimate probabilities of reachability based on statistical simulation of models. The optimizer, UPPAAL STRATEGO, can identify optimal control strategies for stochastic priced timed games through simulation and machine learning and even do optimization under safety constraints defined by winning game strategies [10].

Processes communicate through channels and shared variables. A channel synchronizes a sender and a receiver process. A transition involving synchronization on a channel can only be taken when both the sender and receiver are able to perform the synchronization. Broadcast channels can also be used to synchronize a sender with multiple receivers. To exchange values between processes, synchronous value passing [9] is used. Synchronous value passing is a modelling technique where a sender and a receiver synchronize on a channel and exchange a value by using a shared variable. Transitions can be guarded by boolean expressions over clocks and discrete variables. A transition with a guard can only be taken when the guard evaluates to true. Transitions can also update the values of variables through update statements and reset clocks. Finally, hybrid phenomena can be expressed in models via differential equations as shown in Figure 1, which models the temperature development in a specific room taking various heat exchange relations into account.

In the toolchain, UPPAAL is first used to model the rooms through a graphical user interface. The optimizer UPPAAL STRATEGO is then used to generate a near-optimal strategy at regular time intervals through a series of steps to be explained in the technology section. The generated strategy is executed via the HomePort gateway on the actuators of the floor heating system and the current room temperatures are received from the sensors. The sensor readings are then applied as a basis for the next strategy generation.

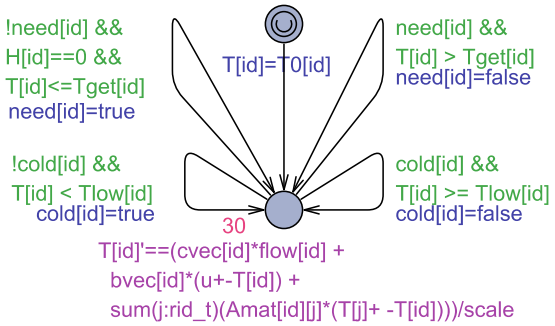

Fig. 1. Room model

B. HOMEPORT

HOMEPORT [3] is a middleware implementing a common interface to heterogeneous home automation networks. It facilitates the tasks of the control system by enabling it to access and modify the services of the environment in a unified manner, regardless of the protocol or technology that a specific device is using. Moreover, its REpresentational State Transfer (REST) application is easy to interface with. It also provides an event mechanism allowing the control system to receive notifications when the state of a service changes. The overall architecture is described in Figure 2.

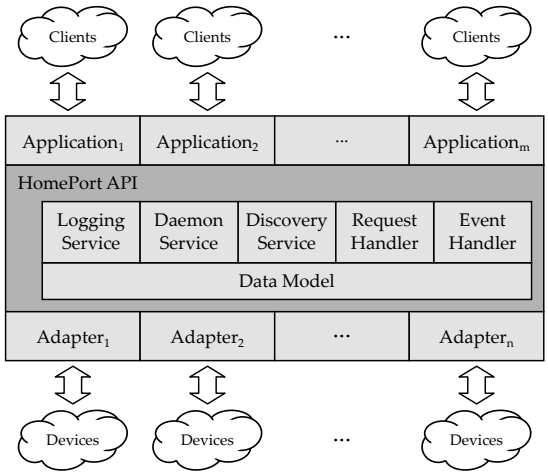

Fig. 2. Homeport architecture

HOMEPORT plays two roles in the toolchain: to provide a list of available services and to provide read and write access to these services. Access to the services of a HOMEPORT instance is available through its web interface and the available services may be linked to simulated UPPAAL time automata through dedicated channels as described in [3]. However, in this case study, the generated strategy simply consists of temperature readings followed by a timed sequence of open/close actuator commands, so a more simplistic approach has been chosen by interpreting the strategy directly.

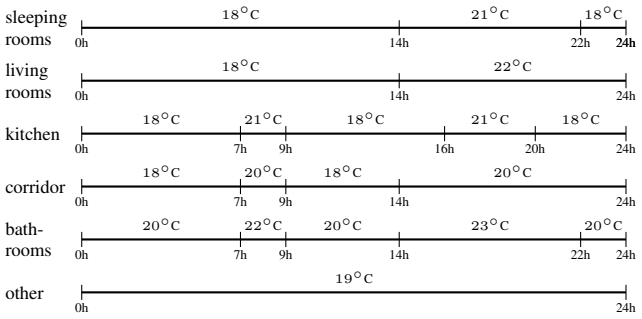

Fig. 3. User defined temperature profile for 24 hour cycle

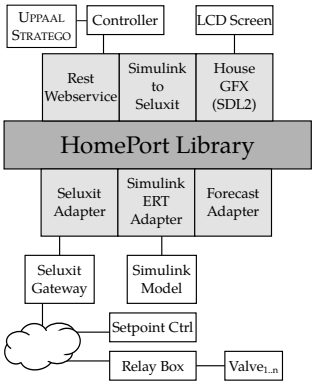

Fig. 4. Toolchain architecture

III. Floor Heating Case Study

The heating system consists of a number of heated water pipes going through the rooms under the floor. The water supply is controlled via valves at the pipe entrance to the building and the heating can be directed to each room individually. Some older houses may have poorer pipe insulation and hence the heat may leak to neighboring areas on transit, thus limiting the direct control. The heat can be lost to the outside environment through the roof, windows and walls, and it is also exchanged between the rooms through walls and doors. The purpose of the heat controller is to maintain the temperature in each individual room based on the user demands. We assume that the user can supply a temperature profile for each room. Figure 3 shows an example room temperature profile provided by the user: a desired temperature setting specified for each day time interval in each room. Our experiments use exactly this profile, however in practice it can be adjusted on-demand. As the outside temperature has the major impact on the inner climate, we also equip the controller with a weather forecast web-service so that it can prepare and adjust its heat supply in advance.

Most of control context (temperatures and forecast) cannot be fixed for long time horizons in advance, therefore the controller must be adaptive. We propose an online strategy synthesis where the controller is synthesized periodically based on the current situation and the new forecast information, details about the underlying game theoretical foundation of the model can be found in [1].

The controller fitness is then evaluated after time interval τ based on the importance W_i of the room i and the distance between the goal T_i^g and actual T_i temperatures:

$$D(\tau) \quad = \quad \int_0^\tau \sum_i W_i \cdot \left(T_i(t) - T_i^g(t)\right)^2 \cdot dt \qquad (1)$$

IV. Tool Chain

We shall now describe the components of the toolchain used for automatic synthesis of a smart controller for the floor heating, focusing on the communication with real devices used in the house. Figure 4 shows an overview of the architecture.

The system is based around the HomePort middleware [3] mentioned earlier, used for solving heterogeneity issues in distributed systems, and in particular home automation systems. Heterogeneity arises in home automation due to the difference in the requirements of the subsystems and the variety of incompatible communication protocols and web-service interfaces. Heterogeneity is thus not only a problem, but also a necessity in order to satisfy different sets of requirements. To solve this issue, there is a need to adapt the information transmitted from one subsystem to another. HomePort provides this functionality. In our case, we have devices from the real house including the Seluxit gateway, temperature sensors, valve actuators as well as weather forecast web-services and front-end GUI designed by Seluxit for setting up the target temperatures and the temperature profiles for each room. The chosen approach enables also future support for devices from other vendors (e.g. ZigBee or Z-Wave), and allows for easy switching between real and virtual devices (as shown in our demonstrator that replaces the actual house with a Simulink simulator of the house termodynamics).

In HomePort, all devices from the underlying protocols are abstracted and exposed through a heterogeneous RESTful HTTP interface. The translation from the messages transmitted over the Web to the ones transmitted in the subnetworks is performed by adapters. The graphics application provides graphical interface showing the floor plan. It monitors the other adapters and updates the interface based on the dynamics of the system. The Simulink adapter runs the simulations of the floor heating (it is replaced by the actual temperature readings in the real house). The Seluxit adapter provides access to real physical devices. In our case it includes the temperature sensors, set-point control for target temperatures via graphical user interface or stand-alone set-point devices placed in the house and a relay box for controlling the floor heating valves.

To generate control strategies for the floor heating, UPPAAL STRATEGO [2] is interacting with HomePort on the HTTP interface, and is intended to run either locally on the Seluxit gateway or centrally outside the house in a cloud. Through the HomePort interface UPPAAL STRATEGO can interact with the devices, read temperature sensors and set valve positions.

The HTTP interface can also be used to develop a graphical web interface for showing e.g. the live temperatures in the

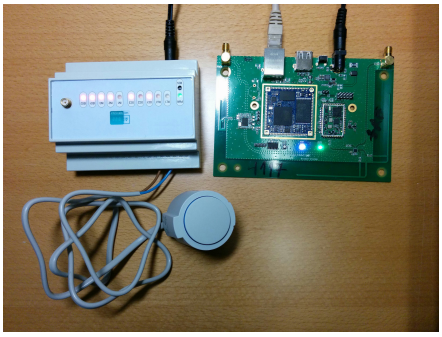

Fig. 5. Seluxit devices: gateway (top right), relay box (top left), valve (bottom)

house, as exploited by the company Seluxit. This is then accessible through smartphones or tablets, such that the inhabitants can monitor and control the house.

A. The Setup

HomePort including all adapters and the graphical interface runs on a Raspberry Pi. To illustrate the connection with real devices a Seluxit gateway is wirelessly connected with a relay box for controlling the valves. These valves are set synchronously with the virtual valves in the simulation. Figure 5 shows the Seluxit gateway (without cover), the relay box and one connected valve.

B. Graphical User Interface

For the use in the demonstrator, the graphical user interface is written in C++ using SDL2. Figure 6 shows the GUI. A

Fig. 6. The GUI showing the floor plan

floor plan of the house contains gray walls separating brown floors. The floors are colored according to the temperature; blue rooms are cold, red rooms are warm as illustrated in Figure 6 (the scenario is not a realistic one and it is used only for presentation purposes). The pipes of the floor heating system are shown throughout the floor. When a valve is open the pipe turns red, illustrating that the pipe is heating the room. When the valve is closed the pipes turn gray and slightly transparent.

For each room two temperatures are shown. On top is the target temperature, shown in integer numbers without decimals. Below is the actual room temperature shown with one decimal. The temperature gauge next to the numbers also illustrates the temperature. In the bottom right is the outside temperature (currently 20.5°) and the time of day in hours in 24-hour format (currently 12h). In the middle top the status of the valves is shown as light gray boxes. A red flame icon illustrates that the valve is on, an empty gray box means that the valve is off. The green area at the bottom shows the lawn in front of the house.

C. Simulink Adapter

The thermo-dynamics of the floor heating system are modeled in Simulink in the demonstrator (these are replaced by the actual temperature readings in the real house). This model includes the thermal capacity of the walls and the heat dissipation between rooms and from the outside. The doors between rooms open and close stochastically (in the current version this is not shown on the GUI). Since a real floor heating system runs very slowly, the simulation is running at an increased speed. Currently the speed is set to one minute per second. This can be changed, and the demonstrator is planned to support replaying of previously simulated scenarios, at speeds in the order of days per minute.

From the Simulink model, code is automatically generated using Embedded Coder. This outputs C code which is compiled into a dynamically linked library. A small HomePort adapter is written which interacts with this library and exposes it as HomePort devices. This allows us to monitor the temperature developments and control the target temperatures and valve positions in the simulation.

D. Seluxit Adapter

The Seluxit adapter exposes the Seluxit devices though the HomePort interface, allowing the monitoring and control. As physical devices are connected, these can be used instead of the virtual Simulink devices. HomePort connects to the Seluxit gateway through an Ethernet connection, using Seluxit's new JSON web interface. This interface allows reading and setting values using simple JSON messages. For example, turning on valve with id `"1234"` is done by sending value 1, as shown below.

```
{ ":type": "urn:seluxit:xml:bastard:value-1.0",
  ":id": "1234", "data":"1" }
```

The real temperature sensors intended to run with the system are wireless, and thus need to conserve energy. They therefore operate in a sleep mode, where they wake up every 15 minutes and report the current temperature. The gateway caches this value and presents it to the HomePort adapter when it asks.

E. Weather Forecast Adapter

Weather forecast data as well as real measured outside temperature is exposed through a weather forecast adapter. The data is fetched from the HTTP API from Weather Underground (www.wunderground.com). Historical data is fetched from five different locations, in order to try different weather scenarios as discussed in our experiments later on. The adapter can also be set to deliver live data. During the simulation, the forecast

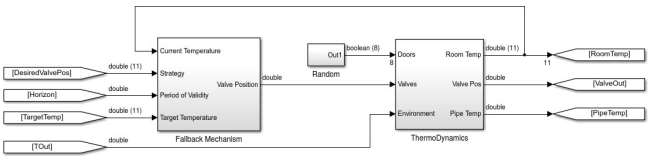

Fig. 7. Overview of Simulink® model.

adapter reads the simulated time from the Simulink adapter and provides forecast and temperatures for the matching time, this way the historical data is replayed at simulated time.

F. Stratego Interface

UPPAAL STRATEGO is intended to run off-site and therefore does not run on the same Raspberry Pi as the rest, typically on a server in a real setup. UPPAAL STRATEGO connects to HomePort through the web interface. It checks for available devices by getting the devices list, an excerpt is shown below.

```
<configuration>
  <adapter id="dt" network="Simulink">
    <device desc="Valve1" id="EjSb"
     location="Room1" type="Valve">
      <service desc="Valve Position" id="9vPF"
               uri="/Valve/EjSb/ValvePos/9vPF"
               isActuator="0" type="ValvePos">
        <parameter max="100" min="0"
         type="double" unit="%"/>
      </service>
    </device>
    ...
  </adapter>
</configuration>
```

This shows a valve device available on URI `"/Valve/EjSb/ValvePos/9vPF"`. UPPAAL STRATEGO can change the position of this valve by making a `PUT` request on this URI with the following XML message:

```
<?xml version="1.0" encoding="UTF-8"?><value>1.0</value>
```

UPPAAL STRATEGO continuously reads the current temperatures and target temperatures and uses these to compute a good strategy for valve configurations, that will optimally keep the rooms as close to the target as possible. The HomePort web interface is easily accessible and allows other control strategies to access the devices. This opens possibilities to try different strategies and visualize the results. Once deployed in a real setting, this can also be used for dynamical switching of control strategy.

V. DEMONSTRATOR

A physical demonstrator around the architecture shown in Figure 4 has been developed. HomePort including SIM-, SLX adapters and the GFX-, SDL2 graphics visualizing the floor plan are compiled and deployed as one binary on a raspberry PI running Linux. The Simulink® model is deployed as a shared library using Embedded Coder® and a slightly customized standard Linux target. The customization integrates the interface to Homeport and the ability to specify the execution interval. For demonstratrion purposes, a Simulink®

TABLE I
EVALUATION IN DIFFERENT WEATHER SCENARIOS.

Weather	Distance			Energy		
	Bang-Bang	STRATEGO	imp.	Bang-Bang	STRATEGO	imp.
Aalborg	14583	8342	43%	14180	12626	10%
Anadyr	2385515	1483272	37%	23040	22475	2%
Ankara	17985	10464	41%	17468	15684	10%
Minneapolis	22052	12175	44%	18165	15882	12%
Murmansk	399421	187941	52%	22355	21011	6%

model of the house thermodynamics is derived based on the following differential equations:

$$\frac{d}{dt}T_i(t) = \sum_{j=1}^{n} A^d_{i,j}(T_j(t) - T_i(t)) + B_i(T_e(t) - T_i(t)) + H^v_{j,i} \cdot v_j \tag{2}$$

where T_i is the room temperature of room i, $A^d_{i,j}$ contains the heat exchange coefficients between room i and room j, given the door configuration d. The vector B contains the heat exchange coefficients between the outside temperature T_e and each room, and H^v contains the heat exchange coefficients for each pipe and the rooms it traverses.

An overview of the Simulink® model is given in Figure 7, showing the input/output, implementation of fallback controller and thermodynamics in the subsystems. The majority of the signals used in the Simulink® model are vectors representing e.g. 11 room temperatures in one signal. To interface the model, when deployed as a shared library, the HomePort adapter utilizes root-level input/output blocks, which are intentionally keep off the figure and represented as signal routing blocks. Differential equations are implemented as a part of the ThemoDynamics block. The core model is implemented as an m-function taking valve position, door opening, outside temperature and the room temperature as input, calculating delta temperature for each model time step which is feed into an integrator to calculate the current room temperature. Having all major components of the demonstrator connected through HOMEPORT makes it easy to share data. For instance, the Simulink® model room temperatures are accessible for UPPAAL STRATEGO. Similarly, the GFX adapter is capable of visualizing data from all sources, including states of physical connected valves and the modeled room temperature.

STRATEGO interfaces the simulated house and real valves through HomePort RESTful HTTP interface, fetching current state each 15 min, calculating a new control strategy and applying the optimal valve configuration and the period of validity of the given strategy. In case the period of validity is exceeded, a simple fallback control strategy (bang-bang) is enforced by the Simulink® model. In normal circumstances, STRATEGO applies a new strategy within the period of validity.

VI. EXPERIMENTS

The applicability and efficiency of the toolchain has been evaluated on five different weather scenarios for different cities in the world and for a fixed user temperature profile presented in Figure 3. The weather forecast was dynamically updated by

TABLE II
EVALUATION WITH APPROXIMATE CONSTANTS (DEVIATING UP TO 20%) IN THE MATRICES A, B AND H FROM EQUATION 2.

Weather	Distance			Energy		
	Bang-Bang	STRATEGO	imp.	Bang-Bang	STRATEGO	imp.
Aalborg	14583	8552	41%	14180	12590	11%
Anadyr	2385515	1503448	36%	23040	22371	2%
Ankara	17985	10511	41%	17468	15697	10%
Minneapolis	22052	12725	42%	18165	15837	12%
Murmansk	399421	191441	52%	22355	20923	6%

repeatedly consulting a Weather Underground web-service and sending the weather forecast data to UPPAAL STRATEGO. The actual data that show the computed distance (see Equation (1) for the bang-bang controller implemented by the company Seluxit and the strategy synthesized by UPPAAL STRATEGO are presented in Table I. We would like to emphasize that the actual distance function (user discomfort Equation 1) was computed using the real historical temperature data for the three day period that was explored (January 29th to February 2nd, 2016) and not the data sent to UPPAAL STRATEGO about the weather forecast. The actual house thermodynamics was then simulated using Simulink in order to evaluate the actual performance at several different weather scenarios. Clearly, UPPAAL STRATEGO produces a strategy that follows more closely the user set temperature profile with the relative improvement between 37 to 52 percent. This is due to the predictive strategy that is aware of the user temperature profile and the expected weather conditions.

As a side effect we measured also the energy requirements for the bang-bang strategy versus UPPAAL STRATEGO strategy. The energy corresponds to the volume of hot water used to heat up the rooms (larger rooms require more water volume than smaller rooms). Also here we can see considerable savings ranging from 2 up to 12 percent.

Finally, in the actual house we might not have the access to the precise heat exchange constants among the rooms and the environment. Hence in Table II we randomly change these constants in our model by up to 20% and the achieved performance of our controller is still almost as good as in the system with the exact knowledge of these constants. This demonstrates the robustness of our approach.

VII. CONCLUSIONS AND FUTURE WORK

We have presented a framework for intelligent floor heating control based on a chain of tools that allows us to connect the different hardware and software components via a unified interface and hence enable an online use the state-of-the-art synthesis tool UPPAAL STRATEGO for automatically producing efficient control strategies. The synthesized strategies are then executed on the actual hardware as documented by a physical demonstrator that we built. As a result, our approach significantly increases the user satisfaction with the system (the distance from the user target temperatures in the 24 hour temperature profile is smaller than in the currently employed control system designed by Seluxit engineers). As a side effect, taking into account the weather forecast and the heat exchange among the rooms depending on their actual placement in the house, we are able to decrease also the energy needed for heating up the house, often saving around 10% of the heated water consumption.

While our method scales well for the short term floor heating control, some optimization questions (like e.g. optimizing in 70% for the user comfort and 30% for energy savings as a combined criterium) require to look several days ahead in order to produce a globally optimal controller strategy. The tool does not currently scale to such long look-ahead horizons and the construction of the composed short term strategies may not necessarily achieve an optimal behaviour. As a future work we are working on improving the learning algorithms used in UPPAAL STRATEGO in order to handle even more complex problems with longer time horizons.

Acknowledgments. The research leading to these results has received funding from the EU FP7 FET projects CASSTING and SENSATION, the project DiCyPS funded by the Innovation Fund Denmark, the Sino Danish Research Center IDEA4CPS and the ERC Advanced Grant LASSO. The author J. Srba is partially affiliated with FI MU, Brno.

REFERENCES

[1] K. Larsen, M. Mikucionis, M. Muniz, J. Srba, and J. Taankvist, "Online and compositional learning of controllers with application to floor heating," in *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16)*, ser. LNCS, vol. 9636. Springer-Verlag, 2016, pp. 244–259.

[2] A. David, P. Jensen, K. Larsen, M. Mikučionis, and J. Taankvist, "Uppaal stratego," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. LNCS. Springer, 2015, vol. 9035, pp. 206–211.

[3] T. Le Guilly, P. Olsen, A. Ravn, J. Rosenkilde, and A. Skou, "Homeport: Middleware for heterogeneous home automation networks," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, 2013, pp. 627–633.

[4] A. Le Coënt, L. Fribourg, N. Markey, F. De Vuyst, and L. Chamoin, "Distributed Synthesis of State-Dependent Switching Control," *ArXiv e-prints*, Apr. 2016.

[5] E. Camacho, D. Ramirez, D. Limon, D. M. de la Pea, and T. Alamo, "Model predictive control techniques for hybrid systems," *Annual Reviews in Control*, vol. 34, no. 1, pp. 21–31, 2010.

[6] P. Roy, P. Tabuada, and R. Majumdar, "Pessoa 2.0: a controller synthesis tool for cyber-physical systems," in *14th ACM International Conference on Hybrid Systems: Computation and Control (HSCC'11)*, M. Caccamo, E. Frazzoli, and R. Grosu, Eds. ACM, 2011, pp. 315–316.

[7] E. Hahn, G. Norman, D. Parker, B. Wachter, and L. Zhang, "Game-based abstraction and controller synthesis for probabilistic hybrid systems," in *Eighth International Conference on Quantitative Evaluation of Systems (QEST'11)*. IEEE Computer Society, 2011, pp. 69–78. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6041098

[8] A. David, D. Du, K. Larsen, A. Legay, and M. Mikučionis, "Optimizing control strategy using statistical model checking," in *NASA Formal Methods (NFM'13)*, ser. Lecture Notes in Computer Science, G. Brat, N. Rungta, and A. Venet, Eds., vol. 7871. Springer, 2013, pp. 352–367. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38088-4

[9] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on UPPAAL," in *SFM-RT 2004*, ser. LNCS, M. Bernardo and F. Corradini, Eds., no. 3185. Springer, 2004, pp. 200–236.

[10] A. David, P. Jensen, K. Larsen, A. Legay, D. Lime, M. Sørensen, and J. Taankvist, "Automated technology for verification and analysis (ATVA'14)," ser. LNCS, vol. 8837. Springer, 2014, pp. 129–145.