Modelling of Control Logic for Start-Up and Shut-Down Sequences of the Heat Pump

Dragan Pršić *, Aleksandar Vičovac **

* University of Kragujevac, Faculty of Mechanical and Civil Engineering in Kraljevo, Department of Automatic Control, Fluid techniques and Robotics, Dositejeva 19, 36000 Kraljevo, Serbia
** SM Engineering, Inc., Topličin Venac 21, 11000 Belgrade, Serbia
e-mail: prsic.d@mfkv.kg.ac.rs, aleksandar@sm-inzenjering.rs

Abstract—To increase energy efficiency, heat pumps are becoming part of an integral solution for residential heating, ventilation, and air-conditioning. Depending on the electricity tariff, coefficient of performance (COP) and thermal load, the heat pump can also be combined with other heat sources (smart grid). All these new possibilities additionally complicate the heat pump control. In On-Off control, special attention is paid to start-up/shut-down sequences which repeat periodically during the entire life of the heat pump. For the design of control logic for start-up/shut-down sequences, the paper uses the model-driven approach instead of the traditional method. The design is organized around the model which is developed iteratively and incrementally through several use cases. Harel's statecharts implemented by Matlab tool Stateflow are used in all phases of development: from the specification of requirements up to implementation.

I. INTRODUCTION

Ever since the beginning of application of heat pumps (HPs), one of the primary goals has been to increase their energy efficiency. The modern control theory offers new possibilities in this field [1-3]. The lower prices of sensors, actuators and computers allow practical realization of advanced control algorithms. Domestic HP systems have, more or less, a standard hardware configuration, but control algorithms, most commonly provided by programmable logic controllers, are the key factor which makes a system advantageous over the others.

In On-Off control of heat pumps, special attention is paid to start-up/shut-down (SU/SD) sequences. The realization of these sequences without errors is particularly critical during commissioning of the system. The scope and type of problems which arise at that point are a good indicator of the success of design. Their solution may sometimes require only minor modifications, but sometimes it can lead back to the design goals. The modifications in that case are expensive and demanding. The problems detected after a successful start-up can usually be solved through several iterations, but they do not affect the basic operation of the system. The start-up of the system is also important from the aspect of safety of both the employees and the equipment.

In HP systems with On-Off control, the SU/SD process repeats periodically during the entire life. Optimization of these sequences, and not only the correct operation, may contribute to the energy efficiency of the whole system.

The control of SU/SD sequences is not an easy task because it realizes through parallel processes with certain synchronization between them. The problem is even bigger when the existing processes should be modified. For instance, if we want to improve the existing functionality or add a new one.

In order to get to know a system better, it is necessary to create its model. Simulation of such a system can give us answers to various questions. Different design solutions and different use cases can be tested. It is desirable for such a model to be the backbone through all phases of development, from the definition of requirements to the final realization. It enables an iterative and incremental development of the model. Each phase, starting from the easier ones, the model is corrected through several iterations. Each phase is tested by simulation. After that, the model is added by a new increment, a new use case.

Depending on the complexity of such a model, engineers of different profiles can participate in its development. At least two profiles are necessary: process engineer (thermodynamics specialist) and software engineer (programmer). Mutual communication between domain specialists and software specialists is particularly important. The first phase in the development of a control system (CS) - proper defining and understanding of requirements, is especially critical. On one side there is a domain specialist who uses 'natural language' to describe functionalities. Textual specification can be imprecise, incomplete and inconsistent. On the other side there is a software specialist who uses, for example, a language from the IEC 1131-3 family. Such a language does not have to be understandable to a domain specialist. Verification of requirements in that case is more difficult. Every misunderstanding of requirements leads to bigger problems in later phases of design.

There may arise a question which language to use in the development of such a model. The essential thing which defines a modelling methodology is the language for model description. What can be expressed in models is determined by the language in which they are stated. The language specifies what terms can be incorporated in a model (the vocabulary) and how these can be meaningfully combined (the grammar) [4].

The control logic of a heat pump is event-driven [5]. CS responds to discrete events, such as mode switching by the customer, switching on/off by the thermostat, switching on/off by the pressure switch, mode switching by timetable, alarm occurrence, etc. The Statechart notation established by Harel [6] is used for describing discrete logic. Statecharts are based on the finite state machine...
theory. The model consists of a set of states, a set of transitions between states, and a set of actions. The Harel’s statechart introduces three new mechanisms into the conventional state-transitions diagram: hierarchy, concurrency, communication. Thanks to these mechanisms, statecharts can be used for modelling complex discrete-event systems in a compact and modular manner [7]. The Matlab® tool, Stateflow® [8], is used for realization of statecharts diagrams in this paper.

II. HEAT PUMP SYSTEM

The HP system which is considered in the paper is schematically shown in Fig. 1.

A more detailed description of operation of the installation can be found in [3], and the start-up/shut-down process is considered in this paper.

The start-up/shut-down of an HP system consists of a series of sequential steps performed one after another in a precisely defined order. The processes cover the following devices: BP, ESP, EMV, EXV, Comp. In each step, a new device is started up/shut down. The first step is known, as well as each next step up to the last one. The time that passes between two steps is variable and has final values. The SU/SD process is shown by means of the activity diagram in Fig. 2.

In regular conditions, the SU/SD of the system is realized from beginning to end, without interruptions. In other words, the control signal StartUp initiates the first action from Fig. 2a (BP_On), which initiates the next one (ESP_On) and thus until the end (Comp_On). It is similar for the signal ~StartUp (ShutDown), but the activities here flow in the opposite direction (Fig. 2b). However, both activities can be interrupted at any moment. For example, in case of occurrence of an alarm or if the user changes the desired temperature of the thermal buffer. Even in case of any interruption of power supply, the system should continue its operation where it has stopped. In that case, the system should not execute the SU/SD process until the end. The flow of activities is changed after the last completed action. For example, if the signal ShutDown appeared after the action EMV_On, the shut-down process starts from that device. Similarly, the start-up may begin during the shut-down sequence, too.

After the power is supplied, the control system can be described by four parallel processes which are executed simultaneously (Fig. 3).

The state Alarms detects irregular events in the system. They are most often connected with the exceeding of values of certain physical values in relation to their boundary values. However, this module does not treat alarms. Its task is only to forward the information about the alarm to the module ControlLogic.

The state ControlLogic has the central role in the control of the system. The communication with the environment and the user is performed in it. The signals from the sensor are read in real time and, based on them,
the control strategy is determined according to the defined algorithm.

The communication with the user is performed by the module UserInterface. It covers the communication by means of the programmable graphic display or by means of the embedded web server.

The module SUSD is responsible for the start-up/shut-down of the system according to the activity diagram in Fig. 2. In the Power On state (Fig. 3), the SUSD state communicates only with the ControlLogic state. There is no direct communication among the Alarms, UserInterface, and SUSD states. The messages are exchanged only through ControlLogic. All events in the outside world (sensors, user interface) are manifested through the one-way message (StartUp) sent by ControlLogic to the SUSD state. StartUp has two logic values: True, which denotes starting up, and False, which shuts down the system. This paper gives a more detailed presentation of the SUSD state, which is responsible for the realization of start-up/shut-down sequences of the HP system.

III. START-UP/SHUT-DOWN SEQUENCES

The composite SUSD state consists of five orthogonal substates shown in Fig. 4.

Each substate is responsible for the start-up/shut-down of one component of the HP system (Fig. 1) which participates in the SUSD sequences from Fig. 2. The processes in them are parallel, and they communicate by sending messages.

One of three design patterns is used for the creation of each substate from Fig. 4, regardless of their number.

The first pattern is used for modelling the state which is first in the activity diagram (Fig. 2). Only the message (StartUp) which arrives from the external environment (out of the SUSD state) is sufficient for activation (Off→On). In our case, that state is BP. The excitation for activation arrives from the external ControlLogic state. After a certain time, according to the activity diagram (Fig. 2), the message for activation is sent to the internal ESP state. The structure of such a state is shown in Fig. 5.

By default, after the power is supplied, the buffer pump is in the Off state. The message StartUp leads to the transition from Off into On, but only after the time \( T_{dOn} \) expires. The adding of this condition aims at preventing a too frequent shut-down of the device and introduces a kind of time hysteresis. In order to shut down BP, three conditions should be fulfilled:
- first, that a minimum time has passed after entering the On state (temporal logic operator \( \text{after}(T_{dOn}) \)). This condition aims at preventing a too frequent shut-down of the pump;
- second, that the system is in the shut-down mode (~StartUp);
- third, that the signal for the shut-down of the buffer pump has been received (~BP_On).

![Figure 5. First state in the chain](image)

The self-loop transition from Fig. 5 designates that BP, after a certain period spent in the On (\( T_{dOn} \)) state, if the system is in the StartUp mode, sends a signal for activation of the ESP pump (\( ESP_{On}=\text{True} \)).

The second design pattern is used for modelling a middle state from Fig. 4. They are the states which receive the excitation for activation from a previous internal state. Also, they send the activation message to a next internal state. For example, the structure of the ESP state is presented in Fig. 6.

![Figure 6. Middle state in the chain](image)

In this case, there is an additional condition for transition from the Off state into the On state. The flag for starting up (\( EXV_{On}=\text{True} \)) is raised by the previous state in the chain (EMV). Unlike the other devices, entering the On state in \( EXV \) values does not mean a discrete change of the state. Instead of that, the PID algorithm which continuously opens and closes \( EXV \) in order to provide desired superheat temperature [3] is initiated. The transition \( \text{On} \rightarrow \text{Off} \) takes place under similar conditions as in the previous state. The novelty is that there is self-loop transition even in the Off state. Provided that the system is in the shut-down mode, after the expiration of the time \( T_{dprev} \), the previous component in the chain shuts down.

It should be noted that if we want to add a new device to the start-up/shut-down process, e.g. Domestic Hot Water (DHW) pump, it can be done relatively easily. Its position should be located in the activity diagram (Fig. 2), and then, based on one of the previous patterns (Fig. 5, Fig. 6), a new state in the composite SUSD state should be created (Fig. 4).
The third design pattern is used for the state where the condition ~\textit{StartUp} is sufficient for its shut-down (On\textrightarrow Off). For all other devices, it is only one of two conditions. As in the first pattern, there is nonsymmetry between the conditions for start-up/shut-down. In the HP system, this pattern is usually used for the \textit{Comp} state (Fig. 7). The compressor is the last device which is started in the start-up sequence and the first to be shut down in the shut-down sequence.

Like the other devices, the compressor has its own protection time which protects it from frequent start-up/shut-down. The message for transition from Off into On state is sent by another component (in our case \textit{EXV}). However, the message ~\textit{StartUp} which is received from the \textit{ControlLogic} state is sufficient for the transition On\textrightarrow Off. As in the first pattern, there is only one self-loop transition, but here it is connected with the Off state.

The results of simulation presented in Fig. 8, Fig. 9 and Fig. 10 are obtained based on the model from Fig. 4, by using the Stateflow® (Matlab®) tool. Each of the signals shown in these figures has only two logic values (True or False). However, they are presented with different amplitudes so that their changes could be noticed more easily.

Fig. 8 shows complete sequences of the start-up/shut-down of the HP system. After the variable \textit{StartUp} obtains the value True, BP, ESP, EMV, EXV and Comp are switched on one after another, with a certain time delay. During the shut-down, the order is opposite. It can be noticed that the compressor is shut down immediately after the variable \textit{StartUp} obtains the value False because the compressor spends the time greater than \(T_{dOn}\) in the On state.

Fig. 9 shows the case when the signal \textit{StartUp} switches on the pumps BP and ESP. However, before it switches on the valve EMV, the signal obtains the value False. That is why the start-up sequence is cancelled, and after some time ESP and BP are shut down one after the other.

Fig. 10 shows two start-up sequences and one incomplete shut-down sequence. After the appearance of the signal \textit{StartUp}=True at time=1 [sec], all devices are switched on. The shut-down process starts at time=17 [sec]. Comp, EXV and EMV are switched off. However, at time=25 [sec] the variable \textit{StartUp} again obtains the value True and that is why the pumps ESP and BP are not switched off. Instead of that, the new start-up process is continued.

IV. CONCLUSION

The control logic of a heat pump is event-driven. Modelling of the control logic of the heat pump by using the model-driven method has a series of advantages over the traditional method. The traditional method is
Implementation-based and results in quick ad hoc solutions.

However, trouble shooting and the change of specification in such solutions can be very difficult. The method used is model-based and it results in long-lasting solutions. The Statechart notation implemented by Stateflow® is used for describing discrete logic. The design is based on executable models which are developed iterative and incrementally throughout several iterations. Every new increment introduces new functionality. The fidelity of solutions is increased in every new iteration. Such solutions are modular and can be modified more easily. A special advantage is easier communication among specialists of different profiles.

ACKNOWLEDGMENT

The authors wish to express their gratitude to the Serbian Ministry of Education, Science and Technological Development for supporting this paper through projects TR33026 and TR33027.

REFERENCES