IMPLEMENTATION OF DATA PROCESSING AND AUTOMATED ALGORITHM BASED FAULT DETECTION FOR SOLAR THERMAL SYSTEMS

Stefan Küthe, Corry de Keizer, Reza Shahbazfar and Klaus Vajen

Institute of Thermal Engineering, Kassel University (Germany)

1. Introduction

Solar thermal systems are designed to function for about 25 years, but faults and malfunctions are likely to occur at a certain time causing reduced solar gains and economic losses. Due to the fact that usually a backup heater is installed to guarantee the supply of the heat demand, faults and malfunctions of the solar thermal systems are often not recognized or only with a delay. Hence, fault detection is an important issue. Because a manual monitoring would be time and cost intensive, an automated detection method is needed. In this paper an automated algorithm based fault detection is presented. The data transfer and storage is done via data emails, which are written into a server based database guaranteeing the usage of different structures of data files. Data can then be accessed everywhere, so that the data analysis can be handled on different computers. This leads to the possibility to use several computers at once in order to increase the performance of the fault detection. The fault detection itself is carried out by functions (algorithms) analyzing the data for deviations from normal operating conditions. The structure of the database and the different levels of algorithms used for the fault detection as well as an example implemented on a monitored solar thermal system are presented in this paper.

2. Data processing

2.1. Transfer and storage

Good data storage and easy and fast access to the gathered data of monitored systems are important issues for long term monitoring and fault detection. Hence, a database is used to handle the large amount of resulting data in an efficient way. To ensure flexible access the database is server based. So, stored data is generally available everywhere via an internet connection. Restrictions are set by applying user rights to the database. Figure 1 shows the data administration.

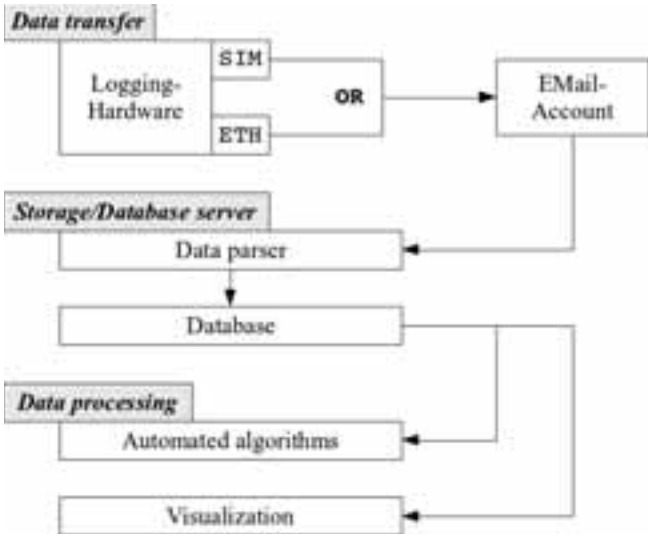

Fig. 1: Data administration. Data is transferred via email from the logger to a given email account. The Emails can either be sent via a mobile communication network or via an Ethernet connection. The data parser fetches the data files from the emails and fits them to the database structure before inserting the data into the database. The data is then accessible and can be used for visualization etc.

Data transfer from the monitored systems to the database is done via emails with attached data files. The emails are fetched and parsed by server side software in order to make the transferred data files suitable for the given structure of the database. Therefore, different data formats from varying logging hardware can be handled easily. Emails can either be sent via a mobile communication network or, if available, directly via an Ethernet connection. There is a redundant backup system: on the one hand the emails are stored for a given period and on the other hand the database is backed up daily.

2.2. Database structure

Beside the data itself there are different levels in the database structure to store additional information about the monitored systems and their measuring equipment. In the first level administrative and general information of the systems like details of contact person, system specifications, etc. are stored. In the next step details of installed sensors are given, e. g. name, min/max values and system reference. Data is saved in the last level of the structure containing timestamp, value and sensor reference. Figure 2 shows the simplified database structure.

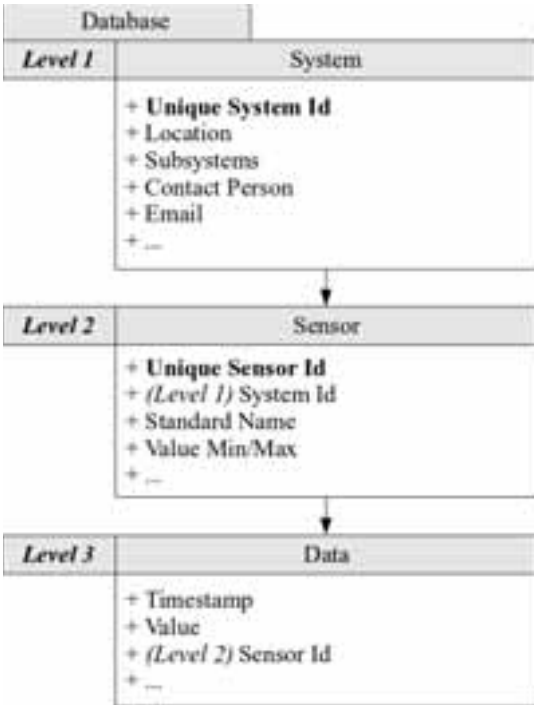

Fig. 2: Database structure. In Level 1 administrative and general information as well as the description based on subsystems is stored. Level 2 holds information about the sensors while the data itself is stored in Level 3. Levels are linked with each other via identification numbers (ids), e. g. each data line has a reference to the respective sensor id.

As one can see the layers are linked with each other via references, implemented as so called *identification numbers* (ids), e. g. in order to get all sensors of a monitored system, the sensor table is searched for all sensors with given system id, afterwards data of a sensor is extracted by filtering the data table with a chosen sensor id. Ids are always unique, whereas names can be used as often as needed.

Another important aspect regarding automated fault detection is the standardization of the sensor names in the database. The fault detection is carried out by programmed algorithms analyzing the data, whereas the algorithms itself are described in more detail in the next section. So, if there is a standardization of sensor names, it can be checked whether all sensors needed by a certain algorithm are available for a respective monitored system. In case this is true, the algorithm is applicable to the system. Because a sensor can have as many attributes as needed, alias names can be defined beside the standard names. This is an important feature when visualizing data for the user, because it is easier to understand. Also conversion functions for the sensor units are stored as additional sensor information. Table 1 shows a possible naming of sensors on

the primary side of a solar loop; standard names are used by the algorithms and aliases for visualization.

Tab. 1: Sensor attributes stored in the sensor table of the database structure.

Id	Standard Name	Alias	Function
1	SOL_T_FL_PRI	T_Flow_Solar_Loop_Primary_Side	None
2	SOL_T_RT_PRI	T_Return_Solar_Loop_Primary_Side	None
3	SOL_V_PRI	Flow_Rate_Solar_Loop_Primary_Side	Value*0.06

The system description of the monitored systems is specified as a connection of defined subsystems based on a subsystem approach developed in Dröscher et al. (2009). Algorithms for fault detection can be linked with the subsystems and therefore, applicable algorithms for a monitored system can be filtered by querying the available subsystems. In Figure 3 two examples of subsystems of solar loops are presented.

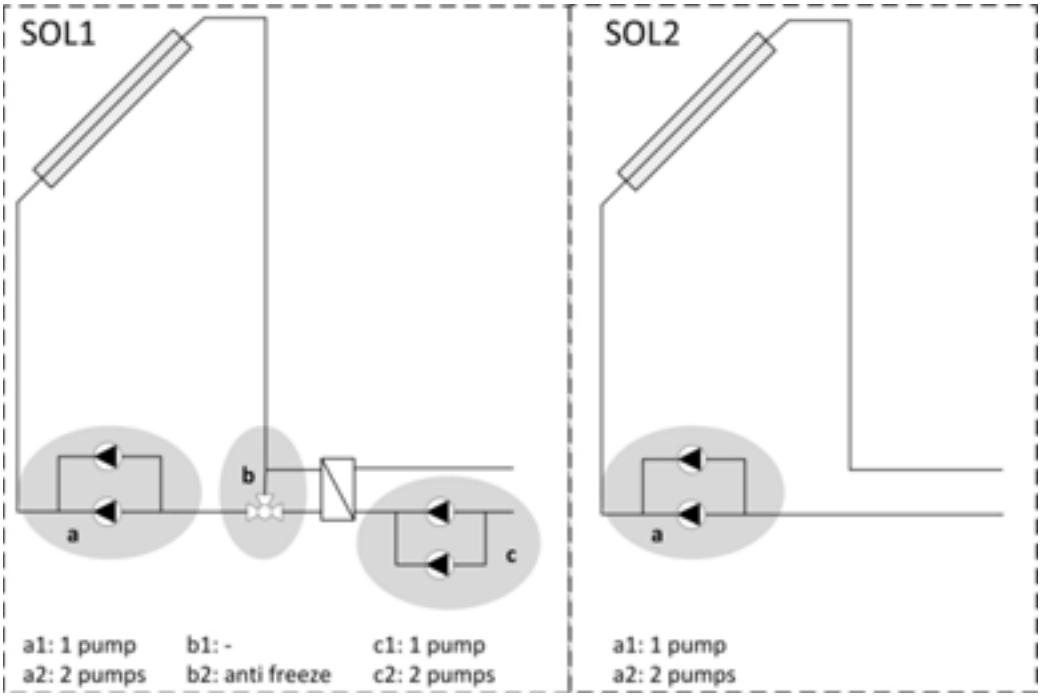

Fig. 3: Subsystems of solar loops with details. Every monitored system is described clearly by the connection of several subsystems. SOL1 and SOL2 are different variants of a solar loop.

2.3. Access and visualization

Data is accessible via programmed software interfaces and can be extracted to files for further use with other programs or handled directly. The only requirement for the used programming language is an available connector to the database server. Therefore, data can also be visualized easily, e. g. in a web browser implementing a PHP/HTML-interface with the advantage that no additional software needs to be installed by external users in order to view and analyze plots. Such an interface was implemented at Kassel University but will not be discussed in this paper.

3. Algorithms

Because of data quantity and the fact that an analysis by hand would be cost intensive and time consuming, an automated method for fault detection is unavoidable. This is done by implemented *software objects*, afterwards called algorithms. The algorithms run in defined time intervals fetching needed data from the

database in order to check the correct function of the respective solar thermal system. They can run on the server where the database is located as well as on other computers or rather servers. Hence, if needed for a better performance, it is possible to have several servers doing the data analysis. Thereby, the algorithms vary from simply calculating key figures and comparing them to references to a complex linking of several algorithms with each other. Another task is to examine the data regarding quality and missing time series. Nevertheless the structure for each algorithm is always the same: an algorithm itself is a so called *class* with *n attributes* and main *functions* like *run*, *calculate* and *get_result*. So, new algorithms can be integrated in the fault detection in an easy and flexible way by more or less simply overwriting or rather implementing the *calculate function*. Figure 4 shows the structure.

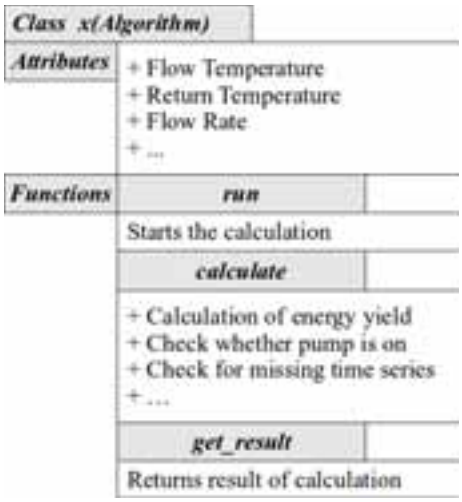

Fig. 4: Algorithm structure. An algorithm is started by calling the *run function*. When calculation is finished the result is returned by the *get_result function*. In the calculation step also other algorithms can be called in order to use their results. In this way algorithms can be linked with each other. The attributes correspond to the sensors needed for the calculation.

There are several categories or levels of algorithms. While in the first level only data quality and quantity is checked, the last level deals with the final detection of faults. So, the classification is described below and based on approaches in (Isermann, 1994, 2006), whereas example algorithms are presented in a simplified syntax showing the calculation step or rather the *calculation function*.

Level 1: sensor value/measured value

Algorithms in level 1 examine the quality and quantity of the data transferred by the sensor. It is checked whether the values of the sensors are in their measuring range. Therefore, broken sensors can be found. Quantity checks mean that missing time series are identified and handled. If too many data is missing in a certain time interval, it is not used for the fault detection. Another task can be, to get a not directly logged/measured value by calculating it with the help of other sensor values. The following example checks the measuring range and returns *false*, if the value is out of range; the range is stored in the database as sensor attributes *min* and *max*:

```
CLASS *VALUE_RANGE*:
      IF value < sensor.get_attribute("min")
      OR value > sensor.get_attribute("max"):
            RETURN false
```

Level 2: measured value/characteristic

In this step the measured values from level 1 are used to generate system characteristics. Characteristics can be key figures like energy yields, number of charging-cycles of the tank or temperature differences. As a result the algorithms only return states of the system, but no faults yet. In the following example the temperature difference between flow and return in a primary solar loop is returned:

```
CLASS *CHAR_DELTA_T_SOL_PRI*:
      CHARACTERISTIC = SOL_T_FL_PRI - SOL_T_RT_PRI
      RETURN CHARACTERISTIC
```

Level 3: characteristic/symptom

In this step extracted system characteristics are compared to reference values. Thereby, deviations from normal conditions are identified and handled as so called symptoms. Often multiple characteristics are linked with each other for the plausibility checks in level 3. In the following simple example the temperature difference between flow and return temperature in the solar loop is compared to a given reference variable:

```
CLASS *SYMP_DELTA_T_SOL_PRI*:
      IF *CHAR_DELTA_T_SOL_PRI* > REF_DELTA_T_SOL_PRI_MAX
      OR *CHAR_DELTA_T_SOL_PRI* < REF_DELTA_T_SOL_PRI_MIN:
            RETURN false
```

As one can see the algorithm is linked with the example from level 2 in order to get the temperature difference as characteristic.

Level 4: symptom/fault

If a plausibility check returns *false* and therefore, a symptom is generated, in level 4, a fault diagnosis is carried out in order to determine a possible malfunction or fault of the system. In the simplest way, a certain symptom matches exactly one unique fault. But usually this is not the case, so that multiple symptoms must be taken into account and maybe also be linked with already extracted characteristics of the system.

Figure 5 shows the different levels of algorithms.

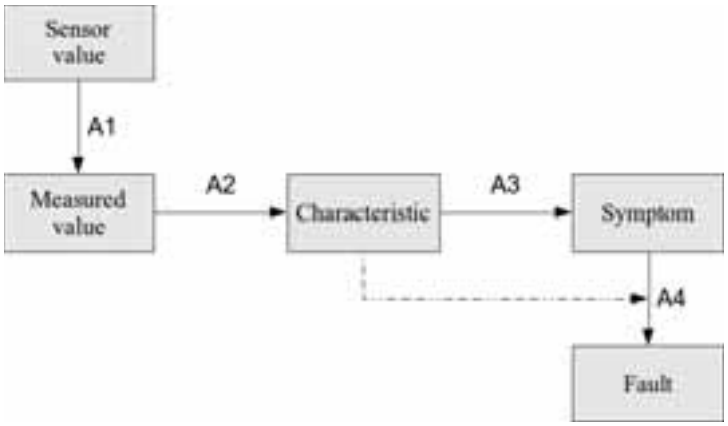

Fig. 5: Levels of fault detection. A1 to A4 correspond to the 4 levels of algorithms described above.

4. Case Study/Example

In this section a simple fault detection is shown for a solar thermal system with space heating and domestic hot water support. Figure 6 reveals a data gap of three days between January 2 and January 5, which corresponds with the email generated by a level-1-algorithm checking data quantity:

```
*Missing Time Series, Location 1*
guessed time step: 60 seconds

from_date, to_date, missing_time_steps
----------
2011-01-01 12:00:00, 2011-01-01 12:04:00, 3
2011-01-02 12:03:00, 2011-01-05 00:02:00, 3598
2011-01-05 22:48:00, 2011-01-05 22:52:00, 3
```

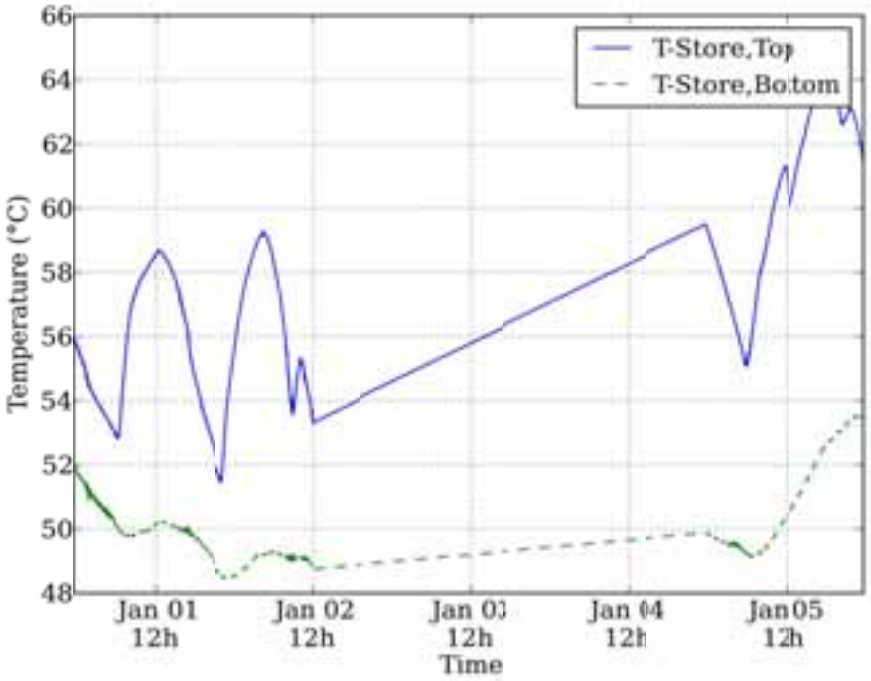

Fig. 6: Temperatures in the store. The linear line shows a data gap between January 2 and January 5.

Figure 7 shows an identified symptom (level 3) regarding too low temperatures in the upper part of the store; this was detected by an algorithm doing a reference check:

```
CLASS *SYMP_T_STORE_TOP*:
      IF *CHAR_T_STORE_TOP* < REF_T_AUX_SET - REF_DEAD_BAND:
            RETURN false
```

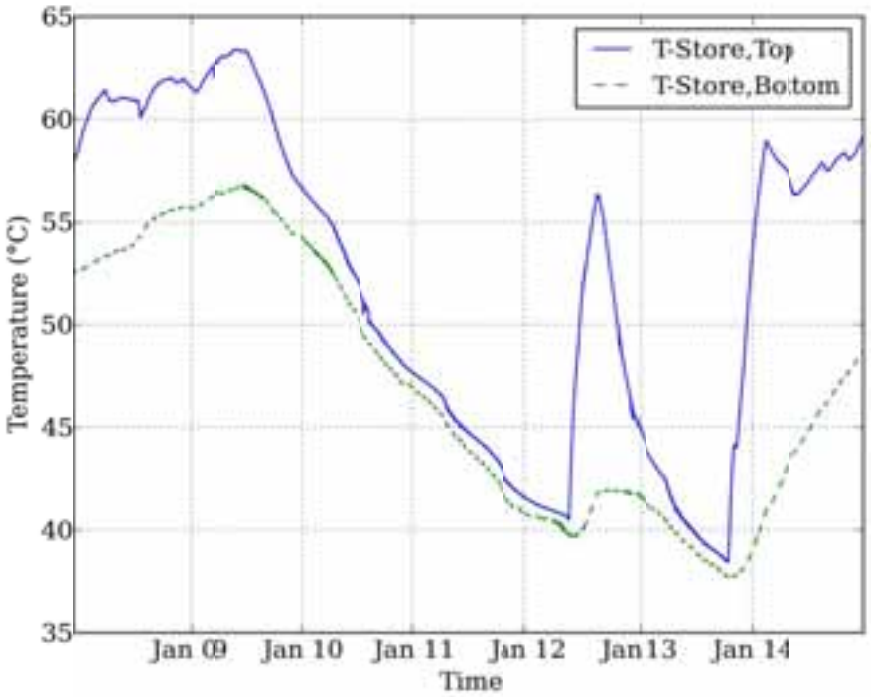

Fig. 7: Temperatures in the store. From January 10 to January 14 the temperatures in the upper part are too low; the set point temperature is not reached.

In the next step a fault diagnosis is carried out. Therefore, a characteristic algorithm is called to check whether the auxiliary heater is running correctly in order to supply the needed temperature. As can be seen in Figure 8, this is not the case: although there is a hot water demand and the supply temperature is not reached, neither the solar loop nor the auxiliary heater supply the set point temperature in the upper part of the store.

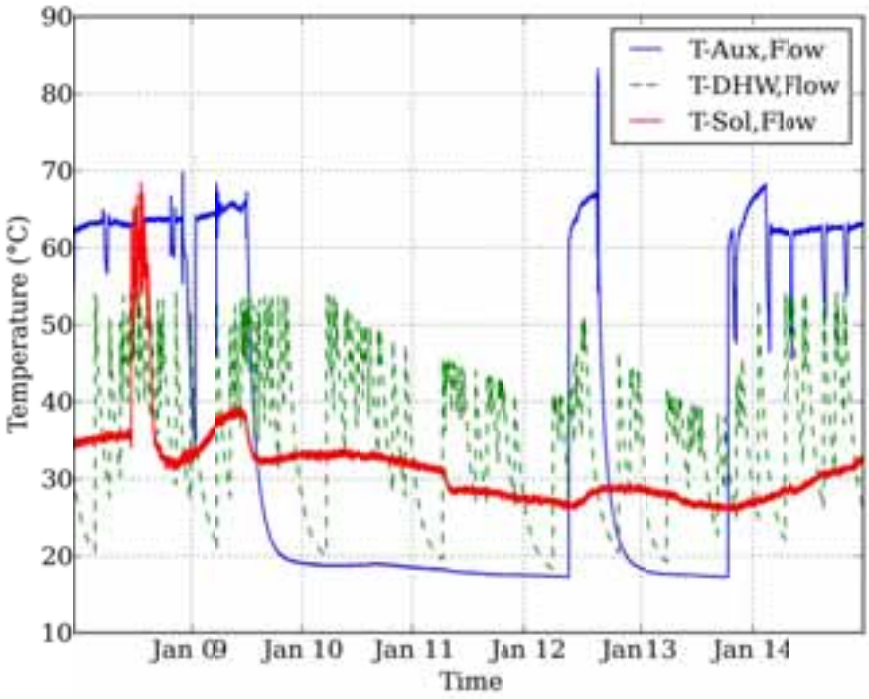

Fig. 8: Although the needed flow temperature for domestic hot water is not reached, neither the solar loop nor the auxiliary heater supply the set point temperature in the upper part of the store.

Depending on the settings of the fault detection algorithm further characteristics are taken into account to get more information about the malfunction, or a message is sent to a responsible person in order to check, if e.g. the auxiliary heater was switched off by accident or something is wrong with the controller. In this example it is important that the malfunction is detected as early as possible, because otherwise there will be a negative effect in the comfort for the user. In this case it turned out that the heater was switched off manually by the user for testing purposes.

5. Summary

In this paper a fault detection method based on algorithms for solar thermal systems was presented. As a basis, data storage of monitored systems is implemented as a server based database, which holds beside the data itself all information needed by the fault detection method. The data is transferred via emails with attached data files, whereas the structure of the data files can have any format, because data is parsed by software before it is written into the database. The algorithms for the fault detections are classified into four levels: in the first level only the quality and availability of data is checked, e.g. missing time series, transferred values of the sensors are not in the measuring range, however, in level 4 the final diagnosis is carried out by examining identified so called symptoms. A symptom is detected in case a characteristic like energy yield or a temperature difference does not match a reference value or interval. Additionally the data can be visualized in a simple way to analyze a found fault more in detail, e. g. via a web browser. In a further step stored data can also be used to compare simulated and measured energy yields in order to detect faults as done in (Keizer, 2010).

6. Acknowledgements

This project (FKZ-0325975A) is supported by the „Federal Ministry for the Environment, Nature Conservation and Nuclear Safety“.

7. References

Keizer, A.C. de; Kuethe, S; Vajen, K; Jordan, U; Ohnewein, P., 2010: Simulation based fault detection for large solar thermal systems. In Proceedings of Eurosun. Eurosun. Graz, Austria.

Dröscher, A., Ohnewein, P., Haller, M., and Heimrath, R., 2009. Modular specification of large-scale solar thermal systems for the implementation of an intelligent monitoring system. In ISES Solar World Congress, Johannesburg, South Africa.

Isermann, R., 2006. Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance. Springer, Berlin Heidelberg New York.

Isermann, R., 1994. Überwachung und Fehlerdiagnose. Moderne Methoden und ihre Anwendung bei technischen Systemen. VDI Verlag, Düsseldorf.