Brandenburgische
Technische Universität
Cottbus - Senftenberg

A Knowledge Discovery Cycle for Monitoring Mobile Cyber-Physical Systems

Von der Fakultät für Mathematik, Naturwissenschaften und Informatik
der Brandenburgischen Technischen Universität
Cottbus – Senftenberg

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation

vorgelegt von

Master of Science

Tino Noack

geboren am 01.07.1981 in Königs Wusterhausen

Gutachter: Prof. Dr.-Ing. I. Schmitt, BTU Cottbus – Senftenberg
Gutachter: Prof. Dr.-Ing. H. T. Vierhaus, BTU Cottbus – Senftenberg
Gutachter: Prof. Dr.-Ing. A. Nürnberger, OVGU Magdeburg

Tag der mündlichen Prüfung: 03.02.2015

I dedicate this to my lovely little family.

Abstract

Mobile cyber-physical systems (MCPSs) such as motor vehicles, railed vehicles, aircraft, or spacecraft are commonly used in our life today. These systems are location-independent and embedded in a physical environment which is usually harsh and uncertain. MCPSs are equipped with a wide range of sensors that continuously produce sensor data streams. It is mandatory to process these data streams in an appropriate manner in order to satisfy different monitoring objectives, and it is anticipated that the complexity of MCPSs will continue to increase in the future. For instance, this includes the system description and the amount of data that must be processed. Accordingly, it is necessary to monitor these systems in order to provide reliability and to avoid critical damage. Monitoring is usually a semi-automatic process while human experts are responsible for consequent decisions. Thus, appropriate monitoring approaches are required to both provide a reasonably precise monitoring process and to reduce the complexity of the monitoring process itself.

The contribution of the present thesis is threefold. First, a knowledge discovery cycle (KDC) has been developed, which aims to combine the research areas of knowledge discovery in databases and knowledge discovery from data streams to monitor MCPSs. The KDC is a cyclic process chain comprising an online subcycle and an offline subcycle. Second, a new data stream anomaly detection algorithm has been developed. Since it is necessary to identify a large number of system states automatically during operation, data stream anomaly detection becomes a key task for monitoring MCPSs. Third, the KDC and the anomaly detection algorithm have been prototypically implemented and a case study has been performed in a real world scenario relating to the ISS Colombus module.

Kurzfassung

Mobile cyber-physikalische Systeme (MCPS) wie z.B. Kraft-, Schienen-, Luft- oder Raumfahrzeuge sind in der heutigen Zeit sehr weit verbreitet. Diese Systeme sind ortsunabhängig und in eine raue bzw. unsichere physikalische Umwelt eingebettet. MCPS sind mit einer Vielzahl von Sensoren ausgestattet, die kontinuierlich Sensordatenströme erzeugen. Um den unterschiedlichen Zielsetzungen der Überwachung gerecht zu werden, ist eine angemessene Verarbeitung dieser Sensordaten wichtig. Es zeichnet sich ab, dass die Komplexität von MCPS in der Zukunft weiter steigt. Damit sind die Systembeschreibung sowie die Menge der zu verarbeitenden Daten gemeint. Aus diesem Grund und um die Zuverlässigkeit der Systeme zu gewährleisten sowie kritische Systemausfälle zu vermeiden, ist eine Überwachung zwingend notwendig. Der Überwachungsprozess ist üblicherweise semi-automatisch und menschliche Experten sind verantwortlich für die getroffenen Entscheidungen. Daher werden Überwachungsansätze benötigt, die eine angemessen präzise Überwachung ermöglichen und gleichzeitig die Komplexität des Überwachungsprozesses verringern.

Der Beitrag der vorliegenden Arbeit ist dreigeteilt. Zunächst wird ein Knowledge Discovery Cycle (KDC) entwickelt, der die Forschungsgebiete Wissensentdeckung in Datenbanken und Wissensentdeckung aus Datenströmen miteinander kombiniert. Weiterhin wird ein neuer Algorithmus für die Anomalieerkennung in Datenströmen vorgestellt. Für die Überwachung von MCPS ist eine Unterscheidung in verschiedene Systemzustände notwendig. Daher wird die Anomalieerkennung in Datenströmen zu einer Schlüsselaufgabe. Zuletzt werden der KDC und der vorgestellte Algorithmus zur Anomalieerkennung prototypisch implementiert und eine Fallstudie durchgeführt. Diese Fallstudie basiert auf einem realen Szenario, welches sich auf das Columbus-Modul der internationalen Raumstation ISS bezieht.

Contents

CHAPTER 1

Introduction

"If we knew what it was we were doing, it would not be called research, would it?"

Albert Einstein (1879-1955)

A large number of today's products contain *embedded systems*. Many of these products are safety-critical and rely on real-time requirements. A *system* is a compound structure consisting of objects which cohere due to interaction and interdependency [Bac00]. An embedded system is a *mechatronic system* which is embedded into a product, processes information, and interacts with the product and the *system environment* by *sensors* and *actuators* [Wol02]. The system environment describes the immediate surroundings of a system [Bos89]. Sensors collect information about the system environment and actuators influence the system environment [Mar07]. A mechatronic system is a system which comprises mechanic, electronic, and software domains [BGJ+09]. A *real-time system* works under timing constraints, while the correctness of the computational results is time-dependent [Kop11]. A *safety-critical system* is a system which is subject to the consequences of failure [Kni02].

Basic approaches of contemporary *cyber-physical systems (CPSs)* [Lee08] are widely disseminated in application domains such as manufacturing, home entertainment, healthcare, transportation, or aerospace. As stated by Lee et al. [LS11], the acronym 'CPS' is inspired by the term *cybernetics*. The term 'cybernetics' was coined over a half century ago and describes the conjunction of physical processes, computation, and communication. The acronym 'CPS' redefines the term 'cybernetics' and is intended to bring 'cybernetics' into a context of today's technologies. Amongst others, these include digital computing, software intensive computation, and massive use of networks.

1.1 Mobile Cyber-physical Systems (MCPSs)

It is possible to distinguish between *stationary CPSs* and *mobile CPSs* [NS13]. Stationary CPSs such as power grids, manufacturing plants, or research facilities are strongly tied to a specific location. Whereas *mobile cyber-physical systems (MCPSs)* [GD06], such as motor vehicles, railed vehicles, aircraft, or spacecraft, are location-independent and embedded into a *physical environment*. A physical environment is a system environment which is subject to physical conditions. The present thesis focuses on MCPSs. Figure 1 depicts a sketch of an MCPS as it is used in this thesis. Sensors and actuators are used for interaction between an MCPS and the embedding physical environment. MCPSs consist of hardware such as electronic assemblies (e.g. processors, memory, and batteries). These electronic assemblies are components of embedded systems. Moreover, software is an integral and substantial part of embedded systems. These embedded systems are usually connected via an internal network. Embedded systems and the internal network are commonly used for *monitoring* and controlling physical processes. An external network connects an MCPS with *external information systems*. Embedded systems and electronic assemblies are subject to resource restrictions such as processor capacity, memory, and power consumption. In general, external information systems are stationary parts of MCPSs and are used to compensate resource restrictions. Accordingly, MCPSs are very complex systems [NS13] (see also [Noa11a, Noa11b, NS12a]).

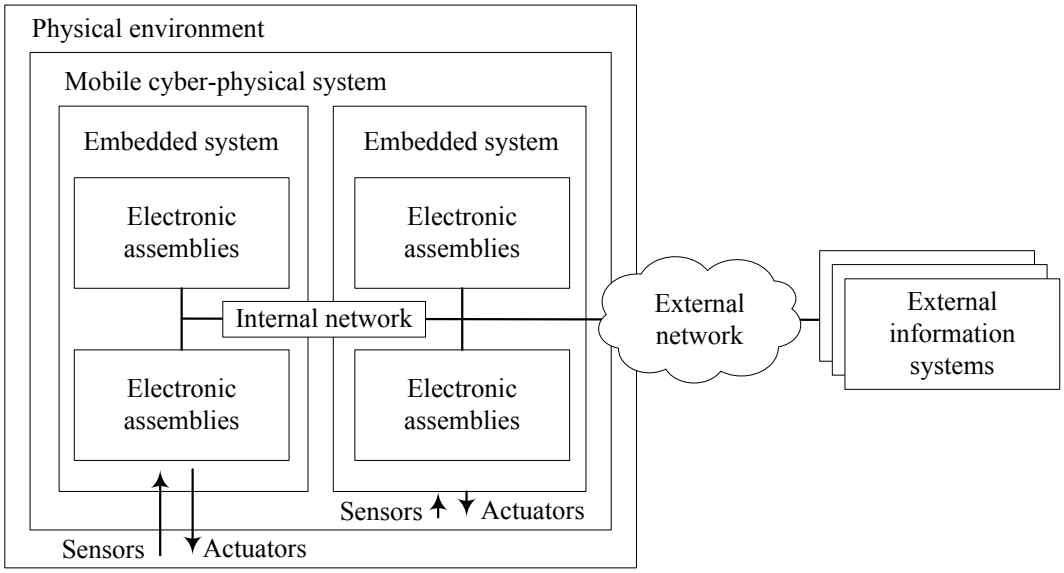

Figure 1: A sketch of an MCPS (based on [NS13])

1.2 Life-time Phases of MCPSs

The life-time of MCPSs can be separated into four phases: *design*, *test*, *dispatch and commissioning*, and *operation*. The 'design phase' relates to the conceptional engineering. The 'test phase' relates to the implementation of the conceptional design and the subsequent tests. The 'dispatch and commissioning phase' relates to the shipment and activation to the application environment. Finally, the 'operational phase' relates to the runtime of an MCPS until its deactivation [NNS+13].

A general *bathtub curve* separates the operational phase of MCPSs into three more phases. As depicted in Figure 2, these include: *wear in*, *normal wear*, and *wearout*. The 'wear in phase' follows after the dispatch and commissioning phase. It is characterized by a relatively high failure rate and infant mortality. This typically includes design and manufacturing defects, assembly mistakes, or installation and commissioning errors. The 'normal wear phase' is a relatively long period where the failure rate is proportionally low. The 'wearout phase' is characterized by a gradually increasing failure rate at the end of the expected life-time due to metal fatigue, wear on mechanisms between moving parts, corrosion, and obsolescence. However, the shape of a specific bathtub curve is machine-dependent [Sil05] (see also [KKW03]).

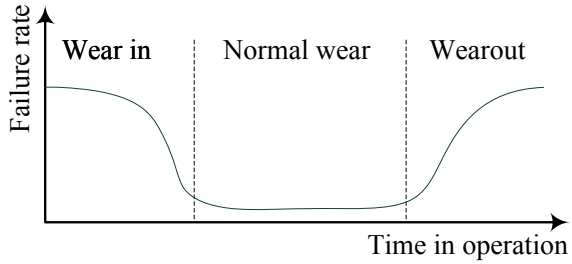

Figure 2: General bathtub curve (based on [Sil05])

The application of MCPSs demands a high degree of *reliability*. The term 'reliability' describes the continuity of correct service and probability of default [BM00]. As described by the above-mentioned bathtub curve, current *system components* as subparts of a system can become increasingly unreliable during operation due to wear and tear or aging effects. Additionally, design and manufacturing errors, which may occur during the design and test phases, can compromise the reliability. Accordingly, monitoring MCPSs is essential and necessary to ensure reliability and to avoid critical damage. After the dispatch and commissioning phase, an appropriate monitoring approach must provide initial reliability and be flexible and dynamically adaptable during operation (wear in, normal wear, and wearout phases).

1.3 Key Challenges for Monitoring MCPSs

Monitoring MCPSs is very challenging. MCPSs are under constant remote monitoring and control to keep these systems reliable, healthy, and stable. MCPSs are being monitored by telemetry and commands. To fulfill these monitoring tasks, such systems are equipped with sensors that produce continuous sensor *data streams*. Generally, data streams consist of data items and are produced in real-time with a potentially infinite length [BW01]. These data streams are commonly examined for the purpose of online monitoring and control, and to a lesser degree for offline monitoring. Online and offline monitoring are prerequisites for recognizing specific event occurrences, *anomaly detection*, and early detection of sensor degradation. *Anomalies* are patterns in data which deviate from normal behavior [CBK09]. An *event* is an incident that has occurred within a particular system or domain [EN10]. Monitoring and control tasks are very time consuming and resource intensive, and for remote systems any equipment failure must be avoided [NSS13b]. The key challenges for monitoring MCPSs are identified as: *change*, *time dependence*, *continuity*, *data processing*, and *autonomy* [NS12a, NS13].

1. **Change**: Changes in the system behavior of MCPSs are specific event occurrences which must be monitored and detected. Thus, variation in system behavior is one of the main challenges for monitoring MCPSs. The physical environment, in which MCPSs are embedded, is usually harsh and uncertain. Conditions of the physical environment such as heat or humidity influence the system behavior continuously. Hence, the system behavior changes during operation and may be significantly different from the behavior which has been observed during the design and test phases. For example, outer space is a physical environment which differs from terrestrial conditions such as zero gravity, no atmospheric pressure, or the absence of the earth's magnetic field. Moreover, chemical elements inside of spacecraft behave differently in contrast to terrestrial conditions. For instance, water or dust collection is very difficult under gravity-free conditions. Such variant behavior cannot always be sufficiently analyzed during the design and test phases under terrestrial conditions. Consequently, it is necessary to adapt the applied monitoring process to the emerging conditions continuously during operation.

   As depicted in Figure 3, a distinction is usually made between *sudden change* and *gradual change*. 'Sudden change' is also known as *concept shift*, whereas 'gradual change' is also known as *concept drift* [BHKP11]. For the sake of clarity, the terms 'sudden change' and 'gradual change' are used hereinafter. Sudden change such as a crash (e.g. a collision with space debris) cannot be excluded and must be immediately followed by an appropriate action. Conditions of the physical environment can also induce gradual change, such as wear

and tear. Gradual change can in turn cause sudden change. Hence, monitoring gradual change is necessary to forecast sudden change which relates to gradual change.

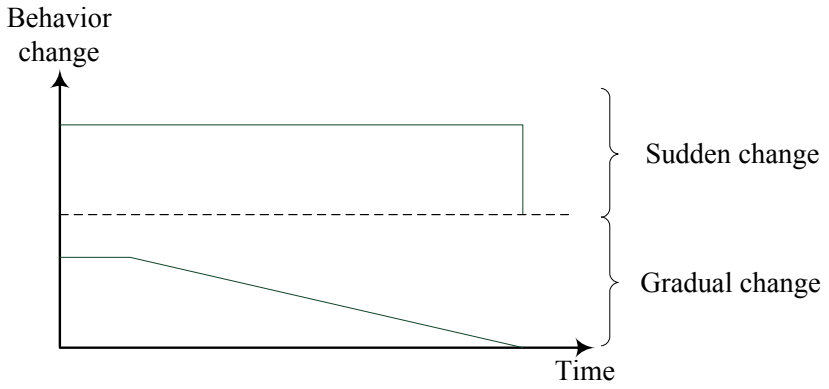

Figure 3: Distinction between gradual and sudden change

2. **Time dependence**: Depending on the resulting effects, the appearance of gradual and sudden change need to be detected within a previously defined time frame. Subsequently, the monitoring process is subject to real-time requirements. The time dependence is a key challenge due to the prevailing resource restrictions of MCPSs.

3. **Continuity**: Continuity is a key challenge and refers to the process of knowledge discovery. It is necessary to discover knowledge during the test, dispatch and commissioning, and operational phases of MCPSs. Such discovered knowledge must be continuously brought into connection with expert knowledge which has been carefully deliberated upon during the design phase. This is required since the monitoring process presupposes knowledge about the *target system*, the belonging physical environment, and the intended or related system behavior. A target system is an MCPS which have to be monitored.

4. **Data processing**: As aforementioned, the conditions of the physical environment and the system behavior are changing over time. Thus, it is also important to discover knowledge continuously during operation and to associate new derived knowledge with historical information obtained during design, test, or dispatch and commissioning phases. It is necessary to process streaming data and historical data to provide continuous monitoring of MCPSs. Therefore, appropriate mechanisms are required which consider processing of *persistent* and *transient* data. 'Data persistence' describes the ability for data to outlive

the operational phase and possibly the overall life-time of an MCPS [CB10]. 'Data transience' constitutes the opposite.

5. **Autonomy**: Autonomy relates to the online monitoring process and the external network connection. The monitoring process is commonly a semi-automatic process with the support of *human experts*. Johnson describes the term 'expert' as follows: "An expert is a person who, because of training and experience, is able to do things the rest of us cannot; experts are not only proficient but also smooth and efficient in the actions they take." [Joh83, p. 78]. As stated by Huebscher et al.: "Autonomic computing seeks to improve computing systems with a similar aim of decreasing human involvement." [HM08, p. 1]. Processing data streams in real-time and interaction with human experts are mutually exclusive. Moreover, the external network connection is usually unreliable and can be interrupted from time to time. Hence, the online monitoring process must be able to act autonomously in order to benefit from knowledge which is discovered during the life time phases of MCPSs and to detect sudden changes in real-time.

1.4 Real World Scenario

This section presents a real world scenario which relates to the *International Space Station (ISS)* Columbus module and is structured as follows. First, a particular view on the current implementation of the *ISS Columbus failure management system* [NBW$^+$10] is given. Second, the *ISS Columbus air loop* [NBW$^+$10] is explained. Third, a failure event is presented which has occurred on the ISS Columbus module during operation [NBW$^+$10]. This failure event emphasizes the relation of gradual and sudden change. This section is related to the author's publication [NS13].

1.4.1 ISS Columbus Failure Management System

As depicted in Figure 4, the ISS Columbus failure management system is distributed over various instances. Each instance is responsible for specific functions, capabilities, and constraints, with a distinction between *on-board* and *ground* instances. The ISS Columbus module is an MCPS and contains on-board instances. On-board instances are embedded systems which are connected via an internal network. On-board instances work automatically and in real-time. However, on-board instances suffer from resource restrictions. Ground instances are stationary parts of the ISS Columbus module and represent external information systems. They are applied semi-automatically and without real-time constraints. Furthermore, ground instances are used for long-term analysis and provide support for human experts. Nevertheless, they suffer from bandwidth limitations and the availability of the external network connection (see also [NBW$^+$10, NNP$^+$11, NS12b, NLS$^+$12]).

1. **ISS Columbus**: The ISS Columbus module is embedded into outer space and employs an air loop as a life-support system. The on-board failure management system is responsible for crew health and mission success. The system works automatically and is applied for monitoring the ISS Columbus air loop. Aboard the ISS Columbus module, automatic detection of time critical failures is a vital necessity. This process encounters difficulties, since the existing on-board instances are subject to resource restrictions. For example, the ISS Columbus module consists of a variety of sensors, and the on-board failure management acquires approximately 3000 analogue and digital measurements per second. Due to resource restrictions, only a small proportion (about 230) of these measurements can be adequately monitored on-board and in real-time. The acquired measurements are sent to the ground control when the external network is available. However, they cannot be sent directly in case of a *loss of signal (LOS)*. Consequently, acquired measurements have to be stored temporally on-board.

2. **Ground control**: The ground control is a ground instance and works semi-automatically. Human experts are responsible for manual failure detection and recovery. As with the on-board failure management system, the ground control is responsible for crew health and mission success. The ground control is also used to plan short-term corrective actions. Not all on-board measurements are available at the ground control because of the external network and its bandwidth limitations. Delays can occur from time to time due to LOS. Accordingly, on-board acquired measurements do not arrive in real-time at the ground control.

3. **Engineering support center**: The engineering support center is another ground instance which is used for root cause failure analysis and to plan long-term corrective actions. Human experts work offline and use data analysis tools intensively. Furthermore, data and information available from persistent storage (the mission archive) and the ground control are used.

4. **Assembly, integration, and test facility**: The assembly, integration, and test facility is amongst others used for engineering tests, troubleshooting, and validation tests. Troubleshooting of on-board issues and validation of updated or new functions and services, such as experiment facilities along with failure detection methods, are very important. The assembly, integration, and test facility uses data and information which are available from persistent storage (the mission archive) and the ground control. Moreover, this facility is used to adapt and reconfigure on-board instances.

5. **Mission archive**: The mission archive is a persistent storage which contains historical data and information which were obtained during all life-time phases

of the ISS Columbus module. Data transferred from the ISS Columbus module to the ground control is stored entirely in the mission archive. Furthermore, the mission archive contains annotations, expert knowledge, and additional information.

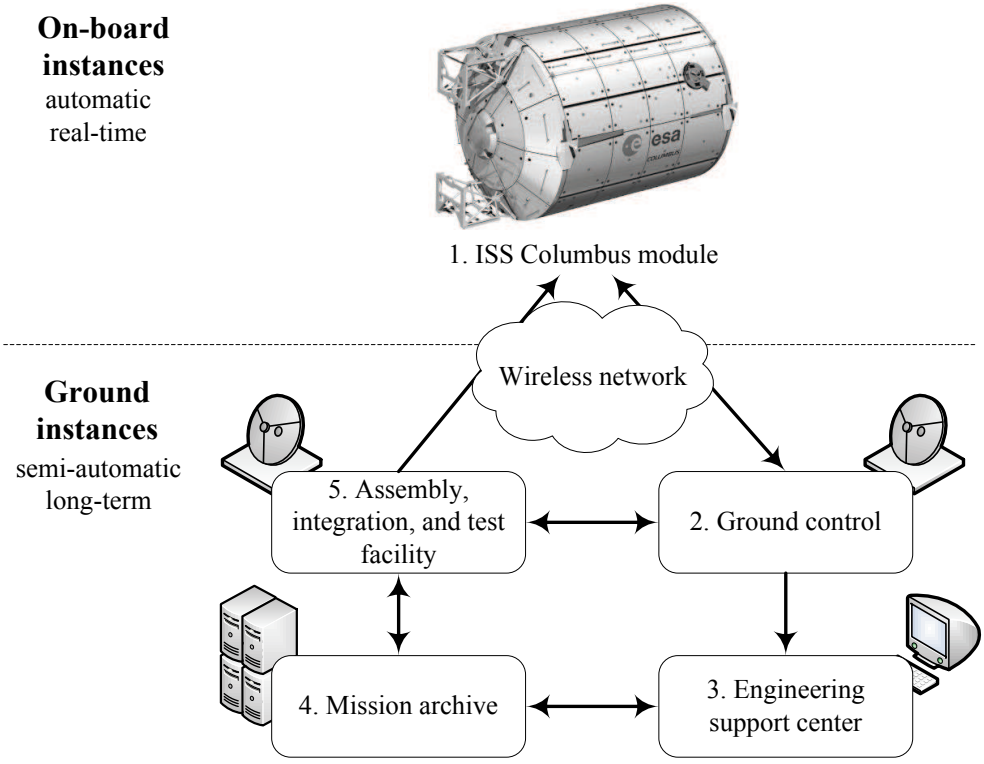

Figure 4: ISS Columbus failure management system (based on [NBW+10, NS13])

1.4.2 ISS Columbus Air Loop

The ISS Columbus air loop is an embedded system and part of the life-support system of the ISS Columbus module. The air loop is monitored by the ISS Columbus failure management system. As depicted in Figure 5, the air loop contains an *inter module ventilation supply fan assembly (ISFA)* that is used to aspirate air from the ISS. The ISS Columbus air loop also consists of two redundant *cabin fans (CFAs)* which are used for air circulation inside of the ISS Columbus module. These cabin fans work redundantly to provide a failover if one of them breaks down. The air pressures which are produced by the ISFA and a single cabin fan are joined after-

wards. The resulting air flow passes a *condensate heat exchanger (CHX)* that is used for temperature regulation and dehumidification. Afterwards, the air flow passes a diffuser and is blown into the ISS Columbus module. The forced air flow leaving the diffuser prevents dead air pockets and enables fire and smoke detection of the ISS Columbus center aisle. Furthermore, the ISS Columbus air loop contains an *inter module ventilation return fan assembly (IRFA)*. The IRFA aspirates air from the ISS Columbus module while the air passes a return grid and smoke detectors. The aspirated air is then forwarded to the ISS where air revitalisation takes place. The fan assemblies are responsible for air exchange between the ISS Columbus module and the ISS (see also [NBW+10, KPD10, PSRD11]).

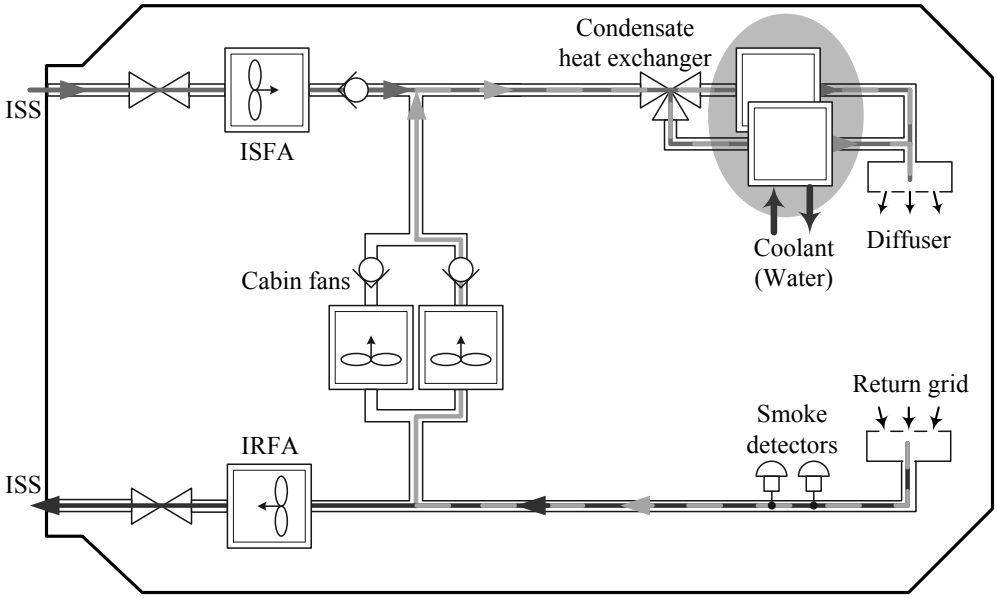

Figure 5: ISS Columbus air loop (based on [NBW+10])

1.4.3 Failure Event

The ISS Columbus air loop is intended to be at service 24/7. Due to continuous operation, aging effects, such as wear and tear, occur and cause a wide range of problems. As described by Noack et al. [NBW+10] and Parzianello et al. [PSRD11], bearing wearout has induced a functional loss of the IRFA. The occurred wearout effects refer to gradual change (cf. Section 1.3) of the system behavior.

The IRFA is monitored by sensors. For the sake of simplicity, only three sensor *attributes* are selected initially. An attribute is at least one measurable property or

a combination of several measurable properties of a system or a system component [Cha09]. An *attribute value* is a specific measurement which relates to an attribute. These include *speed*, *current*, and *pressure*. Figure 6 depicts the collected measurements. The speed relates to the rotating speed of the fan assembly, and the unit of measurement is $1/min$. The current relates to the electrical input current of the IRFA and is equivalent to the produced air flow and the mechanical friction losses. The unit of measurement is *ampere*. The pressure relates to the pressure head that is generated by the IRFA, and the unit of measurement is *kilopascal*.

The uppermost diagram (i) of Figure 6 shows gradual changes which relate to the wearout effects. The undermost diagram (ii) shows sudden changes of the system behavior and the consequent failure event. The gradual change of the system behavior (bearing wearout) led to a continuously increasing input current from the beginning of February to the beginning of April. However, the pressure and the speed were uninfluenced (i). The consequent failure event, which lasted 210 seconds, occurred on day 106 in 2008 (ii). The failure event led to an erratic speed of the IRFA and consequently to erratic air flow. Currently, there are two existing implementations for automatic failure detection and deactivation of the IRFA. Though, neither of these implementations covered the unknown failure signature. For that reason, the failure event was manually detected and manually recovered by the flight control team instead of automatic detection. In the worse case scenario, such failure situations could remain undetected for long periods of time.

1.5 Characteristics of MCPSs

It is possible to identify four main characteristics of MCPSs. This identification is based on the aforementioned challenges for monitoring MCPSs (cf. Section 1.3). The previously stated real world scenario (cf. Section 1.4) is used as an example. These characteristics include: *physical environment*, *restricted computing resources*, *real-time constraints*, and *external network*. This section is closely related to the author's publication [NS13].

1. **Physical environment**: The physical environment of MCPSs is usually harsh and uncertain. For instance, various climate zones on earth where the temperature or humidity varies greatly. Moreover, conditions of a physical environment can change within short periods of time. For example, the physical environment of outer space differs considerably from terrestrial conditions. These variable conditions can induce sudden or gradual change (cf. Section 1.3).

2. **Restricted computing resources**: MCPSs are usually subject to resource restrictions. External information systems are commonly used to compensate these restrictions. However, external information systems are not always

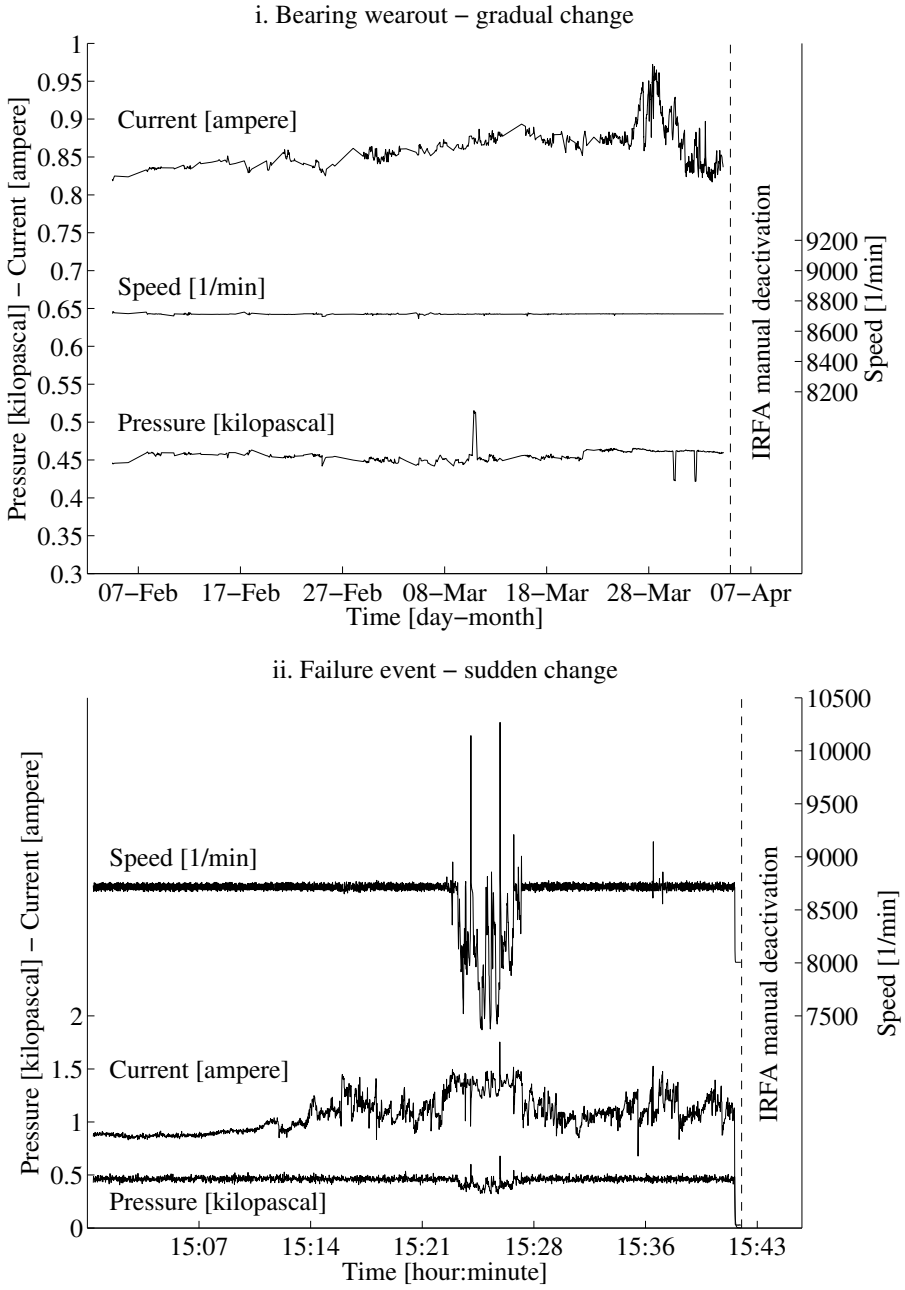

Figure 6: Gradual and sudden change (based on [NBW+10, Noa11b, NS13])

available due to an unreliable network connection and LOS. For example, the existing computational resources of the ISS Columbus module, such as processor capacity and memory or power consumption, are limited (cf. Section 1.4.1). Accordingly, only a small proportion of the acquired measurements can currently be adequately monitored on-board and in real-time. Hence, computing resources have to be used as efficiently as possible.

3. **Real-time constraints**: MCPSs are prevailingly subject to real-time constraints. Sudden changes and unforeseeable situations, such as the aforementioned bearing wearout (cf. Section 1.4.3), can occur at any time during operation. Considering such uncertainty, the computational processing and the resulting actions must also be performed in real-time. Such failure situations have to be detected reliably and must be immediately followed by an appropriate action to ensure reliability and to avoid critical damage.

4. **External network**: The external network connection is uncertain and subject to bandwidth limitations. One reason for this is the mobility of MCPSs. The external network of MCPSs is a temporal connection and not always available. Thus, appropriate mechanisms for providing a quality of service are required. However, providing a high quality of service is quite difficult. For example, the downlink of the ISS Columbus module (cf. Section 1.4.1) suffers from bandwidth limitations and is continuously interrupted due to LOS.

1.6 Scope

Monitoring MCPSs is the principal aim of this thesis. The author of the present thesis therefore proposes to combine the research areas of *knowledge discovery in databases (KDD)* [FPSM92] and *knowledge discovery from data streams (KDDS)* [GKA06] in order to attain this aim. KDD is a well studied and widely acknowledged research area. Fayyad et al. described the attempt of KDD as follows: "KDD is an attempt to address a problem that the digital information era made a fact of life for all of us: data overload." [FPSS96, p. 38]. KDD is a process of identifying valid, novel, and potentially useful patterns in data and is intended to facilitate and speed-up the extraction of knowledge from persistent data sources (see also [ES00, KZ02, MR05]). The digital information era is constantly evolving and reinventing itself. Thus, it is necessary to reconsider existing data processing approaches and to adapt them to the demands of upcoming challenges [NS13].

KDDS, contrary to KDD, is a relatively new research area. The perspective on data has changed significantly since Fayyad et al. [FPSS96] described the attempt of KDD. Currently, data is more dynamic and produced rapidly by continuous data sources (e.g. automatic stock trading, Internet click streams, online advertisement, or

sensor data streams). This development motivates the new research area of KDDS. The following three intentions of KDDS can be identified. First, KDDS attempts to extract knowledge from data streams (e.g. *data stream mining* [DH00]) so that computer systems are able to communicate using real-time events or information in order to trigger real-time decisions (e.g. *complex event processing (CEP)* [Luc02]). Second, KDDS can be used to slow down the speed of information by aggregating data so that relevant information can be sufficiently presented to human experts (e.g. cockpits or round robin databases [Oet13]). Third, KDDS can be applied to sample the data down so that they can be adequately stored into historical data repositories (e.g. *streaming data warehouse (SDW)* [GJSS09]) [NS13].

1.7 Combining KDD and KDDS

Figure 7 associates relevant technologies to the research areas of KDD and KDDS and illustrates the differences between both research areas. The x-axis relates to the complexity of the applied queries and the availability of system resources. The x-axis starts with complex queries which necessitate many system resources and ends with restricted system resources where only simple queries can be executed. The y-axis relates to the data flow, starting with static data, persistently stored in archives, and ending with dynamic data which arrive in a continuous and potentially infinite stream of data items. The lower section of Figure 7 relates to KDD where knowledge is extracted from persistent storages. The uppermost section of Figure 7 relates to KDDS where knowledge is extracted from data streams. The complexity of the used queries relates to the provided system resources.

As depicted in Figure 7, *data warehouses (DWHs)* [Inm02] and *database management systems (DBMSs)* [Cod82] relate to the KDD process where many system resources are intensively used to extract knowledge from persistent repositories. Embedded or lightweight DBMSs [Egy85] address the challenges of data storage under restricted computing resources. Moreover, SDWs are used to store dynamic and streaming data into persistent repositories and universal *information flow processing (IFP)* [CM12] systems are applied for *data stream processing* [BBD+02] where many system resources are available. Furthermore, lightweight IFP systems can be used for data stream processing under resource restrictions. As illustrated in Figure 7, the current thesis intends to combine KDD and KDDS for monitoring MCPSs.

Concerning the KDD process, data is stored persistently, queries are complex, and computational resources are almost unrestricted. Concerning the KDDS process, data arrives in a continuous stream of data items, queries are mostly simple, and computational resources are restricted. In order to proceed with this intention, pre-existing technologies are used. Therefore, it is possible to identify pitfalls and challenges of existing technologies for future work. The scope of the current thesis

is very interdisciplinary and covers a variety of research areas. More information about the above-mentioned technologies and a discussion on related work is given in Chapter 2 on page 17.

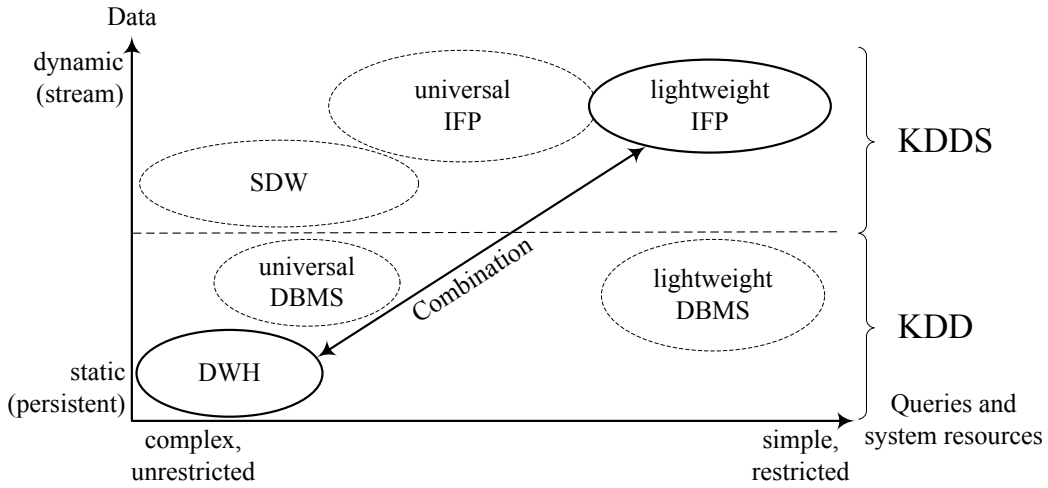

Figure 7: Combining KDD and KDDS for monitoring MCPSs (based on [GO10])

1.8 Contribution

The contribution of this thesis is threefold. First, the principal contribution is the definition of a *knowledge discovery cycle (KDC)* [NS13]. The KDC aims to combine the research areas of KDD and KDDS for the purpose of monitoring MCPSs with support of IFP. Second, a novel approach for data stream anomaly detection is explained. This approach is based on the KDC. Third, the implementation of the KDC identifies pitfalls and challenges for future work in the area of IFP.

1.9 Selected Publications

This section presents a list of selected publications. A complete list of publications is shown in the appendix on page 151. Publications 1 and 2 describe the real world scenario which is related to the ISS Columbus module. Moreover, these publications are the foundation of the cooperation with EADS Astrium Space Transportation. Furthermore, publication 3 describes the KDC in more detail, which constitutes the background of the current thesis. Publication 4 presents a case study which is based on the KDC and IFP. In addition, publication 5 provides a discussion on ISS Columbus data streams. Lastly, publication 6 discusses open challenges for data stream mining research.

1. E. Noack, T. Noack, V. Patel, I. Schmitt, M. Richters, J. Stamminger, and S. Sievi. Failure Management for Cost-Effective and Efficient Spacecraft Operation. In *Proceedings of the 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. IEEE Computer Society, 2011
2. E. Noack, A. Luedtke, I. Schmitt, T. Noack, E. Schaumlöffel, E. Hauke, J. Stamminger, and E. Frisk. The Columbus Module as a Technology Demonstrator for Innovative Failure Management. In *German Air and Space Travel Congress*, Deutscher Luft- und Raumfahrtkongress, 2012
3. T. Noack and I. Schmitt. Monitoring Mobile Cyber-Physical Systems by Means of a Knowledge Discovery Cycle. In *Seventh IEEE International Conference on Research Challenges in Information Science (RCIS)*, 2013
4. T. Noack and I. Schmitt. Monitoring Mobile Cyber-Physical Systems by Means of a Knowledge Discovery Cycle - A Case Study. In *Workshop on Knowledge Discovery, Data Mining, and Machine Learning (KDML)*, 2012
5. T. Noack, E. Noack, I. Schmitt, S. Sievi, and S. Mirzakhyl. A Discussion on ISS Columbus Data Streams. In *ECML/PKDD Workshop on Real-World Challenges for Data Stream Mining (RealStream)*, 2013
6. G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, and J. Stefanowski. Open Challenges for Data Stream Mining Research. *ACM SIGKDD Explorations Newsletter – Special Issue on Big Data*, 16(1):1–10, 2014

1.10 Structure and Organization

The present thesis is structured as follows. Chapter 2 outlines foundations of the present thesis. Moreover, related work referring to the KDC is presented. In Chapter 3, the KDC is described in more detail and compared with the presented related work. Chapter 4 introduces a novel algorithm for data stream anomaly detection. In Chapter 5, experiments are performed to evaluate the introduced data stream anomaly detection approach and a case study is carried out. Finally, Chapter 6 concludes the present thesis and outlines subjects for future work.

This manuscript is written in American English and the Oxford comma is used. New and important terms are highlighted once in italics and are explained. Terms which are referenced and explained once again are highlighted by means of single quotation marks. In addition, important terms which are intensively used throughout this theses are listed in a glossary on page 157. References are sorted in ascending order by the publication year if more than one reference is used. Direct citations provide a page number.

CHAPTER 2

Foundations and Related Work

> "*If you have built castles in the air, your work need not be lost; that is where they should be. Now put the foundations under them.*"
>
> *Henry David Thoreau (1817-1862)*

This chapter describes foundations of the present thesis along with related work and is structured as follows. First, a brief overview on the adjacent research areas is given and the term *data* is described in more detail since the term 'data' is frequently used throughout the present thesis. Moreover, a distinction is made between the terms *error*, *fault*, and *failure*. This distinction is necessary in order to ensure a consistent use of the terminology within this thesis. Next, the KDD process is exposed, and, based on this, the research area of *data mining* [FPSS96] is considered. Furthermore, selected topics of IFP are described and requirements for data stream classification are listed. The KDD process and the selected topics of IFP form the basis for the development of the KDC are described in Chapter 3 on page 41. The research area of data mining forms the basis for a new data stream anomaly detection algorithm presented in Chapter 4 on page 63. Following this, a particular view on related work is given. This includes specialized IFP approaches: the *massive online analysis (MOA) data stream classification cycle* [BHKP11], *expert systems* [HRWL83], and the *monitor, analyse, plan, execute, and knowledge (MAPE-K) reference model* [IBM03]. Lastly, monitoring is described in more detail since monitoring MCPSs is the scope of the present thesis.

2.1 Brief Overview

The present thesis is very interdisciplinary and covers a variety of subjects. Thus, it is impossible to describe each research area in detail. Accordingly, this section

attempts to give a brief overview on the adjacent research areas. This section relates to Section 1.6 along with Section 1.7 and is closely related to the author's publication [NS13].

From the author's point of view, KDD is the basement for appropriate system monitoring. KDD incorporates storage mechanisms such as DBMSs and DWHs. A DBMS is a software which is used to manage databases. As stated by Inmon: "A data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data [...]" [Inm02, p. 31]. Data mining as an independent research area constitutes an essential component of KDD (see also [DHS01, TSK05, Bis06, HK06, Cha09, WFH11]).

MCPSs are equipped with sensors which continuously produce sensor data streams (cf. Section 1.3). The research area of 'data stream processing' addresses the machining of data streams and is mandatory to provide online monitoring in a real-time manner. Based on the definition of the stream model, *data stream management systems (DSMSs)* [MWA+02] such as STREAM [BBD+02], Aurora [CcC+02], or PIPES [KS04] have been developed for data stream processing (see also [CJ09, GO10]). As reported by Stonebraker et al. [ScZ05], such DSMSs should provide high level query languages. Examples are the continuous query language [ABW06] and the stream query algebra [ACc+03]. Data stream processing engendered new storage mechanisms, such as SDWs (see also [GJSS09, GO10, GJ11]), and concepts for data stream mining. 'Data stream mining' is the application of data mining and machine learning algorithms directly onto data streams (see also [GZK05, WYH05, HK06, Agg07, BHKP11]). Several data stream mining frameworks such as MOA [BHKP10], VEDAS [KBL+04], or SMM [TLM+11] have been developed. The aforementioned concepts for data stream processing and mining can be grouped together under the acronym 'KDDS' (see also [GKA06, Gam10]).

For some time, data stream processing has been reconsidered under a certain perspective. This perspective aims to extend stream query languages by pattern definitions and action parts. This endeavor is already known from active databases as the *event-condition-action (ECA)* paradigm [DGG96, PD99]. Based on the ECA paradigm it is possible to formulate *ECA-rules* which imply: "[...] when an event occurs, check the condition and if it holds, execute the action [...]" [DGG96, p. 2]. The combination of the ECA paradigm and data stream processing is called CEP (see also [Luc02, EB09, EN10]). The acronym 'CEP' describes the deduction of complex events from fundamental or underlying events in a data stream context. Several CEP engines such as SASE [WDR06], Cayuga [DGH+06], Esper [Esp13], StreamBase [Str13], or Drools Fusion [JBo14] have been developed. Standards for CEP in combination with ECA-rules has been discussed by Paschke et al. [PVS11]. CEP has induced new storage strategies such as event data warehouses [RSOR10]. However,

an exact differentiation between DSMSs, frameworks for data stream mining, and CEP engines is not always possible. Accordingly, the acronym 'IFP' attempts to combine these approaches by means of a uniform base (see also [CM12]).

Amongst others, further information about the term 'system' is provided by several authors, including Gordon [Gor72], Bossel [Bos89], Backlund [Bac00], and Imboden et al. [IK03]. More detailed descriptions of 'embedded systems' are provided by several authors, including Marwedel [Mar07], Peckol [Pec07], Bertsche et al. [BGJ+09], and Berns et al. [BST10]. More information about 'CPSs' is provided by several authors, including Tan et al. [TGP08], Sha et al. [SGLW08], Rajkumar et al. [RLSS10], Lee et al. [LS11], and Shi et al. [SWYS11]. Additional information about 'MCPSs' is provided by several authors, including Xu et al. [XLZ+08] and Fok et al. [FPS+11]. Further information about the term 'monitoring' is provided by several authors, including Snodgrass [Sno88], Schroeder [Sch95], and Junior et al. [JR08]. More information about embedded or lightweight 'DBMSs' is provided by several authors, including Ortiz [Ort00], Nori [Nor07], and Rosenmüller et al. [RLAS07, RALS09].

2.2 Data

The term 'data' is broadly used over a variety of research domains. Moreover, it is frequently used throughout the present thesis. Accordingly, this section aims to describe this term in more detail and to introduce data scales. Data scales are introduced in order to ensure a consistent use of the terminology in this thesis.

2.2.1 Definition

It is necessary to describe the term 'data' (singular: data item) in more detail. As stated by Chattamvelli: "Data are basic facts on an entity." [Cha09, p. 2]. Fayyad et al. describe the term 'data' as follows: "[...] data are a set of facts (for example, cases in a database) [...]" [FPSS96, p. 41]. This results in the following definition of the term 'data' in the context of the present thesis for monitoring MCPSs.

Definition 2.1 (data):
Data are a set of attribute values which describe a system (or a system component) within a particular time frame.

Furthermore, there is a distinction between *univariate data* and *multivariate data*. Univariate data refer exclusively to one attribute, and multivariate data refer to more than one attribute [Ste09, Han10].

2.2.2 Data Scales

Attribute values can be subject to a variety of scales. Amongst others, these include: *nominal scale*, *ordinal scale*, *interval scale*, and *ratio scale*. It is always possible to scale-down from a higher scale to a lower scale, although down-scaling is tantamount to loss of information [Cha09] (see also [BW08, Ste09, Han10]).

Nominal Scale

The nominal scale is used to categorize attribute values by means of a name convention. This name convention can consist of numbers, characters, or character strings (e.g. names). Binary scaled attributes constitute only two outcomes of a name convention (e.g. on and off).

Ordinal Scale

Ordinal scaled attribute values exhibit a meaningful order. Consequently, comparison operations (e.g. $>$, $\geq$, $<$, $\leq$, $=$, $\neq$) can be applied between these attribute values. Ordinal scaled attribute values can always be mapped to the numbers 1 to n (e.g. school marks).

Interval Scale

The interval scale is a metric scale where the intervals are appropriately interpretable. The zero point is chosen indiscriminately (e.g. temperature or loudness).

Ratio Scale

The ratio scale is a metric scale where a fixed zero point exists and pre-defined intervals are interpretable (e.g. weight and size).

2.3 Error, Fault, and Failure

It is necessary to describe the terms 'error', 'fault', and 'failure' in more detail. Lapire defines these terms as follows: "A system failure occurs when the delivered service deviates from the specified service, [...] an error is that part of the system state which is liable to lead to failure, [...] The cause [...] of an error is a fault." [Lap95, p. 3]. Avižienis et al. describe these terms as follows: "[...] failure, is an event that occurs when the delivered service deviates from correct service. [...] The adjudged or hypothesized cause of an error is called a fault. [...] an error is the part of the total state of the system that may lead to its subsequent service failure." [ALRL04, p. 13]. Moreover, Schuster gives the following definition of these terms:

"The manifestation of a fault will produce errors in the state of the system, which could lead to a failure [...]." [Sch08, p. 26] (see also [SS94, BM00]).

Based on these statements, it is possible to deduce the following descriptions. The term 'error' describes a system state where the system does not work as expected. The term 'fault' constitutes the root cause of an error. Finally, the term 'failure' describes a situation where the intended functioning of the system (or the subsystem) cannot be guaranteed any longer.

The correlation of the above-mentioned terms is exemplified by means of the failure event of the IRFA which has occurred on the ISS Columbus module (cf. Section 1.4.3). Bearing wearout is the fault because it constitutes the root cause. The manifestation of bearing wearout led to the following errors: a continuously increasing input current, an erratic speed of the IRFA, and consequently to erratic air flow. Finally, the delivered service deviated from the specified service. This constitutes the failure.

2.4 KDD Process Model

The KDD process model is described in more detail since it constitutes the basis for the KDC which is described in Chapter 3 on page 41. As stated by Fayyad et al. [FPSS96], the KDD process is an interactive and iterative process which entails several processing steps where many decisions must be made by human experts. As depicted in Figure 8, the KDD process includes five processing steps: *selection*, *preprocessing*, *transformation*, *data mining*, and *evaluation*.

1. **Selection**: The first processing step is used to identify the objectives of a planned discovery process. Therefore, it is necessary to understand prior knowledge from the viewpoint of the related application domain. A target data set is selected which focuses on a relevant subset of variables and data samples.

2. **Preprocessing**: The second processing step encompasses data cleaning and preprocessing. Amongst others, this includes noise reduction, handling missing data fields, and accounting for time-sequence information. As a result, preprocessed data are created.

3. **Transformation**: The third process step involves data reduction and projection. Therefore, useful features are selected to represent relevant data depending on the previously identified discovery objectives. This processing step includes dimensionality reduction along with transformation methods and produces transformed data.

4. **Data mining**: The fourth processing step comprises the selection of particular data mining methods such as clustering, classification, or regression analysis. These methods are applied to search for patterns of interest. Data mining is described in more detail in the following section.

5. **Evaluation**: The fifth processing step is used for interpretation and evaluation of the mined patterns by human experts. This also includes visualization, documentation, and storage of the patterns and the discovered knowledge.

As depicted in Figure 8, the KDD process involves iteration steps and loops between any two steps. For example, a specific discovery process could return from the processing step 'evaluation' to any of the previous processing steps for further iteration and adaptation.

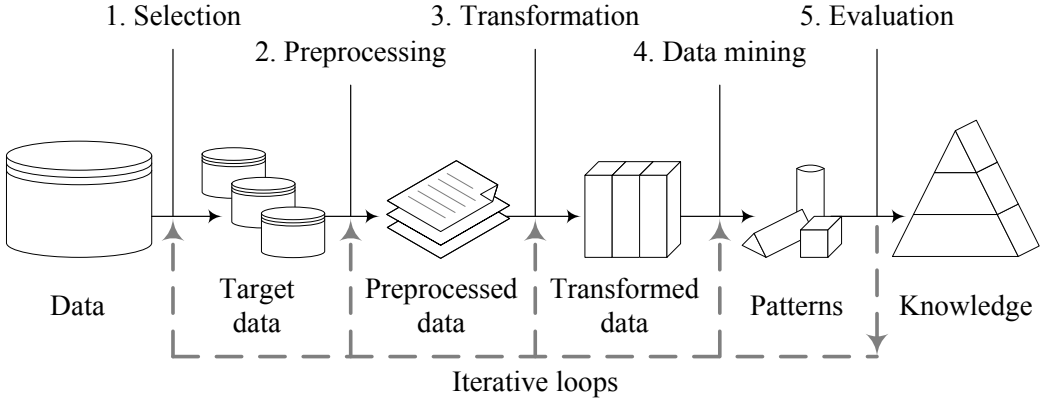

Figure 8: The KDD process model (based on [FPSS96])

2.5 Data Mining

Data mining is an important part of the aforementioned KDD process and is described by Fayyad et al. as: "[...] the application of specific algorithms for extracting patterns from data." [FPSS96, p. 39]. This section attempts to describe selected topics of data mining in more detail. This description constitutes the basis for a new data stream anomaly detection algorithm which is presented in Chapter 4 on page 63. The current section is structured as follows. Initially, learning methods are described, followed by data mining types. Table 1 summarizes the relation between learning methods and data mining types. Furthermore, two methods for the combination of classifiers are described. The descriptions of the learning methods,

data mining types, and combination of classifiers are essential to understand the presentation of the new data stream anomaly detection algorithm.

2.5.1 Learning Methods

In the research areas of data mining and statistical machine learning, a distinction is usually made between *supervised learning* and *unsupervised learning*. Supervised learning methods use *labeled data*, while unsupervised learning methods use *unlabeled data*. Labeled data provide class labels which indicate the membership to *classes*, whereas unlabeled data do not provide them [Cha09]. A 'class' represents a logical grouping of data and a *cluster* is a homogeneous group of data. This correlation is known from the cluster assumption [See00].

2.5.2 Data Mining Types

Amongst others, a distinction can be made between the following two data mining types: *clustering* and *classification*. The aforementioned data mining types are described as follows.

Clustering

Clustering is a technique to break a large heterogeneous set of data into a small number of homogeneous groups or clusters so that data items of a cluster are more similar to each other than to those in other clusters. Clustering is commonly an unsupervised learning method [Cha09] (see also [JMF99]).

Classification

Classification is a technique to assign data into a set of distinct classes. It is usually a supervised learning method and includes three steps: *training*, *assessing*, and *classifying*. Training is used to learn a classifier model (short form: a classifier) from a set of labeled training data. Assessing is used to evaluate and refine the learned model by data items of the training data (e.g. cross validation [Efr83]). Finally, classifying is used to assign unlabeled data items to the distinct classes by means of the trained classifier model [Cha09] (see also [JMF99]).

However, classifiers are often used to distinguish between just two classes (e.g. support vector machines (SVMs) [CV95]), while many real world scenarios entail a larger number of classes. Such a classification problem is called *multi-class classification (MCC)* [TD02, HL02] hereinafter. Thus, it is necessary to combine these classifiers in an appropriate manner in order to solve such a multi-class classification problem.

One-class classification (OCC) is a specific classification technique. As stated by Moya et al.: "We call a classifier that can recognize new examples of target patterns and distinguish those from non-target patterns a *one-class classifier*. A desirable feature in a one-class classifier is the ability to learn to characterize the target class by examining only target data without requiring training samples of non-target data." [MH96, p. 1]. OCC can be appropriately applied when only a small proportion of training instances of a single class is available. Therefore, the classifier model usually comprises a trained boundary around the present class. The resulting classifier model is consequently used to decide if an unlabeled data item is either a member or a non-member of this class [Tax01].

		Methods	
		unsupervised	supervised
Types	Clustering	✓	
	Classification		✓
	OCC		✓

Table 1: Relation between learning methods and data mining types

2.5.3 Combining Classifiers for Multi-class Classification

The combination of classifiers is a well-studied research area, and a large number of combining mechanisms are already available. Therefore, the individual decisions of all classifiers are combined in some way, typically by voting or averaging, to classify unlabeled data items [Kun04] (see also [Die97, KR00, Gam10]). Amongst others, there are two general mechanisms for combining classifiers to solve a multi-class classification problem: *one-against-rest (OAR)* and *one-against-one (OAO)* [TD02, HL02] (see also [HYMK09]). Two sets of three classes $\{\omega_1, \omega_2, \omega_3\}$ are considered in an example shown in Figure 9. The variable $h \in \mathbb{N}$ denotes the number of classes (in this example $h = 3$). As stated by Tax et al. [TD02], both mechanisms assume that the applied classifies output binary decisions.

One-against-rest

As depicted on the left in Figure 9 (i), an OAR-based multi-class classification model is trained between each class and the $h-1$ remaining classes. Consequently, h many classifiers must be trained. Each discriminant classifier of the OAR-based mechanism outputs two decisions: ω_i (member of class ω_i) or ω_i^{C} (complement of class ω_i) with $i = \{1 \,..\, h\}$.

One-against-one

As depicted on the right in Figure 9 (ii), an OAO-based multi-class classification model is trained between each pair of classes. Consequently, $h(h-1)/2$ classifiers must be trained. Each classifier of the OAO-based mechanism outputs two decisions: ω_i (member of class i) or ω_j with $j = \{1 \,..\, h\}$ (member of class j).

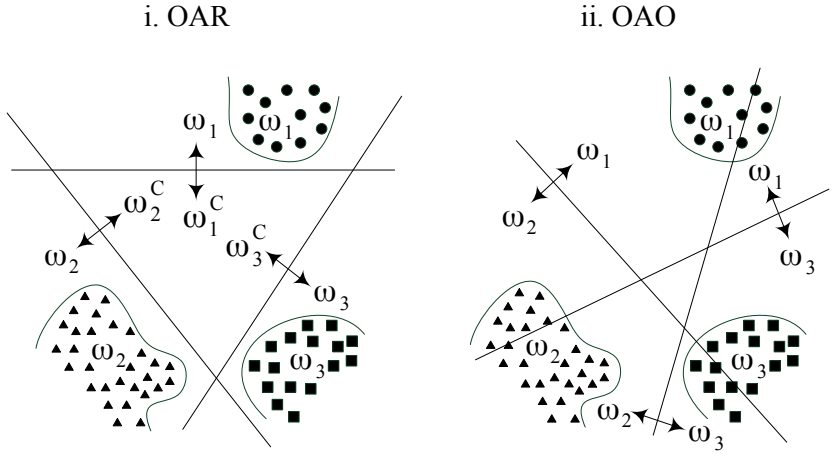

Figure 9: OAR versus OAO (based on [TD02])

2.6 Information Flow Processing (IFP)

This section discusses selected topics of the IFP domain which form the basis for the development of the KDC. First, the stream model is described. Based on this, differences between DBMSs and IFP engines are outlined. Further on, reference architectures of a DSMS, an IFP engine, and a SDW are presented. Moreover, specialized data stream processing approaches are briefly outlined. Furthermore, *windows* are described. Lastly, the *MOA data stream classification cycle* [BHKP10] is explained and requirements for data stream classification are listed.

2.6.1 Stream Model

As described by Babcock et al. [BBD+02], input data that arrive as one or more continuous data streams, are not available for random access from memory and differ from the conventional stored relation model in different ways (see also [GO10]).

- Data items arrive online, and the sequence is potentially infinite.
- The processing system cannot control the order of the arriving data items.
- An already processed data item is discarded or stored.

2.6.2 DBMSs versus IFP engines

As listed in Table 2, DBMSs process persistently stored data while the update rate is relatively low. IFP engines process data streams while the update rate is very high and sometimes bursty. During query execution, DBMSs have random access to all stored data, whereas IFP engines process data streams at one sequential pass. The process model of DBMSs is usually passive, query-driven, and short-dated or ad hoc queries are executed once by pulling an answer by means of persistently stored data. Results are presented only when explicitly asked by users or applications. This process model is called *human-active database-passive (HADP)* [ACc+03]. However, the process model of IFP engines is active, data-driven, and long-dated or continuous queries are deployed, pushing new results as new data arrive. This process model is called *database-active human-passive (DAHP)* [ACc+03]. DBMSs process one query at a time and use a fixed query plan. Consequently, an exact result is anticipated. Since IFP engines process many continuous queries at a time, adaptive query plans are necessary, and only approximate results are anticipated [GO10].

	DBMSs	IFP engines
Data	persistent data	data streams
Update rates	relatively low	high, bursty
Data access	random	sequential or one-pass
Process model	HADP (query-driven, pull)	DAHP (data-driven, push)
Queries	one-time, ad hoc	continuous
Query plans	fixed	adaptive
Query results	exact	approximate

Table 2: Differences between DBMSs and IFP engines (based on [GO10])

2.6.3 DSMS Reference Architecture

Figure 10 depicts a reference architecture of a DSMS. It comprises the following components: *streaming inputs*, *input buffer*, *input monitor*, *working storage*, *local storage*, *query repository*, *query processor*, and *streaming outputs*. The input buffer is used to capture the streaming inputs. The input monitor is an additional component which can be used to collect various statistics such as inter-arrival times. The working storage stores recent portions of the streaming inputs temporarily. The stored data is then used for query execution. The local storage contains meta data, and the query repository is used to register continuous queries. Continuous queries are transformed into execution plans after registration. The query processor executes the registered queries and produces streaming outputs which can be used for further processing. The query processor may communicate with the other components to optimize query execution [GO10].

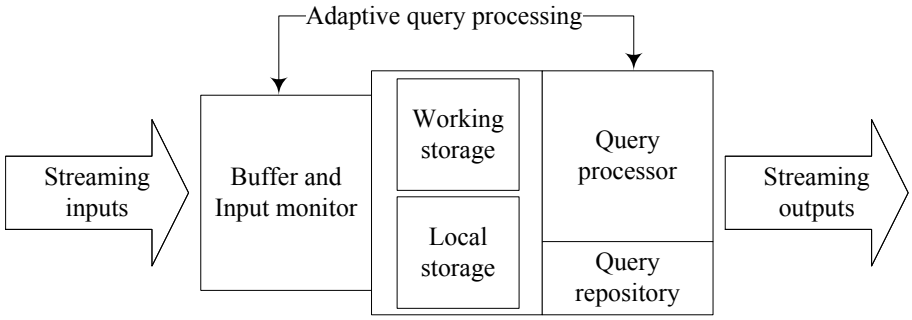

Figure 10: Reference architecture of a DSMS (based on [GO10])

2.6.4 IFP Engine Reference Architecture

Figure 11 depicts a reference architecture of an IFP engine. It includes the following components: *streaming inputs*, *receiver*, *clock*, *decider*, *history*, *rules*, *knowledge base*, *producer*, *forwarder*, and *streaming outputs*. This IFP reference architecture is used to explain the main functional components and to compare existing IFP engines. Since this reference architecture aims to provide a uniform base, it extends the reference architecture of DSMSs from the viewpoint of IFP.

As a DSMS, an IFP engine captures and processes streaming inputs approaching from different sources and produces streaming outputs. Rules are applied to process the streaming inputs. Amongst others, this includes functions such as filtering, combining, or aggregation. The receiver is used to manage the streaming inputs

and to forward them to the next component. The clock is an additional component which can be connected to the receiver to create special data stream elements (e.g. time stamps) and to allow periodic processing of streaming inputs.

From the viewpoint of IFP, rules (or ECA-rules) are logically composed by the following two parts: *condition* and *action*. The condition part defines the constraints that must be satisfied by incoming data items (e.g. streaming inputs and time stamps), and the action part defines what to do. Based on this distinction, the reference architecture entails two phases: *detection* and *production*. The detection phase is realized by a decider, while the production phase is realized by a producer. The decider gets data items from the receiver and compares them with the condition part of the existing rules. The decider may need to accumulate or access synopsis information and historical data in order to process the incoming data items. Consequently, the action part of each triggered rule is passed on to the producer for execution. The producer sends the results to the forwarder. Moreover, there exists a loop which forwards selected results to the receiver. The knowledge base constitutes a read-only memory which contains information used during the detection and production phases. Lastly, the forwarder sends the generated streaming outputs to expected sinks for further processing [CM12].

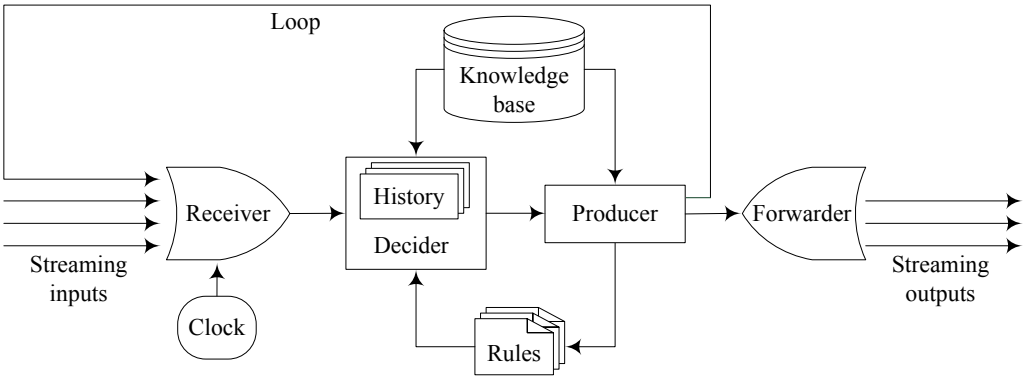

Figure 11: Reference architecture of an IFP engine (based on [CM12])

2.6.5 SDW Reference Architecture

Golab et al. define the acronym 'SDW' as follows: "[...] an SDW faces the same challenges as standard data warehouses, among them the need to store massive amounts of data on disk for offline-analysis." [GO10, p. 39]. Figure 12 depicts an abstract reference architecture of an SDW. It includes the following components: *data feeds*, *data files*, *extract, transform, and load (ETL) process*, *update scheduler*, *base tables*, and *derived tables*. Along with streaming inputs, data feeds arrive continuously from various data sources. In some cases, these inputs are in the form of text or zipped files. The update scheduler is used to decide which data or files should be integrated next. As in DWHs, the data then pass through an ETL process and are subsequently stored into persistent databases. For example, an ETL process includes simple data cleaning, converting, or time stamp standardization. The persistent databases include base tables and derived tables. Base tables are sourced directly from data input, and derived tables refer to materialized views. Additionally, the update scheduler decides which derived table must be updated next [GO10].

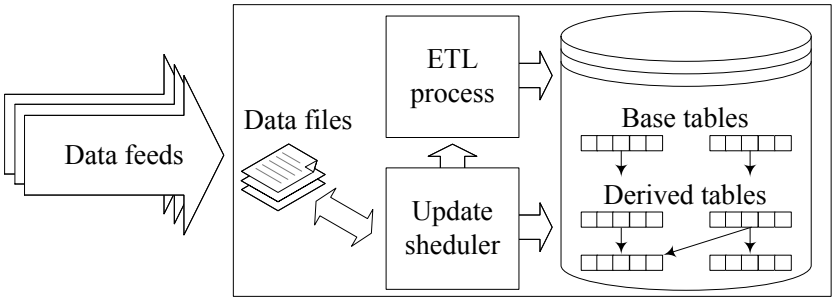

Figure 12: Reference architecture of an SDW (based on [GO10])

As summarized in Table 3, there are differences between DWHs and SDWs. A fundamental difference relates to the frequency and asynchronism of updates. In contrast to DWHs, SDWs are subject to a high update frequency and asynchronous update propagations. Furthermore, SDWs attempt to integrate arriving data continuously rather than refreshing the entire warehouse periodically. Additionally, SDWs make recent and historical data available for analysis and further processing. Consequently, a fast ETL process and efficient update propagation across derived tables are required in order to keep them up with the streaming inputs [GO10].

	DWHs	SDWs
Update frequency	low	high
Update propagation	synchronized	asynchronous
Data	historical	recent and historical
ETL process	complex	fast, lightweight

Table 3: Differences between DWHs and SDWs (based on [GO10])

2.6.6 Stream Windows

As stated by Golab et al. [GO10] and Cugola et al. [CM12], stream windows are constructs of data stream query languages that are applied to language operators to restrict the scope of continuous queries. Stream windows can be classified by the way they bind the move. Amongst others, these include: *fixed windows*, *landmark windows*, and *sliding windows*.

Fixed Windows

Fixed windows define starting and ending points and do not move. For example, they can be used to process data items received within a defined time frame.

Landmark Windows

Landmark windows provide one fixed and one moving point. For example, the starting point is fixed and the ending point moves forward in time. This configuration can be used to process data items which arrive since a fixed time.

Sliding Windows

Sliding windows are the most commonly used window type. Typically, they provide a fixed size where the starting and ending points slide forward in time. For example, sliding windows can be used to process the last ten data items received.

2.6.7 Requirements for Data Stream Classification

Data stream classification is a part of KDDS and data stream mining. Based on the stream model (cf. Section 2.6.1), Bifet et al. identified four requirements for data stream classification [BHKP11].

1. **Processing**: The first requirement claims to "Process an example at a time, and inspect it only once (at most)" [BHKP11, p. 8]. This requirement implies that data items arrive in a continuous stream, random access is not available, and processed data items are discarded. Although, this very strong requirement is softened by two limitations: an algorithm which provides a short-term storage can remember previous data items, and a data stream can be resent.

2. **Memory**: The second requirement claims to "Use a limited amount of memory" [BHKP11, p. 8]. This requirement implies that a single data stream is many times larger than the accessible memory.

3. **Time**: The third requirement claims to "Work in a limited amount of time" [BHKP11, p. 9]. This requirement implies that the runtime complexity of an algorithm must be linear in the number of processed data items.

4. **Preparation**: The fourth requirement claims to "Be ready to predict at any point" [BHKP11, p. 9]. This requirement implies that a data stream classification algorithm must be able to prepare a classification model during operation as efficiently as possible.

2.6.8 Specialized IFP Approaches

There are several IFP approaches which refer to specialized application domains [NS13]. One of such approaches is *Odysseus* [Bol09]. Odysseus is a data stream management framework intended to combine and integrate different techniques for data stream processing and CEP. Based on Odysseus, Geesen [Gee13] provides a discussion on the integration and optimization opportunities of machine learning mechanisms into DSMS. A second approach is *VEDAS* [KBL+04]. VEDAS is a mobile and distributed data stream mining system for real-time vehicle monitoring. A third approach is described by Bell et al. [BKZ10]. This approach presents a sensor event abstraction language for spacecraft monitoring. A fourth approach is called *Mini-ME* [CEW01]. Mini-ME is a rule-based fault monitoring system for spacecraft monitoring. A fifth approach is mentioned by Madden et al. [MF02]. It describes an architecture for queries over streaming sensor data. A sixth approach is presented by Li et al. [LPV+08]. It reflects real-time storm detection and weather forecasting by means of data mining and event processing. A seventh approach is mentioned by Stojanovic et al. [SA11]. This approach is related to the application of CEP in real-time situations. An eighth approach is presented by Weigert et al. [WHF11]. It is related to mining large distributed log data in near real-time. Moreover, this approach also considers real-time fault diagnoses. Yet another approach is called *Pharos* [FPS+11]. Pharos is a testbed for the validation of MCPSs.

However, the above-mentioned approaches consider key challenges for monitoring MCPSs (cf. Section 1.3) and complex characteristics of MCPSs (cf. Section 1.5) more or less insufficiently.

2.7 MOA Data Stream Classification Cycle

The MOA data stream classification cycle extends the general classification process for handling data streams and is based on the requirements for data stream classification (cf. Section 2.6.7). As depicted in Figure 13, the MOA data stream classification cycle contains three steps. First, a labeled data item is caught from the data stream. Second, the labeled data item is processed and is used to adapt the present model. Third, the classification cycle is ready to process the next data item. The learned model can be used to predict the class of an unlabeled data item on request [BHKP10, BHKP11].

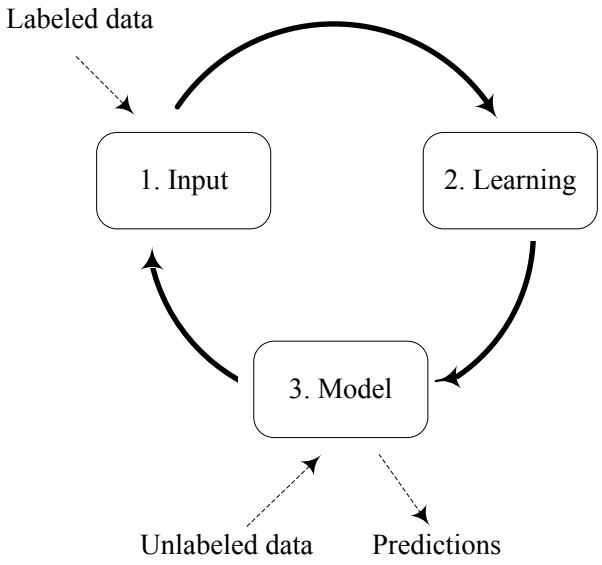

Figure 13: The MOA data stream classification cycle (based on [BHKP11])

2.8 Expert Systems

Expert systems technology derives from the research area of artificial intelligence. As stated by Hayes-Roth et al.: “The area of expert systems investigates methods and techniques for constructing man-machine systems with specialized problem-solving expertise.” [HRWL83, p. 3]. Jackson describes the term ’expert system’

as follows: "An *expert system* is a computer program that represents and reasons with knowledge of some specialist subject with a view to solving problems or giving advice." [Jac90, p. 3]. In other words, expert systems are computer systems which aim to combine human expertise with artificial expertise. An expert system can be used to apply expert knowledge to difficult real world problems and to augment and enhance the user's skills [Wat86] (see also [Ped89, Ign90]).

Figure 14 depicts an abstract structure of an expert system. The *knowledge base* of an expert system comprises the knowledge about the problem domain or the domain knowledge. The knowledge base involves facts and rules which are the basis for decision making. The *inference engine* of an expert system contains an interpreter and a scheduler and draws conclusions from the domain knowledge. The interpreter decides how to apply the rules to infer new knowledge, whereas the scheduler decides the order in which the rules should be applied [Wat86].

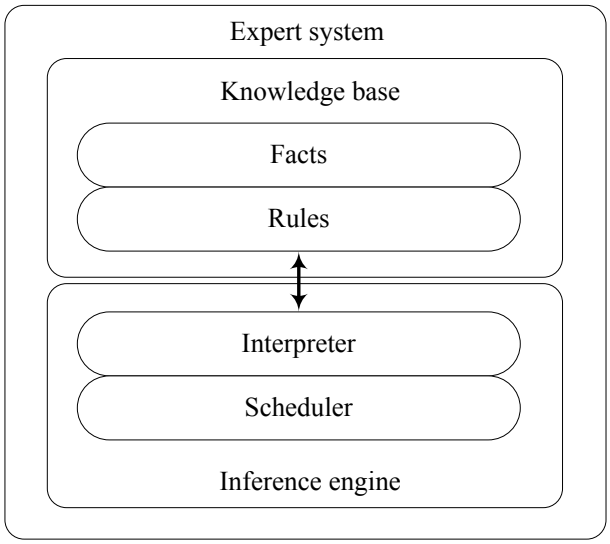

Figure 14: An abstract structure of an expert system (based on [Wat86])

2.9 MAPE-K Reference Model

The MAPE-K reference model is intended to provide an autonomic control loop with the purpose to achieve autonomic computing. As depicted in Figure 15, the MAPE-K reference model comprises a *managed element* and an *autonomic manager*. The managed element is a system component which represents any software or hardware resource and is controlled through its sensors and actuators. The sensors provide mechanisms to collect information about the state of an element, and

the actuators are mechanisms to change the state of an element. The autonomic manager is a software component that implements the control loop. The autonomic manager should ideally be configurable by human experts. Moreover, the control loop is dissected into four parts that share knowledge: *monitor*, *analyze*, *plan*, and *execute*. The monitor part is used to collect, aggregate, filter, manage, and report information. The analyze part is used to learn correlations and to model complex situations. The plan part is used to structure actions. The execute part is used to control the execution of the previously constructed plan [IBM03, HM08].

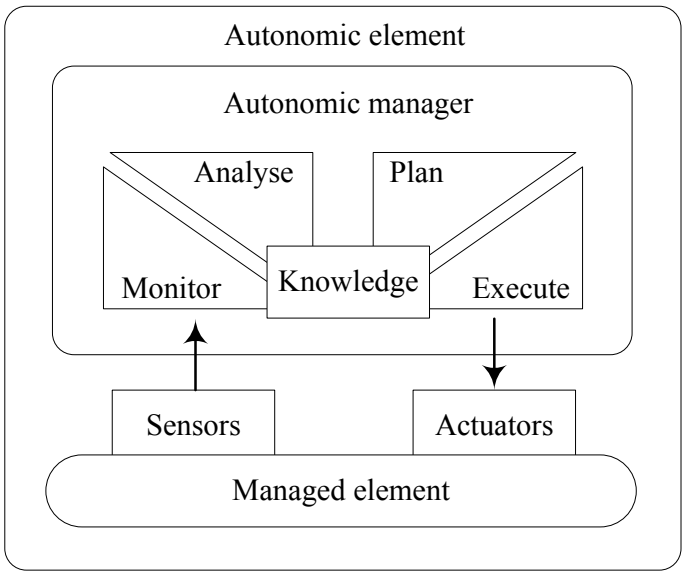

Figure 15: The MAPE-K reference model (based on [IBM03])

2.10 Monitoring

The term 'monitoring' is broadly used over many application domains, and there exist a lot of specific terms which are related monitoring terms. Therefore, this chapter discusses foundations for monitoring and attempts to discuss and to structure these terms in the context of the current thesis. Accordingly, monitoring is discussed in more detail and monitoring characteristics are explained. Furthermore, monitoring types and monitoring objectives are outlined.

2.10.1 Definition

It is necessary to describe the meaning of the term 'monitoring' in more detailed. Schroeder specifies it as follows: "Monitoring gathers information about a compu-

tational process as it executes [...]" [Sch95, p. 72]. Snodgrass gives the following definition: "Monitoring is the extraction of dynamic information concerning a computational process, as that process executes." [Sno88, p. 157]. Kopetz formulates it as follows: "An important function of a real-time computer system is the continuous monitoring [...] to detect abnormal process behaviors." [Kop11, p. 5]. Tsai et al. describe monitoring as: "[...] the recording of specified event occurrences during [...] execution in order to gain runtime information that cannot be obtained merely by studying [...]" [TY95, p. 9]. Dvorak et al. describe it as follows: "[...] monitoring is a continuous real-time task of recognizing anomalies in the behavior of a dynamic system and identifying the underlying faults." [DK89, p. 1]. This results in the following definition of the term 'monitoring' in the context of the present thesis.

Definition 2.2 (monitoring):
Monitoring is a continuous task of recognizing specific event occurrences in the behavior of MCPSs and the identification of underlying faults during operation.

2.10.2 Monitoring Variants

Amongst others, it is possible to identify the following six monitoring variants: *periodic monitoring* [Sil05], *continuous monitoring* [Sil05], *local monitoring*, *global monitoring*, *online monitoring* [Sch95], and *offline monitoring*. These variants are explained further.

Periodic Monitoring

The term 'periodic monitoring' describes a monitoring variant where data are only collected at specific times. This involves intermittent data gathering and analysis where removable monitoring equipment can be used. Due to periodic interruptions during the monitoring process, it is mostly used to monitor uncritical system components. The focal points of periodic monitoring are trend analysis and severity level checks [Sil05] (see also [PP09]).

Continuous Monitoring

The term 'continuous monitoring' describes a monitoring system which is permanently installed. In contrast to periodic monitoring, a very frequent data collection is continuously analyzed in an automatic fashion. It is carried out on critical equipment such as the ISS Columbus air loop (cf. Chapter 1.4.2). Changes of the system behavior should trigger more detailed investigation or possibly an automatic deactivation of the system components such as the failure event of the IRFA (cf. Chapter 1.4.3) [Sil05] (see also [PP09]).

Local Monitoring

The term 'local monitoring' describes a monitoring variant where only a few system components of an MCPS can be monitored. Failures or events that relate to few system components should be detected by the use of local monitoring. Local monitoring is widely applicable, and the implementation costs for local monitoring are reasonably low [NS13].

Global Monitoring

The term 'global monitoring' describes a monitoring variant where an MCPS is monitored entirely. Complex interrelations and influencing factors exist between an MCPS and the physical environment embedded within. Furthermore, complex interrelations and influencing factors exist between the system components due to the complex characteristics of MCPSs (cf. Chapter 1.5). Hence, it is a requirement to gather and to detect such complex interrelations by the use of global monitoring. The monitoring costs for global monitoring are usually much higher than for local monitoring. Depending on the application domain, global monitoring can be more accurate than local monitoring [NS13].

Online Monitoring

The term 'online monitoring' describes a monitoring variant where a monitoring system is synchronously applied with the target system. It relates to monitoring and data analysis which work in an automatic and real-time manner [Sch95].

Offline Monitoring

The term 'offline monitoring' describes a monitoring variant where a monitoring system is asynchronously applied from a target system. It relates to monitoring and data analysis which work in a semi-automatic and near real-time or long-term manner.

2.10.3 Monitoring Types

Amongst others, it is also possible to identify the following three monitoring types: *model-based monitoring* [DK89], *limit monitoring* [FYM05], and *condition monitoring* [Sil05]. These monitoring types are described below. Table 4 associates the aforementioned monitoring variants to monitoring types along with the lifetime phases of the bathtub curve (cf. Section 1.2).

Model-based Monitoring

For model-based monitoring, a static and preliminary model of the target system is built which is implemented in hardware. For example, such a preliminary model could be built using a prototype during the design and test phases of an MCPS [DK89]. Model-based monitoring comprises two monitoring variants, continuous and local monitoring. Model-based monitoring is necessary to provide basic reliability of the target system during the normal wear phase (cf. Section 1.2). However, building such a static and preliminary system model is highly expensive and very time consuming. Model-based monitoring approaches suffer from limited knowledge about the target system, the surrounding physical environment, and the related system behavior during the design and test phases (cf. Section 1.2). Moreover, model-based monitoring approaches suffer from the inflexibility of the resulting model during operation, and it is very difficult to adjust a hardware implementation during operation. This includes no dynamic adjustments or revisions of the static model during operation [NS13] (see also [dKW89, HB95]).

Limit Monitoring

Limit monitoring (or limit checking) is a very common monitoring type. Limit monitoring uses thresholds [FYM05, NBW$^+$10] mostly defined by means of one-dimensional functions. A message is generated if an attribute value reaches a previously defined threshold. Limit monitoring supports periodic, continuous, and local monitoring variants. Amongst others, limit monitoring can be implemented by means of intelligent [OC92] or *smart sensors* [Mei08]. An intelligent or smart sensor integrates information gathering with additional signal processing functions. Moreover, the cumulative sum [Pag54] is another possibility for implementing limit monitoring. Limit monitoring is an appropriate monitoring approach that can be easily applied over a variety of application domains. According to the real world scenario, limit monitoring can be used to detect the gradual changes caused by bearing wearout (cf. Chapter 1.4). Although, limit monitoring ignores complex interrelations between system components and the reliability of the sensor itself. For example, a sensor could be broken and send defective data. According to the aforementioned failure event of the IRFA (cf. Chapter 1.4.3), limit monitoring is not always the best approach. A failure event can constitute complex interrelations of more than one attribute or sensor. Moreover, the monitoring effort increases with the number of sensors which must be monitored [NS13].

Condition Monitoring

Condition monitoring is applied to monitor the conditions or system states of MCPSs. It is used to obtain convenient information on the condition of system components to human experts. Therefore, n-dimensional mathematical models or vector spaces are used to describe the conditions or system states of MCPSs. Condition monitoring includes all monitoring variants and can be appropriately applied over all phases of the bathtub curve (cf. Section 1.2). Furthermore, it includes advantages against the aforementioned monitoring types. These include: increased reliability, improved efficiency, extended operational phase, improved safety, and flexibility and adaptability during operation. Condition monitoring can also be applied for monitoring legacy systems. However, there are some disadvantages such as monitoring equipment costs, operational costs, and costs for training the mathematical models [Sil05] (see also [Rao96, PP09]).

		Variants				Phases		
		Periodic monitoring	Continuous monitoring	Local monitoring	Global monitoring	Wear in phase	Normal wear phase	Wearout phase
Types	Model-based monitoring	✗	✓	✓	✗	✗	✓	✗
	Limit monitoring	✓	✓	✓	✗	✗	✓	✗
	Condition monitoring	✓	✓	✓	✓	✓	✓	✓

Table 4: Monitoring types associated to variants and phases of the bathtub curve

2.10.4 Monitoring Objectives

Figure 16 summarizes three possible monitoring objectives from the angles of data usage and time reference. The first monitoring objective relates to the past where persistent data coming from historical data archives are used. This monitoring objective is applied for long-term and root cause failure analysis. Moreover, it can be used to detect gradual changes of the system behavior. The second monitoring objective relates to the present time where synopsis information of persistent and

transient data such as data streams are computed. This monitoring objective can be used for failure detection, anomaly detection, and state change detection. Moreover, it can be used to detect gradual and sudden change of the system behavior. The third monitoring objective relates to the future where prediction models built by means of persistent and transient data are used. This monitoring objective can be used for failure prediction and to forecast possible changes of the system behavior in the future.

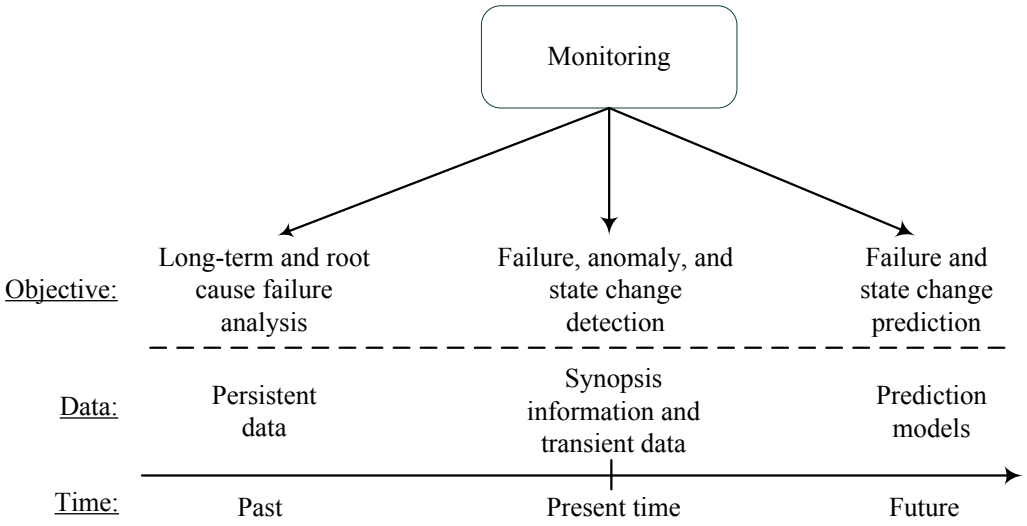

Figure 16: Monitoring objectives (based on [SLM10])

CHAPTER 3

The Knowledge Discovery Cycle (KDC)

> "*The greatest obstacle to discovery is not ignorance – it is the illusion of knowledge.*"
>
> *Daniel Joseph Boorstin (1914-2004)*

In this chapter, the KDC is described in more detail. The structure of this chapter is as follows. First, requirements for monitoring MCPSs are identified. Second, characteristics of the KDC are outlined. Third, key challenges for monitoring MCPSs (cf. Section 1.3) are compared with the aforementioned characteristics of the KDC. Fourth, the stream model (cf. Section 2.6.1) and the requirements for data stream classification (cf. Section 2.6.7) are discussed. Based on this discussion, a *storage-aware stream model* for data stream classification is introduced and training methods are described. Fifth, processing steps for the KDC are identified. Sixth, these processing steps are assigned to pre-existing concepts. Seventh, the KDC is compared with related work. Finally, this chapter is concluded.

3.1 Requirements for Monitoring MCPSs

The following five requirements for monitoring MCPSs can be identified: *time*, *locality*, *knowledge*, *system resources*, and *sharpness*. The identification is based on the abstract architecture of MCPSs (cf. Section 1.1), key challenges for monitoring MCPSs (cf. Section 1.3), characteristics of MCPSs (cf. Section 1.5), and monitoring variants (cf. Section 2.10.2). The previously discussed real world scenario (cf. Section 1.4) is used as an example. As depicted in Figure 17, each requirement contains two conditions. This section is closely related to the author's publication [NS13].

1. **Time**: This requirement refers to behavioral changes of an MCPS (or target system) under a temporal reference. There is a distinction between 'sudden changes' and 'gradual changes' (cf. Section 1.3). Sudden changes such as crashes can be unforeseen and may occur at any time. Depending on the resulting effects, it can be necessary to detect sudden or gradual changes immediately and in real-time. Moreover, gradual changes such as wear and tear can require long-term analysis in order to identify potential system threats.

2. **Locality**: This requirement refers to interrelation effects of influencing factors and the spatial location of monitoring. Two types of locality can be distinguished: 'local monitoring' and 'global monitoring' (cf. Section 2.10.2).

3. **Knowledge**: This requirement refers to available information about an MCPS, the physical environment in which the system is embedded, and the related system behavior. The author of the present thesis distinguishes between two conditions: *known* and *unknown*. This distinction is closely related to the division of known and unknown system states which is described in Section 4.1 on page 64.

   The condition 'known' refers to the existence of knowledge and information about a monitored MCPS. It means that it is necessary to employ available knowledge and information as comprehensively as possible for monitoring.

   The condition 'unknown' refers to the unawareness of an MCPS. Because of unforeseeable conditions, a dynamic, flexible, and adaptable monitoring process is required. The monitoring process should be able to gain knowledge continuously to decrease unawareness over time. Simultaneously, it is required to detect and predict unexpected situations. This requirement considers that previously acquired knowledge can expire. Accordingly, mechanisms able to calculate the degradation of knowledge truthfulness over time are required.

4. **System resources**: This requirement refers to all available computational resources for monitoring, data processing, and data analysis. There are two types of system resources: *restricted* and *unrestricted*. An MCPS is subject to resource restrictions (cf. Section 1.4.1). Thus, external information systems are usually used to compensate these restrictions. Online monitoring (cf. Section 2.10.2) refers to automatic as well as real-time monitoring and should be directly applied onto an MCPS. Offline monitoring (cf. Section 2.10.2) refers to semi-automatic and long-term analysis. Offline monitoring should be applied by means of external information systems. Long-term analysis in particular requires an extreme amount of system resources. Consequently, a combination of online and offline monitoring is required to provide enough system resources for the entire monitoring approach.

5. **Sharpness**: This requirement refers to the interpretation of conditions. Two types of processing can be distinguished: *crisp* and *non-crisp*. System states must be detected exactly and reliably by the use of binary processing (Boolean logic; e.g. if a threshold value is reached). Although, crisp processing can be inadequate for particular problems. Hence, it is necessary to generalize binary processing by means of affiliation degrees between 1 and 0 as fuzzy membership values [Zad73]. Value 1 implies full affiliation, while value 0 implies the opposite.

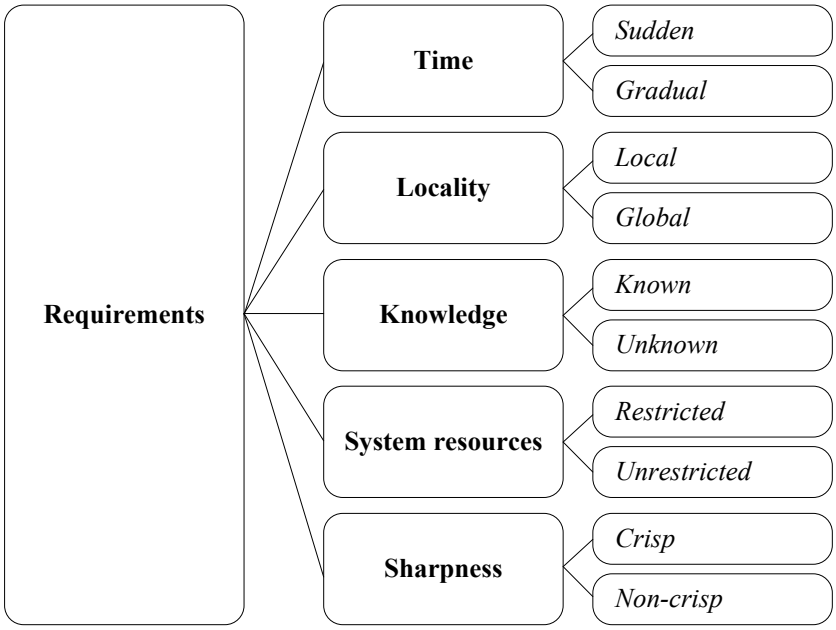

Figure 17: Requirements for monitoring MCPSs (based on [NS13])

3.2 Characteristics of the Knowledge Discovery Cycle

The KDC is a *cyclic*, *dynamic*, and *abstract* arrangement of data processing concepts. Furthermore, it comprises *data-*, *knowledge-*, and *event-oriented* data analysis concepts. Figure 18 depicts a simplified version of the KDC. The characteristics of the KDC are discussed below. This section is based on the author's publication [NS13] (see also [NS12a, NS12b, NSS13b]).

1. **Cyclic**: The KDD process is an iterative process which comprises a set of processing steps and provides loops between any of two processing steps. These loops are used for iteration and adaptation (cf. Section 2.4). Therefore, the

KDD process model can be represented as a cyclic process which includes iterative loops.

The reference architectures of DSMSs (cf. Section 2.6.3) and IFP systems (cf. Section 2.6.4) can be interpreted as preliminary process models for KDDS. However, to the best of the author's knowledge, no concluding process model for KDDS exists as yet. In accordance which the KDD process model, a potential KDDS process model is also an iterative process and must provide loops between processing steps as well. Consequently, a KDDS process model can also be represented as a cyclic process and must include iterative loops.

The KDC is intended to combine the KDD process model with a KDDS process model into a cyclic process model for monitoring MCPSs. This cyclic process model aims to increase information and knowledge about an MCPS, the system environment, and the related system behavior at each *round trip* in an iterative process. A round trip is a complete cyclic pass through the KDC. As depicted in Figure 18, a distinction is made between an *online subcycle* and an *offline subcycle*.

The online subcycle refers to the KDDS process where online monitoring (cf. Section 2.10.2) is applied in an automatic and real-time manner. The support of human experts is usually not needed. Thus, the online subcycle reflects the properties of IFP engines where the DAHP process model is used (cf. Section 2.6.2). The online subcycle reflects the aforementioned three intentions of KDDS (cf. Section 1.6). First, it extracts knowledge from data streams to invoke real-time decisions. Second, it is used to slow down the speed of data so that relevant information can be presented to human experts (e.g. cockpits). Third, it samples data down so that data can be adequately sent to external information systems. The iterative loop is required to adapt the online monitoring process iteratively and automatically during operation. For example, existing data or information can be used, depending on the monitoring objectives (cf. Section 2.10.4), for predictive analysis or for optimizing online monitoring.

The offline subcycle refers to the KDD process where offline monitoring (cf. Section 2.10.2) is applied for semi-automatic and long-term analysis. Human experts plan objectives for knowledge discovery and evaluate discovered knowledge. Thus, the offline subcycle reflects the properties of DBMSs where the HADP process model is used (cf. Section 2.6.2). According to the intention of the KDD process (cf. Section 1.6), the offline subcycle is used to facilitate and speed-up the extraction of knowledge from historical data sources.

The subdivision of the KDC is based on the assumption that automatic decisions are necessary to evoke actions in real-time, even though it is necessary to maintain a human expert in the loop. Hence, the KDC aims to provide automatic and semi-automatic data processing. Both subcycles are weakly coupled and work asynchronously. At certain times, a synchronization of these subcycles becomes necessary. A complete round trip of the KDC requires two synchronizations. For example, the first synchronization can be used for data transfer to the offline subcycle. The second synchronization can be used to submit information and knowledge to the online subcycle.

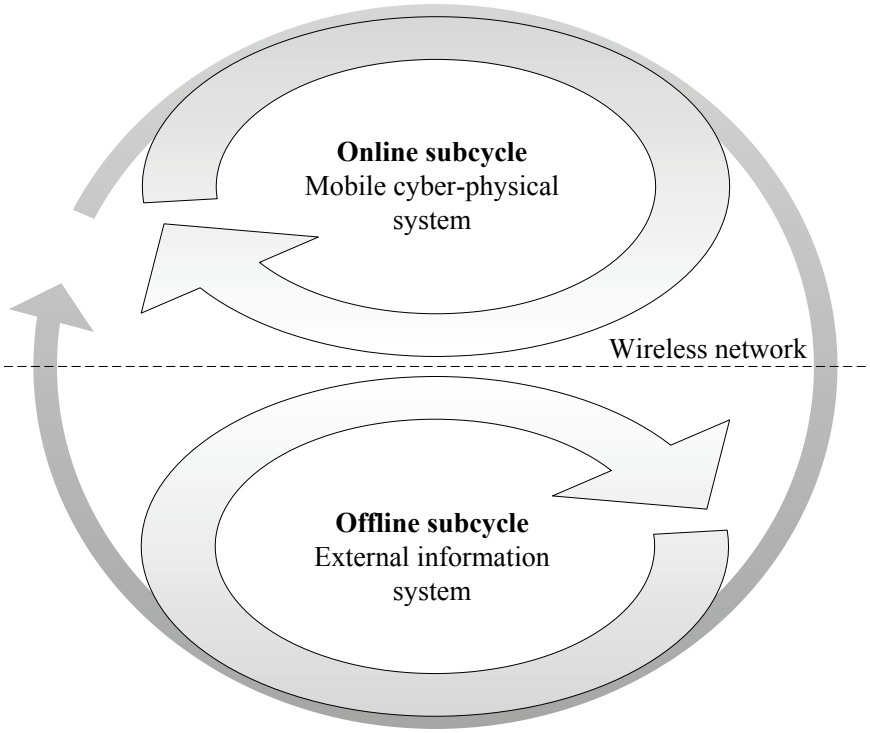

Figure 18: The Knowledge Discovery Cycle [NS13]

2. **Dynamic**: Sudden changes to the system behavior of MCPSs are unforeseeable and can occur at any time during operation (cf. Section 1.5). On that score, the KDC is a dynamic, iterative, and continuous process which reflects changes in the system behavior of MCPSs and data stream processing. It is necessary to adapt the online monitoring process to the changing system behavior during operation.

3. **Abstract**: The KDC is an abstract infrastructure where concepts and technologies only are used for description. The subsequent implementation of the KDC depends on a specific target system. Accordingly, algorithms which appropriately refer to the requested monitoring process can be selected. The KDC becomes very flexible and easily extensible due to the usage of concepts and technologies. Consequently, the KDC can be easily accommodated to the demands of specific monitoring situations and problems.

4. **Data-oriented**: The implementation of the KDC requires many different data storage and data access strategies which must be appropriate for the required application field. On one hand, an appropriate monitoring task is required to process data streams which represent transient data. On the other hand, these transient data must be stored in persistent data repositories. In most cases, the online subcycle reflects streaming data and a small amount of persistent data. The offline subcycle reflects persistent data and some streaming data. At the first round trips of the KDC (during test and operational phases), only a small amount of persistent or historical data exists. Over time, more and more persistent data are gathered. This continuously increasing amount of persistent data can be used for offline monitoring, long-term analysis, and knowledge discovery.

5. **Knowledge-oriented**: One of the main objectives of the KDC is to discover knowledge of an MCPS, the system environment, and the related system behavior during operation to increase knowledge continuously over time. This continuity implies a successive sequence of round trips of the KDC. Meanwhile, knowledge storage and access strategies are used intensively. The offline subcycle is mostly used to discover knowledge by the application of data mining (cf. Section 2.5). The discovered knowledge is consequently stored into persistent knowledge repositories. The online subcycle is mostly used to apply the previously discovered knowledge. For example, the application of knowledge onto the online subcycle can be implemented by means of rule sets. The applied knowledge is used for online monitoring in a real-time manner. In online monitoring, synopsis and summarized knowledge can be applied to decrease the processing overhead for the present system resources. The online subcycle can also be used for knowledge discovery. However, the application of knowledge discovery increases the processing requirements.

6. **Event-oriented**: Based on the discovered knowledge, the KDC is intended to detect events or complex events from continuously arriving data streams. Therefore, adequate event storage techniques are necessary. Additionally, it could also be useful to create event metrics (e.g. cockpits).

3.3 Relation between Key Challenges and KDC Characteristics

The aforementioned characteristics imply that the KDC recognizes the key challenges for monitoring MCPS (cf. Section 1.3). These include change, time dependence, continuity, data processing, and autonomy. Table 5 correlates the characteristics of the KDC with the key challenges for monitoring MCPS.

1. **Change**: The cyclic, dynamic, and abstract characteristics of the KDC refer to the key challenge 'change'. Due to consequent round trips of the KDC, it is possible to adapt the monitoring process to the changing system behavior dynamically and iteratively during operation. The KDC is independent from implementation and can be accommodated to the demands of specific monitoring situations and problems.

2. **Time dependence**: The cyclic and abstract characteristics of the KDC refer to the key challenge 'time dependency'. Apart from synchronizations, the online subcycle works automatically and is decoupled from human interactions. Thus, decisions and actions can be triggered in real-time. The KDC is independent from a specific implementation due to its abstractness, and it is possible to apply algorithms which respect time dependency.

3. **Continuity**: The cyclic and knowledge-oriented characteristics of the KDC refer to the key challenge 'continuity'. Expert knowledge can be continuously brought into connection with the current system behavior during operation. For example, expert knowledge can be used to satisfy such monitoring objectives (cf. Section 2.10.4) as failure and anomaly detection along with state change prediction.

4. **Data processing**: The data-, knowledge-, and event-oriented characteristics of the KDC refer to the key challenge 'data processing'. Data is stored persistently and can be used to discover knowledge. This knowledge can then be implemented to detect events.

5. **Autonomy**: The cyclic characteristic of the KDC refers to the key challenge 'autonomy'. The online subcycle of the KDC works autonomously, asynchronously, and is weakly coupled with the offline subcycle.

Key challenges	Cyclic	Dynamic	Abstract	Data-oriented	Knowledge-oriented	Event-oriented
	Characteristics					
Change	✓	✓	✓			
Time dependence	✓		✓			
Continuity	✓				✓	
Data processing	✓			✓	✓	✓
Autonomy	✓					

Table 5: Relation between key challenges and characteristics

3.4 Stream Model Extension

In this section, the aforementioned stream model (cf. Section 2.6.1) and the requirements for data stream classification (cf. Section 2.6.7) are discussed. Based on this discussion, a storage-aware stream model for data stream classification is introduced and training methods for data stream classification are identified.

3.4.1 Controversial Discussion

There exists a widely adopted assumption that data streams should not be stored entirely. A closer look on this assumption is provided below to discuss the controversies of this assumption in more detail. Amongst others, the following three statements, which were noted by leading experts in the data stream processing research area, can be found in the literature.

1. As stated by Gama: "It is impractical to store all data to execute queries that reference past data. These types of queries require techniques for storing summaries or synopsis information about previously seen data." [Gam10, p. 9].
2. An example, presented by Cugola et al., relates to fire detection: "[...] there is no need to store sensor readings if they are not relevant for fire detection [...]" [CM12, p. 15:2].

3. In accordance to the requirements for data stream classification (cf. Section 2.6.7), Bifet et al. state: "The amount of data that has arrived and will arrive in the future is extremely large; in fact, the sequence is potentially infinite. Thus, it is impossible to store it all. Only a small summary can be computed and stored, and the rest of the information is thrown away. Even if the information could be all stored, it would be unfeasible to go over it for further processing." [BHKP11, p. 109].

The first statement discusses queries and algorithms which are used for data stream processing. However, monitoring of MCPSs requires storage of data streams (e.g. by means of SDWs) and application of algorithms which generate classification models. These classification models provide summaries or synopsis information about historical data and can subsequently be used for data stream processing.

The second statement declares that sensor readings do not have to be stored. However, considering the monitoring objectives (cf. Section 2.10.4), it is necessary to analyze the root cause of the occurred fire event. Currently, it is impossible to provide an in-depth analysis directly on MCPSs due to resource restrictions and the absence of human experts. Consequently, only local monitoring of current system states can be applied aboard an MCPS, whereas global monitoring (cf. Section 2.10.2) can only be provided by analyzing historical data. In-depth analysis requires reasonable computing resources and interaction with human experts. Hence, historical data are a prerequisite for in-depth analysis, and it is necessary to store sensor readings for further processing.

The third statement shows two main concerns. The first concern states that data streams cannot be stored entirely and are discarded. The second concern states that it would be unfeasible to go over the stored data for further processing. These concerns contradict the functioning of monitoring MCPSs, such as aboard the ISS Columbus module (cf. Section 1.4). The presented example shows that data streams are stored in mission archives. Historical data are a prerequisite for long-term failure analysis and for planning long-term corrective actions. Furthermore, these concerns contradict the new and upcoming research area of big data [Jac09]. There is a demand for tools to process big data, and many practical solutions to process big data, such as cluster [BM06], grid [FK03], or cloud computing [AFG+10], are already available.

3.4.2 Storage-aware Stream Model

The assumption that data streams should not be stored in their entirety holds for some application domains, but not for all. Accordingly, a storage-aware stream

model is introduced which aims to extend the current stream model (cf. Section 2.6.1). The storage-aware stream model combines KDDS with KDD for monitoring MCPSs.

In general, monitoring MCPSs is a semi-automatic process. For example, the ISS Columbus failure management system (cf. Section 1.4.1) dispatches incoming data streams almost completely (down-sampled) to an external information system (ground control) to provide historical data (mission archive). However, complete storage is very expensive. Amongst other reasons, human experts are responsible for related decisions and consequent actions. With respect to the aforementioned monitoring objectives (cf. Section 2.10.4), historical data are necessary for long-term failure analysis, adequate monitoring, and to avoid failures which have occurred in the past. The KDD process is used to facilitate knowledge discovery with the support of human experts. The KDDS process is used to process continuous data streams automatically. Thus, it is necessary to consider KDD and KDDS as a combination and to build links between both research areas [NS13].

Based on the KDC, the storage-aware stream model encompasses the following differences to the existing stream model (cf. Section 2.6.1):

1. The online subcycle provides enough memory to manage a defined number of arrived data items.
2. The offline subcycle provides enough memory to store arrived data items entirely.
3. Data stream mining algorithms account for online training, offline training, or hybrid training (cf. Section 3.4.3).

The first difference enables the opportunity to collect a set of data items before further processing. Consequently, it is possible to control the order of the arrived data items, and it becomes feasible to process data items separately or a as set of collected data items by pooled computation. This aspect is already partly addressed by stream windows (cf. Section 2.6.6).

The second difference enables the opportunity to store the arrived data items in their entirety. It becomes possible to provide long-term analysis. Regarding the importance and age of the data, it is possible to subsequently decide which data items must be deleted, down-sampled, or remain unchanged.

The third difference enables the opportunity to apply different training methods depending on the demanded knowledge discovery objectives.

This results in the following definition of the term 'storage-aware stream model' in the context of the present thesis for monitoring MCPSs.

Definition 3.1 (storage-aware stream model):
The storage-aware stream model extends the commonly known stream model by the ability of storing incoming data streams entirely.

3.4.3 Training Methods

Classification presupposes a training phase of a classifier (cf. Section 2.5.2). The requirements for data stream classification (cf. Section 2.6.7) do not include specific training methods. Accordingly, the author introduces three training methods for data stream classification: *online*, *offline*, and *hybrid* training. This section is closely related to the author's publication [NS13].

Online Training

The requirements for data stream classification (cf. Section 2.6.7) refer to online training. Online training neglects the existence of external information systems. According to the definition of the stream model (cf. Section 2.6.1), it is assumed that data streams cannot be stored in their entirety. Online training provides the ability for simultaneous training and classification during operation. Algorithms which refer to online training are online adaptive. For this purpose, online training methods refer to one-pass algorithms only while a small window-based set of training data is available. User interaction is usually impossible in order to provide fast algorithms. Hence, the verification by human experts is very difficult, and the accuracy of the resulting classifier is proved insufficient. Online training reflects online knowledge discovery without any previously extracted knowledge and without any previous round trips of the offline subcycle or long-term analysis.

Offline Training

Offline training benefits from the existence of external information systems which provide many system resources. Thus, resource intensive algorithms can be applied on the offline subcycle while a large set of historical training data is available. Classifier training can be very resource intensive, while the online resources are not adversely affected. Additionally, the resulting classifiers can be easily assessed by human experts. Accordingly, both subcycles of the KDC have to be synchronized from time to time. The online subcycle is used to apply the trained classifiers. Computational resources of the online subcycle are exclusively used for classification and decision making. Hence, classification becomes very fast and resource-saving. However, it becomes necessary to prepare the incoming data streams for persistent storage.

Hybrid Training

Hybrid training benefits from both aforementioned training methods. Previously extracted knowledge can be used to train the required classifiers. Consequently, these classifiers are online adaptive and can be changed during runtime without synchronization between both subcycles of the KDC.

3.5 KDC Processing Steps

As depicted in Figure 19 on page 56, a distinction is made between four types of processing steps. These include *repository management*, *preprocessing*, *transfer*, and *analysis*. Repository management involves data streams and transient or persistent storage techniques. Processing steps for repository management are depicted by means of cylinders. Preprocessing steps, represented by rhombuses, include data, knowledge, or event preprocessing. Transfer preprocessing steps, represented by ellipses, include the transfer of data, knowledge, or events from one subcycle to another. Analysis processing steps, represented by squares, include stream analysis and analysis from persistent repositories.

Afterwards, twelve processing steps are identified, and each processing step is discussed in more detail. The identification starts with the online subcycle followed by the offline subcycle. This section is closely related to the author's publication [NS13] (see also [NS12a, NS12b, NSS13b]).

3.5.1 Online Processing Steps

This section identifies processing steps of the online subcycle which relates to online monitoring. Figure 19 on page 56 summarizes the identified processing steps.

1. **Streaming inputs**: According to the reference architectures of DSMS (cf. Section 2.6.3) and IFP engines (cf. Section 2.6.4), the first processing step reflects the streaming inputs approaching from heterogeneous data sources. As depicted in Figure 19 on page 56, data stream items arrive continuously and asynchronously at the KDC.

2. **Preprocessing**: This processing step includes the extraction of relevant substreams or specific data stream items. Furthermore, it also includes transformations such as dimensional reduction, noise reduction, scaling, time stamp standardization, and correlation.

3. **Online analysis**: The third processing step is the core of the online subcycle and the online monitoring process. Online analysis is intended to identify

the current system behavior and to detect failure situations of the target system. Therefore, data stream processing algorithms are applied which follow the storage-aware stream model (cf. Section 3.4.2). Contrary to the KDD process, data stream processing algorithms provide continuous queries (cf. Section 2.6.2). Hence, the knowledge discovery objectives must be defined in advance before algorithms are applied. Subsequent changes of the knowledge discovery objectives can be considered by the different training methods (cf. Section 3.4.3).

This and the following processing steps are associated with a repository. It is used to store queries, rules, or data stream processing algorithms. Accordingly, it refers to the functionalities provided by the local storage along with the query repository of DSMSs (cf. Section 2.6.3) and by the knowledge base of IFP engines (cf. Section 2.6.4).

4. **Actions**: As depicted in Figure 19 on page 56, the fourth processing step is directly related to online analysis. The deduction of decisions or events necessitate the initiation of actions as known from the ECA paradigm. A distinction is made between *hardware actions* and *software actions*. However, hardware and software actions can also be triggered in combination.

   - Hardware actions, such as toggling of switches, effect hardware directly.
   - Software actions do not directly effect hardware. Amongst their many functions, they can be used for message sending, alarm triggering, or to provoke storage mechanisms. Software actions could also include the preparation of metrics and can be used to initiate the synchronization of both subcycles.

5. **Temporal storage**: The fifth processing step includes temporal storage on the online subcycle. This requires an integration of the arrived data items into an existing data schema. The temporal storage is a window-based (cf. Section 2.6.6), preliminary, and short-term storage. It includes the storage of down-sampled data from data streams as well as the storage of events and metrics. For example, metrics could include the counting of specific events or frequency of arriving data items. The author distinguishes between two fundamental storage strategies: *complete* and *incomplete storage*.

   - Complete storage refers to persistent data storage and assures that data items are not deleted. Several applications, such as the ISS Columbus failure management system (cf. Section 1.4.1), presupposes complete storage.

- Incomplete storage refers to transient data storage strategies and does not assure complete storage. Hence, data and event sketches or samples of the down-sampled data can be used. Previously stored data can be overwritten or deleted if necessary. Incomplete storage can be an adequate storage strategy due to restricted system resources.

The cyclic process model branches after the fifth processing step. This branch demonstrates that the online subcycle and the real-time monitoring process are asynchronous and decoupled from the offline subcycle and from long-term analysis. The online subcycle provides an iterative loop where temporally stored data and metrics can be applied for online training (cf. Section 3.4.3).

6. **Data and event transfer**: The sixth processing step is the last step of the online subcycle. It is used for synchronization and to transmit temporally stored data, events, or metrics from the online subcycle into a persistent memory on the offline subcycle. After the transfer is completed, stored data are deleted from the temporal storage. However, synopsis information, such as metrics, can be preserved on the temporal storage. This synopsis information can be applied for automatic adaptation of online and hybrid training algorithms for online adaptivity which are related to the online analysis processing step. Data transmission is optional and not always performed at each pass-through. It is triggered from time to time on condition that the external network is available. Amongst others, triggering factors can be near exhaustion of the temporal storage, change of the system behavior, or detection of anomalies.

3.5.2 Offline Processing Steps

This section discusses processing steps of the offline subcycle which relates to the offline monitoring process. Figure 19 on page 56 summarizes the identified processing steps.

7. **Persistent storage**: The offline subcycle starts with a data source. The persistent storage is the basis for offline monitoring and long-term analysis. It includes necessary, compacted, and integrated data as well as events and metrics. This processing step also takes the quality of the stored data into account [Ols02]. As depicted in Figure 19 on page 56, it is also possible to import data from external data sources.

8. **Preprocessing**: The eighth processing step reflects the selection and transformation of relevant data and events as known from the KDD process model (cf. Section 2.4). This preprocessing step is supported by human experts and is used to define the knowledge discovery objectives for both offline and online

analysis. On one hand, knowledge discovery objectives for offline analysis are required in case of root cause or long-term analysis. On the other hand, knowledge discovery objectives are required to create or update models for online analysis appropriately.

9. **Offline analysis**: The ninth processing step constitutes the core of the offline subcycle and offline monitoring. Offline analysis is well-known from the KDD process model. It is used to discover knowledge by means of data mining and preprocessed data. The offline subcycle works semi-automatically and is controlled by human experts.

10. **Validation**: The tenth processing step is directly related to offline analysis. This validation step can be used by human experts to interpret the discovered knowledge and to evaluate the results of analysis.

11. **Knowledge storage**: The knowledge storage includes three tasks. First, newly derived knowledge must be stored, combined, and integrated with already existing knowledge. Second, the knowledge is then translated into query language expressions to obtain rule sets. Third, human experts should annotate the derived knowledge with additional information. A distinction can be made between two operational methods: *evolutionary method* and *re-launch method*.

    - The 'evolutionary method' is the most common operational method and is also an evolutionary process. The evolutionary process refers to the 'normal wear phase' of the aforementioned bathtub curve (cf. Section 1.2) and is used to increase knowledge about the target system during the operational phase. It can be used to obtain an overview of existing knowledge and information. Newly derived knowledge can be used to evaluate pre-existing knowledge.

    - The 're-launch method' starts from scratch without any pre-existing knowledge. This operational method relates to the 'wear in phase' and 'wearout phase' of the the aforementioned bathtub curve (cf. Section 1.2) and can be used when the system behavior differs totally from pre-existing knowledge. For example, the re-launch method is useful in the initial round trip of the KDC directly after the 'dispatch and commissioning phase'. Moreover, this operational method can be useful when the failure rate significantly increases during the 'wearout phase'. After the re-launch method, it is possible to switch back to the evolutionary method in order to gather new knowledge over time again. For example, this is necessary after the transition from the 'wear in phase' to the 'normal wear phase'.

The cyclic process model branches after the eleventh processing step. This branch indicates that the offline subcycle is asynchronous, decoupled, and independent from the online subcycle. The offline subcycle provides an iterative loop to adapt or refine discovered knowledge.

12. **Knowledge transfer**: The twelfth processing step is the last step of the offline subcycle and the KDC. It is used for synchronization and to transmit knowledge from the offline subcycle to the online subcycle. This processing step reflects reconfiguration, refinement, or adaptation of the online monitoring process and the online subcycle. Knowledge transfer can affect the following processing steps of the online subcycle: preprocessing (2), online analysis (3), actions (4), temporal storage (5), and data or event transfer (6).

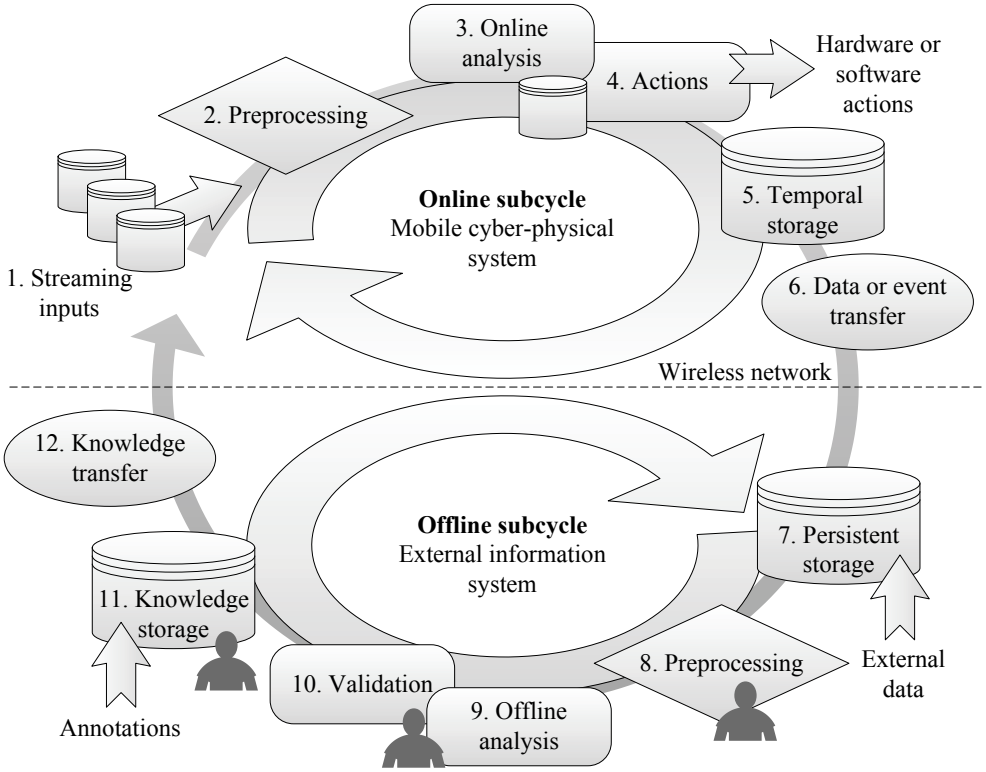

Figure 19: Processing steps of the KDC (based on [NS13])

3.6 KDC Concept Assignments

The KDC is not intended to surrogate the aforementioned monitoring types (cf. Section 2.10.3). These monitoring types have their justifications. For example, model-based monitoring and limit monitoring are necessary to provide basic reliability of an MCPS during the normal wear phase (cf. Section 2.10.3). However, it is hardly possible to apply model-based monitoring and limit monitoring during the wear in and wear out phases. The KDC is intended to be an additional monitoring approach which combines and improves these monitoring types for monitoring MCPSs. Therefore, KDD is used to build a preliminary model during the design and test phases (e.g. by means of a prototype). This preliminary model defines limits and reflects model-based monitoring and is subsequently translated into rule sets. These rule sets represent the preliminary knowledge about the MCPS, the system environment, and the assumed system behavior. Rule sets are applied onto an MCPS for online monitoring by means of KDDS or IFP. This reflects one-dimensional and n-dimensional limit monitoring. Based on the aforementioned 'storage-aware stream model' (cf. Section 3.4.2), more and more data is collected and transmitted to the offline subcycle during operation. The offline subcycle is used for knowledge discovery which is used for revision, adjustment, and adaptation of the existing model. The refined model can be translated into rule sets again and can be used for the refinement of pre-existing rule sets on the MCPS. Hence, the KDC is a dynamic and continuous process which includes revision, adjustment, and adaptation during operation. KDD, KDDS, and IFP provide an appropriate foundation to build a flexible, adaptable, dynamic and continuous monitoring process.

The contribution is twofold. First, the KDC, including processing steps for a combined KDD and KDDS process model, has been defined. In addition, this section assigns the previously identified processing steps to existing concepts which are known from literature. The assignment is intended to provide a uniform arrangement. This consideration starts with the offline subcycle and is followed by the online subcycle. The assignment of concepts is depicted in Figure 20. This section is closely related to the author's publication [NS13] (see also [NS12a, NS12b, NSS13b]).

3.6.1 Concepts of the Online Subcycle

This section assigns concepts to the identified processing steps of the online subcycle which relates to online monitoring.

i. **Data stream management**: The first and second processing steps are grouped together. This group is a fundamental setup for data stream management. Transformations (e.g. noise reduction, time stamp standardization,

or scaling) can be adequately applied by existing data stream query language expressions which are provided by IFP engines.

ii. **Data stream mining/IFP**: The third and fourth processing steps are assigned with data stream mining and IFP. As previously mentioned, data stream analysis is an automatic process and previously derived rules sets can be translated into data stream query language expressions. IFP engines are intended to derive complex events and it is intended to invoke automatic decisions along with triggering actions.

iii. **SDW**: The fifth processing step is associated with an SDW which is used to temporally store the arrived data items. Such an SDW can also be applied as an upstream staging area which integrates the arrived data items into an existing data schema.

iv. **ETL**: The sixth processing step initializes the offline subcycle. An ETL process is required to synchronize both subcycles. However, it is impossible to make an explicit distinction between KDD and KDDS at that point.

3.6.2 Concepts of the Offline Subcycle

This section assigns concepts to the identified processing steps of the offline subcycle which reflects offline monitoring.

v. **SDW**: The seventh processing step is associated with an SDW. It is required to store the arriving data items persistently for further processing. However, it is impossible to make an explicit distinction between KDD and KDDS at that point.

vi. **Data mining and machine learning**: Data mining and machine learning is used to discover knowledge semi-automatically from data, events, and metrics which are provided by an SDW. This concept assignment refers to preprocessing, offline analysis, and evaluation. Data mining and machine learning as well as evaluation are controlled by human experts. Since the KDC is a cycle, this also includes concepts which are known from KDDS. For instance, resource intensive IFP engines such as MOA [BHKP10], Esper [Esp13], or StreamBase [Str13]. Furthermore, it is also possible to apply data stream mining algorithms to provide near real-time analysis of DWH or SDW updates.

vii. **Archiving**: This concept assignment refers to knowledge storage and includes the integration of evaluated knowledge into a persistent repository. For example, the storage of the integrated knowledge can be implemented by an already existing SDW. Additionally, this concept assignment also includes the

representation of knowledge by means of standard CEP vocabularies or semantic ontologies [PVS11]. Moreover, this concept assignment also refers to the annotation of the discovered knowledge by human experts. Further on, it is required to translate the discovered knowledge into specific query language expressions.

viii. **Adaption**: The adaption is used to refine the online subcycle and refers to the transmission of the previously discovered knowledge and the resulting queries to the online subcycle. Adaption reflects offline or hybrid training (cf. Section 3.4.3).

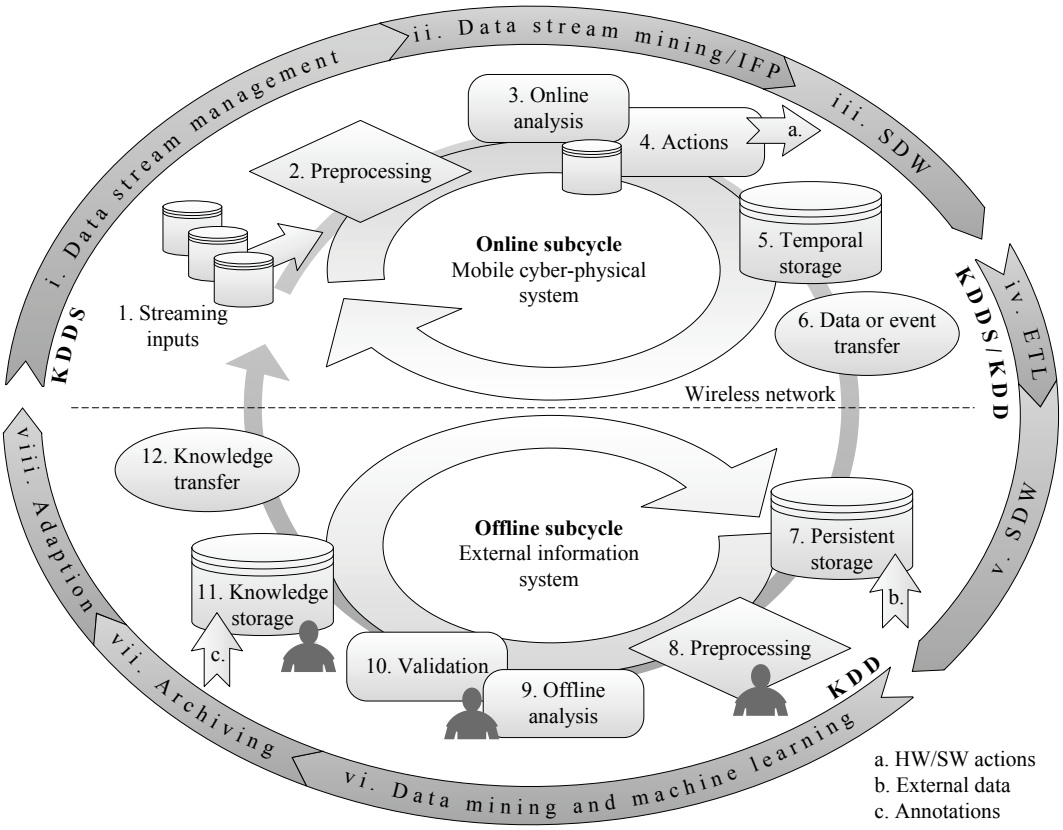

Figure 20: The KDC with associated concepts (based on [NS13])

3.7 Comparison with Related Work

This section is intended to compare the aforementioned approaches: MOA data stream classification cycle (cf. Section 2.7), expert systems (cf. Section 2.8), and the MAPE-K reference model (cf. Section 2.9) with the key challenges for monitoring MCPSs (cf. Section 1.3). Finally, this comparison is summarized.

3.7.1 MOA Data Stream Classification Cycle

The MOA data stream classification cycle is not a priori a monitoring system, but it can be used for monitoring. The key challenge 'change' is considered by the MOA data stream classification cycle. The MOA data stream classification cycle is intended to adapt the present model whenever a new item has been caught from the arriving data stream. The MOA data stream classification cycle considers the key challenge 'time dependence'. Algorithms which are capable of working in real-time can be applied. The MOA data stream classification cycle is undoubtedly a continuous cycle and does not recognize the key challenge 'continuity'. However, the intention of the key challenge 'continuity' is the integration of expert knowledge into the monitoring process continuously during operation. This is not provided by the MOA cycle. The key challenge 'data processing' is not considered by the MOA data stream classification cycle. The intention of the key challenge 'data processing' is the processing of transient and persistent data. This is categorically ruled out by the MOA data stream classification cycle. The key challenge 'autonomy' is considered by the MOA data stream classification cycle because it is intended to work autonomously.

3.7.2 Expert Systems

Expert systems are not a priori monitoring systems. However, it is possible to use an expert system for monitoring. The key challenge 'change' is considered by expert systems. Expert systems provide possibilities to adapt the monitoring process during operation. This could be done by updating the knowledge base. The key challenge 'time dependency' is not directly considered by expert systems. Under certain circumstances, time-critical decisions cannot be appropriately made due to the internal structure and the complex interrelations between the knowledge base and the inference engine. Expert systems recognize the key challenge 'continuity'. It is potentially possible to bring expert knowledge into connection with the current system behavior during operation. Expert systems do not consider the key challenge 'data processing'. The reason for this appraisal lies in the internal structure of expert systems. Data streams which arrive at an expert system have to be integrated into the knowledge base. Subsequently, facts and rules which are already available have

to be updated continuously. Such updates modify the knowledge base significantly. On that score, a clear distinction between transient and persistent data cannot be recognized. The key challenge 'autonomy' is considered by expert systems. Existing rules can be automatically applied by the inference engine.

3.7.3 MAPE-K Reference Model

The MAPE-K reference model is an autonomic computing architecture which can be used for monitoring. However, similarities between expert systems and the MAPE-K reference model can be recognized. The MAPE-K reference model considers the key challenge 'change'. The acquired sensor readings are used to adapt the autonomic manager. The key challenge 'time dependence' is not considered by the MAPE-K reference model. The reason for this appraisal lies in the internal structure of the MAPE-K reference model. Under certain circumstances, time-critical decisions cannot be appropriately made due to the complex structure of the autonomic manager. The MAPE-K reference model does not consider the key challenge 'continuity'. The integration of expert knowledge during operation would interrupt the autonomic manager. The MAPE-K reference model does not consider the key challenge 'data processing'. It is unclear how the data or the knowledge is integrated by the autonomic manager. On that score, a clear distinction between transient and persistent data cannot be recognized. The key challenge 'autonomy' is recognized by the MAPE-K reference model because it is intended to work autonomously.

3.7.4 Summary

Table 6 summarizes the comparison of the existing approaches with the key challenges for monitoring MCPSs (cf. Section 1.3). As this table implies, the existing approaches entail deficits for monitoring MCPSs. On that score, the KDC attempts to eliminate these deficits (cf. Section 3.2) and to provide an appropriate monitoring approach.

3.8 Conclusion

As described in Section 1.7 on page 13, this thesis aims to combine KDD with KDDS for monitoring MCPSs. Therefore, the development of the KDC has been described in this chapter. Furthermore, requirements for monitoring MCPSs (cf. Section 3.1) have been identified, characteristics of the KDC (cf. Section 3.2) have been outlined, and the key challenges for monitoring MCPSs have been compared with the characteristics of the KDC (cf. Section 3.3). Moreover, a storage-aware stream model (cf. Section 3.4.2), which extends the pre-existing stream model (cf. Section 2.6.1) and combines KDD with KDDS for monitoring MCPSs, has been

Key challenges	Approaches: MOA Cycle	Expert systems	MAPE-K	KDC
Change	✓	✓	✓	✓
Time dependence	✓	✗	✗	✓
Continuity	✗	✓	✗	✓
Data processing	✗	✗	✗	✓
Autonomy	✓	✓	✓	✓

Table 6: Comparison of the key challenges with existing approaches and the KDC

introduced. Additionally, processing steps for the KDC (cf. Section 3.5) have been identified, and the KDC has been associated with existing concepts (cf. Section 3.6). Finally, the KDC has been compared with related work. As a result, the KDC is a cyclic, dynamic, and abstract arrangement of data processing concepts and attempts to eliminate existing deficits of pre-existing approaches.

CHAPTER 4

Multi-Class Data Stream Anomaly Detection

> "*The positive heuristic of the programme saves the scientist from becoming confused by the ocean of anomalies.*"
>
> *Imre Lakatos (1922-1974)*

ANOMALY detection is an interdisciplinary and widely disseminated research area. In the literature, terms such as *outlier detection* [RG97] or *novelty detection* [Bis93] are used synonymously with the term 'anomaly detection'. The main emphasis of anomaly detection is to find data subsets or single data items from a set of data that do not conform to expected behavior. Algorithms for anomaly detection are commonly borrowed from research areas such as statistics, machine learning, data mining, or KDD [CBK09] (see also [MS03a, MS03b, HA04, PP07]).

At the beginning, this chapter categorizes conditions of MCPSs into specific *system states* [NS12b]. Further on, the basic anomaly detection model used in the present thesis is introduced. Moreover, techniques for anomaly detection are presented. Based on this, the problem of anomaly detection, subsequently exemplified by means of the ISS Columbus air loop (cf. Section 1.4.2), is stated. To develop a solution for the stated problem, a novel data stream anomaly detection algorithm is presented which attempts to minimize the average time consumption for a multi-class data stream anomaly detection approach. Additionally, the algorithm is discussed from the viewpoint of filtering and a set of selected one-class classifiers which can be used for anomaly detection are described in detail. Finally, this section is concluded.

4.1 System States

It is necessary to look at the general term 'system states' in more detail. As stated by Gordon [Gor72], the term 'system states' is used to describe the conditions of a system under consideration of all system components, attributes, and activities within a specific time period. As described by Kecher [Kec05], the term 'system states' defines situations in which certain and precisely defined conditions hold. However, MCPSs are very complex systems and comprise many system states during the life-time phases (cf. Section 1.2). This results in the following definition of the general term in the context of the present thesis. Figure 21 illustrates the fine-grained division of system states. This section is closely related to the author's publication [NS12b].

Definition 4.1 (system states):
The general term 'system states' abstracts the conditions of an MCPS in order to abstractly describe the system and its components under consideration of relevant and well-defined attributes within a specific time period.

Further on, the author of the present thesis subdivides the general term into *normal system states* and *abnormal system states*. Normal system states are system states that can be interpreted as correct service of an MCPS or a target system during operation. Abnormal system states are complementary to normal system states, and it is assumed that an MCPS works incorrectly during operation.

The main emphasis of monitoring MCPSs is on the detection of current system states. However, the aforementioned division is very preliminary and coarse-grained. Hence, it is necessary to provide a more detailed division of the general term using pre-existing knowledge. Consequently, the author of the present thesis subdivides the terms 'normal system states' and 'abnormal system states' into *known system states* and *unknown system states*, respectively. Figure 21 illustrates this subdivision. Known system states represent knowledge, whereas unknown system states represent the unawareness about an MCPS. This results into a set of four categories of system states. For 'normal system states', these include *normal known system states* and *normal unknown system states*. For 'abnormal system states', these include *abnormal known system states* and *abnormal unknown system states*. This subdivision is discussed more precisely in the following descriptions.

4.1.1 Default and Novel States

The term 'normal known system states' is replaced with the term *default states*. If the system is in a default state, it is assumed that the system works correctly

and knowledge about the system behavior exists. Further on, the term 'normal unknown system states' is replaced with the term *novel states* hereinafter. If the system reaches a novel state, it is assumed that the system works correctly, but no knowledge about the system behavior exists. For example, novel states can occur due to system reconfiguration or as a result of the replacement of system components.

4.1.2 Irregular, Error, and Anomaly States

The term *irregular states* is used synonymously with the term 'abnormal unknown system states'. If the system reaches an irregular state, no knowledge about the system behavior exists . Moreover, the term *error states* is used synonymously with the term 'abnormal known system states'. If the system reaches an error state, it is assumed that the system works incorrectly and knowledge about the faulty system behavior exists.

As depicted in Figure 21, it is not always possible to differentiate between 'novel states' and 'irregular states'. Accordingly, novelty and anomaly detection techniques are often synonymously used in the literature [CBK09]. Thus, human experts must be involved in order to interpret if the current system state relates to a novel or to an irregular state. For the sake of clarity the term 'anomaly states' is used hereinafter to describe both situations.

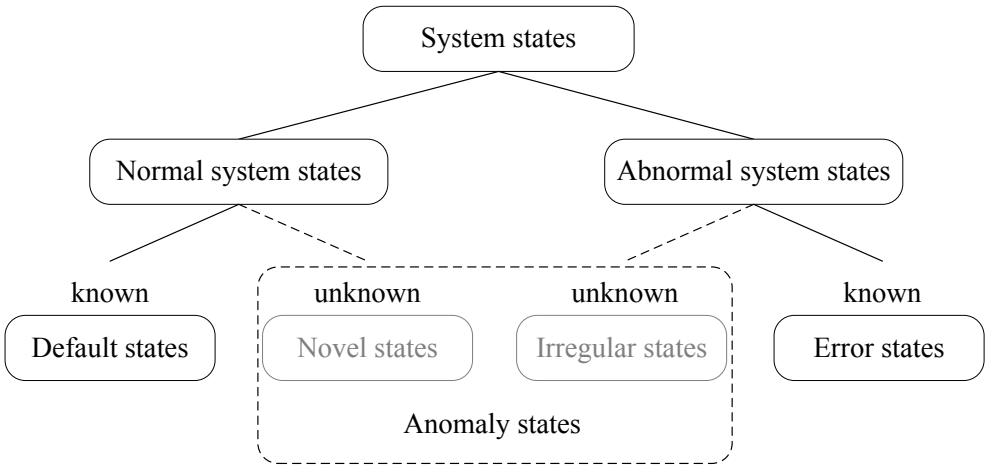

Figure 21: A set of system states (based on [NS12b])

4.2 Basic Anomaly Detection Model

A vector space $S = \mathbb{R}^n$ [Bel61] and the cluster assumption [See00] are the basic foundations of anomaly detection models. A vector space S is spanned by a set of $n \in \mathbb{N}$ mutually independent or orthogonal vectors which are represented by attributes of the target system $\{A_1, \ldots, A_n\}$. Thus, the variable n denotes the number of selected attributes. A multi-class anomaly detection problem is based on a set of known classes $\Omega = \{\omega_1, \omega_2, ..., \omega_h\}$ with $h \in \mathbb{N}$ that can be interpreted as bounded regions $\omega_i \subseteq S$ with $i = \{1 \,..\, h\}$. These classes represent expert knowledge about the system states of an MCPS. A multi-class anomaly detection problem corresponds to the detection of the anomaly class $\Omega^{\mathrm{C}} = S \setminus \bigcup_i \omega_i$. The anomaly class represents the unawareness about the system states. Attribute values are functions over time T, i.e. values of A_i with $i = \{1 \,..\, n\}$ are values of $a_i : T \rightarrow \mathbb{R}$. For the present thesis, the time is denoted as an index and not as an attribute. Consequently, state transitions are not considered, and an unlabeled data item (vector) with index $\tau \in T$ is represented as $\boldsymbol{s}_\tau = (a_{\tau,1}\ a_{\tau,2}\ ...\ a_{\tau,n})'$ where $\boldsymbol{s}_\tau \in S$ [NSS13a].

Figure 22 depicts a two-dimensional vector space which comprises two selected attributes: A_1 and A_2. Moreover, Figure 22 contains the three classes: ω_1, ω_2, and ω_3. Additionally, three anomalies are illustrated which are located outside of these three classes or bounded regions.

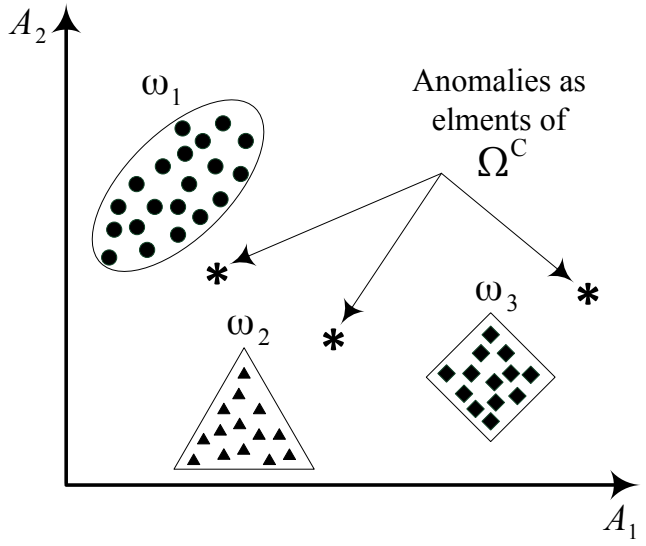

Figure 22: A multi-class anomaly detection problem (based on [CBK09])

4.3 Anomaly Detection Techniques

A variety of anomaly detection techniques exists. Amongst others, these include *classification-based*, *nearest neighbor (NN)-based*, *clustering-based*, and *statistical* anomaly detection [CBK09].

4.3.1 Classification-based

Classification-based anomaly detection operates in a similar fashion to classification (cf. Section 2.5.2). The general assumption states: "A classifier that can distinguish between normal and anomalous classes can be learned in the given feature space." [CBK09, p. 19]. As depicted on the left (i) in Figure 23, classification-based anomaly detection attempts to find minimal boundaries around the present classes.

Classification (cf. Section 2.5.2) can be interpreted as h-class classification. The variable 'h' denotes the number of known classes $h = |\Omega|$. An unlabeled data item is assigned to one of the present classes in any case. In contrast, classification-based anomaly detection can be interpreted as $h+1$-*class classification*. The extension '+1' represents the complement Ω^{C} or an anomaly state. Data items which cannot be assigned to the present classes are declared as anomalies. Classification-based anomaly detection techniques are commonly grouped into two categories: *MCC anomaly detection* and *OCC anomaly detection* [CBK09].

Multi-class Classification Anomaly Detection

As depicted on the left in Figure 23 (i), MCC anomaly detection techniques are based on the assumption that the training data contain labeled data items and that each data item belongs to exactly one of several known classes [DSSV00]. MCC anomaly detection techniques, which train a single classifier to distinguish between set of known classes and the complement, are called *homogeneous multi-class anomaly detection* hereinafter.

One-class Classification Anomaly Detection

In OCC anomaly detection, only one known class exist. As depicted on the right in Figure 23 (ii), OCC anomaly detection assumes that all data items of the training data entail only one class label. Accordingly, a discriminative boundary around the present class is trained using an OCC algorithm. An unlabeled data item that cannot be assigned to the trained boundary is an anomaly.

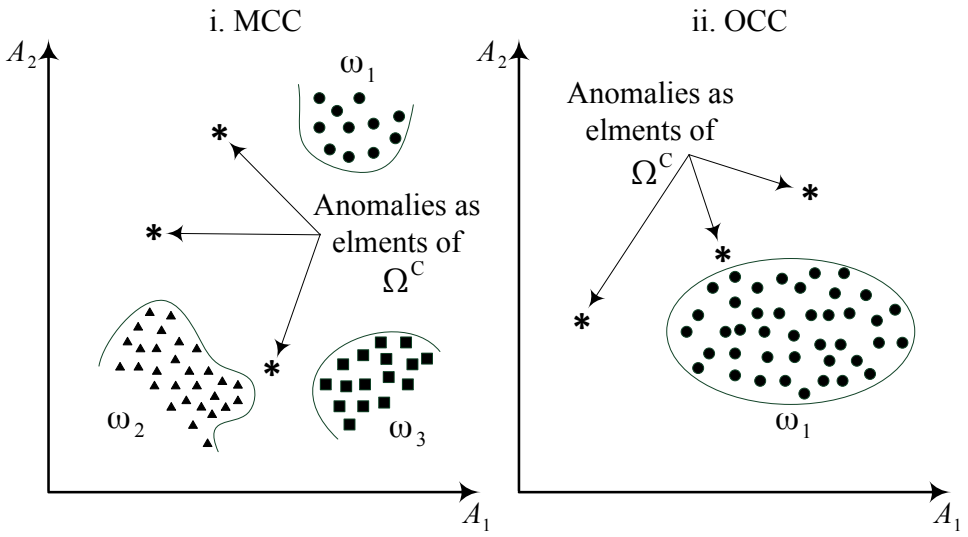

Figure 23: MCC (i.) versus OCC (ii.) anomaly detection (based on [CBK09])

4.3.2 Nearest Neighbor-based

As with others, NN-based anomaly detection techniques require a distance or similarity measure between two data items. The general assumption states: "Normal data instances [items] occur in dense neighborhoods, while anomalies occur far from their closest neighbors." [CBK09, p. 22]. There are different ways to compute the distance or similarity between two data items which are discussed later.

4.3.3 Clustering-based

Clustering-based anomaly detection techniques are grouped by means of two assumptions. These assumptions are discussed as follows.

- The first assumption states: "Normal data instances [items] belong to a cluster in the data, while anomalies do not belong to any cluster." [CBK09, p. 27]. Algorithms which refer to this assumption employ a clustering algorithm (cf. Section 2.5.2) to the data. A data item that cannot be assigned to one of the present clusters is declared as anomalous. A disadvantage of these algorithms is that they attempt to find clusters and are not optimized to find anomalies.
- The second assumption states: "Normal data instances [items] lie close to their closest cluster centroid, while anomalies are far away from their closest cluster

centroid." [CBK09, p. 27]. Anomaly detection techniques which rely on this assumption consist of two steps. First, the data is clustered and second, for each data item the distance to its closest cluster *centroid* is calculated as its anomaly score.

4.3.4 Statistical

For statistical anomaly detection, a statistical model (usually by known system states) is generated using the present data. This model is used to apply a statistical inference test to determine if an unlabeled data item fits to it or not. Data items which are subject to a low probability of being generated from the trained model are declared as anomalies. The general assumption states: "Normal data instances [items] occur in high probability regions of a stochastic model, while anomalies occur in the low probability regions of the stochastic model." [CBK09, p. 29].

4.4 Data Stream Anomaly Detection Algorithms

This section aims to provide a brief overview on existing data stream anomaly detection algorithms. As mentioned earlier, a widely recognized survey on anomaly detection is provided by Chandola et al. [CBK09]. To the best of the author's knowledge, currently a widely recognized survey on data stream anomaly detection does not exist. However, preliminary surveys, such as the survey contributed by Singh et al. [SS13], exist.

An anomaly detection approach, which is called *fast rank-Adaptive row-householder subspace tracking* [dSTM10], is a rank-adaptive algorithm for fast principal subspace tracking. It works in an unsupervised manner and is used to identify anomalies in streaming data of low dimensions. Therefore, the subspaces are built using dimensionality reduction. Observed data that cannot be sufficiently explained by the current model is considered anomalous.

Another anomaly detection approach is called *online novelty and drift detection algorithm (OLINDDA)* [SdLFdCG07]. OLINDDA implements a cluster-based approach for detecting novel classes and gradual change. By default, *OLINDDA* works in an unsupervised manner and uses the *k-means clustering algorithm* [Bis06] to identify unknown clusters. The OLINDDA algorithm is an offline training method and is online adaptive.

A *very fast decision tree for one-class classification of data streams* has been described by Li et al. [LZL09]. During the training phase, this algorithm constructs a tree forest, and then the best tree is chosen as the final output classifier. This approach is an extension of *Hoeffding trees (HT)* [DH00]. The HT approach is an induction algorithm for decision trees. The HT approach is an online training method

which is trained incrementally and aims to process high-speed data streams. This algorithm is based on the assumption that the probability distribution of the data items, which refer to the present classes, do not change over time. In the literature, HT are also known as very fast decision trees.

An anomaly detection method which uses ensembles of streaming *half-space trees (HST)* has been described by Tan et al. [TTL11]. A HST is a binary tree where each node is used to capture a number of data items within a particular subspace of the data stream. The HST method aims to be a fast one-class anomaly detector for evolving data streams and requires only 'normal' data, which excludes the anomaly class, for training and retraining of the anomaly detection model. The used one-class classifier separates the vector space by means of hyperplanes which are parallel to the axes. It is an online training method which is also online adaptive (cf. Section 3.4.3). The model is retrained continuously at the end of a window with a specific size (cf. Section 2.6.6). However, multi-class classification is not supported.

Another algorithm is called multi-class learning algorithm for data streams (MINAS) [FGC13]. Amongst others, the MINAS algorithm takes data stream multi-class classification and novelty detection into account. This algorithm provides two training phases: online and offline. The offline training phase is executed only once and is used to train the decision model. Therefore, the k-means clustering algorithm is used in order to train the decision model. The online training phase is used to classify a data item either as one of the known classes or as unknown. If a previously defined number of data items is classified as unknown, the MINAS algorithm applies a clustering algorithm in order to discover new classes.

Masud et al. [MGK$^+$11] have described an algorithm which takes classification and novel class detection into account. This approach is applied on two different classification techniques: decision tree and k-nearest neighbor. The algorithm is an online training method, and the initial classifier is trained by a chunk of labeled data items. In order to provide novel class detection, this algorithm entails several buffers. The algorithm tries to classify an unlabeled data item. Data items which cannot be classified immediately are stored in a buffer for future processing. Already classified data items are used to retrain the model.

4.5 Example - ISS Columbus Air Loop

Concerning the aforementioned real world scenario (cf. Section 1.4), Figure 24 depicts a data set that relates to the IRFA with three selected attributes: speed, current, and pressure (cf. Section 1.4.3). The data set comprises seven known classes $\Omega = \{\omega_1, \omega_2, ..., \omega_7\}$ and one anomaly class Ω^C. Each of the known classes is associated with a specific system state (cf. Section 4.1) of the IRFA. The values in

the round brackets denote the number of data items. The values of the attributes are ratio scaled (cf. Section 2.2.2), and the data set was previously clustered by human experts. To obtain an appropriate training data set, the data is preprocessed, normalized, scaled, and relevant state vectors are selected.

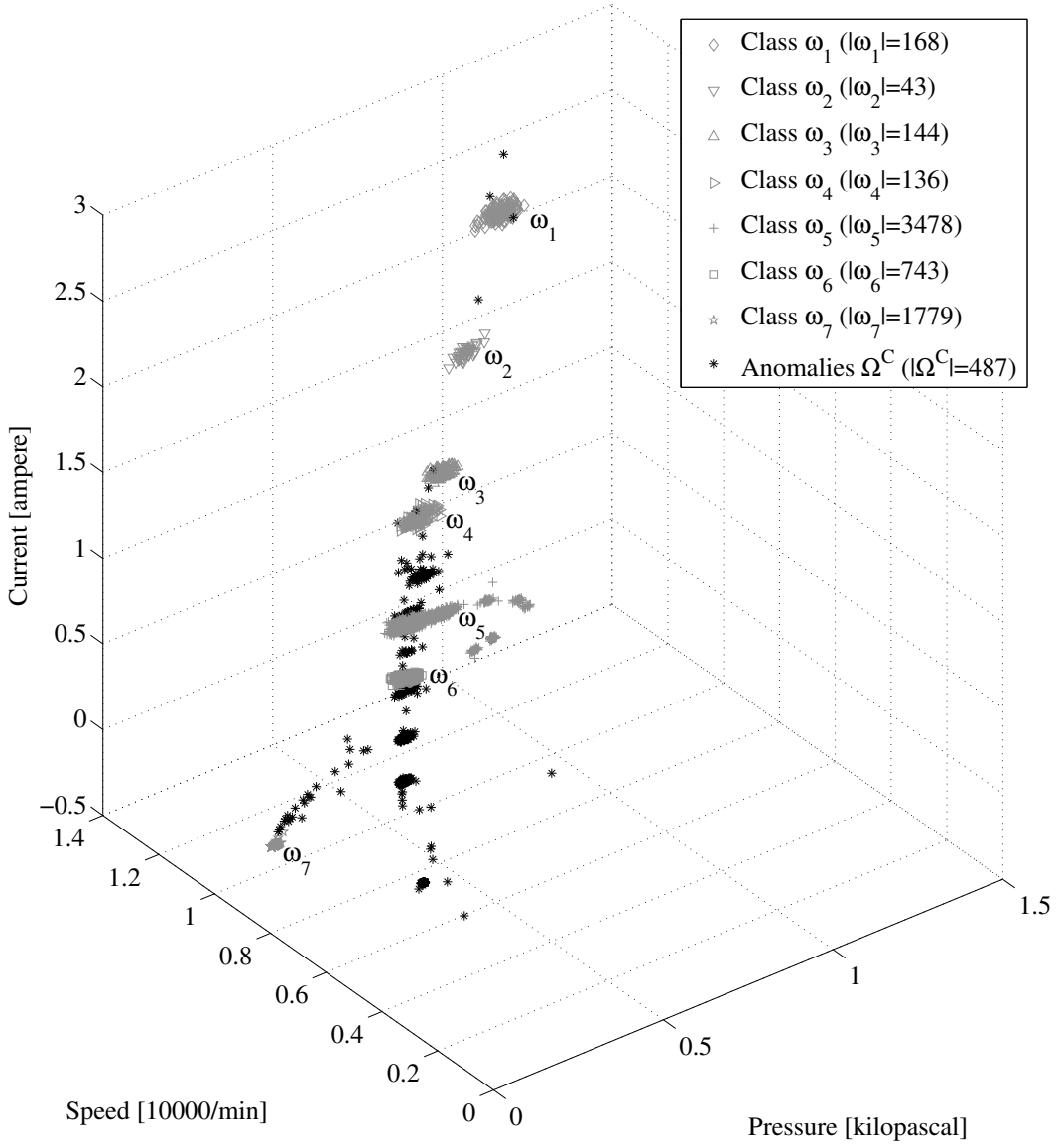

Figure 24: IRFA training data

For example, class ω_1 relates to an error state where the speed of the IRFA is unusually increased. Class ω_5 refers to a default state where the IRFA works as expected. Class ω_7 describes another default state where the IRFA is turned off. Obviously, the shapes or geometrical interpretations of these classes are very heterogeneous.

4.6 Problem Statement

According to the key challenges of monitoring MCPSs (cf. Section 1.3), the monitoring objectives (cf. Section 2.10.4), and the requirements for monitoring MCPSs (cf. Section 3.1), it is possible to identify three main problems for data stream anomaly detection in order to monitor MCPSs. These include *time-efficiency*, *adaptability*, and *heterogeneity*.

1. **Time-efficiency**: With respect to the key challenge 'time dependence' and the requirement 'time', very fast data stream anomaly detection algorithms are required. The time-efficiency relates to the time complexity which is required to process an unlabeled data item. However, only a few studies on time-efficient data stream algorithms have been done so far.

2. **Adaptability**: With respect to the key challenges 'change' along with 'continuity' and the requirement 'knowledge', the shape of the previously identified classes can change during operation. Consequently, it is necessary to adapt the underlying model of the applied anomaly detection approach during operation. Depending on the required monitoring approach, different training methods (cf. Section 3.4.3) can be necessary.

3. **Heterogeneity**: An MCPS is subject to a variety of system states, and the shape of the present classes that must be approximated can diverge widely. Thus, the geometrical interpretations are heterogeneous. Consequently, a monitoring task is a *heterogeneous multi-class anomaly detection* problem, and data stream anomaly detection techniques must identify a large number of system states during operation in order to monitor MCPSs.

4.7 Solution Statement

In order to address the aforementioned problem statement, an approximated solution of a present heterogeneous multi-class anomaly detection problem is required. This solution must be time-efficient and reasonably precise. The author of the present thesis therefore introduces a new data stream anomaly detection algorithm, which uses a distinct and optimized class detector for each of the known classes. This distinct and optimized class detector is called a *dichotomous class detector* hereinafter. A dichotomous class detector is independent from the underlying anomaly detection

technique (cf. Section 4.3). A dichotomous class detector provides two mutually exclusive decisions. It returns *true* (ω_i) if an unlabeled data item was assigned to a class and returns *false* (ω_i^{C}) otherwise. Thus, the heterogeneous multi-class anomaly detection problem becomes a set of binary decisions and falls apart into a set of simple anomaly detection problems [NSS13a]. With regards to the aforementioned example (cf. Section 4.5), the heterogeneous multi-class anomaly detection problem comprises $h = 7$ dichotomous class detectors.

The set of dichotomous class detectors must be combined in an appropriate manner to solve a heterogeneous multi-class anomaly detection problem time-efficiently. As depicted in Figure 25, the author of the present thesis introduces a consecutive arrangement of these dichotomous class detectors by a *detector chain*. For the sake of simplicity, the notation of the classes is used to identify the associated class detectors. The brackets around the index state that the order of the dichotomous class detectors refers to a permutation. The aforementioned problem statement is discussed as follows.

1. **Time-efficiency**: The resulting detector chain provides a very flexible structure due to permutation of the present dichotomous class detectors. Additionally, the administrative overhead is reasonably low. Such a chain offers numerous capabilities for minimizing the processing time.

2. **Adaptability**: During operation, it is easily possible to add, delete, and adapt dichotomous class detectors of the detector chain. Further on, such a detector chain provides the ability to easily rearrange the order of the present class detectors which leads to a set of permutations.

3. **Heterogeneity**: Each dichotomous class detector in the chain is optimized for a single anomaly detection problem and fits best, depending on the underlying anomaly detection technique, to the associated class. Thus, the anomaly detection approach becomes very accurate while the anomaly detection performance is also improved.

Since each dichotomous class detector provides a discriminative boundary, it is possible to identify a combining mechanism (cf. Section 2.5.3). The author of the present thesis has decided to use the OAR-based combining mechanism because a discriminative boundary is trained between the target class and the other classes, including the anomaly class. To the best of the author's knowledge, no substantial research has been conducted so far to study OAR-based data stream anomaly detection. The presented approach is based on the following two main assumptions.

1. **Disjointness**: To achieve a distinct anomaly detection result, the geometrical interpretation of the present classes must be disjoint. Consequently, a chain of

disjointed class detectors leads to a distinct result for a heterogeneous multi-class anomaly detection problem.

2. **Independence**: The present dichotomous class detectors are independent from one another and can be rearranged into different orders (permutations).

The *termination condition* of such an OAR-based multi-class anomaly detection approach entails two possibilities. First, the algorithm terminates when an unlabeled data item has been accepted by a class detector. At most, one of the class detectors yields true due to disjointedness. Second, the algorithm terminates when all class detectors fail. As depicted in Figure 25, the worst case scenario relates to the situation when an unlabeled data item has to be processed by all of the present class detectors. Thus, the worst case relates to a very long runtime of the algorithm. In the following, possibilities to minimize the processing time of the suggested heterogeneous multi-class anomaly detection approach are discussed. This section is closely related to the author's publication [NSS13a].

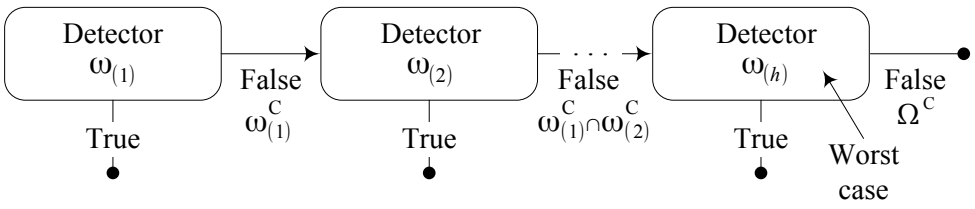

Figure 25: Detector chain (based on [NSS13a])

4.7.1 Minimizing the Processing Time on Average

The main disadvantages of such a detector chain relates to the time consumption until the termination conditions has been reached. The reduction of the overall processing time can also help to reduce power consumption. The time and power consumptions relate to the requirement 'restricted system resources' (cf. Section 3.1) and must be as low as possible. Accordingly, the main purpose is to minimize the overall processing time which is consumed by a detector chain to solve a heterogeneous multi-class anomaly detection problem.

Due to the consecutive application of the dichotomous class detectors, an OAR-based anomaly detection approach becomes to a sequential search problem. Further information about sequential searching is provided by Knuth [Knu98]. Each dichotomous class detector entails a processing time t_i, which is assigned to a probability of occurrence p_i and refers to a specific class ω_i. It is initially assumed that only

one data item is processed at a time. The absolute processing time depends on a specific target machine. The present dichotomous class detectors can be rearranged into $h!$ many different permutations, and the intention is to find the permutation which minimizes the overall processing time on average. This is called the *optimal permutation* hereinafter.

With regards to a set of training data $\mathcal{X}^{tr}$, it is possible to estimate the probabilities of occurrence for each class p_i, where $p_1 + p_2 + ... + p_h + p_{h+1} = 1$. The probability $p_{h+1} = 1 - \sum_i^h p_i$ refers to the complement of known classes (anomaly class). The probabilities, which refer to a known class, are assigned to the corresponding class detectors and can be estimated as follows.

$$p_i = \frac{|\{\boldsymbol{s}_\tau \in \mathcal{X}^{tr} \wedge \boldsymbol{s}_\tau \in \omega_i\}|}{|\mathcal{X}^{tr}|} \tag{4.1}$$

Based on these estimates, it is possible to select a permutation so that the termination condition is reached as early as possible for the majority of the unlabeled data items. According to Knuth [Knu98], it can be achieved when the class detectors are sorted in a descending order by the estimated probabilities.

$$p_1 \geq p_2 \geq \cdots \geq p_h \tag{4.2}$$

However, the main disadvantage is that the individual processing times of the dichotomous class detectors were not taken into account. Consequently, the author of the present thesis integrates the probability of occurrence as product with the processing time. Due to the consecutive chain of class detectors, the *cumulative processing time* $\tilde{t}_{(i)}$ of a dichotomous class detector relates to the sum of the previously executed class detectors. This relationship refers to the expected value μ_ρ of a permutation ρ ($l = \{1...h!\}$). When the processing times of all present dichotomous class detectors are equal $t_{(1)} = t_{(2)} = ... = t_{(h)}$, this reduces to (4.2). Since $\zeta = \tilde{t}_h \cdot p_{h+1}$ is a constant, it does not effect the ordering and can also be neglected.

$$\tilde{t}_{(i)} = \sum_{j=1}^{i} t_{(j)},\ \mu_{\rho_l} = \zeta + \sum_{j=1}^{h} p_{(j)} \cdot \tilde{t}_{(j)} \tag{4.3}$$

The optimal permutation provides the minimal average processing time as long as the probability distribution holds. The empirical calculation of the minimal expected value represents a feasible solution. However, this solution is very resource intensive due to the calculation of $h!$ many permutations. Accordingly, the author of the present thesis introduces the following theorem which is based on the descriptions provided by Smith [Smi56] and Knuth [Knu98].

Theorem 1. *Let $p_{(i)}$, $t_{(i)}$ and $\tilde{t}_{(i)}$ be as defined above. The arrangement of the dichotomous class detectors in an OAR-based detector chain is optimal if*

$$\frac{p_{(1)}}{t_{(1)}} \geq \frac{p_{(2)}}{t_{(2)}} \geq \cdots \geq \frac{p_{(h)}}{t_{(h)}}. \tag{4.4}$$

In other words, the optimal permutation, which provides the minimal expected value over all permutations ρ, provides the minimal average processing time if (4.4) holds.

Proof. Suppose that $p_{(i)} \cdot \tilde{t}_{(i)}$ and $p_{(i+1)} \cdot \tilde{t}_{(i+1)}$ are interchanged and a permutation changes from

$$\cdots + p_{(i)} \cdot \tilde{t}_{(i)} + p_{(i+1)} \cdot \tilde{t}_{(i+1)} + \cdots \tag{4.5}$$

to

$$\cdots + p_{(i+1)} \cdot (\tilde{t}_{(i+1)} - t_{(i)}) + p_{(i)} \cdot \tilde{t}_{(i+1)} + \cdots . \tag{4.6}$$

This results in a change of the expected processing time by $p_{(i)} \cdot t_{(i+1)} - p_{(i+1)} \cdot t_{(i)}$. Under the given assumptions, it follows that the change from (4.5) to (4.6) increases the processing time. Consequently, the permutation (4.6) is not optimal and (4.4) holds for any optimal permutation.

It has been shown that the optimal permutation, which provides the minimal expected value, is locally optimal, and adjacent interchanges lead to no further improvements. Furthermore, it is necessary to show that the permutation is globally optimal. Following the description of Knuth [Knu98], two more proofs are considered.

First proof. The first proof is based on a computer science approach. Assume that (4.4) holds and consider that the dichotomous class detectors are sorted as follows: $\omega_{(1)}, \omega_{(2)}, ..., \omega_{(h+1)}$. Starting from any arbitrary permutation, such an arrangement can be achieved by using a sequence of interchanges so that each interchange replaces $..., \omega_{(j)}, \omega_{(i)}, ...$ by $..., \omega_{(i)}, \omega_{(j)}, ...$ for some $i < j$. This decreases the overall processing time on average by the non-negative amount $p_{(i)} \cdot t_{(j)} - p_{(j)} \cdot t_{(i)}$. Thus, the optimal permutation, which provides the minimal expected value, also provides the minimal average processing time.

Second proof. The second proof uses a mathematical trick ("tie-breaking") which considers the case when some of the quotients of (4.4) are equal. In this case, more than one optimal permutation exists and a solution which excludes equality is sought. Accordingly, each probability $p_{(i)}$ is replaced by

$$p_{(i)}(\epsilon) = p_{(i)} + \epsilon^i - \bar{\epsilon}, \tag{4.7}$$

where ϵ is a really small positive number ($\epsilon \in \mathbb{R}_{>0}$) and $\bar{\epsilon}$ denotes the mean value $\bar{\epsilon} = (\epsilon^1 + \epsilon^2 + ... + \epsilon^h)/(h)$. This replacement assures the main assumption of the probability theory $p_{(1)}(\epsilon) + p_{(2)}(\epsilon) + ... + p_{(h)}(\epsilon) = 1$.

In case ϵ is sufficiently small, it can be excluded that $x_1 p_1(\epsilon) + \cdots + x_h p_h(\epsilon) = y_1 p_1(\epsilon) + \cdots + y_h p_h(\epsilon)$ unless $x_1 = y_1, ..., x_h = y_h$. Consequently, equality will not hold in (4.4). When considering all $h!$ permutations of the present class detectors, it is obvious that there is at least one optimal permutation which satisfies (4.4). Although, due to the exclusion of equality, only one permutation satisfies (4.4), and the Theorem 1 uniquely characterizes the optimal arrangement of class detectors for the probabilities $p_{(i)}(\epsilon)$, whenever ϵ is sufficiently small. By continuity, this also holds if ϵ is set equal to zero.

□

4.7.2 Filter Function

The previous section discusses possibilities to find an optimal permutation which minimizes the overall processing time for the majority of unlabeled state vectors on average. However, the main problem remains: in order to classify an unlabeled state vector, a large number of dichotomous class detectors must be processed. Thus, the entire detector chain must be processed in the worst case scenario, and the overall processing time is relatively high.

In the following outline an additional filter function, which aims to further minimize the overall processing time of an already optimal ordered detector chain (optimal permutation), is discussed. This already optimal ordered detector chain is called *origin chain* ρ_{origin} hereinafter.

Motivation and Conditions

One possibility to further minimize the overall processing time refers to a filter function which excludes dichotomous class detectors time-efficiently from an origin chain. Accordingly, the filter function intends to find dichotomous class detectors in an origin chain so that the found dichotomous class detectors can be excluded without processing. As a result, two detector chains are derived. The first derived detector chain contains excluded dichotomous class detectors. These excluded dichotomous class detectors need no further processing and are not further considered. The second derived detector chain contains pre-selected or candidate dichotomous class detectors. This second detector chain is called *candidate chain* hereinafter.

Thus, the filter approach is a function f_{filter} which maps the origin chain ρ_{origin} to a candidate chain $\rho_{\text{candidate}}$.

$$f_{\text{filter}} : \rho_{\text{origin}} \rightarrow \rho_{\text{candidate}} \tag{4.8}$$

The filter function comprises the following three possible outcomes: *equality*, *reduction*, or *emptiness*. The corresponding processing time of the filter function is denoted as t_{filter}, the corresponding overall processing time of the origin chain is denoted as t_{origin}, and the overall processing time of the candidate chain is denoted as $t_{\text{candidate}}$. The aforementioned outcomes are discussed below.

1. **Equality**: If none of the dichotomous class detectors is excluded, the origin detector chain and the candidate chain are equal $\rho_{\text{candidate}} = \rho_{\text{origin}}$. Accordingly, the overall processing time also equals $t_{\text{candidate}} = t_{\text{origin}}$.
2. **Reduction**: If some of the dichotomous class detectors are excluded, the candidate chain contains a reduced number of dichotomous class detectors $|\rho_{\text{candidate}}| < |\rho_{\text{origin}}|$ which must be processed. Accordingly, the overall processing time is reduced to $t_{\text{candidate}} < t_{\text{origin}}$.
3. **Emptiness**: If all of the dichotomous class detectors are excluded, the candidate chain is empty, and the number of dichotomous class detectors, which must be processed, is equal to zero $|\rho_{\text{candidate}}| = 0$. Accordingly, the overall processing time is reduced to $t_{\text{candidate}} = 0$, and the algorithm terminates immediately. In this case, an unlabeled state vector cannot be assigned to any of the known classes and is directly classified as an anomaly.

In addition, the author of the present thesis introduces the following theorem.

Theorem 2. *Let $p_{(i)}$, $t_{(i)}$ and $\tilde{t}_{(i)}$ be as defined above. The candidate chain is still in optimal order when an applied filter function excludes one or more dichotomous class detectors out of an origin chain and the order is not changed.*

$$f_{filter} : \left(\cdots \frac{p_{(i)}}{t_{(i)}} \geq \frac{p_{(i+1)}}{t_{(i+1)}} \geq \frac{p_{(i+2)}}{t_{(i+2)}} \cdots \right) \rightarrow \left(\cdots \frac{p_{(i)}}{t_{(i)}} \geq \cancel{\frac{p_{(i+1)}}{t_{(i+1)}}} \geq \frac{p_{(i+2)}}{t_{(i+2)}} \cdots \right) \tag{4.9}$$

Proof. Suppose that the following origin chain is in optimal order and the summand $p_{(i+1)} \cdot \tilde{t}_{(i+1)}$ is excluded. Accordingly, the permutation changes from

$$\cdots + p_{(i)} \cdot \tilde{t}_{(i)} + p_{(i+1)} \cdot \tilde{t}_{(i+1)} + p_{(i+2)} \cdot \tilde{t}_{(i+2)} + \cdots \tag{4.10}$$

to

$$\cdots + p_{(i)} \cdot \tilde{t}_{(i)} + p_{(i+2)} \cdot (\tilde{t}_{(i+2)} - t_{(i+1)}) + \cdots . \tag{4.11}$$

The probabilities do not change, and the relation of the cumulative processing times is preserved since the excluded processing time is subtracted from all of the following summands. Consequently, adjacent interchanges lead to no further improvements, and the presented Theorem 2 holds in any case.

□

In accordance with the presented Theorem 2, the filter function is subject to the following two requirements.

1. The filter function must ensure that an unlabeled state vector does not belong to a class which refers to an excluded dichotomous class detector.
2. The filter function must ensure that the ordering of the dichotomous class detectors is not changed.

Filter Approach

The previous section motivates and defines two requirements for the usage of a filter function. This section intends to present a specific filter function which takes the defined conditions into account.

The author of the present thesis introduces a filter function which aims to completely enclose each of the dichotomous class detectors by an additional discriminative boundary. As depicted in Figure 26, a set of hyperspheres is used. A single hypersphere is centered around each class. A single hypersphere b_i provides a centroid $\boldsymbol{c}_i$ and a radius r_i. In general, the centroid is defined as follows.

$$\boldsymbol{c}_i = \frac{\sum\limits_{\tau} \{\boldsymbol{s}_\tau \in \mathcal{X}^{tr} \wedge \boldsymbol{s}_\tau \in \omega_i\}}{|\{\boldsymbol{s}_\tau \in \mathcal{X}^{tr} \wedge \boldsymbol{s}_\tau \in \omega_i\}|} \tag{4.12}$$

Moreover, it is necessary to define a radius. The presented filter function necessitates the maximum Euclidean distance from the centroid of the hypersphere to the boundary of the underlying dichotomous class detector. Consequently, it is concluded that a hypersphere binds an underlying dichotomous class detector minimally.

Contrary to the underlying dichotomous class detectors, these hyperspheres need not to be disjoint. Hence, a filter function can select a set of candidate dichotomous class detectors.

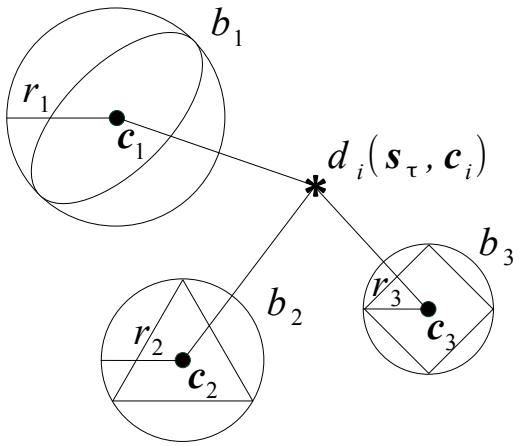

Figure 26: Filter function based on hyperspheres

To process an unlabeled data item $\boldsymbol{s}_\tau$, the Euclidean distances to all of the hypersphere centers $d_i(\boldsymbol{s}_\tau, \boldsymbol{c}_i)$ must be calculated. An underlying dichotomous class detector becomes a candidate if the calculated Euclidean distance is less than or equal to the radius of the referring minimal bounding hypersphere. This is defined as follows.

$$\omega_i \in \rho_{\text{candidate}}, \quad \text{if} \quad d_i(\boldsymbol{s}_\tau, \boldsymbol{c}_i) \leq r_i \tag{4.13}$$

The aforementioned requirements are fulfilled by the presented filter function. The first requirement is ensured due to the application of a minimal bounding hypersphere. If a dichotomous class detector is excluded, it is impossible that an unlabeled state vector belongs to the referring class. The second requirement is also ensured. Since the presented filter function works in a first in, first out manner, it does not change the given order of the dichotomous class detectors.

In order to achieve an appropriate filter result, the author of the present thesis defines the filter function as a function that is executed before the processing of the candidate chain starts.

Furthermore, the filter function should ensure that the overall processing time is effectively reduced $t_{\text{filter}} + t_{\text{candidate}} \leq t_{\text{origin}}$. However, it is very difficult to estimate if the overall processing time is effectively reduced by a filter function. For example, the presented filter function ensures that the processing time is effectively reduced when all minimal bounding hyperspheres are disjoint or only a small set are joint. With an

increasing number of joint minimal bounding hyperspheres, the filter function can no longer ensure that the overall processing time is effectively reduced. The probability that hyperspheres are joint increases in high dimensional spaces. Consequently, the presented filter function must be empirically tested (e.g. by means of the training data set) if it ensures that the overall processing time is effectively reduced.

Moreover, the processing time of the filter function t_{filter} should be very small. The processing time of the presented filter function is very small since only the Euclidean distances to the hypersphere centers must be calculated. To further reduce the processing time of the presented filter function, it is also possible to estimate the distance of an unlabeled state vector to the hypersphere centers by using the triangle inequality [Sam05].

Filter Example

Figure 27 depicts the presented filter approach by means of the example data set (cf. Section 4.5). For better illustration, only three minimal bounding hyperspheres b_4, b_5, and b_6 are depicted. These hyperspheres relate to the three classes ω_4, ω_5, and ω_6. The remaining four hyperspheres b_1, b_2, b_3, and b_7, which relate to the classes ω_1, ω_2, ω_3, and ω_7, are disjoint.

As mentioned earlier, it is not necessary that the hyperspheres of the presented filter approach are disjoint. For example, the hyperspheres b_4 and b_5 are intersecting, and an intersecting plane exists. The hypersphere b_6 lies within the hypersphere b_5. More precisely, it is completely enclosed by the hypersphere b_5. The hyperspheres b_4 and b_6 are disjoint.

In the following description, three issues are discussed in detail. These issues are exemplified by the state vectors $\boldsymbol{s}_1$, $\boldsymbol{s}_2$, and $\boldsymbol{s}_3$.

Based on the current example, the filter function delivers an empty candidate chain when all distances, from an unlabeled state vector to the hypersphere centers, are greater than the radii of the hyperspheres. This issue is exemplified in Figure 27 by means of the state vector $\boldsymbol{s}_1$. Hence, the state vector $\boldsymbol{s}_1$ refers to an anomaly state, and the overall processing time, in the worst case scenario, is significantly reduced.

Furthermore, the filter function delivers a candidate chain with one candidate dichotomous class detector if the distance between an unlabeled state vector and one center of a hypersphere is less than or equal to the referring radius. In Figure 27, this issue is depicted by means of the state vector $\boldsymbol{s}_2$. Thus, the overall processing time, in the worst case scenario, is significantly reduced.

Moreover, the filter function delivers a candidate chain which contains more than one candidate dichotomous class detector when more than one distance between an unlabeled state vector and the centers of the hyperspheres is less than or equal to the referring radii. In Figure 27, this issue is depicted by means of the state vector $\boldsymbol{s}_3$. The maximum number of candidate dichotomous class detectors in a candidate chain is two. Thus, the overall processing time, in the worst case scenario, is significantly reduced.

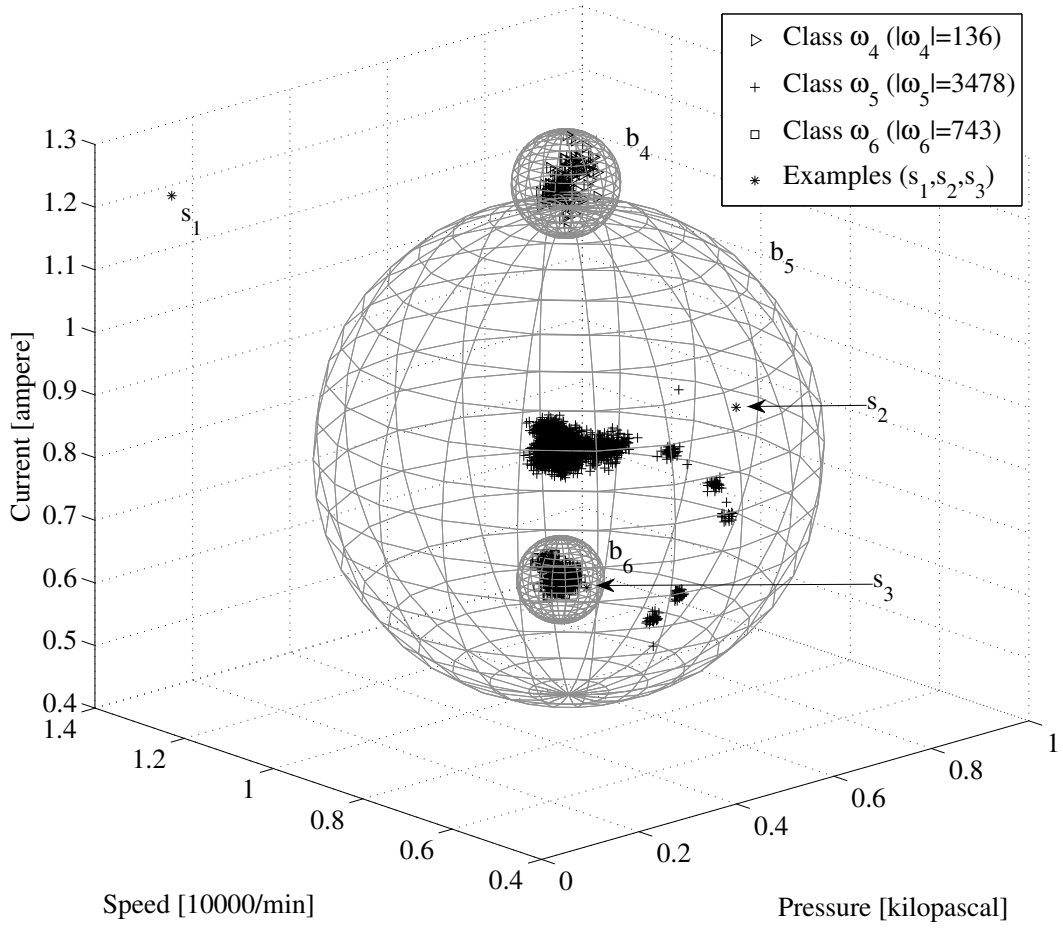

Figure 27: Filter example with three hyperspheres

4.8 One-class Classification for Anomaly Detection

So far, the problem of heterogeneous multi-class anomaly detection was treated on an abstract level. Therefore, it is required to identify a set of algorithms which provide the ability to implement the above stated assumptions. The author of the present thesis suggests to use *one-class classifiers* [Tax01] which are well suited to the given challenges. Amongst others, versatile performance studies of one-class classifiers have been contributed by Janssens et al. [JFP09] and Brereton [Bre11].

As stated by Tax [Tax01], each one-class classifier algorithm comprises two distinct elements: a distance or a resemblance of a data item to a class and a threshold θ on this distance or resemblance. The threshold is used to define a one-class classifier as a dichotomous class detector. The following one-class classifiers are described below: *Gaussian distribution*, *k-centers*, *nearest neighbor*, and *support vector domain description (SVDD)* [TD99b]. In order to prepare example figures, the DDTools [Tax12] and the PRTools [DJP+07] MATLAB [Mat14] packages have been partly used.

4.8.1 Gaussian Distribution

The Gaussian distribution [Bis06], also known as the normal distribution, is the basis for a Gaussian one-class classifier [Tax01]. A Gaussian one-class classifier refers to statistical anomaly detection (cf. Section 4.3.4). Advantages and disadvantages are discussed below.

- **Advantages**: The Gaussian one-class classifier is a very simple OCC approach. It is an appropriate algorithm when the referring data set is Gaussian distributed. Classification consumes very few system resources and is very fast.
- **Disadvantages**: It is preconditioned that the data set is Gaussian distributed.

The multivariate Gaussian distribution in particular is examined below. For an n-dimensional (number of attributes) state vector $\boldsymbol{s}_\tau$, the multivariate Gaussian distribution $\mathcal{N}$ is defined as follows.

$$\mathcal{N}(\boldsymbol{s}_\tau|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} exp\left\{-\frac{1}{2}(\boldsymbol{s}_\tau-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{s}_\tau-\boldsymbol{\mu})\right\} \tag{4.14}$$

The vector $\boldsymbol{\mu}$ entails the expected values of the individual attributes, the symbol $\boldsymbol{\Sigma}$ refers to a covariance matrix, the symbol π denotes the ratio of the circumference of a circle to its diameter, and the symbol $|\boldsymbol{\Sigma}|$ describes the determinant of $\boldsymbol{\Sigma}$. From

the exponent of the multivariate Gaussian distribution follows the *Mahalanobis distance* [Mah36] from an unlabeled state vector $\boldsymbol{s}_\tau$ to $\boldsymbol{\mu}$. The Mahalanobis distance $d_{\text{Mahal}}(\boldsymbol{s}_\tau, \boldsymbol{\mu})$ is a distance measure. It is translationally invariant and is a generalization of the *Euclidean distance*. The squared Euclidean distance is obtained when the matrix $\boldsymbol{\Sigma}$ equals the identity matrix.

When a set of training data is used to create such an one-class classifier, the expected values are unknown. Accordingly, it is necessary to estimate the expected values by an *arithmetic mean* $\bar{x}$ [OSB+72], and it follows $\bar{\boldsymbol{x}} \approx \boldsymbol{\mu}$. The vector $\bar{\boldsymbol{x}}$ constitutes a center, which is also called centroid, of the data set. Further on, each state vector in the training data must be normalized $\boldsymbol{s}_\tau - \bar{\boldsymbol{x}}$ and finally it is possible to estimate the covariance matrix $\boldsymbol{\Sigma}$ from the training data (see also [KM83, JW92, Bis06, DD09, Han10]). The Mahalanobis distance is defined as follows.

$$d_{\text{Mahal}}(\boldsymbol{s}_\tau, \bar{\boldsymbol{x}}) = \sqrt{(\boldsymbol{s}_\tau - \bar{\boldsymbol{x}})' \boldsymbol{\Sigma}^{-1} (\boldsymbol{s}_\tau - \bar{\boldsymbol{x}})} \tag{4.15}$$

Classification

To classify an unlabeled state vector, it is necessary to define a threshold θ_{Mahal}. There are several possibilities for this. One feasible solution is to use one of the quantiles Q of the density function. This results in the following decision function.

$$\boldsymbol{s}_\tau \in \begin{cases} \omega, & \text{if} \quad d_{\text{Mahal}}(\boldsymbol{s}_\tau, \bar{\boldsymbol{x}}) \leq \theta_{\text{Mahal}} \\ \omega^{\text{C}}, & \text{else} \end{cases} \tag{4.16}$$

Since only the distance from an unlabeled state vector to the centroid must be calculated, the processing effort is relatively low. It is necessary to store the threshold θ_{Mahal} along with the vector which defines the centroid $\bar{\boldsymbol{x}}$. Thus, the memory usage is also relatively low.

Example

As a geometrical interpretation, the elliptic shape of the Gaussian distribution or Mahalanobis distance follows from the covariance matrix $\boldsymbol{\Sigma}$. Thus, the training data set is bounded by a convex hull. The vector $\bar{\boldsymbol{x}}$ refers to the centroid of the ellipse, and the eigenvectors of the covariance matrix are the new axes. Figure 29 depicts this circumstances of this case by means of the known class ω_1 (cf. Section 4.5). To illustrate this example better, only two attributes are selected: pressure and current.

In order to train this one-class classifier, class ω_1 is defined as Gaussian distributed. A large number of testing methods to test whether a data set is Gaussian distributed exist. More detailed information and a comparison of these testing methods are provided by D'Agostino et al. [DS86]. A fundamental assumption of a multivariate Gaussian distribution is that the univariate attributes must be Gaussian distributed [JW92]. As depicted in Figure 28, a *quantile-quantile plot* [WG68] is a very simple visual method to verify a data set against the Gaussian distribution. Therefore, the quantiles of the data set are depicted in relation with the quantiles of a Gaussian distribution. Figure 28 shows that the attributes of the given data set are reasonably precisely Gaussian distributed. For the sake of simplicity, it is assumed that class ω_1 refers to a multivariate Gaussian distribution, so further analysis is not carried out. Furthermore, the arithmetic mean is estimated and the class ω_1 (data set) is normalized. Afterwards, the covariance matrix is determined, and, based on this, the eigenvectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are calculated. For the sake of clarity, the eigenvectors are displayed by straight lines in Figure 29.

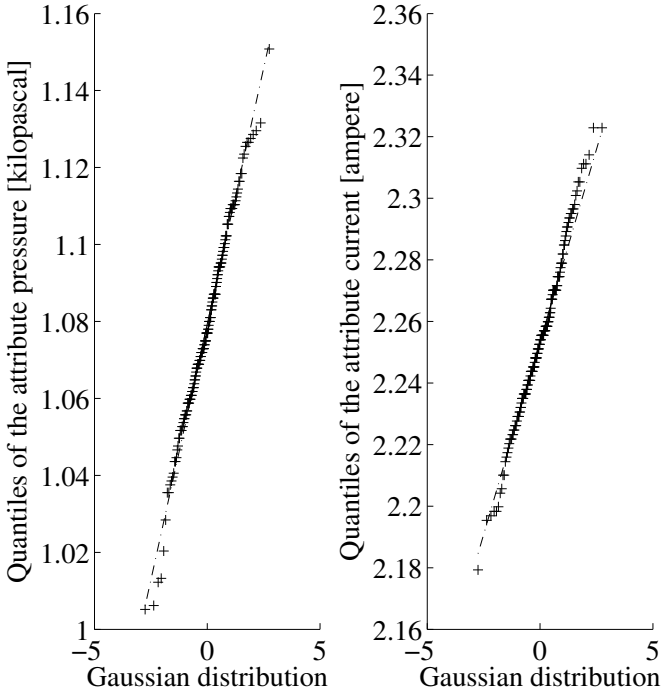

Figure 28: Quantile-quantile plot

As mentioned earlier, a threshold is required in order to classify an unlabeled data item. The present example, depicted in Figure 29, comprises two thresholds. The first threshold $\theta_{\text{Mahal}_1} = Q_1$, depicted by a solid line, refers to a 100% quantile and expresses that 100% of the data items are bounded by the ellipse. The second threshold θ_{Mahal_2}, depicted by a dotted line, refers to a 98.5% quantile and expresses that only 98.5% of the data items are bounded by the ellipse. It follows the expansion of the ellipses in the direction of the eigenvectors.

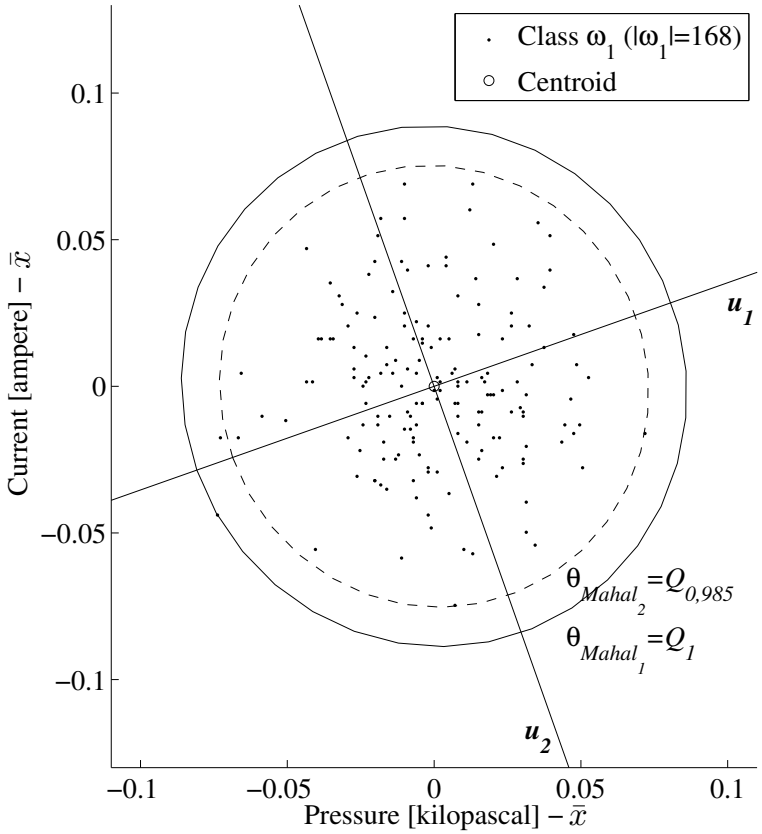

Figure 29: Gaussian distribution

Filter

To construct a reasonably precise filter function for the Gaussian one-class classifier, the centroid of the training data set is calculated. The centroid of the training data set constitutes the centroid $\boldsymbol{c}$ of the hypersphere. Moreover, when the threshold is

based on the 100% quantile, the Euclidean distance to the furthest state vector in the training data set to the centroid constitutes the radius r of the hypersphere.

4.8.2 K-centers

As stated in Tax [Tax01], the k-centers approach is a boundary method which covers the data set with $k \in \mathbb{N}$ many balls. These balls have equal radii, the centers of the balls are medoids, and the number k is user defined. A *medoid* $\boldsymbol{m}$ is a center of a ball where $m \in \mathcal{X}^{tr}$. The k-centers method refers to clustering-based anomaly detection (cf. Section 4.3.3). Advantages and disadvantages are discussed below.

- **Advantages**: The k-centers method is a very simple OCC approach. Generally, classification requires very few system resources because an unlabeled data item must be compared with a fixed number of medoids.
- **Disadvantages**: Usually, the calculation of the k many medoids is very resource intensive. Moreover, it is very difficult to find an appropriate number of k many balls manually, and the algorithm tends to overfitting.

The k-centers methods uses the Euclidean distance d_{Euklid} between an unlabeled data item $\boldsymbol{s}_\tau$ and a medoid m_k. The Euclidean distance is defined as follows.

$$d_{\text{Euklid}}(\boldsymbol{s}_\tau, \boldsymbol{m}_k) = \sqrt{\sum_{i=1}^{n} (\boldsymbol{s}_{\tau,i} - \boldsymbol{m}_{k,i})^2} \tag{4.17}$$

In many cases, the squared Euclidean distance $d_{\text{Euklid}}(\boldsymbol{s}_\tau, \boldsymbol{m}_k)^2$ is used.

$$d_{\text{Euklid}}(\boldsymbol{s}_\tau, \boldsymbol{m}_k)^2 = \sum_{i=1}^{n} (\boldsymbol{s}_{\tau,i} - \boldsymbol{m}_{k,i})^2 = ||\boldsymbol{s}_{\tau,i} - \boldsymbol{m}_{k,i}||^2 \tag{4.18}$$

As mentioned earlier, the k-centers method tries to find k many balls with equal radii to approximate the data set minimally. Therefore, medoids are selected from the data set so that the distance to all state vectors of the data set is minimized. During the training phase an error is minimized $\mathcal{E}_{\text{k-center}}$.

$$\mathcal{E}_{\text{k-center}} = \max_{\tau} \left(\min_{k} ||\boldsymbol{s}_\tau - \boldsymbol{m}_k||^2 \right) \tag{4.19}$$

Classification

To classify an unlabeled state vector, the Euclidean distances to all k many medoids is calculated, and the smallest distance of the resulting set of distances is sought. The distance calculation by means of the k-centers method is defined as follows.

$$d_{\text{k-center}} = \min_{k} ||\boldsymbol{s}_\tau - \boldsymbol{m}_k||^2 \tag{4.20}$$

The threshold $\theta_{\text{k-center}}$ refers to the radii, and the decision function is as follows.

$$\boldsymbol{s}_\tau \in \begin{cases} \omega, & \text{if} \quad d_{\text{k-center}} \leq \theta_{\text{k-center}} \\ \omega^{\text{C}}, & \text{else} \end{cases} \tag{4.21}$$

Since only the distance from an unlabeled state vector to the medoids must be calculated, the processing effort is relatively low. It is necessary to store the threshold $\theta_{\text{k-center}}$ along with the vectors which define the medoids $\boldsymbol{m}_k$. Thus, the memory usage relates to the number k and is, excluding overfitting, relatively low.

Example

Figure 30 depicts the k-centers method by means of the known class ω_3 (cf. Section 4.5). In order to illustrate this better, only two attributes are selected: pressure and current. The number of balls is set to $k = 3$. In order to train this one-class classifier, the medoids and the corresponding radii or thresholds are calculated.

As mentioned above, the number k is a user-defined parameter. An increasing number of k leads to a increasing complexity of the geometrical interpretation of the training data set. It follows that the processing effort increases as well. Accordingly, it is necessary to find an appropriate trade-off between the approximation of the training data and the computational complexity [YD98].

Figure 31 depicts the circumstances of the given example when the known class is ω_3 (cf. Section 4.5) and different values are chosen for the variable k. The upper left diagram (i) shows the k-centers method where $k = 1$. The data set is poorly approximated because a large distance exists between many sections of the circumferences and the next state vector. The k-centers method with $k = 1$ refers to a special case where the training data set is bounded by a single ball. In this case, the boundary is a convex hull. When $k > 1$, the hull is not convex. As depicted in the bottom left corner (iii) and in the bottom right corner (iv) in Figure 31, the k-centers method tends to overfitting when a larger value for k is chosen. This means

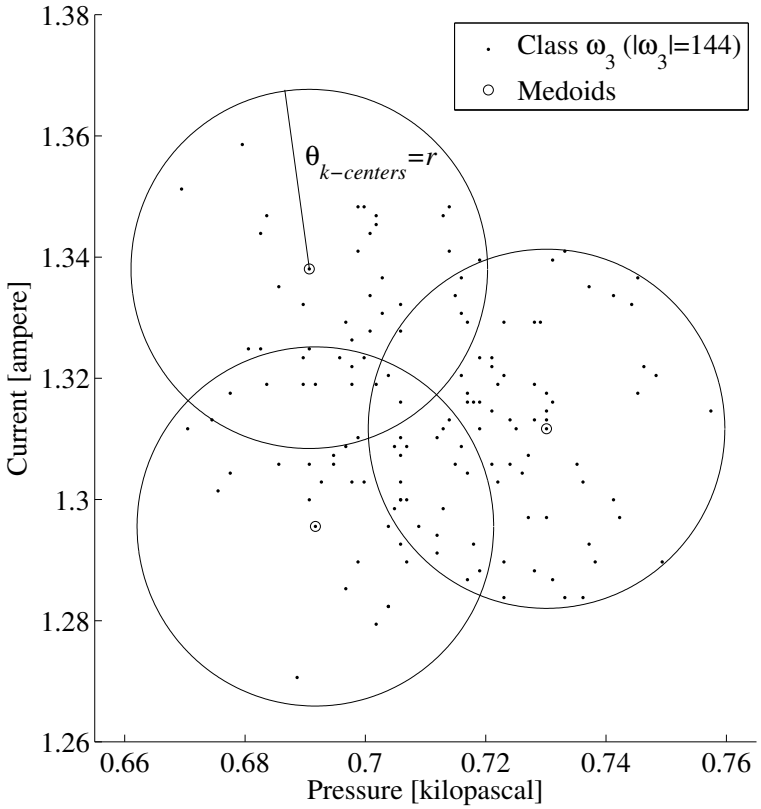

Figure 30: K-centers method with $k = 3$

that only a few state vectors are inside of one ball. For example, the diagram in the bottom right corner (iv) entails balls where only one state vector is inside which, in turn, is the medoid. As depicted in the upper right corner (ii) in Figure 31, an appropriate trade-off exists when the value of $k = 5$. The training data set is better approximated in the diagram in the upper right corner (ii) than in the diagram in the upper left corner (i). In comparison with both diagrams at the bottom (iii and iv), the diagram in the upper right corner (ii) gives no indication that the trained one-class classifier is overfitted.

Filter

To construct a reasonably precise filter function for the k-centers one-class classifier, the mean value of all medoids is calculated. This mean value constitutes the centroid $\boldsymbol{c}$ of the hypersphere. Moreover, the Euclidean distance to the furthest medoid is

calculated. This distance plus the threshold $\theta_{\text{k-center}}$ constitutes the radius r of the hypersphere.

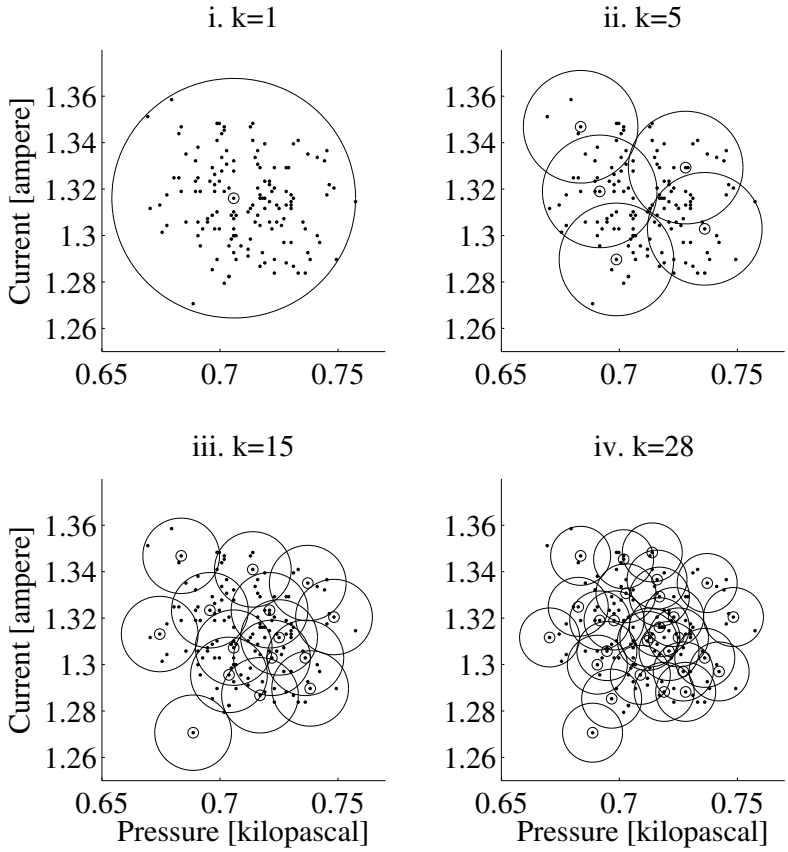

Figure 31: K-centers with different values of k

4.8.3 Nearest Neighbor

As stated by Tax [Tax01], the NN one-class classifier approach is a boundary method and based on a NN classifier which uses a local density estimation. The NN one-class classifier approach uses the distance to the first nearest neighbor rather than the estimation of explicit density. It refers to NN-based anomaly detection (cf. Section 4.3.2). Advantages and disadvantages are discussed below.

- **Advantages**: The NN one-class classifier works in an unsupervised manner and is data-driven. It works very well in high-dimensional spaces (many attributes).

- **Disadvantages**: The NN one-class classifier is a lazy learning method. In general, the entire training data set must be stored for classification. Thus, the computational complexity is very high, system resources are heavily used, and classification becomes relatively slow.

The NN method is introduced here very briefly. More detailed information is provided by Tax [Tax01]. To train a NN one-class classifier, an hypersphere is centered around a state vector, and the volume of the hypersphere is grown until it captures the next state vector (nearest neighbor) in the training data set. The local density $g_{NN}(\boldsymbol{s}_\tau)$ of the considered state vector is estimated as follows.

$$g_{NN}(\boldsymbol{s}_\tau) = \frac{1}{V(||\boldsymbol{s}_\tau - NN^{tr}(\boldsymbol{s}_\tau)||) \cdot |\mathcal{X}^{tr}|} \tag{4.22}$$

The volume of the hypersphere is denoted by the variable V, and the nearest neighbor of the considered state vector is denoted by $NN^{tr}(\boldsymbol{s}_\tau)$. Thus, the local density increases when the volume of the hypersphere decreases.

Classification

To classify an unlabeled data item $\boldsymbol{s}_\tau$, the distance between the unlabeled data item to its nearest neighbor $NN^{tr}(\boldsymbol{s}_\tau)$ is compared with the distance from this nearest neighbor to its nearest neighbor $NN^{tr}(NN^{tr}(\boldsymbol{s}_\tau))$. In general, the threshold θ_{nn} is set to 1. This results in the following decision function.

$$\boldsymbol{s}_\tau \in \begin{cases} \omega, & \text{if} \quad \dfrac{||\boldsymbol{s}_\tau - NN^{tr}(\boldsymbol{s}_\tau)||}{||NN^{tr}(\boldsymbol{s}_\tau) - NN^{tr}(NN^{tr}(\boldsymbol{s}_\tau))||} \leq \theta_{\text{nn}} \\ \omega^{\text{C}}, & \text{else} \end{cases} \tag{4.23}$$

Since the distance from an unlabeled state vector to all other state vectors in the training data set must be calculated, the processing effort is very high. It is necessary to store the threshold θ_{nn} along with the entire training set. Thus, the memory usage relates to the number $|\mathcal{X}^{tr}|$ and is also very high.

Example

Figure 32 shows a sketch of the NN method by means of the known class ω_2 (cf. Section 4.5). In order to better illustrate this example, only two attributes are selected: pressure and current. Since the NN method is a boundary method, a boundary around the data set is generated. The generated boundary is not convex. As depicted in Figure 32, an unlabeled state vector $\boldsymbol{s}_\tau$ is classified as anomaly if

$$\frac{d_1}{d_2} > \theta_{\text{nn}}. \tag{4.24}$$

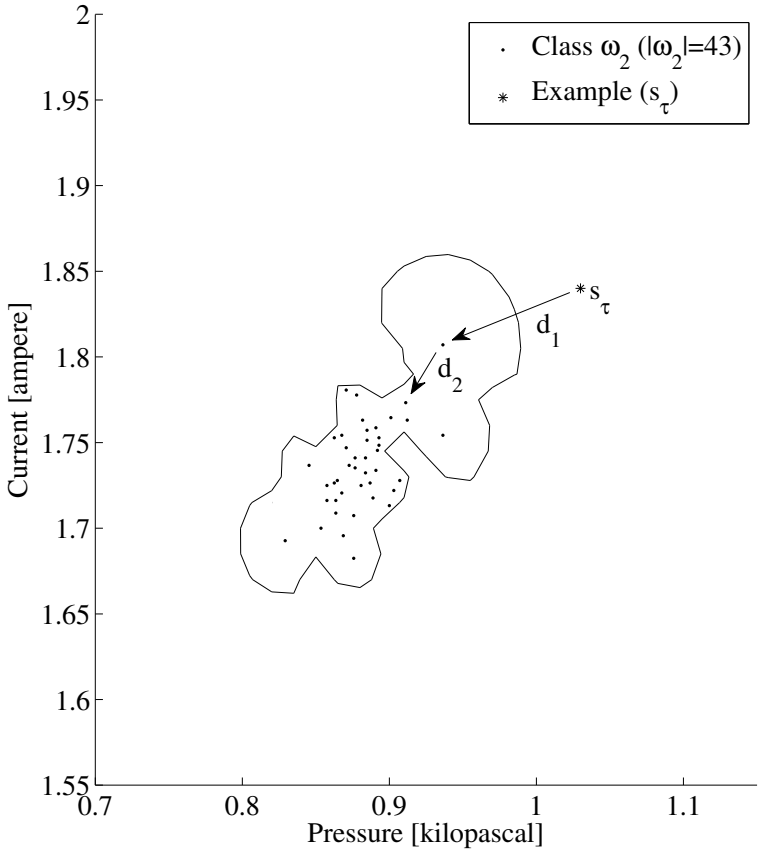

Figure 32: Sketch of the NN one-class classifier

Filter

To construct a reasonably precise filter function for the NN one-class classifier, the centroid of the training data set is calculated. The centroid of the training data set constitutes the centroid $\boldsymbol{c}$ of the hypersphere. Moreover, the Euclidean distance to the furthest state vector in the training data set to the centroid is calculated. This distance plus the value of the local density of furthest state vector constitutes the radius r of the hypersphere.

4.8.4 Support Vector Domain Description

As stated by Tax [Tax01], the SVDD (see also [TD99a, TD99b, TD04]) is another boundary method. The SVDD is a one-class classifier which refers to classification-based anomaly detection (cf. Section 4.3.1). The SVDD tries to find a hypersphere with minimum volume which contains almost all state vectors of the training data set. Moreover, *support vectors* are used. The SVDD is inspired by the SVM [Vap95] which is based on a straight line or hyperplane [SWSST00]. Advantages and disadvantages are discussed below.

- **Advantages**: The SVDD can be used to approximate a training data set very precisely.
- **Disadvantages**: The processing effort relates to the number of support vectors. Thus, the processing effort increases with the number of support vectors and can become very high.

In the following outline, the SVDD is introduced very briefly. More detailed information is provided by Tax [Tax01]. A state vector from the training data set is a support vector $\boldsymbol{z}_\tau$ if the associated Lagrangian multiplier $\alpha_\tau > 0$. As stated above, the SVDD one-class classifier is based on a hypersphere which comprises a radius r and a centroid $\boldsymbol{c}$ with

$$\boldsymbol{c} = \sum_{\tau} \alpha_\tau \boldsymbol{z}_\tau. \tag{4.25}$$

The approximation of a training data set by means of a hypersphere is very inflexible. Accordingly, Tax [Tax01] has introduced a kernel function K ("kernel-trick" [Vap98]) which maps the training data into a higher dimensional feature space. The mapping Φ allows to train a potentially more precise boundary around the training data set. The kernel function relates to the inner product $\boldsymbol{s}_{\tau_i} \cdot \boldsymbol{s}_{\tau_j}$ and is defined as follows.

$$K(\boldsymbol{s}_{\tau_i}, \boldsymbol{s}_{\tau_j}) = \Phi(\boldsymbol{s}_{\tau_i}) \cdot \Phi(\boldsymbol{s}_{\tau_j}) \tag{4.26}$$

A variety of kernel functions exists. In this thesis, the Gaussian kernel or *radial basis function (RBF)* is used exclusively. The RBF is defined as follows where σ denotes the width of the RBF.

$$K(\boldsymbol{s}_{\tau_i}, \boldsymbol{s}_{\tau_j}) = \exp\left(\frac{-||\boldsymbol{s}_{\tau_i} - \boldsymbol{s}_{\tau_j}||^2}{\sigma^2}\right) \tag{4.27}$$

Classification

To classify an unlabeled state vector $\boldsymbol{s}_\tau$, the distance $d_{\text{SVDD\_RBF}}(\boldsymbol{s}_\tau, \boldsymbol{c})$ to the center of the hypersphere is calculated.

$$d_{\text{SVDD\_RBF}}(\boldsymbol{s}_\tau, \boldsymbol{c}) = \sum_i \alpha_i \exp\left(\frac{-||\boldsymbol{s}_\tau - \boldsymbol{z}_i||^2}{\sigma^2}\right) \tag{4.28}$$

The threshold is defined as follows.

$$\theta_{\text{SVDD\_RBF}} = \frac{1}{2}(B - r^2), B = 1 + \sum_{i,j} \alpha_i \alpha_j K(\boldsymbol{z}_i, \boldsymbol{z}_j) \tag{4.29}$$

Finally, this results in the following decision function.

$$\boldsymbol{s}_\tau \in \begin{cases} \omega, & \text{if} \quad d_{\text{SVDD\_RBF}} \leq \theta_{\text{SVDD\_RBF}} \\ \omega^{\text{C}}, & \text{else} \end{cases} \tag{4.30}$$

Since the distance from an unlabeled state vector to the centroid must be calculated, the processing effort depends on the number of support vectors. Classification can be very fast, if the number of support vectors is relatively low, but it can also be very slow, when a larger number of support vectors is trained. It is necessary to store the threshold $\theta_{\text{SVDD\_RBF}}$ and the support vectors. Thus, the memory usage depends on the support vectors as well.

Example

Figure 33 depicts the SVDD method by means of the known class ω_6 (cf. Section 4.5). In order to illustrate this example better, only two attributes are selected: pressure and current. The SVDD is trained with the parameters $\sigma = 0.04$ and $C = 0.6$. The parameter C regulates the trade-off between the volume of the hypersphere and the

number of support vectors which are located outside of the hypersphere. In general, the generated boundary is not convex.

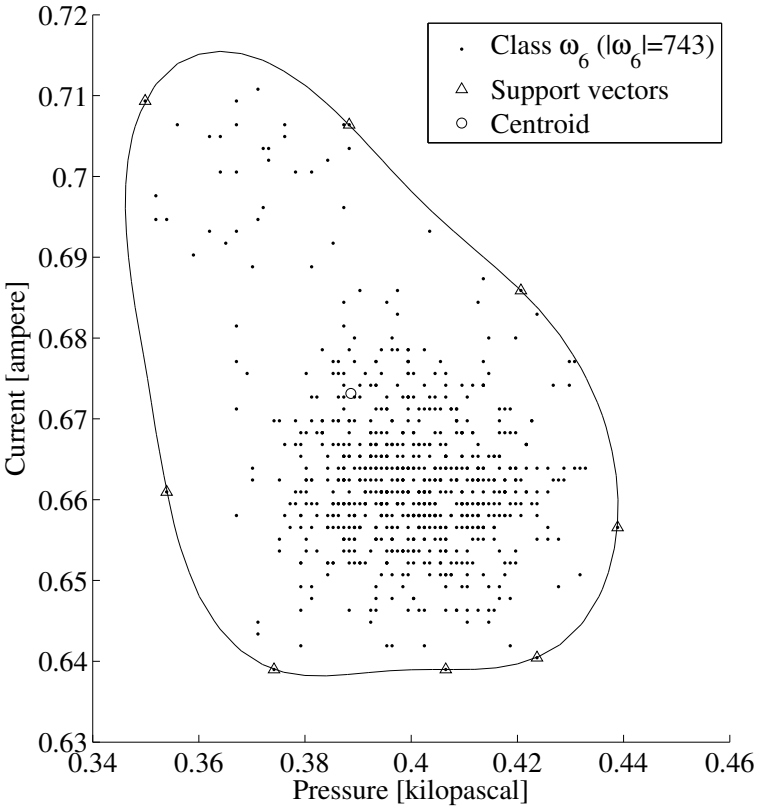

Figure 33: SVDD with RBF, $\sigma = 0.04$ and $C = 0.6$

Filter

To construct a reasonably precise filter function for the SVDD one-class classifier, the centroid of the support vectors which constitutes the centroid $\boldsymbol{c}$ of the hypersphere is calculated. Moreover, the Euclidean distance to the furthest support vector of this centroid is calculated. This distance constitutes the radius r of the hypersphere.

4.8.5 Summary

This section describes very different one-class classifiers. As summarized in Table 7, the differences relate to the geometrical interpretation, convexity of the boundary, and computational effort. The Gaussian one-class classifier describes the training

data by means of an ellipsoid. An ellipsoid as geometrical interpretation is convex and the computational effort of the Gaussian one-class classifier is comparatively low. The k-centers one-class classifier describes the training data by a set of k many balls. A set of k many balls is not convex, and the processing effort, with respect to the number k, is relatively low. The NN one-class classifier describes the training data by means of a density region. A density region as geometrical interpretation is not convex, and the processing effort, with respect to the number of training data items, is very high. The SVDD describes the training data by an adapted hypersphere. Therefore, the training data is mapped into a high dimensional space and a set of support vectors is calculated. An adapted hypersphere as geometrical interpretation is not convex, and the processing effort, with respect to the number support vectors, is relatively high.

Depending on the available training data and the application scenario, it is possible to apply one or more of the presented one-class classifiers in order to create the previously introduced detector chain (cf. Section 4.7).

	Geometry	Convexity	Effort
Gaussian	ellipsoid	convex	low
K-centers	k many balls	not convex	relatively low
NN	density region	not convex	very high
SVDD	adapted hypersphere	not convex	relatively high

Table 7: Summary of selected one-class classifiers

4.9 Conclusion

As described in Section 1.7 on page 13, the present thesis aims to combine KDD with KDDS for monitoring MCPSs. Therefore, a new OAR-based data stream anomaly detection algorithm has been presented in this chapter (cf. Section 4.7). The presented OAR-based data stream anomaly detection algorithm is based on the storage-aware stream model described in Section 3.4.2 on page 49 and the KDC described in more detail in Section 3.5 on page 52. Additionally, an approach to minimize the processing time of an OAR-based detector chain on average has been introduced (cf. Section 4.7.1), and a filter function, which can be applied to reduce the average processing time in the worst case scenario, has been presented (cf. Section 4.7.2). In order to provide a consistent terminology, a set of system states have been explained in more detail (cf. Section 4.1) and a basic anomaly detection model

has been presented (cf. Section 4.2). Furthermore, several anomaly detection techniques (cf. Section 4.3) and existing data stream anomaly detection algorithms (cf. Section 4.4) have been discussed. In order to illustrate the circumstances of heterogeneous multi-class anomaly detection, an example of the ISS Columbus air loop (cf. Section 4.5) referring to the presented real world scenario (cf. Section 1.4) has been presented. Finally, Section 4.8 on page 83 provides a set of selected one-class classifiers for anomaly detection, which can be applied for the presented OAR-based data stream anomaly detection algorithm.

CHAPTER 5

Experiments and Case Study

"If something can go wrong, it will."

Edward Aloysius Murphy (1918 -1990)

In this chapter, experiments and a case study are presented. Two independent KDC implementations are described here. The first KDC implementation is exclusively used to assess and evaluate the *effectiveness* along with the *efficiency* of data stream anomaly detection algorithms. The effectiveness of an anomaly detection algorithm refers to the ability to detect anomalies in the context of a given data set. According to the literature [Faw06], the effectiveness is called *performance* hereinafter. The efficiency refers to the time that an anomaly detection algorithm requires to process a single data item. For the sake of clarity, the efficiency is called *time-efficiency* hereinafter.

The second KDC implementation is used to perform a case study which refers to the above-stated real world scenario (cf. Section 1.4) by means of an existing IFP engine and the presented data stream anomaly detection approach (cf. Section 4.7). Therefore, the Esper CEP [Esp13] engine is used. The distinction between these two KDC implementations is made in order to provide pure performance and time-efficiency assessments without side effects which may occur due to the usage of an additional IFP engine.

The structure of the present chapter is as follows. First, the evaluation scheme is presented, which is subsequently used to compare and assess the presented data stream anomaly detection approach with selected data stream anomaly detection algorithms. Second, the first KDC implementation and the experimental setup, used to compare and assess the presented data stream anomaly detection approach

with selected data stream anomaly detection algorithms, are explained. Third, detailed information about the selected data stream anomaly detection algorithms, subsequently used for comparison with the presented data stream anomaly detection approach, is presented. Fourth, selected data sets are described. Fifth, the experiments are performed and an assessment is made. Sixth, the second KDC implementation is used to perform a case study which relates to the ISS Columbus module and recognizes IFP for monitoring MCPSs. Finally, this chapter is concluded.

5.1 Evaluation Scheme

The evaluation scheme used in this thesis is based on *receiver operating characteristics (ROC)* [Spa89]. As stated by Fawcett [Faw06], ROC graphs are used for visualizing, organizing, and assessing classifiers based on their performance.

Figure 34 depicts a confusion matrix. As shown in the matrix, an anomaly detection algorithm that processes an unlabeled data item can produce four possible outcomes. First, the unlabeled data item is positive and is classified as positive. The first outcome is counted as a *true positive (TP)*. Second, an unlabeled data item is positive and is classified as negative. The second outcome is counted as a *false negative (FN)*. Third, an unlabeled data item is negative and is classified as negative. The third outcome is counted as a *true negative (TN)*. Fourth, an unlabeled data item is negative and is classified as positive. This fourth outcome is counted as a *false positive (FP)*.

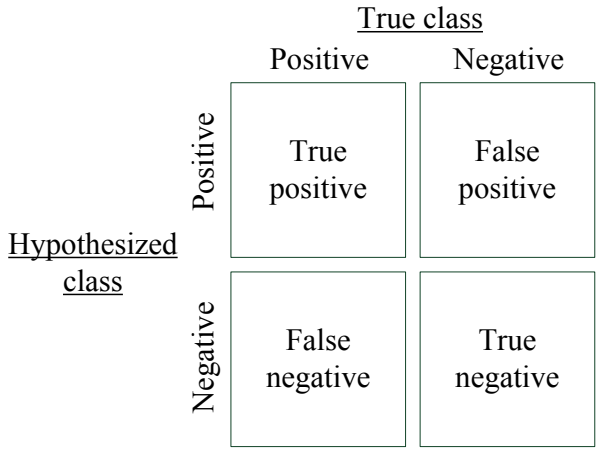

Figure 34: The confusion matrix (based on [Faw06])

It is possible to calculate a wide range of metrics by using the confusion matrix. Amongst others, they include *false positive rate (FPR)*, *true positive rate (TPR)*, *precision*, and *accuracy*.

The FPR corresponds to the ratio between incorrectly classified negative data items and the total number of negative data items. The FPR is sometimes also called false alarm rate or fall-out.

$$\text{FPR} = \frac{FP}{FP + TN} \tag{5.1}$$

The TPR corresponds to the ratio between correctly classified positive data items and the total number of positive data items. The TPR is sometimes also called hit rate, sensitivity, or recall.

$$\text{TPR} = \frac{TP}{TP + FN} \tag{5.2}$$

The precision corresponds to the ratio between correctly classified positive data items and the total number of positive classified data items. The precision is sometimes also called positive predictive value.

$$\text{precision} = \frac{TP}{TP + FP} \tag{5.3}$$

The accuracy corresponds to the ratio between correctly classified data items and the total number of classified data items.

$$\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{5.4}$$

As stated by Fawcett [Faw06], an ROC graph is a two-dimensional plot of the true positive and false positive rates. Accordingly, an ROC graph can be used to depict the performance of classifiers. The *area under an ROC curve (AUC)* [HM82] is commonly used to compare the performance of classifiers. The AUC is a single scalar value and represents the expected performance (see also [Run10]).

5.2 Experimental Setup

The experimental setup is used to evaluate the presented heterogeneous multi-class anomaly detection approach and to compare it with selected data stream anomaly detection approaches. Therefore, the anomaly detection performance and the time-efficiency of each anomaly detection algorithm are measured. The experimental setup is based on the KDC (cf. Chapter 3). The aforementioned KDC implementation, initially described in the author's publication [NSS13a], is used to perform the experiments. An extended version has been described by Kaltschmidt [Kal13].

The time-efficiency of each data stream anomaly detection algorithm and each single one-class classifier algorithm are determined empirically by means of a wall-clock time. Obviously, the measurements of wall-clock times are not very precise. Accordingly, the average of ten runs is calculated in order to determine a reasonably precise time-efficiency.

As depicted in Figure 35, the implementation comprises an offline and an online subcycle. Each of both subcycles is represented by a dedicated computing system. Both dedicated computing systems are weakly coupled, and the communication takes place by an external network.

The offline subcycle comprises a DBMS (PostgreSQL [Pos14]) which is used to store the training data. The offline subcycle is used to train the anomaly detection algorithms if a referring algorithm requires offline training. Preprocessing, clustering, and classifier training are realized using MATLAB [Mat14] along with the DDTools [Tax12] and PRTools [DJP+07] packages. Thereafter, the trained models are stored in the same database.

The communication between both dedicated computing systems is implemented by a protocol. The offline subcycle provides a Java-based implementation to retrieve the trained models from the database and to register these models in the online subcycle. Moreover, the offline subcycle provides the functionality to generate continuous data in order to simulate data streams. Therefore, unlabeled and labeled data items (when labeled data items are required for online training) are retrieved from the database and sent to the online subcycle using the implemented protocol. For the offline subcycle, an off-the-shelf computer system (Intel Core i3 with 2.26 GHz and 4 GB memory) and the Windows 7 personal computer operating system are used.

The online subcycle provides the counterpart of the protocol implementation. When the online subcycle is started, it awaits the handshake with the offline subcycle. If the communication is established, the online subcycle waits for the registration of a data stream and the requested anomaly detection algorithms. Finally, when all necessary parts are registered, the online subcycle awaits the initialization of the

data stream. The anomaly detection algorithms produce results which are sent to the offline subcycle by the protocol. The retrieved results are stored in the aforementioned database. The online subcycle is implemented by means of Java. The Raspberry Pi [Ras13] is used as a target machine to simulate restricted system resources aboard an MCPS. The Raspberry Pi provides a low budget ARM processor (700 MHz with 512 MB memory), and a Debian GNU/Linux wheezy is used as an operating system.

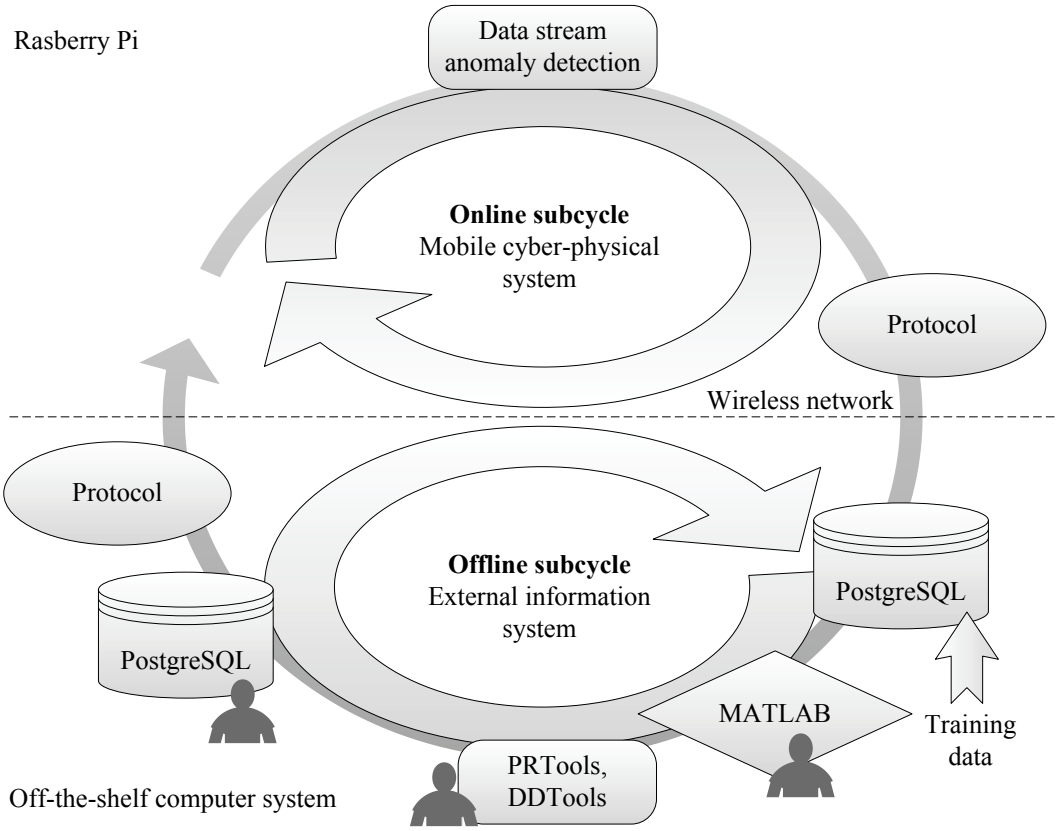

Figure 35: The experimental setup

5.3 Selected Data Stream Anomaly Detection Algorithms

A brief overview of existing data stream anomaly detection algorithms is already given in Section 4.4 on page 69. The current section aims to extend this brief overview in order to give additional implementation-specific information about selected data stream anomaly detection algorithms, which are subsequently used for

comparison with the presented multi-class data stream anomaly detection approach. These include the *k-means one-class classifier*, HT, and HST. Finally, the selected approaches are compared against each other and with the presented approach.

5.3.1 K-means One-Class Classification

The k-means one-class classifier resembles the k-centers one-class classifier (cf. Section 4.8.2). As stated by Tax [Tax01], the important difference refers to the error which is minimized. The k-centers method uses medoids and tries to optimize the centers and the radii of the balls in order to describe the entire training data set. In contrast, the k-means one-class classifier uses centroids, and the distances to the centroids of all objects are averaged. Accordingly, the k-means one-class classifier is more robust against outliers in the training data set.

The OLINDDA algorithm recognizes the problem of novel class detection from a one-class classification perspective and is based on the k-means one-class classifier. Moreover, the OLINDDA approach aims to extend the k-means one-class classifier approach by gradual change (cf. Section 1.3) and novel class detection. Since novel class detection is not considered by the presented multi-class classification approach, the k-means one-class classifier is used separately in this thesis.

There is a difference between the k-means one-class classifier approaches described by Tax [Tax01] and Spinosa et al. [SdLFdCG07]. The k-means one-class classifier method described by Tax [Tax01] uses equal radii for all balls. The k-means one-class classifier method described by Spinosa et al. [SdLFdCG07] uses a different radius for each ball. For the OLINDDA approach, a radius is defined by the calculation of the maximum distance from all data items belonging to a cluster to the referring cluster centroid. By definition, the k-means one-class classifier can be used for data stream anomaly detection. However, multi-class classification is not supported by a single k-means one-class classifier.

5.3.2 Hoeffding trees

The HT (cf. Section 4.4) approach is used for comparison with the presented data stream anomaly detection approach. The HT approach is a widely accepted data stream processing algorithm, and the implementation is taken from the MOA framework [BHKP10]. To predict a referring class, the HT algorithm outputs a *prediction value* for each class of a multi-class classification problem. However, the HT algorithm is not an a priori anomaly detection algorithm. In order to apply the HT approach to an anomaly detection problem, it is necessary to calculate an appropriate *anomaly score* which is independent from the given data set. In this thesis, the anomaly score is calculated as follows. First, the *maximal* and *minimal* prediction

values concerning all correctly classified data items are obtained. Both values are used to normalize the classification result of an unlabeled data item in order to obtain an affiliation probability. The 'maximal prediction probability', corresponding to 100%, constitutes full affiliation to a class, while the 'minimal prediction probability', proportional to the maximal prediction value, constitutes the lowest affiliation to a class used as threshold. An unlabeled data item is classified as an anomaly if the calculated anomaly score is below this threshold.

5.3.3 Half-Space Trees

Moreover, the HST (cf. Section 4.4) approach is used for comparison with the presented data stream anomaly detection approach. As stated by Tan et al. [TTL11], the HST approach requires three important input parameters. The first input parameter specifies the maximum depth of the created trees to capture the data profile. The second input parameter specifies the number of trees to construct. Finally, the third input parameter specifies the size of the update window. The implementation used in this thesis has been described by Kaltschmidt [Kal13] and was inspired by the original HST implementation [Tan14].

5.3.4 Presented Approach

Listing 5.1 shows the implementation of the presented heterogeneous multi-class data stream anomaly detection algorithm (cf. Section 4.7.1) without the application of the filter function. Depending on a specific data set and monitoring objective (cf. Section 2.10.4), various sets of one-class classifiers can be used differently.

```
1  INPUT unlabeled data item
2  OUTPUT labeled data item
3  REPEAT
4    GET next data item from stream
5    SET label of the data item to anomaly
6    FOR each registered one-class classifier
7      CALL classify data item
8      IF classification result is true THEN
9        SET label of the data item equal to the name
10           of the current class
11       EXIT LOOP
12     END IF
13   END FOR
14 UNTIL stream or algorithm stopped
```

List. 5.1: Implementation of the presented anomaly detection algorithm

Moreover, Listing 5.2 shows the same implementation, including the filter function (cf. Section 4.7.2). Rounding errors, which influence the classification result, can occur while the filter function is applied. Thus, the implementation of the filter function includes a tolerance range in order to compensate rounding errors. If the filter function is calculated properly, it does not influence the anomaly detection performances.

```
1  INPUT unlabeled data item
2  OUTPUT labeled data item
3  REPEAT
4    GET next data item from stream
5    SET label of the data item to anomaly
6    CALL filter function
7    FOR each one-class classifier of the candidate chain
8      CALL classify data item
9      IF classification result is true THEN
10       SET label of the data item equal to the name
11           of the current class
12       EXIT LOOP
13     END IF
14   END FOR
15 UNTIL stream or algorithm stopped
```

List. 5.2: Extended implementation (including the filter function)

5.3.5 Comparison

The selected anomaly detection approaches (k-means one-class classifier, HT, and HST) and the presented multi-class data stream anomaly detection approach are very different with respect to the fundamental assumptions. Consequently, the comparison of the presented approach with the selected anomaly detection approaches is very difficult. Table 8 summarizes the properties of the selected algorithms and the previously presented approach. The k-means one-class classifier is an offline training method; it is offline adaptive, and multi-class classification is not supported. The HT is an online training method; it is online adaptive, and multi-class classification is supported. The HST is an online training method; it is online adaptive, and multi-class classification is not supported. The presented multi-class data stream anomaly detection approach is an offline training method; it is offline adaptive, and classification is supported.

The online training methods (HT along with HST) neglect the existence of external information systems and assume that a data stream cannot be stored completely. These online training methods provide the ability for training the anomaly detection model and, simultaneously, for classifying unlabeled data items in real-time

	k-means	HT	HST	Presented approach
Training	offline	online	online	offline
Adaptivity	offline	online	online	offline
Classification	✗	✓	✗	✓

Table 8: Comparison of selected anomaly detection approaches

or near real-time. For this purpose, online training methods are applied by means of one-pass algorithms, while only a small window-based (cf. Section 2.6.6) set of training data items is available. In the context of online data stream anomaly detection, user interaction and fast algorithms are mutually exclusive. Consequently, the assessment of the resulting model is often neglected by online training methods due to the absence of a reasonable number of training data items, since window functions are applied. Hence, the accuracy is insufficiently proved. Besides, the verification of classifying results by human experts is very difficult. These online training methods require an input of labeled training items during operation and at certain time intervals for training or retraining the anomaly detection model. These online training methods entail three drawbacks for monitoring MCPSs. First, the provision of labeled training data during operation cannot be always guaranteed and it contradicts the functioning of several real world applications. Second, model training or retraining during operation expends computational resources actually envisaged for system monitoring. Third, training such an unevaluated model during operation can cause unforeseeable and critical side effects for the entire monitoring process.

Taking these drawbacks into the account, the current thesis presents a multi-class data stream anomaly detection algorithm. This anomaly detection algorithm is an offline training method (cf. Section 3.4.3), offline adaptive, and based on the previously developed KDC. The reason for this is that monitoring MCPSs is a semi-automatic process, and human experts (e.g. the flight control team) are responsible for decisions and consequent actions. Consequently, the presented multi-class data stream anomaly detection approach intends to provide data stream anomaly detection where the human expert needs to be maintained in the loop.

5.4 Selected Data Sets

This section describes selected data sets used to assess the performance along with the time-efficiency of the presented multi-class anomaly detecting approach. The selected data sets are used to compare the presented anomaly detecting approach

with the selected anomaly detection approaches. The first data set relates to the IRFA of the ISS Columbus air loop (cf. Section 4.5). The second data set also relates to the IRFA and, additionally, it includes the sudden changes (cf. Section 1.3) of the presented failure event (cf. Section 1.4.3). The third data set relates to the full ISS Columbus air loop. The fourth data set relates to a space shuttle [BL13]. Finally, the fifth and the sixth data sets are artificial data sets.

5.4.1 ISS Columbus Air Loop - IRFA

The first data set relates to the IRFA of the ISS Columbus air loop, and a comprehensive description is made in Section 4.5 on page 70. The data set includes three selected attributes: speed, current, and pressure (cf. Section 1.4.3). As summarized in Table 9, the data set comprises seven classes with 6978 data items and is denoted as $\mathcal{X}_{\text{irfa}}$ hereinafter.

	$\lvert\omega_1\rvert$	$\lvert\omega_2\rvert$	$\lvert\omega_3\rvert$	$\lvert\omega_4\rvert$	$\lvert\omega_5\rvert$	$\lvert\omega_6\rvert$	$\lvert\omega_7\rvert$	$\lvert\Omega^{\text{C}}\rvert$	Total
$\mathcal{X}_{\text{irfa}}$	168	43	144	136	3478	743	1779	487	6977

Table 9: Summary of data set $\mathcal{X}_{\text{irfa}}$

5.4.2 ISS Columbus Air Loop - Failure Event

The second data set refers to the same case as the first data set. Additionally, it comprises a set of data items which describe sudden changes and were measured during the failure event of the IRFA (cf. Section 1.4.3). The second data set is denoted as $\mathcal{X}_{\text{failure}}$ hereinafter, and a subset of this data set is depicted in Figure 36. For a better illustration, only three classes (ω_4, ω_5, and ω_6) are shown. The additional data items are anomalies and should be detected reasonably precisely by an anomaly detection algorithm. As summarized in Table 10, the data set $\mathcal{X}_{\text{failure}}$ contains seven classes and 9901 data items.

	$\lvert\omega_1\rvert$	$\lvert\omega_2\rvert$	$\lvert\omega_3\rvert$	$\lvert\omega_4\rvert$	$\lvert\omega_5\rvert$	$\lvert\omega_6\rvert$	$\lvert\omega_7\rvert$	$\lvert\Omega^{\text{C}}\rvert$	Total
$\mathcal{X}_{\text{failure}}$	168	43	144	136	3478	743	1779	3410	9901

Table 10: Summary of data set $\mathcal{X}_{\text{failure}}$

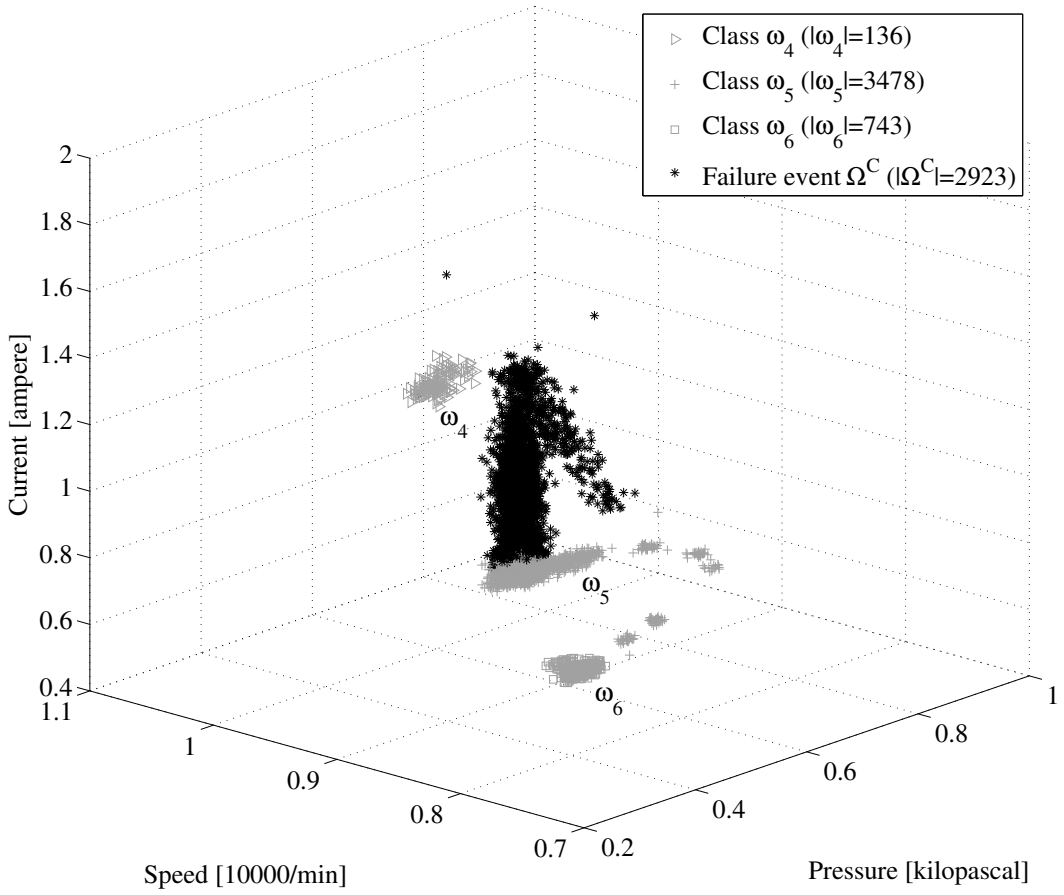

Figure 36: An example of the data set $\mathcal{X}_{\text{failure}}$

5.4.3 ISS Columbus Air Loop - Full

The third data set refers to the full ISS Columbus air loop (cf. Section 1.4.2) and is denoted as $\mathcal{X}_{\text{full}}$ hereinafter. The data set was previously clustered by human experts. To obtain an appropriate training data set, the data is preprocessed, normalized, scaled, and relevant state vectors are selected. The data set includes 16 attributes, and each attribute is ratio scaled (cf. Section 2.2.2). Each fan assembly (CFAs, IRFA, and ISFA) comprises the three attributes: speed, current, and pressure. Additionally, the air pressure and air flow rate at the CHX are measured. More detailed information on the selected data set is provided by Kaufmann [Kau12] and Dietrich [Die12].

The data set comprises five classes $(\omega_{11}, ..., \omega_{15})$. For example, the class ω_{11} refers to a default system state where all fan assemblies are switched off. The classes ω_{12} and ω_{13} refer to default system states where either the first or the second CFA is switched off. The classes ω_{14} and ω_{15} refer to error states where the speed of the ISFA is unusually increased or decreased. The selected anomalies refer to a failure of the ISFA and to a clogging of the return grid. As summarized in Table 10, the data set $\mathcal{X}_{\text{full}}$ contains five classes and 2130 data items. Classification and anomaly detection are the monitoring objectives.

	$\lvert\omega_{11}\rvert$	$\lvert\omega_{12}\rvert$	$\lvert\omega_{13}\rvert$	$\lvert\omega_{14}\rvert$	$\lvert\omega_{15}\rvert$	$\lvert\Omega^{\text{C}}\rvert$	Total
$\mathcal{X}_{\text{full}}$	382	841	256	203	74	374	2130

Table 11: Summary of data set $\mathcal{X}_{\text{full}}$

5.4.4 Space Shuttle

The fourth data set $\mathcal{X}_{\text{shuttle}}$ relates to a space shuttle and is taken from the UCI machine learning repository [BL13]. The original data set contains seven classes. Only three classes are selected in this thesis, while the fourth class is excluded. The data items which belong to the rest of the classes are used as anomalies. Table 12 summarizes this data set.

	$\lvert\omega_{21}\rvert$	$\lvert\omega_{22}\rvert$	$\lvert\omega_{23}\rvert$	$\lvert\Omega^{\text{C}}\rvert$	Total
$\mathcal{X}_{\text{full}}$	1000	132	500	73	1705

Table 12: Summary of data set $\mathcal{X}_{\text{shuttle}}$

5.4.5 Artificial Data Sets

The data sets $\mathcal{X}_{\text{art\_1}}$ and $\mathcal{X}_{\text{art\_2}}$ are generated by means of the DDTools [Tax12] package. For the sake of simplicity, both data sets include only two attributes.

The first artificial data set $\mathcal{X}_{\text{art\_1}}$ comprises six classes, each including 200 data items. As depicted in Figure 37, the shapes of the classes are identically. The anomalies are generated by means of a block-shaped (rectangle or hypercube) uniform distribution covering all classes. Table 13 summarizes the provided classes of the data set $\mathcal{X}_{\text{art\_1}}$, and Figure 37 depicts the described data set.

	$\lvert\omega_{31}\rvert$	$\lvert\omega_{32}\rvert$	$\lvert\omega_{33}\rvert$	$\lvert\omega_{34}\rvert$	$\lvert\omega_{35}\rvert$	$\lvert\omega_{36}\rvert$	$\lvert\Omega^{\mathrm{C}}\rvert$	Total
$\mathcal{X}_{\mathrm{art\_1}}$	200	200	200	200	200	200	268	1468

Table 13: Summary of data set $\mathcal{X}_{\mathrm{art\_1}}$

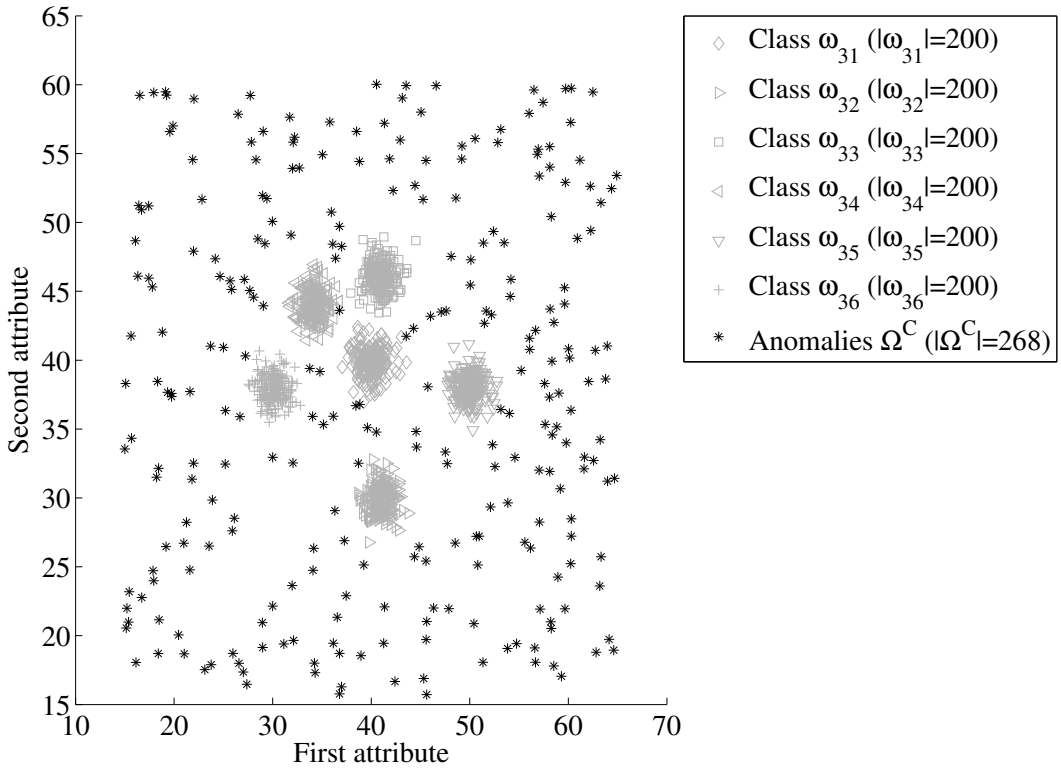

Figure 37: Data set $\mathcal{X}_{\mathrm{art\_1}}$

The second artificial data set $\mathcal{X}_{\mathrm{art\_2}}$ also comprises six classes with very heterogeneous shapes. The anomalies are located between these classes, and it is impossible to separate the classes from the anomalies by parallel lines of axes. Table 14 summarizes the provided classes of the data set $\mathcal{X}_{\mathrm{art\_2}}$, and Figure 38 depicts the described data set.

	$\lvert\omega_{41}\rvert$	$\lvert\omega_{42}\rvert$	$\lvert\omega_{43}\rvert$	$\lvert\omega_{44}\rvert$	$\lvert\omega_{45}\rvert$	$\lvert\omega_{46}\rvert$	$\lvert\Omega^{\mathrm{C}}\rvert$	Total
$\mathcal{X}_{\mathrm{art\_2}}$	150	200	200	150	200	200	360	1460

Table 14: Summary of data set $\mathcal{X}_{\mathrm{art\_2}}$

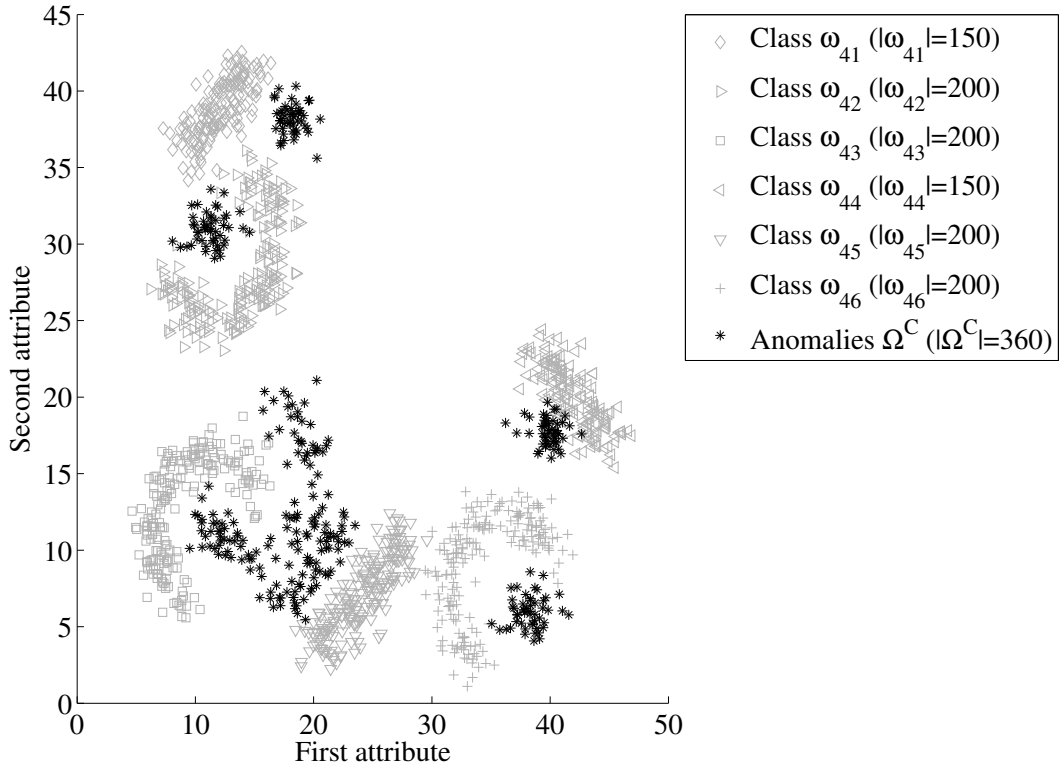

Figure 38: Data set $\mathcal{X}_{\mathrm{art\_2}}$

5.5 Assessments

In this section, the experiments are described in more detail and their results are assessed. First, the anomaly detection performances are discussed. Second, the time-efficiency is discussed.

5.5.1 Anomaly Detection Performance

This section aims to assess the anomaly detection performances of the selected data stream anomaly detection approaches. At first, each data set is considered separately. Thereafter, the performances of the data stream anomaly detection algorithms are compared.

Data Set $\mathcal{X}_{\text{irfa}}$

The data set $\mathcal{X}_{\text{irfa}}$, excluding the anomaly class, is used for classifier training. The k-means one-class classifier is trained and the number k is set to the number of known classes. The HT algorithm does not provide necessary input parameters.

As stated above (cf. Section 5.3.3), the HST approach requires three input parameters. As sated by Tan et al. [TTL11], the maximum depth is a critical parameter, since it used to capture the data profile in a comprehensive manner. Thus, a large number should be preferred. However, this parameter is limited by the present amount of memory. In order to find a trade-off, the variable 'maximum depth' is set to 12. Moreover, the number of trees must be defined. The number of trees is set to 25 (as suggested by Tan et al. [TTL11]). Finally, the size of the update window is set to 10% of the training data set (excluding the anomaly class).

Regarding the presented data stream anomaly detection approach, a single one-class classifier is trained per class. The resulting one-class classifiers are also evaluated by means of a 10-fold cross validation. In accordance with the aforementioned example (cf. Section 4.5), the presented data stream anomaly detection approach aims to provide high classification and anomaly detection performances for all present classes. Accordingly, a well suitable one-class classifier is selected for each class. As summarized in Table 15, this includes two Gaussian one-class classifiers (cf. Section 4.8.1), two nearest neighbor one-class classifiers (cf. Section 4.8.3), a k-centers one-class classifier (cf. Section 4.8.2), and two SVDD one-class classifiers (cf. Section 4.8.4). Based on the measurements presented in Table 15, it is empirically shown that a Gaussian one-class classifier provides the lowest processing time. The processing times of the Gaussian one-class classifiers are approximately equal. The difference occurs due to measurement uncertainties. The processing time of an SVDD one-class classifiers is relatively high and relates to the number of support vectors. Furthermore, the processing time of an NN one-class classifier is very high and increases significantly with the number of training data items. In accordance with the aforementioned Theorem 1 on page 76 (cf. Section 4.7.1), the optimal permutation of the one-class classifiers is as follows: $\omega_7, \omega_5, \omega_6, \omega_1, \omega_3, \omega_2, \omega_4$.

	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7
	Gauss	NN	K-centers	NN	SVDD	SVDD	Gauss
$\lvert\omega_i\rvert$	168	43	144	136	3478	743	1779
p_i	0.026	0.007	0.022	0.021	0.536	0.114	0.274
t_i in ms	0.627	4.469	1.831	19.785	3.369	2.143	0.5918
$p_i/t_i \cdot 10^2$	4.131	0.148	1.212	0.106	15.903	5.341	46.312

Table 15: Summary of one-class classifiers

Table 16 summarizes the performances of all anomaly detection algorithms for the data set $\mathcal{X}_{\text{irfa}}$.

		FPR	TPR	precision	accuracy	AUC
$\mathcal{X}_{\text{irfa}}$	K-means	0.086	0.891	0.438	0.912	0.903
	HT	0	0.620	1.000	0.974	0.810
	HST	0.149	0.959	0.325	0.858	0.905
	Presented approach	0.007	0.906	0.906	0.987	0.949

Table 16: Summary of the anomaly detection performances $\mathcal{X}_{\text{irfa}}$

Data Set $\mathcal{X}_{\text{failure}}$

The data set $\mathcal{X}_{\text{failure}}$, excluding the anomaly class, is used for classifier training. The previously described settings of the data set $\mathcal{X}_{\text{irfa}}$ are used for all data stream anomaly detection algorithms. Table 17 summarizes the performances of all anomaly detection algorithms for the data set $\mathcal{X}_{\text{failure}}$.

		FPR	TPR	precision	accuracy	AUC
$\mathcal{X}_{\text{failure}}$	K-means	0.086	0.833	0.836	0.886	0.873
	HT	0	0.089	1.000	0.686	0.544
	HST	0.152	0.158	0.354	0.610	0.503
	Presented approach	0.007	0.876	0.985	0.953	0.934

Table 17: Summary of the anomaly detection performances $\mathcal{X}_{\text{failure}}$

Data Set $\mathcal{X}_{\text{full}}$

The data set $\mathcal{X}_{\text{full}}$, excluding the anomaly class, is used for classifier training. The settings of the selected data stream anomaly detection algorithms are the same as for the data set $\mathcal{X}_{\text{irfa}}$.

For the presented data stream anomaly detection approach, the setting entails two k-centers one-class classifiers (cf. Section 4.8.2), two SVDD one-class classifiers (cf. 4.8.4), and one NN one-class classifier (cf. Section 4.8.3). It turns out that the Gaussian one-class classifier (cf. Section 4.8.1) is not applicable with the given data set. Table 18 summarizes the performances of all anomaly detection algorithms for the data set $\mathcal{X}_{\text{full}}$.

$\mathcal{X}_{\text{full}}$	FPR	TPR	precision	accuracy	AUC
K-means	0.043	0.516	0.720	0.880	0.737
HT	1.000	1.000	0.176	0.176	0.500
HST	0.155	0.190	0.207	0.730	0.518
Presented approach	0.132	1.000	0.618	0.892	0.934

Table 18: Summary of the anomaly detection performances $\mathcal{X}_{\text{full}}$

Data Set $\mathcal{X}_{\text{shuttle}}$

The data set $\mathcal{X}_{\text{shuttle}}$, excluding the anomaly class, is used for classifier training. The settings of the selected data stream anomaly detection algorithms are the same as for the data set $\mathcal{X}_{\text{irfa}}$. Regarding the presented data stream anomaly detection approach, an SVDD one-class classifier (cf. Section 4.8.4) is used three times. Table 19 summarizes the performances of all anomaly detection algorithms for the data set $\mathcal{X}_{\text{shuttle}}$.

$\mathcal{X}_{\text{shuttle}}$	FPR	TPR	precision	accuracy	AUC
K-means	0.002	0.055	0.571	0.958	0.527
HT	1.000	1.000	0.043	0.043	0.500
HST	0.074	0.685	0.292	0.916	0.805
Presented approach	0.021	1.000	0.680	0.980	0.989

Table 19: Summary of the anomaly detection performances $\mathcal{X}_{\text{shuttle}}$

Data Set $\mathcal{X}_{\text{art\_1}}$

The data set $\mathcal{X}_{\text{art\_1}}$, excluding the anomaly class, is used for classifier training. The settings of the selected data stream anomaly detection algorithms are the same as for the data set $\mathcal{X}_{\text{irfa}}$.

Regarding the presented data stream anomaly detection approach, all classes are trained by means of a Gaussian one-class classifier (cf. Section 4.8.1). Table 20 summarizes the performances of all anomaly detection algorithms for the data set $\mathcal{X}_{\text{art\_1}}$.

		FPR	TPR	precision	accuracy	AUC
$\mathcal{X}_{\text{art\_1}}$	K-means	0.649	0.996	0.255	0.469	0.674
	HT	1.000	1.000	0.183	0.183	0.500
	HST	0.233	0.735	0.414	0.762	0.751
	Presented approach	0.004	1.000	0.982	0.997	0.998

Table 20: Summary of the anomaly detection performances $\mathcal{X}_{\text{art\_1}}$

Data Set $\mathcal{X}_{\text{art\_2}}$

The data set $\mathcal{X}_{\text{art\_2}}$, excluding the anomaly class, is used for classifier training. The settings of the selected data stream anomaly detection algorithms are the same as for the data set $\mathcal{X}_{\text{irfa}}$.

Regarding the presented data stream anomaly detection approach, all classes are trained by means of an SVDD one-class classifier (cf. Section 4.8.4). Table 21 summarizes the performances of all anomaly detection algorithms for the data set $\mathcal{X}_{\text{art\_2}}$.

		FPR	TPR	precision	accuracy	AUC
$\mathcal{X}_{\text{art\_2}}$	K-means	0.013	0.167	0.811	0.785	0.577
	HT	0.020	0.022	0.267	0.744	0.501
	HST	0.468	0.736	0.340	0.582	0.634
	Presented approach	0.318	1.000	0.507	0.760	0.841

Table 21: Summary of the anomaly detection performances $\mathcal{X}_{\text{art\_2}}$

Summary of the Anomaly Detection Performances

As summarized in Figure 39, the k-means one-class classifier (cf. Section 5.3.1) provides appropriate anomaly detection performances for the data sets $\mathcal{X}_{\text{irfa}}$ and $\mathcal{X}_{\text{failure}}$. The anomaly detection performance for the data set $\mathcal{X}_{\text{full}}$ is very low. For the rest of the data sets, the k-means one-class classifier practically does not detect anomalies, and the anomaly detection performances are very poor. However, the number k of the k-means one-class classifier is set to the number of classes. The anomaly detection performance of the k-means one-class classifier might be improved by the selection of a better choice of the number k. Amongst others, possible approaches might be the usage of micro-clusters [AHWY03] or the application of an evolutionary k-means algorithm [NCHC11].

The HT (cf. Section 5.3.2) approach provides a low anomaly detection performance for the data sets $\mathcal{X}_{\text{irfa}}$ and $\mathcal{X}_{\text{art\_1}}$. For the rest of the data sets, the HT approach practically does not detect anomalies, and the anomaly detection performances are very poor. The HT approach is not intrinsically developed for anomaly detection. In this thesis, the anomaly score relates to a calculated threshold. To improve the anomaly detection performance of the HT approach, it might be necessary to improve the calculation of the anomaly score or to reduce the threshold.

The HST (cf. Section 5.3.3) approach provides an appropriate anomaly detection performance for the data set $\mathcal{X}_{\text{irfa}}$. The anomaly detection performances for the data sets $\mathcal{X}_{\text{shuttle}}$ and $\mathcal{X}_{\text{art\_1}}$ are relatively low. For the rest of the data sets, the HST approach practically does not detect anomalies, and the anomaly detection performances are very poor. To improve the anomaly detection performance of the HST approach, it might be necessary to increase the parameter 'maximum depth'.

The presented data stream anomaly detection approach provides appropriate anomaly detection performances for all selected data sets. The presented data stream anomaly detection approach provides the best anomaly detection performance in comparison with the selected data stream anomaly detection approaches. In contrast to the other selected anomaly detection approaches, the presented approach is very flexible and tries to approximate all present classes very precisely in order to provide appropriate anomaly detection performances. However, the selected data stream anomaly detection algorithms are trained automatically. This fact is underscored by the experimental results. However, the good results come at the expense of expert knowledge being used for classifier training.

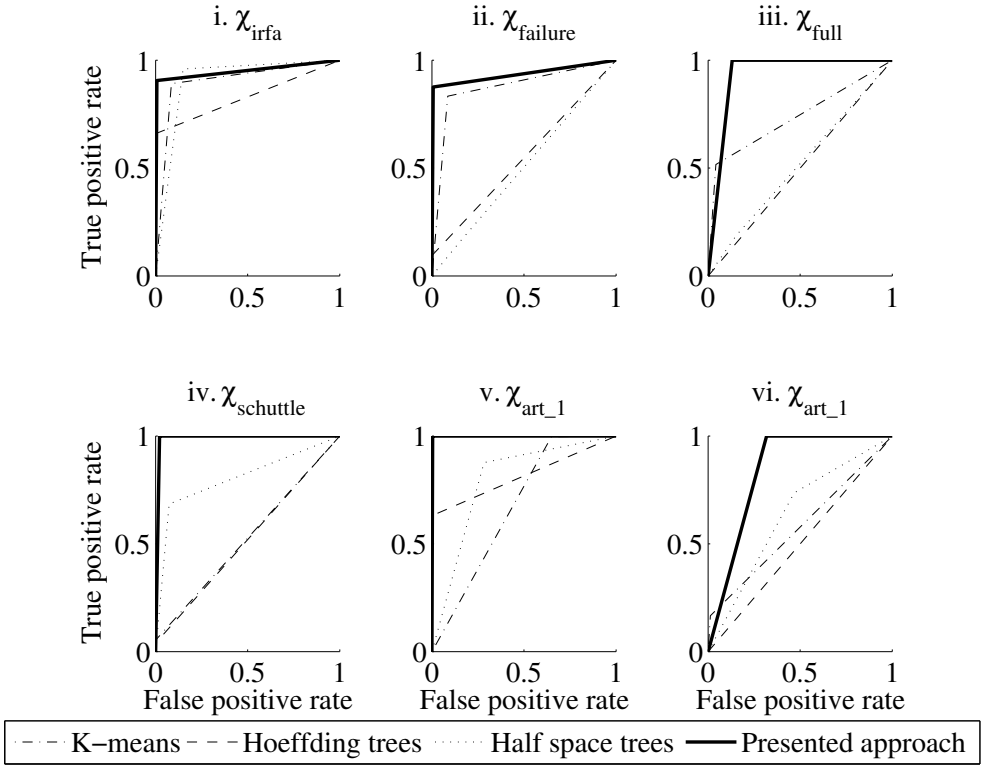

Figure 39: ROC curves

5.5.2 Summary of the Time-efficiency

This section focuses on the the time-efficiency of the selected anomaly detection algorithms. The aforementioned settings are used identically for each data set. Table 22 and Figure 40 summarize the experimental results. The processing times are measured in milliseconds and relate to the average processing time that the anomaly detection algorithm needs to process a single data item.

As summarized in Table 22 and as shown in Figure 40, the k-means one-class classifier requires the lowest average processing time. The HT approach also requires low average processing time. The HST requires the highest average processing time. The average processing time of the presented data stream anomaly detection approach is listed twice. At first, the average processing time is measured without the filter function. Following that, the average processing time is measured while the filter function is applied. As summarized in Table 22 and in Figure 40, the average

processing time of the presented data stream anomaly detection approach lies in the mid-range, and the application of the filter function can be used to significantly reduce the average processing time.

	$\mathcal{X}_{\text{irfa}}$	$\mathcal{X}_{\text{failure}}$	$\mathcal{X}_{\text{full}}$	$\mathcal{X}_{\text{shuttle}}$	$\mathcal{X}_{\text{art\_1}}$	$\mathcal{X}_{\text{art\_2}}$
K-means	0.447	0,585	2.042	0.402	0.663	0.518
HT	4.228	3,242	4.694	4.077	2.827	2.767
HST	44.876	52,625	138.871	109.439	51.075	38.030
Presented approach	5.336	8.869	57.526	7.070	1.749	8.924
Plus filter function	4.117	5.520	26.390	7.530	1.967	3.197

Table 22: Summary of the average processing time per data item in milliseconds

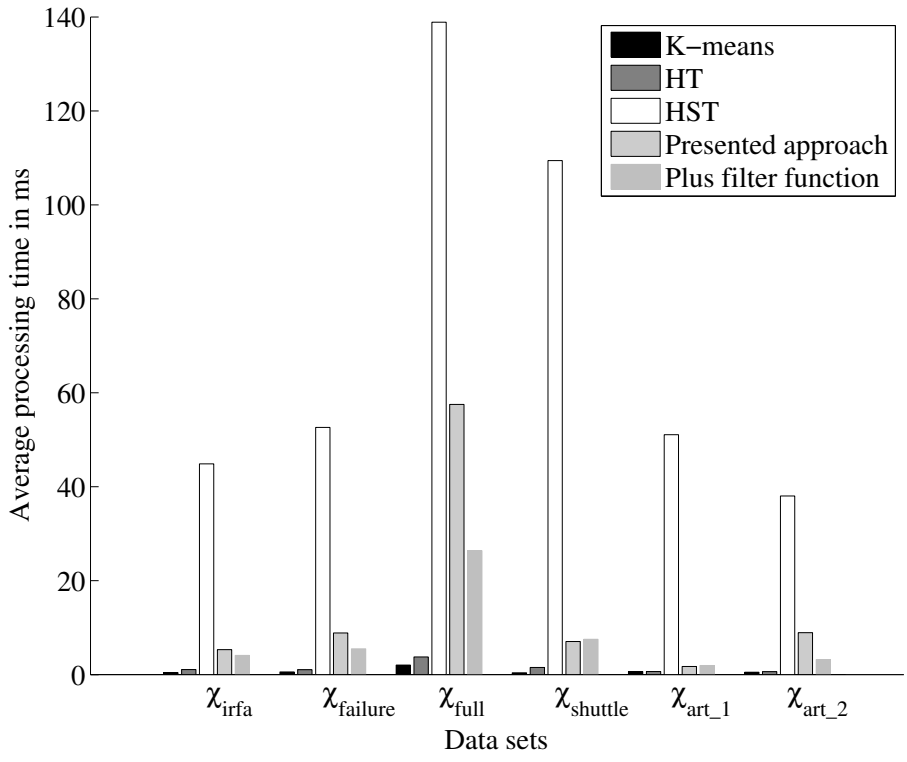

Figure 40: Summary of the average processing time per data item in milliseconds

5.6 Case Study

This section provides a case study of the KDC (cf. Chapter 3), which relates to the aforementioned real world scenario (cf. Section 1.4). The case study is intended to demonstrate the KDC in a real-life situation and to integrate the presented data stream anomaly detection algorithm. Both the offline and the online subcycle are discussed in more detail. The case study is depicted as a float chart in Figure 41 on page 125. This figure aims to extend the aforementioned processing steps of the KDC (cf. Section 3.5) by additional decisions.

5.6.1 Offline Subcycle

The offline subcycle starts with the first processing step: persistent storage.

1. **Persistent storage**: Initially, the persistent storage does not contain any content. In order to perform this case study, a data set relating to the ISS Columbus air loop (cf. Section 4.5) is imported.

2. **Preprocessing**: As depicted in Figure 41 on page 125, the second processing step is executed whenever new data is available. As mentioned earlier in Section 3.5.2 on page 54, preprocessing is used to select and transform relevant data. For the sake of simplicity, only data referring to the IRFA is used. Henceforth, the data set $\mathcal{X}_{\text{irfa}}$ (cf. Section 5.4.1) is used.

3. **Offline analysis**: The third processing step includes clustering and classifier training. As described in Section 5.5.1 on page 113, the selected data set is already clustered, and a well suitable one-class classifier is selected for each class. This includes two Gaussian one-class classifiers (cf. Section 4.8.1), two nearest neighbor one-class classifiers (cf. Section 4.8.3), a k-centers one-class classifier (cf. Section 4.8.2), and two SVDD one-class classifiers (cf. Section 4.8.4).

4. **Validation**: The fourth processing step refers to the validation of the obtained classes and classifiers by automated processes and human experts. For example, the one-class classifiers of a heterogeneous multi-class classifier can overlap. One of the fundamental assumptions of the presented data stream anomaly detection approach is that the dichotomous class detectors must be disjoint (cf. Section 4.7). Consequently, human experts must identify the interpretation of such possible intersections. It is necessary to resolve the occurring intersection in some way in order to avoid ambiguous anomaly detection results.

   The good anomaly detection performance of the presented data stream anomaly detection approach comes at the expense of an increased workload. This includes the extraction of expert knowledge by means of clustering, classifier

training, and validation. However, there are several possibilities to automate classifier training and to reduce the human effort, which must be spent on offline analysis and validation. For example, it is possible to identify a well-suited one-class classification algorithm automatically for a considered class. Moreover, parameters required for classifier training can be iteratively improved by a consecutive application of cross validation. The human expert is still responsible for consequent decisions along with resulting actions and needs to be maintained in the loop. As a consequence, currently there is no other choice for a sophisticated monitoring approach in order to provide reliability and avoid critical damage.

As depicted in Figure 41 on page 125, if the validation process cannot be successfully completed or the obtained classifiers must be refined, the second (preprocessing), third (offline analysis), and fourth (validation) processing steps are repeated in iterative loops until the validation process is successfully completed.

5. **Knowledge storage**: The fifth processing step is executed when the validation is completed successfully and refers to the storage of newly derived knowledge. A difficult decision relates to the choice of an adequate storage strategy of the obtained classifiers. For example, this could be implemented by means of relational tables or object references such as *JavaScript object notation (JSON)* or *extensible markup language (XML)*. Furthermore, it is necessary to translate the dichotomous class detectors into query languages. In order to provide a sophisticated monitoring process, these rule sets must be human-readable and processable by means of existing IFP engines.

   For example, Listing 5.3 presents an obtained query of a Gaussian one-class classifier (cf. Section 4.8.1) using the *event processing language (EPL)* provided by the Esper CEP engine [Esp13]. Moreover, Listing 5.4 presents an EPL query of a SVDD one-class classifier (cf. Section 4.8.4), which includes the RBF kernel function. For the sake of simplicity, both queries comprise only two selected attributes: pressure and current. Each presented query outputs a result whenever an unlabeled state vector is classified as anomaly. Consequently, these queries can be arranged in an appropriate execution sequence. Referring to Theorem 1 on page 76 (cf. Section 4.7.1), the appropriate execution sequence refers to the optimal permutation.

   Obvious disadvantages of such rule sets are the increased complexity and unreadability. The complexity and unreadability of these rule sets increase enormously with the number of selected attributes (dimensionality) and when using more complex one-class classifiers. However, the application of user-defined functions, which are provided by a variety of query languages such as EPL,

can help to reduce the complexity and improve the readability of these rule sets. For example, Listing 5.5 provides an EPL query where the classifier is replaced by a user-defined function.

```
1  select pressure, current
2  from   data_stream
3  where
4  ( ( ((Σ^-1_1,1 * (pressure - x̄_1)) +
5       (Σ^-1_1,2 * (current - x̄_2)) )
6    * (pressure - x̄_1) ) +
7    ( ((Σ^-1_2,1 * (pressure - x̄_1)) +
8       (Σ^-1_2,2 * (current - x̄_2)) )
9    * (current - x̄_2) )
10 ) > θ_Mahal
```

List. 5.3: Translation of a Gaussian one-class classifier into EPL [NS12b]

```
1  select pressure, current
2  from   data_stream
3  where  (
4   (α_1 * java.lang.Math.exp(-1* (
5     ((pressure - x_1,1) * (pressure - x_1,1))
6     + ((current - x_1,2) *
7     (current - x_1,2)) )/σ^2 )) +
8   (α_2 * java.lang.Math.exp(-1* (
9     ((pressure - x_2,1) * (pressure - x_2,1))
10    + ((current - x_2,2) *
11    (current - x_2,2)) )/σ^2 )) +
12  (α_3 * java.lang.Math.exp(-1* (
13    ((pressure - x_3,1) * (pressure - x_3,1))
14    + ((current - x_3,2) *
15    (current - x_3,2)) )/σ^2 )) +
16
17    ⋮ (many more lines)
18
19 ) > θ_SVDD_RBF
```

List. 5.4: Translation of a SVDD one-class classifier into EPL [NS12b]

```
1  select pressure, current
2  from   data_stream
3  where  classifier.classify(pressure, current) > θ
```

List. 5.5: Application of a user-defined function

6. **Knowledge transfer**: As mentioned before, the sixth processing step is the last step of the offline subcycle and is used to synchronize the offline subcycle with the online subcycle. Newly derived knowledge is transferred to the online subcycle on condition that the external network is available.

5.6.2 Online Subcycle

The online subcycle starts with a decision. Whenever new knowledge is available, the following processing steps are adapted: preprocessing (8), online analysis (9), actions (10), temporal storage (11), and data or event transfer (12). When no new knowledge is available or the adaption is successfully finished, it follows the seventh processing step: streaming inputs.

7. **Streaming inputs**: The seventh processing step refers to the streaming inputs approaching from heterogeneous data sources (e.g. sensors). For example, these data sources can provide tuple-like data streams, whereas a tuple is interpreted as a single event. Additionally, it is also possible that more complex data streams are provided. This includes complex events such as objects (e.g. identification of incorrect system behavior or collision detection). In the present case study, the simulated data stream is a tuple-like data stream, and each arriving tuple is construed as a state vector.

8. **Preprocessing**: As mentioned before, preprocessing includes the extraction of relevant data along with the extraction of specific data items or complex events. Preprocessing can be implemented by means of select statements or by stand-alone rules. In this case study, each attribute is an element of the set of real numbers. Additionally, it could be useful to apply principal component analysis [Pea01] in order to reduce the dimensionality (number of attributes).

9. **Online analysis**: As mentioned earlier, the ninth processing step constitutes the core of the online subcycle and the real-time monitoring process. Online analysis reflects automatic, online, and real-time data processing. This includes classification, data stream mining, or a combination of both. For example, classification can be applied by using rule sets. As discussed in the sections on reference architectures of DSMS (cf. Section 2.6.3) and IFP (cf. Section 2.6.4), a component is required in order to store these rule sets. Classification, which is based on offline training methods, presupposes a previous pass through of the offline subcycle. This previous pass through is required for long-term analysis, knowledge extraction, rule set generation, and transfer to the online subcycle. Online training methods are able to extract knowledge without pre-existing knowledge and without any previous pass through of the offline subcycle. The main intention of the online analysis is the assessment

of the current system behavior by the use of single events. Based on these single events, it is possible to detect complex events (e.g. failure and anomaly detection).

This case study is based on the Esper CEP engine, and the aforementioned one-class classifiers are applied. Accordingly, it is possible to classify unlabeled state vectors as belonging to one of the known classes or as anomaly.

10. **Actions**: The tenth processing step is related to the previous processing step. The detection of events necessitates the initiation of actions. Those actions can be used to send alarm messages or to effect hardware (e.g. toggling of switches). Actions could also include the preparation of data or event metrics. In this case study, the action part is represented using a listener. The listener simply writes a message on the screen if an anomaly is detected.

    The processing step 'actions' is followed by another decision, which refers to the adaption of the online subcycle. As described in Section 3.5.1 on page 52, the online subcycle is a loop, which can be used to further adapt the processing steps of the online subcycle. Hence, the online subcycle is asynchronous and decoupled from the offline subcycle. Accordingly, it is possible to automatically adapt anomaly detection algorithms in iterative loops. For example, the presented data stream anomaly detection approach is based on an expected probability of each applied dichotomous class detector. However, the probability distribution can change during operation. In this case, it becomes necessary to rearrange the dichotomous class detectors in order to provide the optimal permutation. Therefore, it is necessary to count the frequency of the arriving data items and the classification results. Such frequency counts have been discussed by Manku [MM02].

11. **Temporal storage**: The eleventh processing step reflects the temporal storage on the online subcycle. The temporal storage is a window-based, preliminary, and short-term storage. This involves storage of data items along with the storage of single events, complex events, and metrics. In this case study, each state vector is simply stored into a data base. This includes a label which includes the classification result for each data item.

12. **Data and event transfer**: The twelfth processing step is the last step of the online subcycle. It reflects the synchronization and transmission of temporally stored data, events, and metrics from the online subcycle into a persistent memory on the offline subcycle. Data and event transfer is an optional processing step executed whenever a data transfer is necessary (e.g. near exhaustion of the temporal storage). For example, the initiation of data and

event transfer could be implemented by action rules. However, the initiation depends on prerequisites such as the availability of the external network.

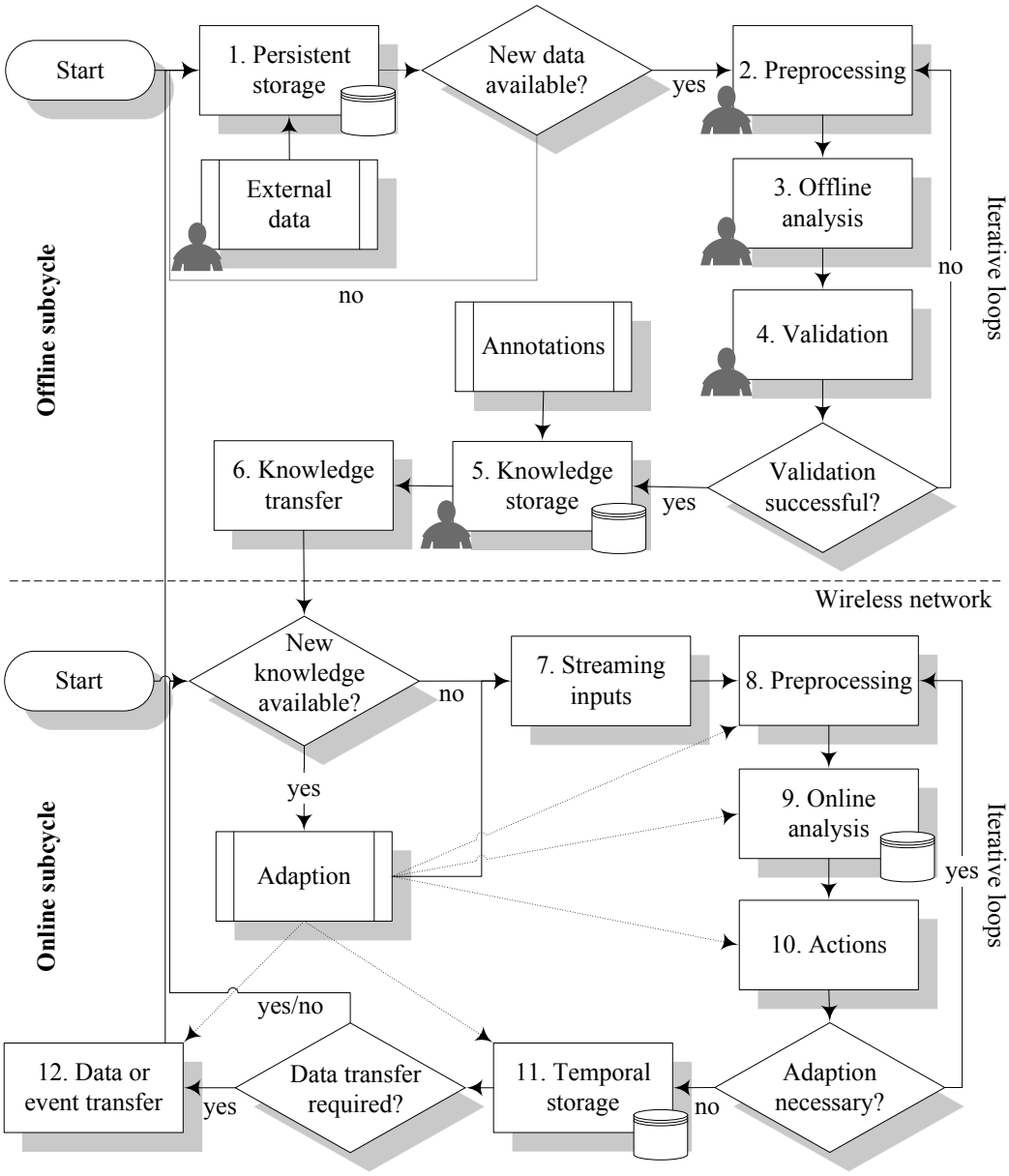

Figure 41: The case study

5.7 Conclusion

In this chapter, the evaluation scheme (cf. Section 5.1) and the experimental setup (cf. Section 5.2) used to perform the experiments have been explained. Moreover, a set of three data stream anomaly detection algorithms have been selected (cf. Section 5.3) which were subsequently used for comparisons with the presented data stream anomaly detection approach. These algorithms include the k-means one-class classifier (cf. Section 5.3.1), the HT approach (cf. Section 5.3.2), and the HST approach (cf. Section 5.3.3). Furthermore, six data sets have been selected and described in more detail (cf. Section 5.4). As a result, the presented data stream anomaly detection algorithm provides a very good anomaly detection performance and outperforms the selected data stream anomaly detection algorithms (cf. Section 5.5.1). Moreover, the time-efficiency of the presented data stream anomaly detection algorithm lies in the mid-range (cf. Section 5.5.2). In contrast to the k-means one-class classifier and the HT approach, the good anomaly detection performance of the presented data stream anomaly detection algorithm comes at the expense of a decreased time-efficiency.

Furthermore, a case study of the KDC has been presented in this chapter (cf. Section 5.6). This case study is related to the aforementioned real world scenario (cf. Section 1.4). The presented case study demonstrates the KDC in a real-life situation and integrates the presented data stream anomaly detection algorithm into the KDC.

CHAPTER 6

Conclusions and Future Work

"Master books, but do not let them master you. Read to live, not live to read."

Edward Robert Lytton Bulwer-Lytton (1831-1891)

This chapter summarizes the individual parts of this thesis and the results of experiments. Moreover, several remaining problems and possible subjects for future works are discussed.

6.1 Summary of the Thesis

As described in more detail in Section 1.6 on page 12, Section 1.7 on page 13, and Section 1.8 on page 14, the contribution of this work is to combine the KDD process with the KDDS process for monitoring MCPSs. Therefore, a KDC described in Chapter 3 on page 41 has been developed. Based on this, a new data stream anomaly detection algorithm has been presented in Chapter 4 on page 63. Finally, experiments and a case study have been presented in Chapter 5 on page 99.

The foundations of this thesis are described in Chapter 2 on page 17. The foundations include a brief description of the KDD process model (cf. Section 2.4), selected topics of data mining (cf. Section 2.5), selected topics of IFP (cf. Section 2.6), and a detailed view on monitoring (cf. Section 2.10). To the best of the author's knowledge, no concluding process model for the KDDS process exists as yet. Accordingly, the reference architectures of DSMSs (cf. Section 2.6.3) and IFP (cf. Section 2.6.4) are interpreted as preliminary process models for the KDDS process in this thesis. Moreover, related work, including specialized IFP approaches (cf. Section 2.6.8),

the MOA data stream classification cycle (cf. Section 2.7), expert systems (cf. Section 2.8), and the MAPE-K reference model (cf. Section 2.9), has been discussed in Chapter 2 on page 17.

The KDC developed in this thesis is described in Chapter 3 on page 41. Therefore, requirements for monitoring MCPSs (cf. Section 3.1), characteristics of the KDC (cf. Section 3.2), and a relation between key challenges for monitoring MCPSs (cf. Section 1.3) and the KDC characteristics have been identified. In addition, a widely adopted assumption that data streams should not be stored entirely has been discussed (cf. Section 3.4.1). Based on this discussion, a storage-aware stream model (cf. Section 3.4.2) which extends the commonly used stream model (cf. Section 2.6.1) has been presented. Also, training methods have been divided into online, offline, and hybrid training methods (cf. Section 3.4.3). The KDC comprises an online and an offline subcycle. The online subcycle refers to KDDS where online monitoring (cf. Section 2.10.2) is applied in an automatic and real-time manner. The offline subcycle refers to KDD where offline monitoring (cf. Section 2.10.2) is applied for semi-automatic and long-term analysis. Moreover, twelve processing steps have been identified for the KDC (cf. Section 3.5). These include six processing steps for the online subcycle (cf. Section 3.5.1) and six processing steps for the offline subcycle (cf. Section 3.5.2). These processing steps have been assigned to existing concepts which are known from literature. Finally, the aforementioned related work, which includes the MOA data stream classification cycle (cf. Section 2.7), expert systems (cf. Section 2.8), and the MAPE-K reference model (cf. Section 2.9), have been compared with the KDC. As a result, the existing approaches entail deficits for monitoring MCPSs, while the KDC attempts to eliminate these deficits (cf. Section 3.2) and to provide an appropriate monitoring approach for monitoring MCPSs.

Furthermore, a new OAR-based data stream anomaly detection algorithm has been presented in Chapter 4 on page 63. The presented data stream anomaly detection algorithm is based on the KDC (cf. Chapter 3) and the above-mentioned storage-aware stream model (cf. Section 3.4.2). To provide a consistent terminology, a set of system states have been explained in detail (cf. Section 4.1) and a basic anomaly detection model has been presented (cf. Section 4.2). Furthermore, several anomaly detection techniques (cf. Section 4.3) and existing data stream anomaly detection algorithms (cf. Section 4.4) have been discussed. In order to illustrate the circumstances of heterogeneous multi-class anomaly detection, an example of the ISS Columbus air loop (cf. Section 4.5) referring to the presented real world scenario (cf. Section 1.4) has been presented, and a problem statement has been stated (cf. Section 4.6). Based on these, the solution statement (cf. Section 4.7) comprises a detector chain which is based on an OAR-based multi-class anomaly detection approach (cf. Section 2.5.3). Additionally, an approach to minimize the average processing time of this detector chain has been introduced (cf. Section 4.7.1).

In addition, a filter function, which can be applied to reduce the average processing time in the worst case scenario, has been presented (cf. Section 4.7.2). Finally, a list of selected one-class classifiers for anomaly detection has been introduced (cf. Section 4.8).

Experiments and a case study have been discussed in Chapter 5 on page 99. Therefore, the evaluation scheme (cf. Section 5.1), the experimental setup (cf. Section 5.2), additional implementation-specific information about selected data stream anomaly detection algorithms (cf. Section 5.3), and selected data sets (cf. Section 5.4) have been explained. Furthermore, the selected data stream anomaly detection algorithms along with the presented data stream anomaly detection approach have been assessed and compared to each other. The anomaly detection performances have been assessed and compared first, followed by the time-efficiency. As a result of the comparison of the anomaly detection performances, the anomaly detection approach presented in this thesis outperforms the selected data stream anomaly detection algorithms. As a result of the comparison of time-efficiency, the presented anomaly detection approach comes at the expenses of time consumption.

Finally, the KDC and the presented data stream anomaly detection approach have been accepted as an experiment [NSS13b] of the *ESA OPS-SAT laboratory* [ESA14]. The ESA OPS-SAT laboratory aims to provide an in-orbit test-bed to promote advanced innovations for future space missions.

6.2 Subjects for Future Work

Requirements for monitoring MCPSs have been described in Section 3.1 on page 41. The requirements 'time', 'locality', 'knowledge', and 'system resources' have been recognized in detail by the KDC (cf. Chapter 3) and the presented data stream anomaly detection approach. However, the requirement 'sharpness' has not been recognized, and it is necessary to integrate this requirement into the presented data stream anomaly detection approach.

Moreover, the KDC has been developed under the assumption that only one MCPS is monitored. However, in many real world scenarios a large number of MCPSs must be monitored at the same time. Consequently, it becomes necessary to combine several KDC implementations in order to integrate the discovered knowledge and to provide a more sophisticated monitoring approach for all target systems. This combination could be applied by a side by side implementation of several KDCs or by extending the KDC processing steps into a larger cycle.

The offline subcycle of the KDC is characterized by a large involvement of human experts. In the future, it could be possible to automate some processing steps and to provide a dynamic history analysis in order to further reduce the effort which must

be provided by human experts. Additionally, it becomes necessary to develop new concepts to provide a sustainable use of the acquired data and derived knowledge for research and future space missions.

The basic data stream anomaly detection model described in Section 4.2 on page 66 denotes the time as an index. Consequently, transitions are not considered. However, it is possible to increase the complexity of the described data stream anomaly detection model and to consider the time as an additional dimension. Then trajectories arise in the vector space (cf. Section 4.2) and it becomes possible to monitor state transitions.

To monitor MCPSs it is necessary to assess the reliability of data stream mining and data stream anomaly detection algorithms. Thus, a quality of service of the algorithmic description and the resulting implementation must be provided by these algorithms in order to avoid critical damage.

Furthermore, it is also possible to extend the presented data stream anomaly detection approach by means of novel class detection. For instance, it might be possible to combine the the presented data stream anomaly detection approach with the OLINDDA approach for novel class detection. Furthermore, it is also possible to rearrange the dichotomous class detectors in order to provide the optimal permutation dynamically during operation.

The performed case study (cf. Section 5.6) presents challenges and open problems for further research. This discussion on further research is closely related to the author's publications [NS12b] and [NS13]. The application of data stream processing on the online subcycle demands lightweight IFP engines which are able to work under resource restrictions. These lightweight IFP engines should provide language expressions for crisp and non-crisp processing of conditions and should be able to integrate data stream mining algorithms such as data stream clustering, classification, or anomaly detection. Moreover, these IFP engines should provide the aforementioned training methods: online, offline, and hybrid training (cf. Section 3.4.3). For example, MOA [BHKP10] already provides online and offline training methods. Although, MOA is not designed to easily handle resource restrictions of MCPSs. Basic examples of such lightweight IFP engines are SwissQM [MAK07] or a data stream management system described by Wolf et al. [WR08]. Furthermore, MOA does not provide an appropriate query language as claimed by the eight requirements for real-time stream processing [ScZ05].

A variety of data stream query languages are already present, however, there exists a lack of standardization [PVS11]. Moreover, the translation of a specific data stream mining task into existing stream query languages is very challenging. As a solution, it could be possible to develop a new abstract stream query language that enables the

possibility to present the data stream mining task and the data stream query in an appropriate manner. Based on this abstract stream query language, an automated translator, which aims to translate the query into several specific data stream query languages, can be built.

Data stream items must be temporally stored on an MCPS. Thus, a lightweight combination of a data stream and event stream warehouse reflecting resource restrictions of MCPSs is required. This combined data and event stream warehouse should be able to provide different storage strategies such as complete and incomplete storage (cf. Section 3.5.1).

Currently, it is impossible to distinguish the ETL process between KDD and KDDS. ETL commonly refers to KDD, whereas data transmitted to the offline subcycle reflects streaming data. Thus, an adequate ETL process, which takes this problem into account, is required.

Data stream mining algorithms can also be applied on the offline subcycle to provide near real-time processing. These algorithms could be massively parallelized onto large computational infrastructures (e.g. computer clusters and grid or cloud computing).

In addition, the adaptation of the online subcycle includes a large number of open problems. How to transmit new queries to an IFP engine without stopping the engine and without evoking critical side effects? First, it is required to define and register streams using a data definition language. Second, it is required to define and register queries by a data manipulation language. In contrast to conventional DBMSs, IFP engines provide continuous queries (cf. Section 2.6.2). Such continuous queries process the data stream continuously. What happens when the stream definition changes during operation? How to register new queries or change and delete already existing queries during operation? Some basic discussions of these problems are provided by Müller et al. [MAK07] and Wolf et al. [WR08]. How to provide transaction protocols and properties such as atomicity, consistency, isolation and durability as known from conventional DBMSs? Furthermore, standardized interfaces such as the open database connectivity or Java database connectivity are not yet available. However, not every query language provides concepts of a data definition language. For example, the event processing language provided by Esper [Esp13] necessitates a Java-based implementation of data streams. Moreover, query languages for stream processing do not provide concepts for procedural extensions such as transact SQL or PL/SQL.

Lastly, an in-depth discussion of open challenges for data stream mining research is provided by Krempl et al. [KvB+14].

Appendix

A Bibliography

[ABW06] A. Arasu, S. Babu, and J. Widom. The CQL Continuous Query Language: Semantic Foundations and Query Execution. *The VLDB Journal*, 15:121–142, 2006.

[ACc+03] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: A New Model and Architecture for Data Stream Management. *The VLDB Journal*, 12:120–139, 2003.

[AFG+10] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A View of Cloud Computing. *Commun. ACM*, 53(4):50–58, 2010.

[Agg07] C. C. Aggarwal, editor. *Data Streams: Models and Algorithms*. Springer, 2007.

[AHWY03] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A Framework for Clustering Evolving Data Streams. In *Proceedings of the 29th international conference on Very large data bases - Volume 29*, VLDB '2003, pages 81–92. VLDB Endowment, 2003.

[ALRL04] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Trans. Dependable Secur. Comput.*, 1:11–33, 2004.

[Bac00] A. Backlund. The Definition of System. *Kybernetes*, 29:444–451, 2000.

[BBD+02] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and Issues in Data Stream Systems. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 1–16. ACM, 2002.

[Bel61] R. Bellmann. *Adaptive Control Processes*. Princeton University Press, 1961.

[BGJ+09] B. Bertsche, P. Göhner, U. Jensen, W. Schinköthe, and H.-J. Wunderlich. *Zuverlässigkeit mechatronischer Systeme*. Springer, 2009.

[BHKP10] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive Online Analysis. *Journal of Machine Learning Research (JMLR)*, 11:1601–1604, 2010.

[BHKP11] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Data Stream Mining - A Practical Approach. Technical report, Centre for Open Software Innovation (COSI) - Waikato University, 2011. Available from: `http://heanet.dl.sourceforge.net/project/moa-datastream/documentation/StreamMining.pdf`.

[Bis93] C. M. Bishop. Novelty Detection and Neural Network Validation. In S. Gielen and B. Kappen, editors, *ICANN*, pages 789–794. Springer, 1993.

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[BKZ10] S. Bell, D. Kortenkamp, and J. Zaientz. A Data Abstraction Architecture for Mission Operations. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2010.

[BL13] K. Bache and M. Lichman. UCI Machine Learning Repository, 2013. Available from: `http://archive.ics.uci.edu/ml`.

[BM00] W. R. Blischke and D. N. P. Murthy. *Reliability: Modeling, Prediction, and Optimization*. John Wiley & Sons, 2000.

[BM06] H. Bauke and S. Mertens. *Cluster Computing*. Springer, 2006.

[Bol09] A. Bolles. A flexible Framework for Multisensor Data Fusion using Data Stream Management Technologies. In *Proceedings of the 2009 EDBT/ICDT Workshops*, EDBT/ICDT '09, pages 193–200. ACM, 2009.

[Bos89] Hartmut Bossel. *Simulation dynamischer Systeme*. Vieweg, 1989.

[Bre11] R. G. Brereton. One-Class Classifiers. *Journal of Chemometrics*, 25(5):225–246, 2011.

[BST10] K. Berns, B. Schürmann, and M. Trapp. *Eingebettete Systeme*. Vieweg+Teubner, 2010.

[BW01] S. Babu and J. Widom. Continuous Queries over Data Streams. *SIGMOD Record*, 30(3):109–120, September 2001.

[BW08] S. Boslaugh and P. A. Watters. *Statistics in a Nutshell*. O'Reilly, 2008.

[CB10] T. Connolly and C. Begg. *Database Systems: A Practical Approach to Design, Implementation, and Management*. Addison Wesley, 5th edition, 2010.

[CBK09] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Comput. Surv.*, 41:15:1–15:58, 2009.

[CcC+02] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring Streams: A New Class of Data Management Applications. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*, pages 215–226. VLDB Endowment, 2002.

[CEW01] S. H. Chung, J. M. Van Eepoel, and B. C. Williams. Improving Model-based Mode Estimation through Offline Compilation. In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics and Automation in Space*, 2001.

[Cha09] R. Chattamvelli. *Data Mining Methods*. Alpha Science International, Ltd, 2009.

[CJ09] S. Chakravarthy and Q. Jiang. *Stream Data Processing: A Quality of Service Perspective*. Springer, 2009.

[CM12] G. Cugola and A. Margara. Processing Flows of Information: From Data Stream to Complex Event Processing. *ACM Comput. Surv.*, 44(3):15:1–15:62, 2012.

[Cod82] E. F. Codd. Relational Database: A Practical Foundation for Productivity. *Commun. ACM*, 25(2):109–117, 1982.

[CV95] C. Cortes and V. Vapnik. Support-Vector Networks. *Mach. Learn.*, 20(3):273–297, 1995.

[DD09] M. M. Deza and E. Deza. *Encyclopedia of Distances*. Springer, 2009.

[DGG96] K. R. Dittrich, S. Gatziu, and A. Geppert. The Active Database Management System Manifesto: A Rulebase of ADBMS Features. *SIGMOD Rec.*, 25(3):40–49, 1996.

[DGH+06] A. Demers, J. Gehrke, M. Hong, M. Riedewald, and W. White. Towards Expressive Publish/Subscribe Systems. In *Proceedings of the 10th International Conference on Extending Database Technology (EDBT)*, 2006.

[DH00] P. Domingos and G. Hulten. Mining High-Speed Data Streams. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 71–80. ACM, 2000.

[DHS01] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2001.

[Die97] T. G. Dietterich. Machine-Learning Research: Four Current Directions. *AI Magazine*, 18:97–136, 1997.

[Die12] M. Dietrich. Anwendung des KDC-Ansatzes zur Überwachung des Frischluftkreislaufs auf der internationalen Raumstation ISS. Master's thesis, Brandenburg University of Technology Cottbus, Institute of Computer Science, 2012.

[DJP+07] R. P. W. Duin, P. Juszczak, P. Paclík, E. Pękalska, D. de Ridder, D. M. J. Tax, and S. Verzakov. PRTools, A Matlab Toolbox for Pattern Recognition, 2007. Available from: `http://prtools.org/`.

[DK89] D. Dvorak and B. Kuipers. Model-based Monitoring of Dynamic Systems. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, volume 2, pages 1238–1243. Morgan Kaufmann Publishers Inc., 1989.

[dKW89] J. de Kleer and B. C. Williams. Diagnosing with Behavioral Modes. In *Proceedings of the 11th IJCAI*, 1989.

[DS86] R. B. D'Agostino and M. A. Stephens, editors. *Goodness-Of-Fit Techniques*. Marcel Dekker, Inc., 1986.

[DSSV00] C. De Stefano, C. Sansone, and M. Vento. To reject or not to reject: that is the question-an answer in case of neural classifiers. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 30(1):84–94, 2000.

[dSTM10] P. H. dos Santos Teixeira and R. L. Milidiú. Data Stream Anomaly Detection through Principal Subspace Tracking. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1609–1616. ACM, 2010.

[EB09] M. Eckert and F. Bry. Aktuelles Schlagwort: Complex Event Processing (CEP). *Informatik-Spektrum*, 32:163–167, 2009.

[Efr83] Bradley Efron. Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation. *Journal of the American Statistical Association*, 78(382):pp. 316–331, 1983.

[Egy85] C. Egyhazy. Using Database Machines in Embedded Computer Systems. *Inf. Manage.*, 8(4):197–203, 1985.

[EN10] O. Etzion and P. Niblett. *Event Processing in Action*. Manning Publications Co., 2010.

[ES00] M. Ester and J. Sander. *Knowledge Discovery in Databases: Techniken und Anwendungen*. Springer, 2000.

[ESA14] ESA. OPS-SAT, 2014. 26.05.2014. Available from: `http://www.esa.int/Our_Activities/Operations/OPS-SAT`.

[Esp13] EsperTech Inc. Esper - Complex Event Processing, 2013. 20.03.2013. Available from: `http://esper.codehaus.org/`.

[Faw06] T. Fawcett. An Introduction to ROC Analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.

[FGC13] E. R. Faria, J. Gama, and A. C. P. L. F. Carvalho. Novelty Detection Algorithm for Data Streams Multi-class Problems. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 795–800. ACM, 2013.

[FK03] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., 2003.

[FPS+11] C.-L. Fok, A. Petz, D. Stovall, N. Paine, C. Julien, and S. Vishwanath. Pharos: A Testbed for Mobile Cyber-Physical Systems. Technical report, University of Texas at Austin, 2011. Available from: `http://pharos.ece.utexas.edu/pubs/TR-ARiSE-2011-001.pdf`.

[FPSM92] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge Discovery in Databases: An Overview. *AI Magazine*, 13:57–70, 1992.

[FPSS96] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37–54, 1996.

[FYM05] R. Fujimaki, T. Yairi, and K. Machida. An Approach to Spacecraft Anomaly Detection Problem Using Kernel Feature Space. In *Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD)*, pages 401–410. ACM, 2005.

[Gam10] J. Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall, 2010.

[GD06] S. Goddard and J. S. Deogun. Future Mobile Cyber-Physical Systems: spatio-temporal computational environments. In *Proceedings of the 2006 National Science Foundation Workshop on Cyber-Physical Systems*, 2006. Position paper.

[Gee13] D. Geesen. *Maschinelles Lernen in Datenstrommanagementsystemen*. PhD thesis, University of Oldenburg, 2013. OlWIR Verlag für Wirtschaft, Informatik und Recht.

[GJ11] L. Golab and T. Johnson. Consistency in a Stream Warehouse. In *CIDR*, pages 114–122, 2011.

[GJSS09] L. Golab, T. Johnson, J. S. Seidel, and V. Shkapenyuk. Stream Warehousing with DataDepot. In *Proceedings of the 35th SIGMOD International Conference on Management of Data*, pages 847–854. ACM, 2009.

[GKA06] J. Gama, R. Klinkenberg, and J. Aguilar, editors. *The Fourth International Workshop on Knowledge Discovery from Data Streams*, 2006. Available from: `http://www.ecmlpkdd2006.org/ws-kdds.pdf`.

[GO10] L. Golab and M. T. Özsu. *Data Stream Management*. Morgan & Claypool Publishers, 2010.

[Gor72] G. Gordon. *Systemsimulation*. Oldenbourg, 1972.

[GZK05] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining Data Streams: A Review. *SIGMOD Rec.*, 34(2):18–26, 2005.

[HA04] V. J. Hodge and J. Austin. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22:85–126, 2004.

[Han10] A. Handl. *Multivariate Analysemethoden*. Springer, 2010.

[HB95] E. Horvitz and M. Barry. Display of Information for Time-Critical Decision Making. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 296–305. Morgan Kaufmann Publishers Inc., 1995.

[HK06] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 2006.

[HL02] C.-W. Hsu and C.-J. Lin. A Comparison of Methods for Multiclass Support Vector Machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

[HM82] J. A. Hanley and B. J. McNeil. The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 143:29–36, 1982.

[HM08] M. C. Huebscher and J. A. McCann. A Survey of Autonomic Computing — Degrees, Models, and Applications. *ACM Comput. Surv.*, 40(3):7:1–7:28, 2008.

[HRWL83] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, editors. *Building Expert Systems*. Addison-Wesley, 1983.

[HYMK09] S. Hashemi, Y. Yang, Z. Mirzamomen, and M. Kangavari. Adapted One-versus-All Decision Trees for Data Stream Classification. *IEEE Transactions on Knowledge and Data Engineering*, 21(5):624–637, 2009.

[IBM03] IBM. An Architectural Blueprint for Autonomic Computing. Technical report, IBM, 2003. Available from: `http://cs.nju.edu.cn/yangxc/autonomic-computing/ACwpFinal.pdf`.

[Ign90] J. P. Ignizio. A Brief Introduction to Expert Systems. *Computers & Operations Research*, 17(6):523 – 533, 1990.

[IK03] D. M. Imboden and S. Koch. *Systemanalyse*. Springer, 2003.

[Inm02] W. H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, Inc., 2002.

[Jac90] P. Jackson. *Introduction to Expert Systems*. Addison-Wesley Longman Publishing Co., Inc., 2nd edition, 1990.

[Jac09] A. Jacobs. The Pathologies of Big Data. *Commun. ACM*, 52(8):36–44, 2009.

[JBo14] JBoss. Drools Fusion, 2014. 22.04.2014. Available from: `http://www.jboss.org/drools/drools-fusion.html`.

[JFP09] J. H. M. Janssens, I. Flesch, and E. O. Postma. Outlier Detection with One-Class Classifiers from ML and KDD. In *Proceedings of the 2009 International Conference on Machine Learning and Applications*, ICMLA '09, pages 147–153. IEEE Computer Society, 2009.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Comput. Surv.*, 31:264–323, 1999.

[Joh83] P. E. Johnson. What Kind of Expert Should a System Be? *Journal of Medicine and Philosophy*, 8(1):77–97, 1983.

[JR08] J. C. Junior and D. Renaux. Efficient Monitoring of Embedded Real-Time Systems. *Information Technology: New Generations, Third International Conference on*, 0:651–656, 2008.

[JW92] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice-Hall, 1992.

[Kal13] S. Kaltschmidt. Entwurf eines Anomalie-Erkennungssystems für Sensordatenströme. Master's thesis, Brandenburg University of Technology Cottbus, Institute of Computer Science, 2013.

[Kau12] M. Kaufmann. Datenbankentwurf und Datenauswertung von Telemetriedaten des Columbus-Moduls der ISS. Diploma thesis, Brandenburg University of Technology Cottbus, Institute of Computer Science, 2012.

[KBL+04] H. Kargupta, R. Bhargava, K. Liu, M. Powers, P. Blair, S. Bushra, J. Dull, K. Sarkar, M. Klein, M. Vasa, and D. Handy. VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, 2004.

[Kec05] C. Kecher. *UML 2.0 Das umfassende Handbuch*. Galileo Computing, 2005.

[KKW03] G.-A. Klutke, Peter C. Kiessler, and M. A. Wortman. A Critical Look at the Bathtub Curve. *IEEE Transactions on Reliability*, 52(1):125–129, 2003.

[KM83] B. Krause and P. Metzler. *Angewandte Statistik*. VEB Deutscher Verlag der Wissenschaften, 1983.

[Kni02] J. C. Knight. Safety Critical Systems: Challenges and Directions. In *Proceedings of the 24th International Conference on Software Engineering (ICSE)*, pages 547–550. ACM, 2002.

[Knu98] D. E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc., 2nd ed. edition, 1998.

[Kop11] Hermann Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Springer, 2 edition, 2011.

[KPD10] A. O. Kohlhase, N. Porth, and J. Doyé. Operational Engineering of the Columbus Thermal and Environmental Control System: Achievements, Optimizations. In *AIAA SpaceOps 2010 Conference*, 2010.

[KR00] J. Kittler and F. Roli, editors. *Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy, June 21-23, 2000, Proceedings*, volume 1857 of *Lecture Notes in Computer Science*. Springer, 2000.

[KS04] J. Krämer and B. Seeger. PIPES – A Public Infrastructure for Processing and Exploring Streams. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 925–926. ACM, 2004.

[Kun04] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[KvB+14] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, and J. Stefanowski. Open Challenges for Data Stream Mining Research. *ACM SIGKDD Explorations Newsletter – Special Issue on Big Data*, 16(1):1–10, 2014.

[KZ02] W. Klösgen and J. M. Zytkow, editors. *Handbook of Data Mining and Knowledge Discovery*. Oxford University Press, Inc., 2002.

[Lap95] J. C. Laprie. Dependable Computing and Fault Tolerance: Concepts and Terminology. In *Fault-Tolerant Computing, 1995, Highlights from Twenty-Five Years., Twenty-Fifth International Symposium on*, 1995.

[Lee08] E. A. Lee. Cyber Physical Systems: Design Challenges. In *Proceedings of the 2008 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing*, pages 363–369. IEEE Computer Society, 2008.

[LPV+08] X. Li, B. Plale, N. Vijayakumar, R. Ramachandran, S. Graves, and H. Conover. Real-time Storm Detection and Weather Forecast Activation Through Data Mining and Events Processing. *Earth Science Informatics*, 1:49–57, 2008.

[LS11] E. A. Lee and S. A. Seshia. *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. Published by Authors, 2011.

[Luc02] D. Luckham. *The Power of Events*. Addison-Wesley Longman, 2002.

[LZL09] C. Li, Y. Zhang, and X. Li. OcVFDT: One-class Very Fast Decision Tree for One-class Classification of Data Streams. In *Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data*, SensorKDD '09, pages 79–86. ACM, 2009.

[Mah36] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science*, volume 2, pages 49–55, 1936.

[MAK07] R. Müller, G. Alonso, and D. Kossmann. SwissQM: Next Generation Data Processing in Sensor Networks. In *Biennial Conference on Innovative Data Systems Research (CDIR)*, 2007.

[Mar07] P. Marwedel. *Eingebettete Systeme*. Springer, 2007.

[Mat14] MathWorks. MATLAB, 2014. 10.03.2014. Available from: `www.mathworks.com/products/matlab/`.

[Mei08] G. Meijer, editor. *Smart Sensor Systems*. Wiley, 2008.

[MF02] S. Madden and M. J. Franklin. Fjording the Stream: An Architecture for Queries Over Streaming Sensor Data. In *Proceedings of the 18th International Conference on Data Engineering*. IEEE Computer Society, 2002.

[MGK+11] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham. Classification and Novel Class Detection in Concept-Drifting Data Streams under Time Constraints. *Knowledge and Data Engineering, IEEE Transactions on*, 23(6):859–874, 2011.

[MH96] M. M. Moya and D. R. Hush. Network Constraints and Multi-objective Optimization for One-class Classification. *Neural Netw.*, 9(3):463–474, 1996.

[MM02] G. S. Manku and R. Motwani. Approximate Frequency Counts over Data Streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 346–357. VLDB Endowment, 2002.

[MR05] O. Maimon and L. Rokach, editors. *Data Mining and Knowledge Discovery Handbook*. Springer, 2nd edition, 2005.

[MS03a] M. Markou and S. Singh. Novelty detection: a review—part 2: neural network based approaches. *Signal Processing*, 83(12):2499 – 2521, 2003.

[MS03b] M. Markou and S. Singh. Novelty Detection: A Review Part 1: Statistical Approaches. *Signal Processing*, 83(12):2481 – 2497, 2003.

[MWA+02] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma. Query Processing, Resource Management, and Approximation in a Data Stream Management System. Technical Report 2002-41, Stanford InfoLab, 2002. Available from: `http://ilpubs.stanford.edu:8090/549/`.

[NBW+10] E. Noack, W. Belau, R. Wohlgemuth, R. Müller, S. Palumberi, P. Parodi, and F. Burzagli. Efficiency of the Columbus Failure Management System. In *AIAA 40th International Conference on Environmental Systems*, 2010.

[NCHC11] M. C. Naldi, R. J. G. B. Campello, E. R. Hruschka, and A. C. P. L. F. Carvalho. Efficiency Issues of Evolutionary K-means. *Appl. Soft Comput.*, 11(2):1938–1952, 2011.

[NLS+12] E. Noack, A. Luedtke, I. Schmitt, T. Noack, E. Schaumlöffel, E. Hauke, J. Stamminger, and E. Frisk. The Columbus Module as a Technology Demonstrator for Innovative Failure Management. In *German Air and Space Travel Congress*, Deutscher Luft- und Raumfahrtkongress, 2012.

[NNP+11] E. Noack, T. Noack, V. Patel, I. Schmitt, M. Richters, J. Stamminger, and S. Sievi. Failure Management for Cost-Effective and Efficient Spacecraft Operation. In *Proceedings of the 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. IEEE Computer Society, 2011.

[NNS+13] T. Noack, E. Noack, I. Schmitt, S. Sievi, and S. Mirzakhyl. A Discussion on ISS Columbus Data Streams. In *ECML/PKDD Workshop on Real-World Challenges for Data Stream Mining (RealStream)*, 2013.

[Noa11a] T. Noack. Echtzeitüberwachung und Langzeitanalyse mittels eingebetteter Systeme. In *Proceedings of the 23nd GI-Workshop on Foundations of Databases*, 2011.

[Noa11b] T. Noack. Real-Time Monitoring and Long-Term Analysis by Means of Embedded Systems. In *Proceedings of the CAiSE Doctoral Consortium*, 2011.

[Nor07] A. Nori. Mobile and Embedded Databases. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 1175–1177. ACM, 2007.

[NS12a] T. Noack and I. Schmitt. A Cyclic Process Model for Monitoring Mobile Cyber-Physical Systems. Technical Report 04/12, Brandenburg University of Technology Cottbus, Institute of Computer Science, 2012. Available from: `http://www-docs.tu-cottbus.de/dbis-informatik/public/paper/TechRep_PP_Noack_Final.pdf`.

[NS12b] T. Noack and I. Schmitt. Monitoring Mobile Cyber-Physical Systems by Means of a Knowledge Discovery Cycle - A Case Study. In *Workshop on Knowledge Discovery, Data Mining, and Machine Learning (KDML)*, 2012.

[NS13] T. Noack and I. Schmitt. Monitoring Mobile Cyber-Physical Systems by Means of a Knowledge Discovery Cycle. In *Seventh IEEE International Conference on Research Challenges in Information Science (RCIS)*, 2013.

[NSS13a] T. Noack, I. Schmitt, and S. Saretz. OVA-based Multi-Class Classification for Data Stream Anomaly Detection. Technical Report 01/13, Brandenburg University of Technology Cottbus, Institute of Computer Science, 2013. Available from: `http://opus.kobv.de/btu/volltexte/2013/2818/`.

[NSS13b] T. Noack, I. Schmitt, and S. Sievi. Knowledge Discovery Cycle for OPS-SAT. Technical report, ESA/ESOC OPS-SAT Open Day, 2013.

[OC92] Y. Ohba and Y. Chba, editors. *Intelligent Sensor Technology*. John Wiley & Sons, Inc., 1st edition, 1992.

[Oet13] T. Oetiker. Rrdtool, 2013. 18.07.2013. Available from: `http://oss.oetiker.ch/rrdtool/`.

[Ols02] J. Olson. *Data Quality: The Accuracy Dimension*. Morgan Kaufmann Publishers Inc., 2002.

[Ort00] S. Ortiz. Embedded Databases Come out of Hiding. *Computer*, 33(3):16–19, 2000.

[OSB+72] G. Ose, G. Schiemann, H. Baumann, F. Stopp, W. Körner, and G. Lochmann. *Ausgewählte Kapitel der Mathematik*. VEB Fachbuchverlag Leipzig, 1972.

[Pag54] E. S. Page. Continuous Inspection Schemes. *Biometrika*, 41:100–115, 1954.

[PD99] N. W. Paton and O. Díaz. Active Database Systems. *ACM Comput. Surv.*, 31(1):63–103, 1999.

[Pea01] K. Pearson. On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine*, 2(6):559–572, 1901.

[Pec07] J. K. Peckol. *Embedded Systems: A Contemporary Design Tool*. John Wiley & Sons, 2007.

[Ped89] K. Pedersen. *Expert Systems Programming – Practical Techniques for Rule-based Systems*. Jo, 1989.

[Pos14] PostgreSQL Global Development Group. PostgreSQL, 2014. 10.03.2014. Available from: `http://www.postgresql.org/`.

[PP07] A. Patcha and J.-M. Park. An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends. *Comput. Netw.*, 51:3448–3470, 2007.

[PP09] A. G. Piersol and T. L. Paez. *Harris' Shock and Vibration Handbook*. McGraw-Hill, 6 edition, 2009.

[PSRD11] G. Parzianello, M. Schmid, E. Roberts, and O. Duforet. Investigation of an Anomalous Failure of Bearing on a Fan within the Columbus Module of the ISS. In *14th European Space Mechanisms and Tribology Symposium (ESMATS)*, 2011.

[PVS11] A. Paschke, P. Vincent, and F. Springer. Standards for Complex Event Processing and Reaction Rules. In F. Olken, M. Palmirani, and D. Sottara, editors, *Rule - Based Modeling and Computing on the Semantic Web*, volume 7018 of *Lecture Notes in Computer Science*, pages 128–139. Springer, 2011.

[RALS09] M. Rosenmüller, S. Apel, T. Leich, and G. Saake. Tailor-made Data Management for Embedded Systems: A Case Study on Berkeley DB. *Data Knowl. Eng.*, 68:1493–1512, 2009.

[Rao96] B. K. N. Rao. *Handbook of Condition Monitoring*. Elsevier Science, 1996.

[Ras13] Raspberry Pi. Raspberry Pi, 2013. 10.03.2014. Available from: `http://www.raspberrypi.org/`.

[RG97] G. Ritter and M. T. Gallegos. Outliers in Statistical Pattern Recognition and an Application to Automatic Chromosome Classification. *Pattern Recognition Letters*, 18(6):525–539, 1997.

[RLAS07] M. Rosenmüller, T. Leich, S. Apel, and G. Saake. Von Mini- über Micro- bis zu Nano-DBMS: Datenhaltung in eingebetteten Systemen. *Datenbank-Spektrum*, 7(20):33–43, 2007.

[RLSS10] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-Physical Systems: The Next Computing Revolution. In *Proceedings of the 47th Design Automation Conference (DAC)*, pages 731–736. ACM, 2010.

[RSOR10] H. Roth, J. Schiefer, H. Obweger, and S. Rozsnyai. Event Data Warehousing for Complex Event Processing. In *Fourth International Conference on Research Challenges in Information Science (RCIS)*, pages 203–212, 2010.

[Run10] T. A. Runkler. *Data Mining: Methoden und Algorithmen intelligenter Datenanalyse*. Vieweg+Teubner, 2010.

[SA11] N. Stojanovic and A. Artikis. On Complex Event Processing for Real-Time Situational Awareness. In Nick Bassiliades, Guido Governatori, and Adrian Paschke, editors, *Rule-Based Reasoning, Programming, and Applications*, volume 6826 of *Lecture Notes in Computer Science*, pages 114–121. Springer, 2011.

[Sam05] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann Publishers Inc., 2005.

[Sch95] B. A. Schroeder. On-Line Monitoring: A Tutorial. *Computer*, 28:72–78, 1995.

[Sch08] A. Schuster. *Robust Intelligent Systems*. Springer, 2008.

[ScZ05] M. Stonebraker, U. Çetintemel, and S. Zdonik. The 8 Requirements of Real-Time Stream Processing. *ACM SIGMOD Record*, 34(4):42–47, 2005.

[SdLFdCG07] E. J. Spinosa, A. P. de Leon F. de Carvalho, and J. Gama. OLINDDA: A Cluster-based Approach for Detecting Novelty and Concept Drift in Data Streams. In *Proceedings of the 2007 ACM symposium on Applied computing*, SAC '07, pages 448–452. ACM, 2007.

[See00] M. Seeger. Learning with Labeled and Unlabeled Data. Technical report, University of Edinburgh, 2000. Available from: `http://lapmal.epfl.ch/papers/review.pdf`.

[SGLW08] L. Sha, S. Gopalakrishnan, X. Liu, and Q. Wang. Cyber-Physical Systems: A New Frontier. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, pages 1–9, 2008.

[Sil05] C. W. de Silva, editor. *Vibration and Shock Handbook*. Taylor & Francis, 2005.

[SLM10] F. Salfner, M. Lenk, and M. Malek. A Survey of Online Failure Prediction Methods. *ACM Comput. Surv.*, 42:10:1–10:42, 2010.

[Smi56] W. E. Smith. Various Optimizers for Single-Stage Production. *Naval Research Logistics Quarterly*, 3(1-2):59–66, 1956.

[Sno88] R. Snodgrass. A Relational Approach to Monitoring Complex Systems. *ACM Trans. Comput. Syst.*, 6:157–195, 1988.

[Spa89] K. A. Spackman. Signal Detection Theory: Valuable Tools for Evaluating Inductive Learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 160–163. Morgan Kaufmann Publishers Inc., 1989.

[SS94] M. Singhal and N. G. Shivaratri. *Advanced Concepts in Operating Systems*. McGraw-Hill, Inc., 1994.

[SS13] V. Singh and D. Singh. Survey on Pattern Optimization for Novel Class in MCM for Stream Data Classification. *Int. Journal of Engineering Sciences and Research Technology*, 2:2767–2771, 2013.

[Ste09] A. Steland. *Basiswissen Statistik*. Springer, 2009.

[Str13] StreamBase Systems. StreamBase, 2013. 20.03.2013. Available from: `http://www.streambase.com/`.

[SWSST00] B. Schölkopf, R. Williamson, A. Smola, and J. Shawe-Taylor. SV Estimation of a Distribution's Support. In *Proceedings of Neural Information Processing Systems, NIPS'99*, pages 582–588. MIT Press, 2000.

[SWYS11] J. Shi, J. Wan, H. Yan, and H. Suo. A Survey of Cyber-Physical Systems. In *International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6, 2011.

[Tan14] S. C. Tan. Fast Anomaly Detection for Streaming Data, 2014. 27.03.2014. Available from: `https://sites.google.com/site/analyticsofthings/recent-work-fast-anomaly-detection-for-streaming-data`.

[Tax01] D. M. J. Tax. *One-Class Classification: Concept-learning in the Absence of Counter-Examples*. PhD thesis, TU Delft, 2001. Available from: `http://homepage.tudelft.nl/n9d04/thesis.pdf`.

[Tax12] D. M. J. Tax. DDTools, the Data Description Toolbox for Matlab, May 2012. version 1.9.1. Available from: `http://prlab.tudelft.nl/david-tax/dd_tools.html`.

[TD99a] D. M. J. Tax and R. P. W. Duin. Data Domain Description using Support Vectors. In *European Symposium on Artificial Neural Networks*, pages 251–256, 1999.

[TD99b] D. M. J. Tax and R. P. W. Duin. Support Vector Domain Description. *Pattern Recognition Letters*, 20(11-13):1191–1199, 1999.

[TD02] D. M. J. Tax and R. P. W. Duin. Using Two-class Classifiers for Multiclass Classification. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 124–127, 2002.

[TD04] D. M. J. Tax and R. P. W. Duin. Support Vector Data Description. *Mach. Learn.*, 54:45–66, 2004.

[TGP08] Y. Tan, S. Goddard, and L. C. Pérez. A Prototype Architecture for Cyber-Physical Systems. *ACM SIGBED Review*, 5:26:1–26:2, 2008.

[TLM+11] H. Thakkar, N. Laptev, H. Mousavi, B. Mozafari, V. Russo, and C. Zaniolo. SMM: a Data Stream Management System for Knowledge Discovery. In *Proceedings of the 27th International Conference on Data Engineering (ICDE)*, pages 757–768. IEEE Computer Society, 2011.

[TSK05] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[TTL11] S. C. Tan, K. M. Ting, and T. F. Liu. Fast Anomaly Detection for Streaming Data. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 1511–1516. AAAI Press, 2011.

[TY95] J. J. P. Tsai and S. J. H. Yang. *Monitoring and Debugging of Distributed Real-Time Systems*. IEEE Computer Society Press, 1995.

[Vap95] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[Vap98] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[Wat86] D. A. Waterman. *A Guide to Expert Systems*. Addison-Wesley, 1986.

[WDR06] E. Wu, Y. Diao, and S. Rizvi. High-Performance Complex Event Processing over Streams. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 407–418. ACM, 2006.

[WFH11] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier, 2011.

[WG68] M. B. Wilk and R. Gnanadesikan. Probability Plotting Methods for the Analysis of Data. *Biometrika*, 55(1):1–17, 1968.

[WHF11] S. Weigert, M. Hiltunen, and C. Fetzer. Mining Large Distributed Log-Data in Near Real-Time. In *Managing Large-Scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques (SLAML/SOSP)*, pages 5:1–5:8. ACM, 2011.

[Wol02] F. Wolf. *Behavioral Intervals in Embedded Software: Timing and Power Analysis of Embedded Real-Time Software Processes*. Kluwer Academic Publishers, 2002.

[WR08] B. Wolf and M. Rosjat. A Dynamic OSGi-Based Data Stream System. In *Proceedings of the 5th Middleware Doctoral Symposium (MDS)*, pages 31–36. ACM, 2008.

[WYH05] H. Wang, P. S. Yu, and J. Han. Mining Data Streams. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 777–792. Springer, 2005.

[XLZ+08] Z. Xu, X. Liu, G. Zhang, W. He, G. Dai, and W. Shu. A Certificateless Signature Scheme for Mobile Wireless Cyber-Physical Systems. In *28th International Conference on Distributed Computing Systems Workshops (ICDCS)*, pages 489–494. IEEE Computer Society, 2008.

[YD98] A. Ypma and R. P. W. Duin. Support Objects for Domain Approximation. In *ICANN 98*, Perspectives in Neural Computing, pages 719–724. Springer, 1998.

[Zad73] L. A. Zadeh. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-3(1):28–44, 1973.

B List of Publications

1. [NNP+11] E. Noack, T. Noack, V. Patel, I. Schmitt, M. Richters, J. Stamminger, and S. Sievi. Failure Management for Cost-Effective and Efficient Spacecraft Operation. In *Proceedings of the 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. IEEE Computer Society, 2011

2. [Noa11a] T. Noack. Echtzeitüberwachung und Langzeitanalyse mittels eingebetteter Systeme. In *Proceedings of the 23nd GI-Workshop on Foundations of Databases*, 2011

3. [Noa11b] T. Noack. Real-Time Monitoring and Long-Term Analysis by Means of Embedded Systems. In *Proceedings of the CAiSE Doctoral Consortium*, 2011

4. [NS12a] T. Noack and I. Schmitt. A Cyclic Process Model for Monitoring Mobile Cyber-Physical Systems. Technical Report 04/12, Brandenburg University of Technology Cottbus, Institute of Computer Science, 2012. Available from: `http://www-docs.tu-cottbus.de/dbis-informatik/public/paper/TechRep_PP_Noack_Final.pdf`

5. [NS12b] T. Noack and I. Schmitt. Monitoring Mobile Cyber-Physical Systems by Means of a Knowledge Discovery Cycle - A Case Study. In *Workshop on Knowledge Discovery, Data Mining, and Machine Learning (KDML)*, 2012

6. [NLS+12] E. Noack, A. Luedtke, I. Schmitt, T. Noack, E. Schaumlöffel, E. Hauke, J. Stamminger, and E. Frisk. The Columbus Module as a Technology Demonstrator for Innovative Failure Management. In *German Air and Space Travel Congress*, Deutscher Luft- und Raumfahrtkongress, 2012

7. [NS13] T. Noack and I. Schmitt. Monitoring Mobile Cyber-Physical Systems by Means of a Knowledge Discovery Cycle. In *Seventh IEEE International Conference on Research Challenges in Information Science (RCIS)*, 2013

8. [NSS13a] T. Noack, I. Schmitt, and S. Saretz. OVA-based Multi-Class Classification for Data Stream Anomaly Detection. Technical Report 01/13, Brandenburg University of Technology Cottbus, Institute of Computer Science, 2013. Available from: `http://opus.kobv.de/btu/volltexte/2013/2818/`

9. [NSS13b] T. Noack, I. Schmitt, and S. Sievi. Knowledge Discovery Cycle for OPS-SAT. Technical report, ESA/ESOC OPS-SAT Open Day, 2013

10. [NNS+13] T. Noack, E. Noack, I. Schmitt, S. Sievi, and S. Mirzakhyl. A Discussion on ISS Columbus Data Streams. In *ECML/PKDD Workshop on Real-World Challenges for Data Stream Mining (RealStream)*, 2013

11. [KvB+14] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, and J. Stefanowski. Open Challenges for Data Stream Mining Research. *ACM SIGKDD Explorations Newsletter – Special Issue on Big Data*, 16(1):1–10, 2014

C Acronyms

AUC	area under an ROC curve
CEP	complex event processing
CFA	cabin fan
CHX	condensate heat exchanger
CPS	cyber-physical system
DAHP	database-active human-passive
DBMS	database management system
DSMS	data stream management system
DWH	data warehouse
ECA	event-condition-action
EPL	event processing language
ETL	extract, transform, and load
FN	false negative
FP	false positive
FPR	false positive rate
HADP	human-active database-passive
HST	half-space trees
HT	Hoeffding trees
IFP	information flow processing
IRFA	inter module ventilation return fan assembly
ISFA	inter module ventilation supply fan assembly
ISS	International Space Station

JSON	JavaScript object notation
KDC	knowledge discovery cycle
KDD	knowledge discovery in databases
KDDS	knowledge discovery from data streams
LOS	loss of signal
MAPE-K	monitor, analyse, plan, execute, and knowledge
MCC	multi-class classification
MCPS	mobile cyber-physical system
MINAS	multi-class learning algorithm for data streams
MOA	massive online analysis
NN	nearest neighbor
OAO	one-against-one
OAR	one-against-rest
OCC	one-class classification
OLINDDA	online novelty and drift detection algorithm
RBF	radial basis function
ROC	receiver operating characteristics
SDW	streaming data warehouse
SVDD	support vector domain description
SVM	support vector machine
TN	true negative
TP	true positive

TPR	true positive rate
XML	extensible markup language

D Glossary

abnormal system states
Complementary to normal system states where it is assumed that an MCPS works incorrectly during operation.

actuator
A system component which influences the system environment.

anomaly
A pattern in data which deviates from normal behavior.

anomaly detection
A research area which aims to find data subsets or single data elements from a set of data which do not conform to expected behavior.

attribute
At least one measurable property or a combination of several measurable properties of a system or a system component.

attribute value
A specific measurement which relates to an attribute.

bathtub curve
A graph which subdivides the operational phase of MCPSs into three further phases.

centroid
The arithmetic mean of a data set.

class
A logical grouping of data.

classification
A technique to assign data into a set of distinct classes.

cluster
A homogeneous group of data.

clustering
A technique to break a large heterogeneous set of data into a small number of homogeneous groups or clusters.

complex event processing
The deduction of complex events from fundamental or underlying events in a data stream context.

condition monitoring
A monitoring type used to monitor the conditions or system sates of MCPSs.

continuous monitoring
A monitoring characteristic while a permanently installed monitoring system is used.

cyber-physical system
A very complex cybernetic system.

cybernetics
A conjunction of physical processes, computation, and communication.

data
A set of attribute values which describe a system (or a system component) within a particular time frame.

data mining
The application of specific algorithms for extracting patterns from data.

data persistence
The ability of data to outlive the operational phase and possibly the overall life-time of an MCPS.

data stream
A potentially infinite sequence of data items.

data stream mining
The application of data mining and machine learning algorithms directly onto data streams.

data stream processing
A research area which addresses the machining of data streams.

data transience
The opposite of data persistence.

data warehouse
A subject-oriented, integrated, nonvolatile, and time-variant collection of data.

database management system
A software used to manage databases.

design phase
The conceptional engineering of an MCPS.

detector chain
A consecutive arrangement of dichotomous class detectors.

dichotomous class detector
A distinct and optimized class detector for each class providing two mutually exclusive decisions.

dispatch and commissioning phase
The shipment of an MCPS to the application environment and its activation.

embedded system
A mechatronic system which is embedded into a product, processes information, and interacts with the product and the product environment by means of sensors and actuators.

error
A term which describes a system state where the system does not work as expected.

event
An incident that has occurred within a particular system or domain.

expert system
A computer system which aims to combine human expertise with artificial expertise.

external information system
A stationary part of an MCPS.

failure
A situation where the intended functioning of the system or subsystem cannot be guaranteed any longer.

fault
The root cause of an error.

global monitoring
A monitoring characteristic while the entire system is monitored.

gradual change
Long-term changes of the system behavior.

heterogeneous multi-class anomaly detection
A set of classifiers which addresses a multi-class anomaly detection problem.

homogeneous multi-class anomaly detection
A single anomaly detection algorithm which addresses a multi-class anomaly detection problem.

human expert
A person who, because of training and experience, is able to do things the rest of us cannot.

knowledge discovery cycle
An abstract process cycle for monitoring MCPSs.

knowledge discovery from data streams
A process of identifying valid, novel, and potentially useful patterns from streaming data.

knowledge discovery in databases
A process of identifying valid, novel, and potentially useful patterns in persistently stored data.

known system states
System states that represent knowledge about an MCPS.

labeled data
Data that provide class labels which in turn indicate the membership to classes.

limit monitoring
A monitoring type which uses one-dimensional functions as thresholds.

local monitoring
A monitoring characteristic while only a few system components are monitored.

mechatronic system
A system which comprises mechanic, electronic, and software domains.

medoid
A center m of a ball where $m \in \mathcal{X}^{tr}$.

mobile cyber-physical system
A location-independent CPS.

model-based monitoring
A monitoring type based on a static and preliminary model of the target system.

monitoring
A continuous task of recognizing specific event occurrences in the behavior of MCPSs and the identification of underlying faults during operation.

multivariate data
Data which refer to more than one attribute.

normal system states
System states that can be interpreted as a correct service of an MCPS or a target system during operation.

normal wear phase
A phase of an MCPS during operation while the failure rate is proportionally low.

offline monitoring
A monitoring characteristic while a monitoring system is applied semi-automatically and asynchronously from a target system.

offline subcycle
A part of the KDC which references offline monitoring.

one-class classification
A specific classification type used to assign data into two distinct classes.

online monitoring
A monitoring characteristic while a monitoring system is applied automatically and synchronously with a target system.

online subcycle
A part of the KDC which references online monitoring.

operational phase
The runtime of an MCPS until its deactivation.

performance
The ability of an anomaly detection algorithm to detect anomalies in the context of a given data set.

periodic monitoring
A monitoring characteristic when data is only collected at specific times.

physical environment
A system environment subject to physical conditions.

real-time system
A system which works under timing constraints while the correctness of the computational results is time-depended.

reliability
The continuity of correct service and probability of default.

round trip
A complete cyclic pass through the KDC.

safety-critical system
A system which is subject to the consequences of failure.

sensor
A system component which collects information about the system environment.

smart sensor
A sensor which integrates the gathering of information with additional signal processing functions.

storage-aware stream model
The storage-aware stream model extends the commonly known stream model by the ability of storing incoming data streams entirely.

streaming data warehouse
A DWH which is used to store massive amounts of streaming data into a persistent repository.

sudden change
Immediate change of the system behavior.

system
A compound structure of objects which cohere due to interaction or interdependency.

system component
A subpart of a system.

system environment
Immediate surroundings of a system.

system states
: A general term which aggregates the conditions of an MCPS in order to abstractly describe the system and its components under consideration of relevant and well-defined attributes within a specific time period.

target system
: An MCPS which have to be monitored.

test phase
: The implementation of the conceptional design and subsequent tests of an MCPS.

time-efficiency
: The time that an anomaly detection algorithm requires to process a single data item.

univariate data
: Data which exclusively refer to one attribute.

unknown system states
: System states that represent the unawareness about an MCPS.

unlabeled data
: Data that do not provide class labels.

wear in phase
: A phase of an MCPS during operation characterized by a relatively high failure rate.

wearout phase
: A phase of an MCPS during operation characterized by a gradually increasing failure rate.

window
: A construct of data stream query languages applied to language operators to restrict the scope of continuous queries.

E Notation

A	An attribute.
a	An attribute value $a \in \mathbb{R}$.
α	A Lagrangian multiplier.
b	A minimal bounding hypersphere.
C	An SVDD regularization parameter.
$\boldsymbol{c}$	Centroid of a hypersphere.
$\mathcal{X}$	A data set.
$\mathcal{E}$	An error.
ϵ	A really small positive number $\epsilon \in \mathbb{R}_{>0}$.
f	A filter function.
g	A function to estimate the local density of a state vector.
h	The number of known classes $h = \lvert\Omega\rvert$.
i	A control variable $i \in \mathbb{N}$.
j	A control variable $j \in \mathbb{N}$.
K	A kernel function.
k	A number of balls $k \in \mathbb{N}$.
l	A control variable $l \in \mathbb{N}$.
$\boldsymbol{m}$	Medoid of a hypersphere.
μ	The expected value.
$\mathbb{N}$	The set of natural numbers.
$\mathcal{N}$	The multivariate Gaussian distribution.
n	The number of attributes (dimensions) $n \in \mathbb{N}$.
Ω	A set of known classes.
Ω^{C}	The complement of known classes (anomaly class).
ω	A specific known class.
p	A probability of occurrence referring to a class.
Φ	Mapping into a higher dimensional feature space.

π	The ratio of the circumference of a circle to its diameter.
Q	Quantile of a density function.
$\mathbb{R}$	The set of real numbers.
r	Radius of a hypersphere.
ρ	A permutation.
S	The corresponding vector space.
$\mathbf{\Sigma}$	The covariance matrix.
σ	The width of the RBF in the SVDD.
$\boldsymbol{s}_\tau$	A state vector with index τ.
T	Time as a function.
t	The processing time of an anomaly detector $t \in \mathbb{R}$.
$\tilde{t}$	The relative processing time of a class detector $\tilde{t} \in \mathbb{R}$.
τ	Time as an index $\tau \in \mathbb{N}$.
θ	A threshold $\theta \in \mathbb{R}$.
$\boldsymbol{u}$	An eigenvector.
V	The volume of a hypersphere.
$\bar{x}$	The arithmetic mean.
ζ	A constant.
$\boldsymbol{z}$	A support vector.

F List of Figures

G List of Tables

H List of Listings

I Acknowledgements

First of all, I would like to thank my doctoral adviser Prof. Ingo Schmitt for the patient guidance. I greatly admire his commitment and devotion for reading my manuscripts meticulously. This always helped me to improve my writings and realize my ideas. Further on, I would like to thank the representative of the IGS ZUSYS class (dependable hardware and software systems) Prof. H. T. Vierhaus for the endeavor to provide a fruitful cooperation within the PhD class. I would also like to thank Dr. Georg Krempl and Dr. Martin Strehler for improvement suggestions.

Furthermore, I would like to thank my brother Enrico Noack, who is a system engineer of EADS Astrium Space Transportation, for provisioning and preparing the ISS Columbus air loop data. His expert knowledge has been central for understanding important issues and associations. Along with this, his encouragement and vision have laid the foundation of the cooperation between the Brandenburg University of Technology and EADS Astrium Space Transportation. In addition, I would like to thank Sonja Sievi, another employee of EADS Astrium Space Transportation, for participating in the OPS-SAT project.

I would also like to thank the members of the chair of database and information systems. The secretary S. Schneider and the administrator E. Schwaar helped a lot to reduce the administrative burden. Marcel Zierenberg, Alexander Schulze, David Zellhöfer, Sebastian Lehrack, Sascha Saretz, and Dr. Adrian Giurca were always available for discussion. Many thanks go to my students for preparing their diploma or master's theses in my field of research.

In addition, I would like to thank my parents for their financial support during my studies. Many thanks go to my brother Andy Noack for his assistance and many productive conversations during my studies. I would have never gone this far without him.

Finally, I would like to thank my large circle of friends for listening my lengthly monologues. Many thanks go to my long-standing friend and fellow student Markus Ulbricht – we have successfully mastered a lot together. I would also like to express my gratitude to my Skat group and my old school friend Christoph for the chess evenings. Last but not least, I would like to thank my friends from Blankenfelde – I can always count on them.

This work was supported by the Brandenburg Ministry of Science, Research, and Culture as part of the International Graduate School (IGS) at Brandenburg University of Technology.