Title
From fault-detection to automated fault correction: A field study

Authors
Pritoni, Marco
Lin, Guanjin
Chen, Yimin
et al.

Publication Date
2022-04-01

DOI
10.1016/j.buildenv.2022.108900

From fault-detection to automated fault correction: a field study

Marco Pritoni[1], Guanjin Lin[1], Yimin Chen[1], Raphael Vitti[1], Christopher Weyandt,[1] Jessica Granderson[1]

[1] Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

Abstract

A fault detection and diagnostics (FDD) tool, as addressed by this study, is a tool that continuously identifies the presence of faults and efficiency improvement opportunities through a one-way interface to the building automation system and the application of automated analytics. Although FDD tools can inform operators of building operational faults, currently an action is always required to correct the faults to generate energy savings. Fault auto-correction integrating with commercial FDD technology offerings can close the loop between the passive diagnostics and active control, increase the savings generated by FDD tools, and reduce the reliance on human intervention. This paper presents the field study of seven fault auto-correction algorithms implemented in commercial FDD platforms. Implementation includes software changes in the FDD tools and additional controls hardware or software changes in the BAS that were required to enable the execution of different types of auto-correction algorithms in real buildings. The routines successfully and automatically correct faults and improve the operation of large built-up Heating, Ventilation, and Air Conditioning (HVAC) systems, common in most commercial buildings. The auto-correction algorithms are tested across four buildings and three different building automation systems, following a rigorous procedure to make sure they work properly and do not negatively impact the system and building occupants. Technology benefits, market drivers, and scalability changes are drawn from the implementation effort and test results, to drive future research and industry engagement.

Keywords: fault correction; fault detection and diagnostics; energy efficiency; field testing; building HVAC system; smart building

1. Introduction and background

Buildings use 40% of primary energy globally, and account for 33% of direct and indirect carbon emissions from fuel combustion (Economidou M, 2011). Based on an analysis of the most common faults in building systems, studies estimate that the energy savings achievable from correcting these faults ranges from 5 to 30% whole building savings (Fernandez, 2017, Roth et al. 2005). Commercial fault detection and diagnostics (FDD) tools automate the process of detecting faults and suboptimal performance of building systems and help to diagnose potential causes (Dexter et al. 2001). They offer several interrelated benefits including energy savings and improved operational efficiency, utility cost savings, persistence in savings over time, streamlining operations and maintenance processes, and supporting continuous energy management practices such as monitoring-based commissioning.

As buildings become more data rich, FDD technologies are increasingly adopted in commercial buildings. There are over 30 full-featured FDD software product offerings in today's market in the US and new software products continue to enter the market (Kramer et al., 2020). Building operators at the forefront of technology adoption are using FDD to enable median whole-building portfolio savings of 9% (Kramer et al., 2020). A FDD tool usually is a software layer on top of the existing building automation systems (BAS). It integrates with BAS to obtain building system or equipment operational data (e.g. temperature, pressure, flow rate). Extensive libraries of detection logic are continuously run against the data, and results are presented through a graphical user interface for resolution by operations and maintenance staff. A number of possible causes or recommendations for correcting each fault are listed, requiring either additional data analysis by the user or on-site inspection. In addition, tools may provide a report of the duration and frequency of faults, cost and/or energy impacts, and relative priority levels (Granderson et al. 2017).

Although FDD tools are being used to enable cost-effective energy savings, there is a capability gap. Today's FDD technologies operate in an open loop manner. Faults are identified by the FDD tools, however, the identified faults must be corrected through manual human intervention (Kramer et al., 2020). In practice, the need for human intervention to fix faults once they are identified often results in delay or inaction, causing additional operations and maintenance costs or deteriorating comfort conditions. This capability gap is not only technical, but also represents market-relevant desired functionality on behalf of FDD users and technology providers (Granderson et al., 2017). Therefore, realizing automated fault correction in commercial FDD technology offerings closes the loop between passive diagnostics and active control, increase the savings realized through the use of FDD tools, and reduce the extent to which savings are dependent upon human intervention.

Kim and Katipamula (2018) indicate that since 2004, more than 100 FDD research studies associated with building systems have been published. However, the academic publication has extensively focused on the development of new FDD algorithms for HVAC systems (Katipamula et al. 2005, Zhao et al. 2013, Wang et al. 2017). Limited studies have been found on fault-tolerant or self-correcting controls for building HVAC systems. The purpose of fault-tolerant controls is to help good system operation despite the presence of faults (Zhang and Jiang, 2008). Wang et al. (2002) developed a supervisory control strategy that adapts to the presence of outdoor air flow rate sensor error. Hao et al. (2005) applied principal component analysis to develop fault tolerant HVAC controls. Bengea et al. (2015) developed a fault-tolerant optimal control schema for a HVAC system integrating FDD and model predictive control. These advanced controls are still in the early research and development stage and are not yet readily deployed in today's BAS. Additionally, these fault-tolerant controls typically do not integrate with or make use of modern FDD technologies, which are becoming increasingly present in commercial buildings.

Regarding the topic of automated fault correction, Fernandez et al. (Fernandez et al., 2009a; Fernandez, et al., 2009b) and Brambley et al. (Brambley et al., 2011) developed both passive and proactive fault auto-correction algorithms for an air-handler unit (AHU) and a variable-air-volume (VAV) box. The developed algorithms correct the following faults: temperature and humidity sensor bias, incorrect damper operation, control hunting, and manual overrides. A subset of these algorithms (sensor bias and minimum outdoor air damper position) were tested in a laboratory experiment. They have not been validated in real buildings or integrated into existing BAS and commercial FDD products.

Lin et al. (2020a) complemented and extended the work of Fernandez and Brambley (Fernandez et al., 2009a; Fernandez, et al., 2009b, Brambley et al., 2011), by developing fault auto-correction algorithms designed to be integrated with commercial FDD tools. The new auto-correction algorithms afford the FDD technology a certain degree of control capability, as the autonomous correction of faults are enabled by opening 2-way interfaces between the BAS and the FDD tool. These algorithms target incorrectly programmed schedules, override not released, sensor bias, control hunting, rogue zone, and suboptimal setpoints in HVAC systems. All the algorithms developed in the study follow a general auto-correction process, with different control variables overwritten in the BAS, and different ways to determine the correct or improved value of these variables (Lin et al. 2020a). In this process, after the FDD algorithm generates a fault flag for a specific fault, the auto-correction algorithm is initiated to correct this fault. Having a variable in the BAS that is accessible by the FDD tool is the key element in the process. The developed auto-correction algorithms were integrated into commercial FDD tools and some preliminary integration challenges and solutions were documented in Lin et al. (2020b). Among the auto-correction algorithms, three algorithms (rogue zone, improve AHU supply air temperature setpoint reset, and improve AHU static pressure setpoint reset) were deployed in a single commercial FDD software and tested in two office buildings. These preliminary field testing results are presented in Lin et al. (2020a) and Lin et al. (2021). The enhanced FDD tool with these three auto-correction algorithms was able to correct faults successfully. While these preliminary results are encouraging, they fall short of demonstrating all the algorithms and they are limited to a single software platform.

This article presents and discusses the final results of the project introduced in Lin et al. (2020a), focusing on their implementation in two commercial FDD tools, and extensive field testing performed in four buildings from late 2019 to 2021. The research team is composed of researchers, FDD implementation partners and facility managers testing the new FDD features in their buildings. This paper presents modifications to the FDD tools and the BAS that were required to enable the execution of different types of auto-correction algorithms in real buildings. It also presents the field testing procedure and results of seven auto-correction algorithms across four buildings and three different BASs. This study aims to answer the following three questions:

1. Can auto-correction algorithms be successfully implemented in modern FDD tools and field tested in real buildings?
2. Are the enhanced FDD solutions able to correct faults in real buildings without adverse operational effects?
3. What are the benefits, adoption drivers, and scalability challenges of fault auto-correction capability?

The rest of the paper is organized as follows: section 2 describes the method used for implementing, deploying and testing the algorithms, section 3, 4, and 5 present the FDD tool implementation results, the field tests results, and benefits and challenges of fault auto-correction, respectively. Section 6 summarizes the conclusions.

2. Method

In this study, a set of seven fault-correction algorithms for HVAC systems were tested in real buildings by two FDD partners (Table 1). The development of these routines is described in detail in Lin et al (2020a). The variables corrected by the algorithms span schedules, setpoints, sensor readings, commands, heating/cooling requests, and proportional, integral, derivative (PID) parameters. The algorithms were created based on a detailed literature review and domain expertise of the research and implementation team.

The routines can be broadly divided into three types (Table 1):

- <u>One-time correction</u>. After detecting and identifying the fault, the algorithm corrects it automatically. The correction is not re-triggered until the fault is detected again (Section 3.2.1). Algorithms #1-3 are one-time corrections of faults.
- <u>Active testing + one-time correction.</u> Similar to the one above, with the addition of one or more active tests performed before the variable is overridden in the BAS. The active tests perturb the system in specific operating conditions to determine the best values of parameters to be corrected (Section 3.2.2). Algorithm #4: Control Hunting is in this category.
- <u>Continuous optimization</u>. After identifying the opportunity, these routines act continuously to optimize system operation, behaving similarly to a “continuous” control algorithm. The overwriting of BAS variables happens with higher frequency than the other two types of routines and human intervention is not required to authorize each BAS variable update, although the algorithm may require initial operator’s approval (Section 3.2.3). Algorithms #5-7 belong to this type.

The selected auto-correction routines were implemented into two FDD products (Section 2.1). Then field tests were performed in four commercial buildings (Section 2.2) following the same testing procedure (Section 2.3).

Table 1: Summary of the seven auto-correction algorithms implemented and tested in this study

#	Fault/Opportunity Name	Fault/Opportunity Description	Type of Correction	Variables Corrected
1	HVAC schedules are incorrectly programmed	HVAC equipment doesn't turn on/off according to intended schedule due to error in control programming	One-time correction	Schedule
2	Override not released	Operator unintentionally neglects to release what was intended to be a short-term override of setpoints or other control commands (e.g. fan VFD speed, valve control command).	One-time correction	Override property of setpoint or command
3	Improve zone temperature setpoint setback	The zone temperature cooling setpoint is lower than needed or the heating setpoint is higher than needed while the space is scheduled occupied or unoccupied.	One-time correction	Zone temperature setpoint
4	Control hunting	The actuator operates under oscillation due to improper PID parameter setting	Active testing + one-time correction	PID parameters
5	Rogue zone	A zone continuously sends cooling/heating requests, due to zone-level equipment problems like a leaky reheat valve, a dysfunctional supply air damper, or unachievable zone temperature setpoints.	Continuous Optimization	Number of ignored requests from zones
6	Improve AHU static pressure setpoint reset	Non optimized AHU static air pressure setpoint	Continuous Optimization	Supply static pressure setpoint
7	Improve AHU supply air temperature setpoint reset	Non optimized AHU supply air temperature setpoint	Continuous Optimization	Supply air temperature setpoint

2.1 Implementation of auto-correction routines into FDD tools

The research team first developed the high-level algorithms in the form of flow charts (Lin et al 2020a). Later the FDD partners selected a subset of them to implement and deploy them based on desired new features for their platforms and the interest of their clients (Table 2). Partner 1 is an end-user with the staff and internal capability to customize the platform for their needs. Therefore, the routines developed by Partner 1 are site-specific customizations of the standard vendor platform. The FDD tool used by partner 1 is located on the premises and has direct access to the BAS network. This allowed Partner 1 to more easily implement continuous optimization routines and more complex BAS modifications. Partner 2, instead, is a FDD provider with a centralized, cloud-based platform without direct access to the BAS network. The algorithms were developed in a "sandbox" environment where initial testing of the auto-correction functionality took place. Once functionally tested and validated, these new platform features were incorporated into the "production" version of the software for deployment to the test buildings, making this capability also available to other customers while focusing on easily scalable algorithms.

Implementation activities included: (1) Modifying the FDD tool and the BAS to enable write capability into the BAS and to set up user interfaces for building operators. These changes are typically software modifications (creation of new points, interface programming, BAS logic changes), but can also include hardware changes or additions (e.g., a new auto-correction device.) (2) Coding the algorithms in the analytics engine of the FDD tool (3) Commissioning the algorithms, including a review of the auto-correction algorithm outputs. The results of this implementation step are described in Section 3.

2.2 Testing sites and equipment

FDD Partner 1 deployed the algorithms on two buildings in the same campus, while Partner 2 deployed them in two separate locations. The testing equipment includes AHUs, variable-air-volume boxes (VAV), fan coils (FC) and a heat recovery ventilation (HRV) unit for a total of 225 distinct pieces of equipment. The routines were also integrated with three different BAS vendor's platforms: Automated Logic Controls[1] (ALC), Johnson Controls Inc.[2] (JCI), and Delta Controls[3] (DC). Table 2 summarizes sites, equipment, BAS and tested algorithms. The tests were performed between the end of 2019 and the beginning of 2021.

Table 2: Summary of the field testing sites and equipment

FDD Partner	Site	Location	Equipment Tested	Algorithm Tested	BAS
Partner 1	Site A	Berkeley, CA, US	2 AHU, 48 VAV	4. Control hunting 5. Rogue zone	ALC
	Site B	Berkeley, CA, US	2 AHU, 163 VAV	5. Rogue zone, 6. Improve AHU supply air static pressure setpoint reset 7. Improve AHU supply air temp. setpoint reset	JCI
Partner 2	Site C	Vancouver B.C. Canada	3 FC and 1 HRV	1. HVAC schedules are incorrectly programmed 2. Override not released 3. Improve zone temp. setpoint setback	DC
	Site D	Atlanta, GA	1 AHU and 6 VAVs		DC

2.3 Testing procedure

After implementing the algorithms and deploying them into the buildings, their operation was tested using the following procedure. The procedure aimed at assessing the ability of an FDD tool to automatically correct a fault, and therefore allowed the "correction" capability of the FDD tool to be decoupled from its ability to perform detection and diagnostics. In this way, potentially confounding factors can be ignored, associated with false negative/positive detection or incorrect/missed diagnosis.

For each fault and automated fault correction procedure, each implementation partner did:

1. Verify the ability to override all setpoints/parameters to be tested.

[1] https://www.automatedlogic.com/en/
[2] https://www.johnsoncontrols.com/
[3] https://deltacontrols.com/

2. Use a naturally occurring fault or impose the fault in “clean” (fault-free) equipment or "assume" the fault is present if physical presence of the fault is not necessary to validate the behavior of the corrective action.
3. Observe and document the FDD tool output, i.e., its detection and diagnosis results (not applicable for assumed faults).
4. Execute the FDD-embedded correction routine.
5. Observe and document the effect of the automated fault correction.

The results of the tests are described in Section 4.

2.4 Interviews with partners, facility managers and industry advisors

At the end of the project, the research team conducted a series of interviews with seven different FDD providers and two facility managers. The researchers asked questions about perceived benefits of auto-correction, as well as market barriers and potential drivers of adoption of this technology. The interviews were transcribed and their content is summarized in Section 5, to support answering the third research question.

3. FDD tool and BAS modifications for auto-correction

Partner 1 implemented one *active testing + one-time correction algorithm* and *three continuous optimization algorithms*. Partner 2, instead, implemented three *one-time correction algorithms*. These are listed in Table 2.

3.1 FDD-BAS infrastructure update

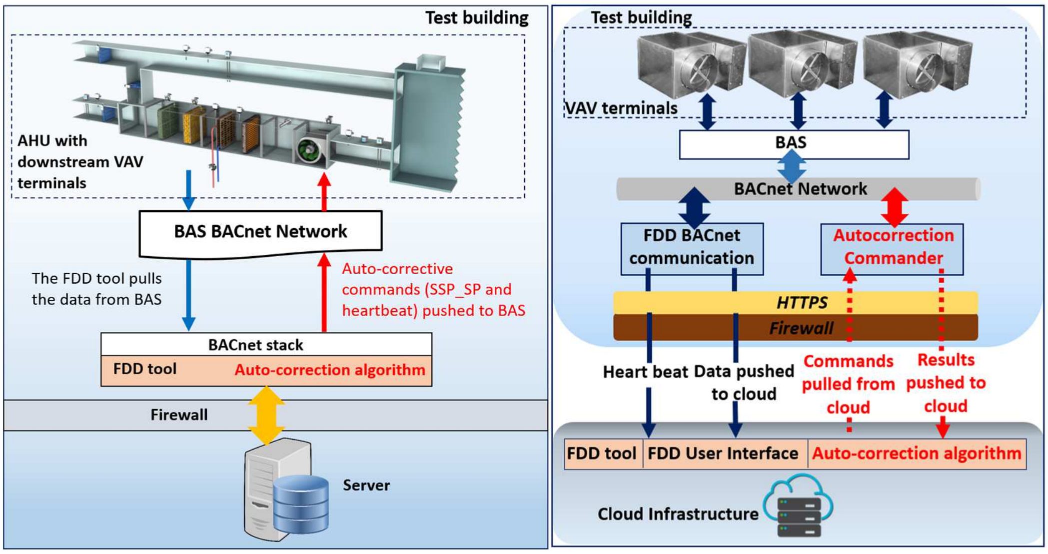
The current state-of-art FDD systems typically use one-way communication with the BAS, reading operational data, running analytics, and flagging faults on the software interface (Lin et al., 2020b). The first step in the software development for both partners consisted of enabling secure 2-way communication between the FDD tool and the BAS.

3.1.1 Partner 1 implementation

Due to cybersecurity requirements of the site for Partner 1, the FDD software is hosted on a server within the site’s firewall protected internal networks. The server collects data directly from BACnet/IP networks, and by existing on those networks it is also capable of issuing BACnet commands[4]. For this reason, access to the server is restricted to administrators. Data from the FDD software is replicated to a separate server for end-user access to visualization and reporting tools, accessible remotely. All FDD auto-correction routines, applications, and point mappings reside on the internal server. The architecture of the FDD tool and the BAS developed by partner 1 is presented in Figure 1a. The blue line shows the original infrastructure and the red line shows the upgrade. Before the start of the project the FDD tool already included a BACnet module, which implemented the BACnet communication protocol (ASHRAE, 2021), to extract data from the BAS. This BACnet module already enabled two-way communication, but each writable setpoint or

[4] For more information about known cybersecurity vulnerabilities related to the BACnet protocol, the reader should consult Holmberg and Evans, (2003) and Peacock et al. (2018).

command needed to be further configured in the FDD tool to enable writing operations and to define BACnet priority levels[5]. Several modifications also needed to be made in the BAS to make sure that the auto-corrected controls could operate even if the connection with the FDD tool was lost, described in Section 3.2.3.



(a) (b)

Figure 1: New FDD-BAS architecture created to support auto-correction (a: Partner 1, b: Partner 2)

3.1.2 Partner 2 implementation

The architecture of the new system is shown in Figure 1b. In traditional deployments, the most common FDD integration pathway for Partner 2 involves the installation of a local device within the BAS infrastructure. Once online, this device is tasked with systematically polling the networked devices to retrieve configuration and operational data, continuously delivering these data sets to the cloud servers for storage and analysis. The data-collection device is securely connected to the cloud platforms by limiting its interaction with a specified IP address and only initiating outbound messages from the site to the cloud. The existing FDD algorithms of the standard platform are run on the cloud and accessed via a web interface from any computer with the proper credentials.

To enable two-way communication to the FDD platform, Partner 2 opted for adding a new device – Autocorrection Commander, which manages the execution of the auto-correction algorithms. After BAS data is collected and pushed to the cloud platform, the new auto-correction algorithms are run in the cloud and the correction commands are prepared for execution. The new device periodically pulls these commands from the cloud and executes them on the BAS network. When the BAS receives the correction command, correction actions are implemented. The correction results are collected via the new device and pushed back to the cloud FDD platform. Compared to Partner 1, this implementation requires more attention to be paid to synchronization between cloud intelligence and local execution, because loss of connectivity is more likely to occur. Partner

[5] BACnet uses priority levels as a mechanism to assign priority to specific entities to prevent conflict between control actions.

2 implemented two features in the new device to avoid synchronization issues. The first feature is “value validation” which means the device will validate the value that has been collected (in case this value is changed after the FDD results are delivered) before it attempts to auto-correct it. If the value is as expected, the auto-correction would proceed, if it is different, it would deny auto-correction and insert an explanation of this in the activity log. The second feature is "command expiration" in case loss of connectivity delays the ability for the auto-correction device to communicate to the cloud and get the latest correction commands from the queue.

3.2 Software development for algorithms, BAS integration and UI

In addition to modifying the platform to enable two-way communication, each partner translated the research-grade algorithms generated by the research team into platform-specific auto-correction algorithms. This was accomplished by using the native scripting language of each platform. Other software modifications were required in both the FDD tool and the BAS, for example to create and integrate new points, to generate user interfaces, or to modify the BAS logic. Sections 3.2.1-3.2.3 describe the translation and Table 3 summarizes other software modifications.

Table 3: Software modifications in the FDD tool and the BAS in addition to the translation of the algorithms

#	Fault/Opportunity	FDD tool modification	BAS modification
1	HVAC schedules are incorrectly programmed	- Create new FDD writable point[6] (schedule) - Modify the user interface - Create an action log	- No algorithm-specific modification
2	Override not released	- Create new writable point (override) - Modify the user interface - Create an action log	- No algorithm-specific modification
3	Improve zone temperature setpoint setback	- Create new writable point (setpoint) - Modify the user interface - Create an action log	- No algorithm-specific modification
4	Control hunting	- Integrate PID parameters as new points - Create test management application - Create new database tables for PID loop info - Create test log	- Expose PID parameters to BACnet (when not available by default)
5	Rogue zone	- Create 1 new ignored requests[7] point for each zone (if desired for logging) - If not already present, add FDD rules related to rogue zone detection (e.g. leaky reheat valve) - Create ignore calculation application - Integrate new writable AHU ‘ignore’ points	- Add BACnet-exposed ‘ignore’ inputs to existing AHU control logic. - If logic doesn’t already exist, calculate new effective heating and cooling requests, using provided ‘ignore’ inputs

[6] Note: BAS and FDD tools typically store time-series data into a database sometimes called “historian”. By “point” we mean new variables linked to these time-series data.
[7] Requests and ignored requests (also called ‘ignore’) are defined in section 3.2.3

6	Improve AHU static pressure setpoint reset	- Create FDD writable point (setpoint) - Integrate required zone points if not already available (e.g. cooling PID output)	- Create new BAS point (setpoint) - Create new point for FDD heartbeat - Modify control sequence to use the new static pressure/supply air temp. setpoint when heartbeat is present - Modify BAS graphics to provide operator transparency and control.
7	Improve AHU supply air temp. setpoint reset		

3.2.1 One-time Correction

Partner 2 developed platform-specific algorithms #1: *HVAC schedules are incorrectly programmed*; #2: *Override not released* and #3: *Improve zone temperature setpoint setback* described in Table 1. The underlying FDD tool already saves several parameters describing the intended operation of the buildings, including schedules, control modes, and setpoints. These "recommended" parameters are actively determined from the operation history or selected by the facility managers. In the FDD tool, these parameters are continuously compared with current schedules and operation, and when the operation deviates from the recommended values, the facility managers are notified. In the standard implementation of the software, the user is required to use the BAS interface to revert the parameters back to the saved value or change this saved value at the FDD if the BAS value is deemed the more appropriate value. With auto-correction, the software was modified to allow updates of these parameters from the FDD tool. In order to accomplish that, the user interface was modified to show the recommended value and a log of the previous auto-correction actions (a and b in Figure 2 respectively). The end-users have the option to either approve the auto-correction action that reverts the values back to the recommended value or confirm the latest value is correct and update it to be the new recommended value. This additional evaluation step was adopted to earn facility staff's trust. The software also includes an option to enable automatic correction of faults, bypassing the approval, once the building manager has gained trust in the system. These two alternative paths are also represented in Figure 9 (i.e., correction evaluation and auto-approval). When executing the auto-correction actions for all three algorithms, the FDD tool changes the value of the related BACnet variable (i.e., Weekly_Schedule for #1, Out_Of_Service property for #2 and Present_Value for #3 respectively) to recommended values. On the BAS side, no change was necessary for each algorithm, aside from opening two-way communication between the FDD and BAS platforms.

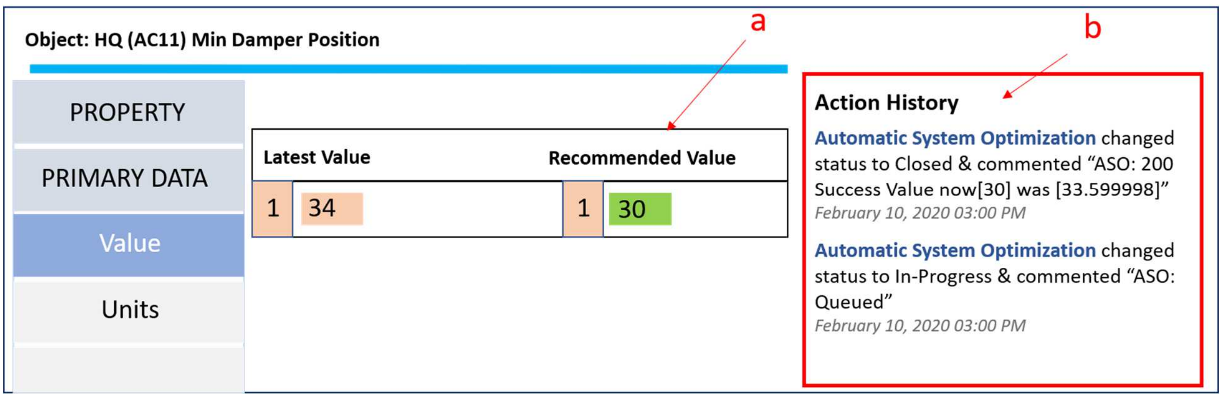


Figure 2: Update to the Partner 2's user interface[8] to accommodate automatic fault correction

3.2.2 Active testing + one-time correction

Algorithm #4: Control Hunting corrects the fault by overwriting the values of PID parameters in the control loop of the BAS. Partner 1 implemented the algorithm as three separate FDD software modules (b), (c), (d), in addition to a standard fault detection algorithm (a):

a. Fault detection algorithm
b. Management of active tests
c. Calculation of improved PID parameters
d. Database and interface to access test results

As shown in Figure 3, after the hunting fault is detected by module (a), the auto-correction is initiated by the facility manager. Modules (b), (c), (d) are executed to obtain the improved PID parameters through the designed active tests. In this prototype implementation, the improved PID parameters are shown through a custom interface in the FDD tool, then the facility manager manually enters the new parameters in the BAS.

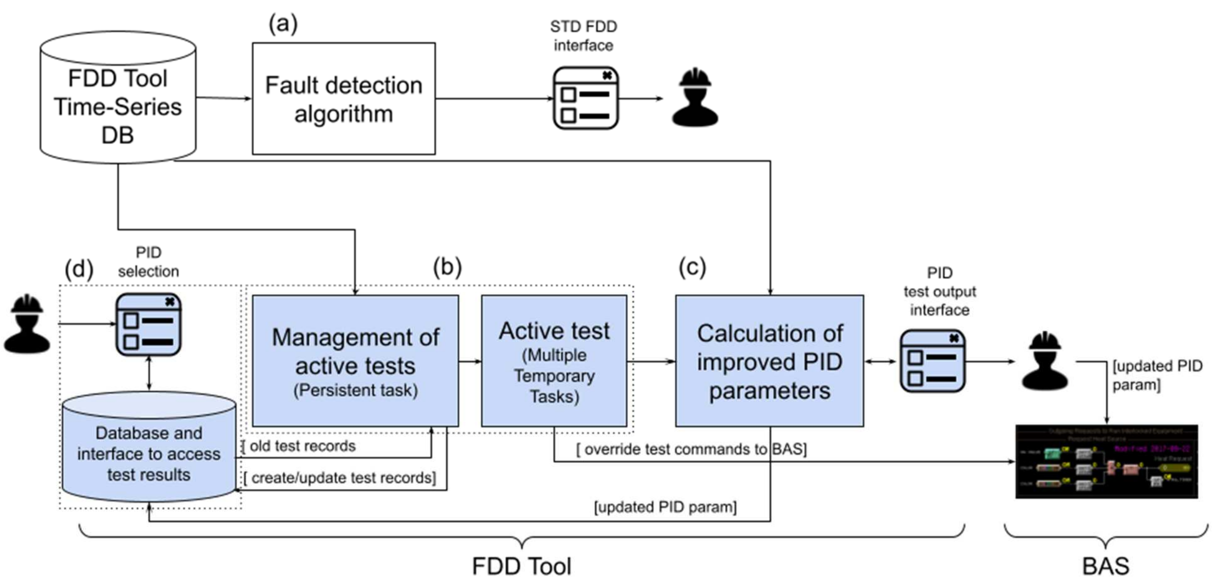


Figure 3: Software modules created (in blue) and updated (in white) in the FDD tool to implement the auto-correction algorithm

<u>(a) Fault Detection</u>

An existing fault detection algorithm is run in the background to identify what command variables are hunting[9] and on which equipment. The detection conditions look at the rate of change of the variable that is hunting. If the rate of change exceeds a certain threshold (e.g., 5%/min) continuously or more than twice during a certain period of time (e.g., 30 min), a fault is generated. The above algorithm is meant to detect bad cycling behaviors with minimal false positives.

[8] Image modified from FDD interface
[9] "Hunting" is a term used in the HVAC industry to indicate variables that keep oscillating with a higher frequency than expected.

(b) Management of active tests

As the start of the auto-correction process, this module initiates the active test following the Lambda open-loop tuning rule (Pruna et al, 2017) which determines the improved PID parameters based on the open-loop reaction of the process variable (e.g. temperature) to a change in the control variable (e.g. valve control command).

- A persistent task runs every hour to review all the PID loop records that have auto-testing enabled. If a PID loop requires a test (number of successful tests < target) and a test wasn't recently completed (in the last 24 hours), the software queues up a new test by creating a PID test record, and a new temporary, dedicated task.
- Temporary, dedicated tasks run every 5 minutes to perform the following steps sequentially:
    - Load its dedicated PID test record,
    - Synchronize time-series data to have the most recent data available for the process and control variables,
    - Check if testing conditions are met (e.g. airflow is detected for a reheat valve test),
    - If testing conditions are not met, the current test fails.
    - If testing conditions are met, monitor previous changes in the control variable and the corresponding reaction of the process variable to determine what action, if any, needs to be taken:
        - Override control variable to achieve stable state (start of test)
        - Override control variable to perform step change
        - Release override (end of test)

When enough data has been collected, module (c) is called.

(c) Calculation of improved PID parameters

This module is invoked during testing, if enough data has been collected after the execution of module (b). In this module, the improved PID parameters are calculated using the collected data of the control variable, the process variable, and the time between the step change in control variable and the response from the process variable. If the module then successfully returns improved PID parameters, the test is considered to be complete and successful.

(d) Database and interface to access test results

This module stores and views the results from modules (b) and (c). Information about each test is recorded in a database, including start and end timestamps, whether the active test is successful or failed, and the PID parameters' results from the successful tests. A user interface is also created that allows for viewing these results.

3.2.3 Continuous Optimization

Partner 1 implemented three algorithms #5, #6, and # 7 aimed at improving AHU operation using supply air temperature and static pressure resets, enhanced by an evaluation of rogue zones. These strategies are ranked in the top ten efficiency measures implemented by organizations using FDD technology based on FDD analytics results. (Kramer et al., 2020). The auto-correction algorithms for this opportunity are closely related to ASHRAE High-Performance Sequences of Operation Guideline 36 (ASHRAE, 2018), but deployed via the FDD tool instead of the BAS (Lin et al., 2020a). These algorithms determine the values of setpoints depending on the number of cooling “requests” generated by downstream zones that are served by the same AHU and write the improved setpoints into the BAS every five minutes. Details about their implementation are described in Lin et al. 2020a and Lin et al. 2020b. Table 3 summarizes the modifications necessary to their operation. These include creation of new points and new logic in the FDD tool and modification of interfaces and control sequences in the BAS. Heartbeat signals were also added in the FDD tool and sent to the BAS to constantly monitor connectivity between the two systems. If the BAS lost connection with the FDD tool it would revert back to the output of the old control sequence.

4. Field testing results

After implementing the routines and debugging them, each partner conducted formal field tests in real buildings, following the procedure highlighted in Section 2.3. The implementation partners successfully tested the algorithms, without adverse consequences, in at least one building and HVAC system. In two cases (algorithms #1 and #4) the code had to be modified to address problems identified during the field test. In seven cases, the faults were artificially imposed on the system, in order to test the procedure and in nine cases, the faults were successfully detected by the FDD tool. In two cases the FDD tool did not have the detection algorithms and the faults or opportunities were practically determined by the facility staff (N/A). The results of these field studies are summarized in Table 4 and described by type of algorithm in Section 4.1-4.3.

Table 4: Summary of test results of field testing in four buildings

#	Algorithm Tested	Site	Equipment	Artificially imposed	Fault detected	Auto-correction without adverse impact
1	HVAC schedules are incorrectly programmed	Site C	1 FC	Y	Y	Y
		Site D	1 AHU	Y	Y	Y
2	Override not released	Site C	3 FCs and 1 HRV	Y	Y	Y
		Site D	3 VAVs	Y	Y	Y
3	Improve zone temp. setpoint setback	Site C	3 FCs	Y	Y	Y
		Site D	6 VAVs	Y	Y	Y
4	Control hunting	Site A	1 VAV	Y	Y	Y
5	Rogue zone	Site A	2 AHU, 48 VAV	N	Y	Y
		Site B	2 AHU, 163 VAV	N	Y	Y
6	Improve AHU static pressure setpoint reset	Site B	2 AHU, 163 VAV	N	N/A	Y
7	Improve AHU supply air temp. setpoint reset	Site B	2 AHU, 163 VAV	N	N/A	Y

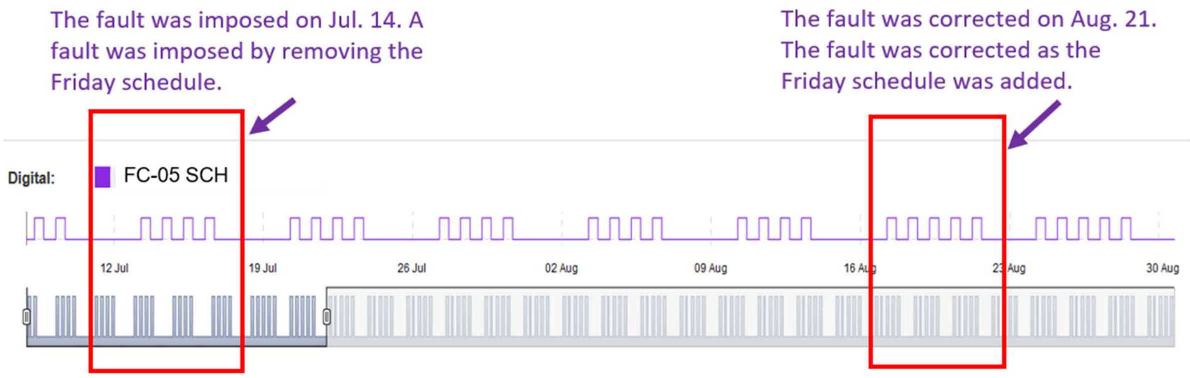
4.1 Testing results of one-time correction algorithms

Partner 2 tested algorithm #1 on two pieces of equipment in two sites, algorithm #2 on seven pieces of equipment and algorithm #3 on nine pieces of equipment across two sites (Table 4). All the faults were artificially imposed on the equipment and then the facility manager executed corrections after the faults were flagged in the FDD tool.

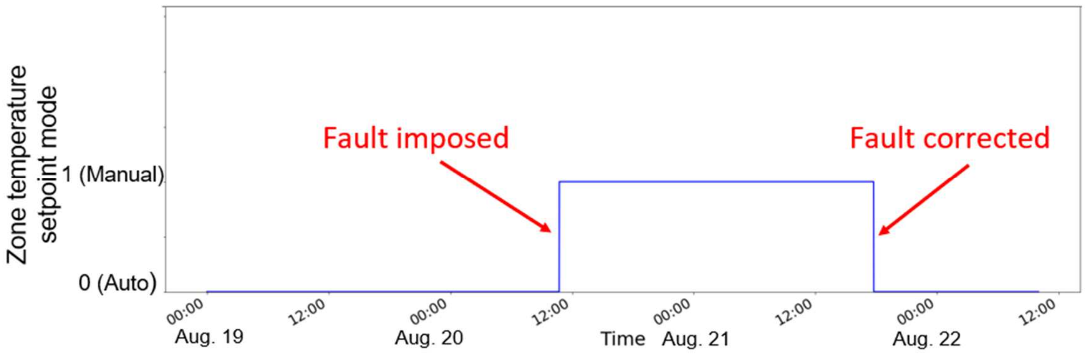
For algorithm #1: *HVAC schedules are incorrectly programmed,* two cases failed and two were successful in the two testing sites. The two early tests failed due to inconsistencies in the implementation of the schedule object in the BAS. Two additional tests were successful after updates in the auto-correction code. An example of the successful results is presented in Figure 4 (a). The equipment schedule was modified by purposely deleting Friday's schedule set to 6:30 AM - 5:00 PM on the BAS object property 'TEMP_SCH'. The fault was correctly identified by the FDD tool on July 15. Figure 4 (a) shows the Friday schedule disappeared after the fault was implemented and reappeared on August 21 after the correction action was authorized by the user. Between the start of the test on July 1 and the successful correction on August 21, the first test failed due to an integration issue between the BAS and the FDD platform. This required changes in the FDD software that was resolved the second week of August. Following this software update, the correction operated as expected.

For algorithm #2: *Override not released,* the correction succeeded in all test cases at the two test sites. The results of an example test case "The zone temperature setpoint mode of a FC was overridden from auto to manual" is presented in Figure 4 (b). The value of "zone temperature setpoint mode" was changed from 0 (auto) to 1 (manual) when the fault was imposed on August 20. Auto-correction was executed at 18:00 on August 21 after the fault was detected, and successfully changed back the value from 1 (manual) to 0 (auto). In the other test cases, the mode of other setpoints (i.e., maximum, minimum, or actual space temperature setpoints, the night-heating setback enable temperature setpoint, and the CO_2 differential setpoint) were overridden from auto to manual, and the algorithms also successfully converted them back to auto.

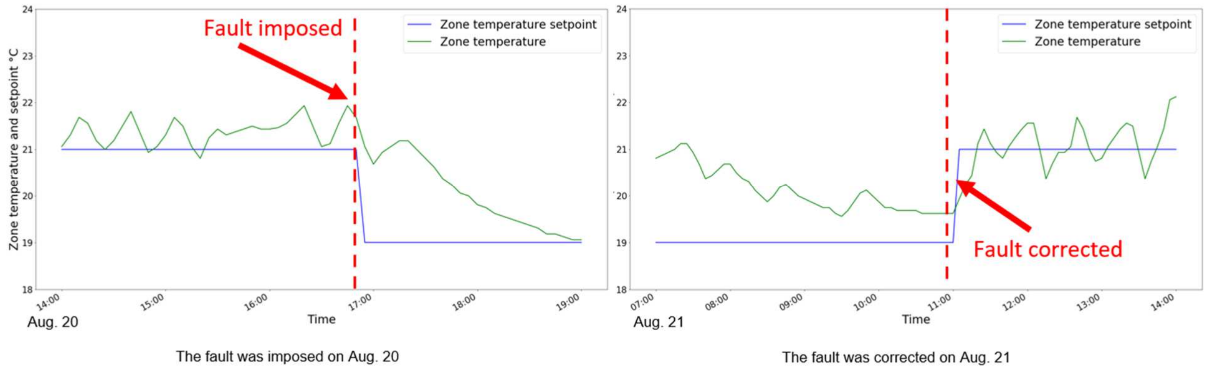
For algorithm #3: *Improve zone temperature setpoint setback,* the values of zone temperature setpoints were changed to impose faults. All the faults were successfully corrected without adverse impact. For example, during the test of a FC in Site C, the actual zone temperature setpoint was changed from 21 °C to 19 °C at 5:00 PM August 20 to impose the fault. Figure 4 (c) shows the zone temperature and setpoint between August 20 and August 22. After the fault was imposed at 5:00 PM, the zone temperature setpoint was decreased from 21°C to 19°C. As a result, the zone temperature dropped until it reached the new setpoint of 19°C two hours later. The next day (i.e., August 21), the zone temperature tracked the wrong zone temperature setpoint until the correction action was executed at 11:00 AM August 21. Consequently, zone temperature reached the corrected setpoint value 21°C.



(a)



(b)



(c)

Figure 4: Example auto-correction test results (a): #1 *HVAC schedules are incorrectly programmed,* (b): #2 *Override not released, and* (c): #3 *Improve zone temperature setpoint setback (left: the fault was imposed on Aug. 20; right: the fault was corrected on Aug. 21)*

4.2 Testing results of active testing + one-time correction algorithm

Partner 1 tested algorithm #4 *Control hunting* on a VAV box discharge air temperature control. In this control loop, the PID controller compares the setpoint to the discharge air temperature (process variable) to obtain the error, then the reheat valve command (control variable) is determined based on the error and PID parameters. The reheat valve command inputs to the actuators to generate actual control actions so that the discharge air temperature reaches the setpoint (Figure 5)

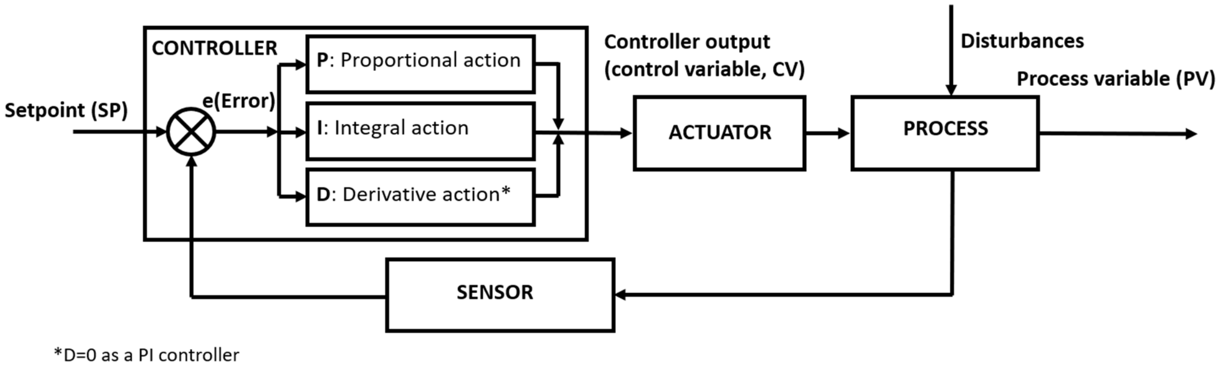
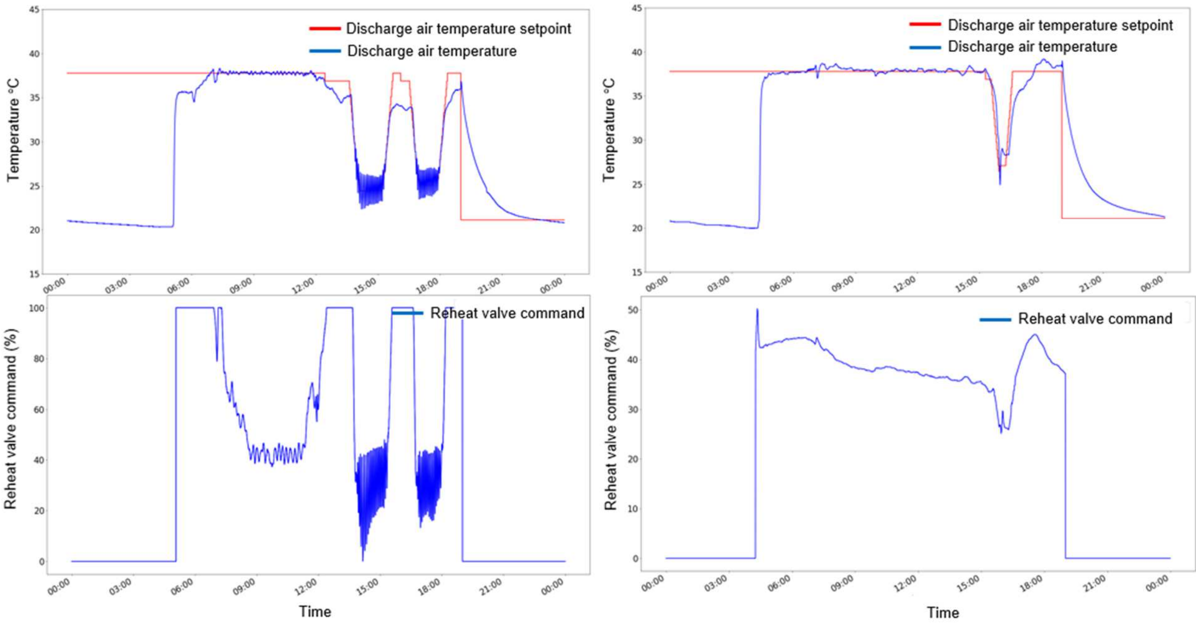


Figure 5: Control loop, control variable and process variable for test of algorithm #4

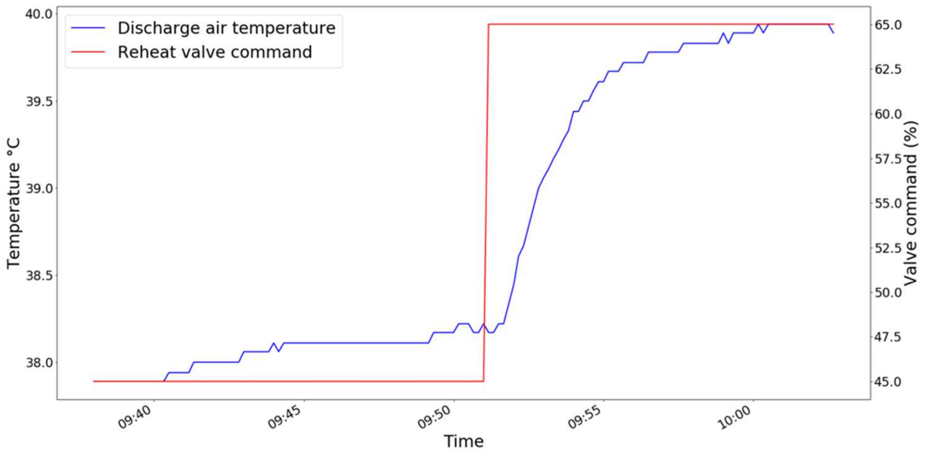
The successful test was conducted in January 2021. The behavior of the control and process variables on January 26th, 2021, before auto-correction, is displayed in Figure 6a. The top panel shows the Discharge Air Temperature (process variable, in blue) and the Discharge Air Temperature Setpoint (in red). The temperature oscillated more than 15 times per hour between 2-3pm and between 5-6pm. The oscillations were caused by the control variable, the Reheat Valve Command, displayed in the bottom panel (in blue). This hunting behavior was caused by improper PID parameters. Figure 6b shows trends from the same points on January 29th, 2021, after the auto-correction routine was executed. The oscillations of control and process variables (i.e., hunting) disappeared after the update of the parameters, and a hunting fault was no longer detected by the FDD tool.

To calculate the improved PID parameters for correction, the implementation team performed an active perturbation test, as described in Section 3.2.2. Figure 7 displays trends and the active test results. The results include proposed PID parameters - proportional and integral gains (Kp and Ki) determined from the test. To perform the open-loop step change test, the FDD tool increased the control variable (Reheat Valve Command of the VAV) from 45% to 65%. As a result, the Discharge Air Temperature increased from 38.4 °C to 40 °C. The improved PID parameters, calculated from Lambda open-loop tuning rules (Pruna et al, 2017), were proportional gain(Kp) = 0.5 and integral gain (Ki) = 0.1. Derivative gain is zero for PI controls, frequently used in HVAC control systems.



(a) Jan. 21 Before correction (b) Jan. 29 After correction

Figure 6: Comparison of behavior of the Discharge Air Temperature (process variable) and Reheat Valve Command (control variable) in Site A before (Jan 21th 2021, a) and after (Jan 29th 2021, b) the update of the PID parameters.



ALC Kp: proportional gain	ALC Ki: integral gain	ALC integral
0.5	0.1	1 min

Figure 7: Process variable (Discharge Air Temperature), control variable (Reheat Valve Command) and derivative of the process variable for the test of algorithm #4. The bottom table shows the results of the calculation of improved PID parameters in the FDD interface (ALC K_p and K_i)

4.3 Testing results of continuous optimization correction algorithms

Partner 1 tested algorithm #5 in two buildings and #6 and #7 in a single building. For each building, the routines were implemented on two large AHUs serving tens to hundreds of VAV boxes.

Detailed results of the preliminary tests of algorithms #5, #6, #7 are presented in Lin et al., 2020a, and Lin et al., 2020b. The new control strategies have been permanently adopted by the site beyond the testing requirements of the project and have now been running for over a year. The new control sequence did not cause any occupant complaints, and it worked more efficiently than the previous ones, although precise savings estimates were beyond the scope of the test. Figure 8 shows results when the FDD tool successfully changed the supply air temperature setpoint of one of two test AHUs based on the algorithms described in Section 3.2.3. The bottom of Figure 8 describes the number of calculated requests R', defined as:

Calculated (Heating or Cooling) Requests (R') = Incoming Requests (R) - Ignored Requests (I) (2)

When the number of R' became larger than zero starting at 10:05 a.m., the algorithm slowly reduced the SAT setpoint by 0.06 °C for each request every five minutes. Starting at 11:50 a.m., the requests remained at zero and the routine slowly increased the supply air temperature setpoint by 0.12 °C every five minutes until it reached a max value (SATmax=18.3 °C). The setpoint remained at SATmax until R' was positive again at 14:50 p.m. Then, the supply air temperature setpoint again slowly decreased when R' was positive and slowly increased when R' became zero. The strategy saved energy compared to the legacy control algorithm as illustrated in Lin et al., 2020a.

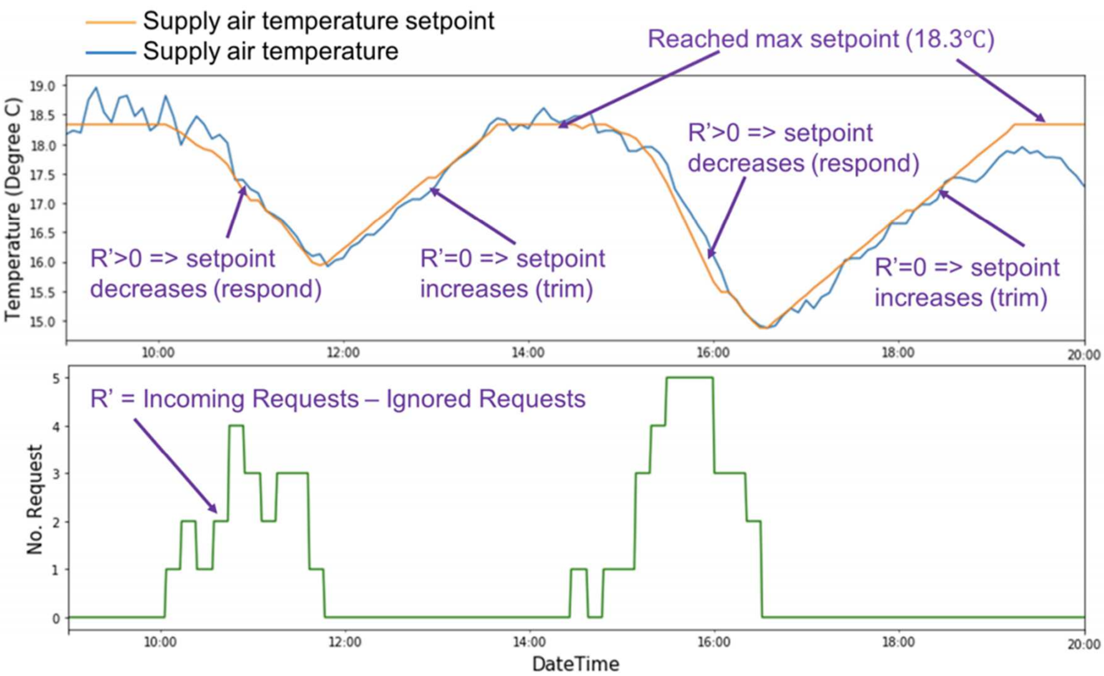


Figure 8: The SAT setpoint of an AHU after the execution of the auto-correction algorithm (Lin et al. 2020a)

5. Benefits and challenges of fault auto-correction

5.1 Technology benefits and market drivers

Commercial FDD platforms help to continually identify operational inefficiencies in building equipment. However, these FDD tools generate recommendations that need to be implemented by service technicians or other staff, resulting in delays, operations and maintenance costs or lost opportunities. All the FDD providers and facility managers interviewed during the project (Section 2.4) recognized these shortcomings and agreed that fault auto-correction integrating with commercial FDD technology offerings can close the loop between the passive diagnostics and active control. Several providers highlighted that many buildings with small operations teams may struggle to respond to FDD fault reports in a timely manner. The ability to auto-correct faults, even if it is only a subset of the total faults list identified by an FDD tool, can make a significant difference in the realized savings. One interviewee asserted that, for many organizations, auto-correction will be the primary way they can scale their ability to act on FDD findings.

In particular, the interviewees identified several benefits of this technology:

1. reducing the extent to which savings are dependent upon human intervention
2. scaling building operators' ability to act on FDD findings (especially for facilities with small operations teams),
3. tracking the changes executed on the BAS
4. applying consistent fixes for a subset of fault conditions,
5. saving a significant amount of energy from the routines related to optimal controls

To better understand how the different algorithms tested in this paper enable these benefits, the research team abstracted the workflow of each category of algorithm and compared them to the standard FDD workflow (Figure 9). Each box in Figure 9 represents a step in the process and the arrows indicate the data transferred between them. The steps that involve facility staff are indicated by a human icon, while the automated steps have no icon. The colored boxes represent changes compared to the standard workflows. The tasks in blue are automated by the algorithms, while tasks in red add a new step to the traditional process. The second and third group of algorithms show two parallel paths, because different options may fully automate the task or require human confirmation.

In the traditional case, the FDD tool identifies the faults using data from the BAS (step B in Figure 9). A human (e.g., facility manager) evaluates these faults and plans a set of actions to fix the identified issues (step C). After this phase, other actors fix physical problems in the underlying systems or reprogram the BAS (step F). The resulting actions are typically, but not always, recorded in a system different from the FDD tool (step G), for instance a computerized maintenance management system (CMMS) (Wireman, 1994). The manual steps in the implementation of the corrective actions and the difficulties in tracking their outcomes are often recognized as limitations of current FDD platforms (Granderson et al., 2017).

The algorithms proposed in this paper improve over this base workflow through some degree of automation, but they differ in some of the steps. All the algorithms automate the correction of faults (step F: action execution), thus contributing to reducing the dependency of savings from human intervention (benefit #1) and increasing the ability of the facility team to act on FDD findings (benefit #2). However, they also replace the manual evaluation of faults (step C, in the basic workflow) with additional steps that depend on the algorithm group. For the *one-time correction* algorithms, the detection of the fault triggers a proposed action. The user can manually approve it or decide to approve it automatically (step E). Partner 2 plans to implement options in the interface in future that allow users to auto-approve certain corrections, after gaining trust in the system (Step E, in blue). This feature will allow to apply consistent fixes to a subset of fault conditions (benefit #4). After this decision, the FDD tool generates a command, pushes it to the BAS (step F) and tracks it in a log (step G).

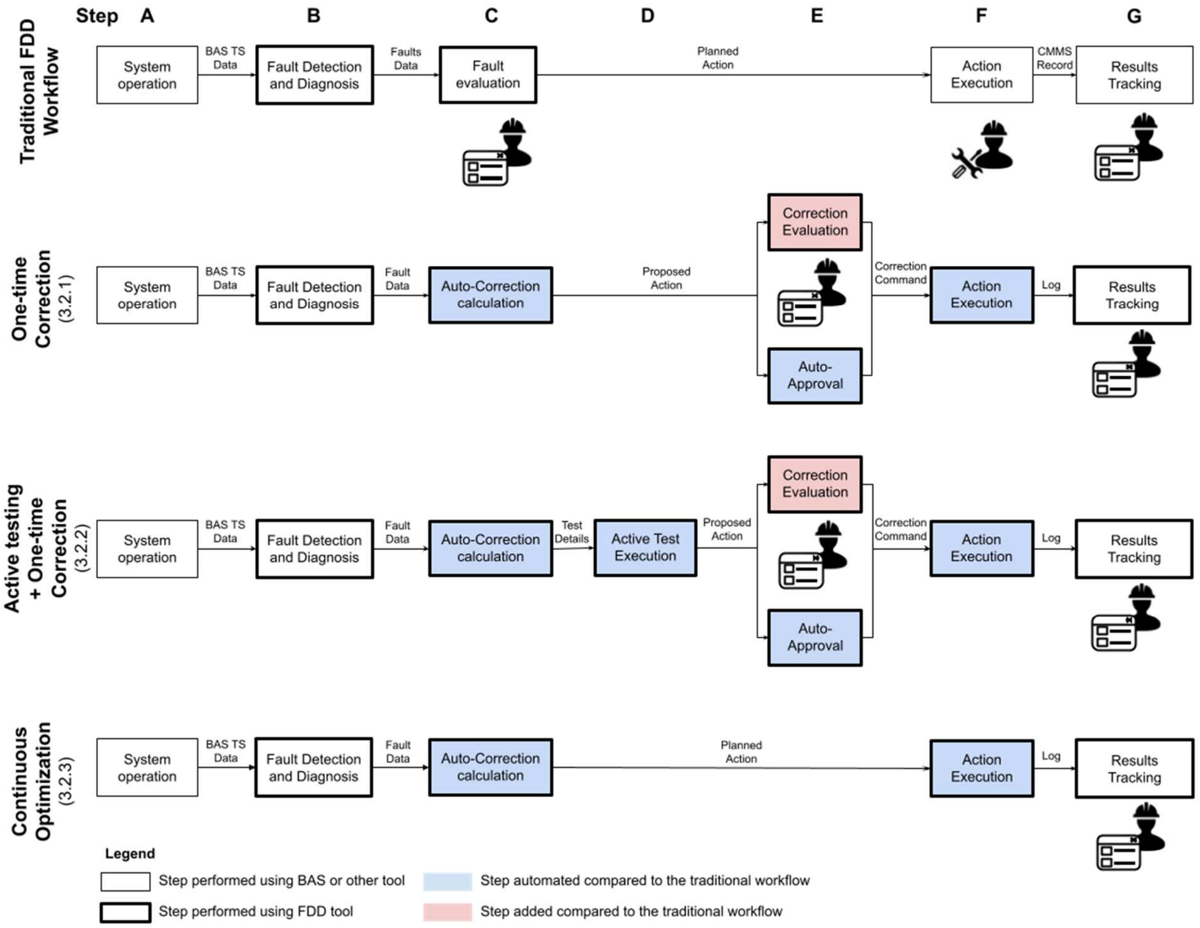


Figure 9: Traditional FDD workflow and enhanced process with the three fault correction types

The workflow for *active testing + one-time correction* algorithms add an additional step to the previous process, to gather additional information (step D, Figure 9) used to calculate parameters and recommend them to the user. Auto-correction routines that involve active testing have promising applications. For instance, automated tuning of PID loops could save operators the time to perform trial and error tests of parameters in the field. This is useful for control systems that don’t already have such functionality, or for which out-of-the-box results are not satisfactory.

The algorithms belonging to the *continuous optimization* category automate both fault evaluation (step C) and execution of the correction (step F). Since the correction takes place continuously, the operator is only involved in initial approval and periodic evaluations of the outcome of the strategy. Based on the interviews and preliminary test results (Lin et al, 2020a), these strategies have the highest potential for energy savings (benefit #5). Further, these routines may be especially cost-effective on sites where the underlying control infrastructure is obsolete and heterogeneous, because they allow the deployment of supervisory control algorithms more with less labor.

In addition, all the algorithms log the corrections enacted by the FDD tool, tracking the changes executed on the BAS (benefit #3).

The interviewees agreed that the drivers for market adoption of auto-correction features will be similar to those of FDD tools. For example, energy efficiency and conservation goals are likely to be increasingly important in the future. Labor shortages within the operation and management industry, caused by many facility staff approaching retirement age, may also favor solutions that automate parts of the traditional operation workflows. With common experience of electronics and other consumer devices, facility staff may also expect a better user experience with HVAC controls. A new driver may also emerge as building occupancy patterns are more dynamic post-pandemic, whereby auto-correction via the FDD tool can be a good option to implement the occupancy-based supervisory control strategies.

An additional driver is that FDD adoption continues to increase, meaning there is a larger market of established FDD users who will be looking for ways to extend their benefits beyond one-way fault detection.

5.2 Scalability challenges

While the benefits of this technology are significant, several challenges have to be addressed in future research to enable scalable deployment of these algorithms. The first common challenge is enabling secure two-way communication between the FDD tool and the BAS, allowing the FDD tool to override the BAS. The required effort for this integration varies depending on the IT/BAS network architecture. During this project this step was successful on two FDD tools and three types of BAS in three office buildings and one university student center, as described in Section 3.1. Additional field testing with more FDD tools and BAS types will be conducted in future to prove the generalizability of these solutions.

The second common challenge is overcoming cybersecurity and accountability concerns, when systems are controlled by a third party. The building owners and operators interviewed indicated that, similar to other supervisory control software, they are concerned about remote changes to BAS settings, especially for some building segments like military and healthcare. Interviewees also noted that many owners may be reluctant to hand over any portion of their building's control to a third party. The acceptability of this control overwritten will eventually be determined by the balance between risks and benefits perceived by the organizations using them. To mitigate this challenge, interviewees suggested ensuring the correction routines and corresponding control action be transparent, implementing proposed auto-correction routines only after the confirmation from onsite operations staff, enhancing auto-correction interface to build trust and confidence, and starting with owners who are well established with using FDD and looking for additional benefits. Close attention should be paid to clearly communicating to all the facility staff so that they are aware of the changes to the BAS made by the FDD tool.

The research team then evaluated each type of algorithm in relationship to the two dimensions of scalability, 1) required effort, 2) generalizability.

The *one-time fault correction* algorithms require low effort (coding and integration) and has high generalizability. They were the simplest to implement, because they modify schedules, setpoints, commands and sensor values, most of which are standard BACnet objects. For the same reason, the implementation partners believe these routines will be easy to scale up across multiple buildings using different BAS, given the growing adoption of BACnet in the control industry. However, the field test demonstrated that even when BACnet is used, different versions of the protocol or proprietary/custom objects used by different BAS vendors may require customization of the code. For example, Partner 2 discovered different implementations of schedule objects in different buildings, some starting the week from Monday, while others from Sunday. These discrepancies in data formatting were found between different versions of BACnet devices as well as in different implementations of the BACnet protocol stack. To ensure user acceptability, Partner 2 updated the user interfaces of the FDD tool to allow operators to revise the proposed corrections and to automate the process even further. During the tests, users provided positive feedback and seem to accept these algorithms without problems.

The *one-time correction + active testing* algorithm requires high effort and has medium-low generalizability. Several interviewees highlighted the benefits of auto-correction of “control hunting”. Many PID loops in the buildings are out of tune, and it would take significant operator time to manually perform trial and error tests of parameters in the field. In spite of great potential, the effort required to develop this type of *active testing* algorithm was significant, since neither the FDD nor the BAS offered tools to manage the periodic tests needed. As described in Sections 3.2.2 and 4.2, three modules and a new interface were added in the FDD tool to understand and manage these tests. Timing and scope of the active testing were managed programmatically, by setting allowed testing times and other conditions based on available trends (e.g. zone temperature) in order to ensure that the desired perturbation of the system did not adversely affect occupant safety or comfort. The generalizability of the auto-correction of PID parameters is medium to low. While the single test with Lambda open-loop tuning rule was successful in a reheat valve - discharge air temperature VAV-box control loop in this study, further work has to be done to fully automate this procedure, prove its robustness, and make it applicable to additional types of equipment. Partner 1 reported that the BAS controller tested did not expose PID loop parameters via BACnet and exposing them required significant manual work. Without proper standardization of BACnet objects and properties describing PID parameters, implementing these routines will require customization to interface with different implementations of PID loops.

The *continuous optimization* algorithms also require high effort and has medium-low generalizability. The development was time-intensive, because it required the modification of the BAS logic as well as the FDD tool. The BAS logic cannot be accessed via BACnet and its update currently requires dedicated and proprietary tools that depend on the BAS vendor. While recent research has been exploring how to digitize control sequences (Wetter et al, 2022), standardization of such workflows is still underway. Partner 1 successfully implemented these algorithms in the FDD software which is hosted on a server within the site's firewall protected internal networks. To scale up these algorithms in the cloud FDD software, FDD providers should develop methods to ensure the synchronization between the BAS and the FDD tool in controlling the equipment. Any loss of connectivity may cause delays in the control logic with negative consequences on occupant comfort and equipment safety. For example, partner 2 developed two new features "value validation" and "command expiration" to accommodate the asynchronous interaction of the data collection and the auto-correction devices.

6. Conclusion and Future Work

This paper presents the field study of seven fault auto-correction algorithms implemented in commercial FDD platforms. It puts the algorithms in their logical contexts, summarizes their objectives, describes the testing procedure, and shows the successful testing results. The algorithms automatically correct faults and improve the operation of large built-up HVAC systems, focusing on incorrectly programmed schedules, override not released, control hunting, rogue zone, and suboptimal setpoints in HVAC systems. These algorithms were integrated into two commercial FDD platforms and deployed across four buildings and three different building automation systems. The modifications of the FDD tool and the building BAS for auto-correction are summarized in the paper, including FDD-BAS infrastructure update and other software modifications. Each of the seven correction routines was tested in one or two buildings following a rigorous procedure. In general, the enhanced FDD tools were able to correct faults successfully without negatively impacting the system and building occupants. The control hunting correction was tested in a semi-automated way and the schedule correction was successful after some adjustments to the algorithms. Technology benefits, market drivers, and scalability changes are also discussed based on implementation and field testing results, as well as interviews with the FDD providers and facility managers.

Future work will focus on more field testing of the auto-correction algorithms with additional FDD platforms in a larger cohort of buildings to prove their robustness. This will include the evaluation of the technical efficacy and the performance of each correction routine, the evaluation of the operations and maintenance benefits for each site in the cohort and the characterization of challenges and best practices. A second area of future work should enhance the auto-correction interface of these FDD tools. This is needed to overcome the natural concerns among end-users about accountability and loss of control, when the FDD routines correct BAS control parameters automatically. At last, the current testing of the auto-correction algorithms was decoupled from the FDD algorithms embedded in the FDD tools. Faults were artificially induced to validate the correction capability. In future, the FDD and auto-correction process needs to be tested together to mitigate the impact of false positives during fault detection.

7. Acknowledgements

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Building Technologies Office, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The authors wish to acknowledge Harry Bergmann and Hannah Debelius for their guidance and support of the research. We also thank the fault detection and diagnostics technology and service providers who participated in this study.

8. References

American Society of Heating Refrigerating and Air Conditioning Engineers (ASHRAE). (2018). *Guideline 36–2018. High Performance Sequences of Operation for HVAC Systems*. ASHRAE. Akron, OH, USA, 2018.

American Society of Heating Refrigerating and Air Conditioning Engineers (ASHRAE). (2020). *Standard 135-2020—BACnetTM—A Data Communication Protocol for Building Automation and Control Networks.* Available online: https://www.ashrae.org/technical-resources/bookstore/bacnet (accessed on 10 Aug 2021).

Bengea, S.C., Li, P., Sarkar, S., Vichik, S., Adetola, V., Kang, K., Lovett, T., Leonardi, F., Kelman, A.D. (2015). Fault-tolerant optimal control of a building HVAC system. *Science and Technology for the Built Environment*, 21(6), 734–751. https://doi.org/10.1080/23744731.2015.1057085

Brambley, M., Fernandez, N., Wang, W., Cort, K.A., Cho, H., Ngo, H., Goddard, J.K. (2011). *Final project report: Self-correcting controls for VAV system faults filter/fan/coil and VAV box sections*. No. PNNL-20452; Pacific Northwest, National Laboratory (PNNL), Richland, WA, USA.

Dexter A., J. Pakanen (Eds.). (2001) *Demonstrating Automated Fault Detection and Diagnosis Methods in Real Buildings*, Technical Research Centre of Finland, Finland.

Economidou M. (2011). Europe's buildings under the microscope. A country-by-country review of the energy performance of buildings. Technical Report Buildings Performance Institute Europe.

Fernandez, N.; Brambley, M.; Katipamula, S. (2009a). *Self-correcting HVAC controls: Algorithms for sensors and dampers in air-handling units*, PNNL-19104; Pacific Northwest; National Laboratory: Richland, WA, USA.

Fernandez, N.; Brambley, M.; Katipamula, S.; Cho, H.; Goddard, J.; Dinh, L. (2009b). *Self-correcting HVAC controls project final report*, PNNL-19074; Pacific Northwest; National Laboratory: Richland, WA, USA.

Fernandez, N.E., Katipamula, S., Wang, W., Xie, Y., Zhao, M. Corbin, C.D. (2017). *Impacts on commercial building controls on energy savings and peak load reduction*. Pacific Northwest National Laboratory. PNNL Report Number PNNL-25985, Richland, WA, USA.

Granderson, J., Singla, R., Mayhorn, E., Ehrlich, P., Vrabie, D., Frank, S. (2017) . *Characterization and survey of automated fault detection and diagnostics tools*. Report Number LBNL-2001075.

Granderson J, Lin G, Singla R, Mayhorn E, Ehrlich P, Vrabie D, Frank S. (2018). Commercial fault detection and diagnostics tools: What they offer, how they differ, and what's still needed. *2018 ACEEE summer study on energy efficiency in buildings, 12-17 August, 2018, Pacific Grove, CA U.S.* https://escholarship.org/uc/item/4j72k57p

Hao, X., Zhang, G., Chen, Y. (2005). Fault-tolerant control and data recovery in HVAC monitoring system. *Energy and Buildings*, 37(2), 175–180. https://doi.org/10.1016/j.enbuild.2004.06.023

Holmberg, D. (2003). *BACnet Wide Area Network Security Threat Assessment,* NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, Gaithersburg, MD, [online], https://doi.org/10.6028/NIST.IR.7009 (Accessed November 10, 2021)

Johnson Controls Inc, (2020). Help files.

Kramer, H., Lin, G., Curtin, C., Crowe, E., Granderson, J. (2020). *Proving the business case for building analytics*. Lawrence Berkeley National Laboratory. https://doi.org/10.20357/B7G022

Katipamula, S., Brambley, M.R. (2005). Methods for fault detection, diagnostics, and prognostics for building systems—A review, part I. *HVAC and R Research*, 11(1), 3–25.

Kim, W., Katipamula, S. (2018). A review of fault detection and diagnostics methods for building systems, *Science and Technology for the Built Environment*, 24 (1), 3–21. https://doi.org/10.1080/23744731.2017.1318008

Lin G., Pritoni M., Chen Y., Granderson J. (2020a). Development and implementation of fault-correction algorithms in fault detection and diagnostics tools. *Energies*, 13(10). https://doi.org/10.3390/en13102598

Lin, G, Pritoni M, Chen Y, Moromisato R, Kozlen S, Granderson J. (2020b). Can we fix it automatically? Development of fault auto-correction algorithms for HVAC and lighting systems*. 2020 ACEEE Summer Study on Energy Efficiency in Buildings*. 17-21 August, 2020, Pacific Grove, CA U.S. https://doi.org/10.20357/B74C7N

Lin, G, Pritoni M, Chen Y, Moromisato R, Kozlen S, Granderson J. (2021). Fault “auto correction” for HVAC systems, a preliminary study*. 6th International High Performance Buildings Conference*. 24-28 May, 2021, West Lafayette, IN U.S. https://docs.lib.purdue.edu/ihpbc/369/

Peacock M., Johnstone M.N., Valli C. (2018). An exploration of some security issues within the BACnet protocol. In: Mori P., Furnell S., Camp O. (eds) Information Systems Security and Privacy. ICISSP 2017. *Communications in Computer and Information Science*, 867. Springer, Cham. February 19-21, 2017, Porto, Portugal. https://doi.org/10.1007/978-3-319-93354-2_12

Pruna E., Sasig E. R., Mullo S., (2017). PI and PID controller tuning tool based on the lambda method, *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies, CHILECON 2017 - Proceedings, 18-20 Oct. 2017.* Pucon, Chile. https://doi.org/10.1109/CHILECON.2017.8229616

Roth, K.W.,Westphalen, D., Feng, M. Y., Llana, P., Quartararo, L. (2005). *Energy impact of commercial building controls and performance diagnostics:market characterization, energy impact of building faults and energy savings potential*. Report prepared by TIAX LLC for the U.S. Department of Energy.

Wang, S.; Chen, Y. (2002). Fault-tolerant control for outdoor ventilation air flow rate in buildings based on neural network. *Building and Environment*, 37(7), 691–704. https://doi.org/10.1016/S0360-1323(01)00076-2

Wang, Z., Wang, Z., He, S., Gu, X., Yan, Z.F. (2017). Fault detection and diagnosis of chillers using Bayesian network merged distance rejection and multi-source non-sensor information. *Applied Energy*, 188, 200-14. https://doi.org/10.1016/j.apenergy.2016.11.130

Wetter, M., Ehrlich, P., Gautier, A., Grahovac, M., Haves, P., Hu, J., Prakash, A., Robin, D. and Zhang, K. (2022). OpenBuildingControl: Digitizing the control delivery from building energy modeling to specification, implementation and formal verification. *Energy*, 238, 121501. https://doi.org/10.1016/j.energy.2021.121501

Wireman, T. (1994). Computerized maintenance management systems. Industrial Press Inc.

Zhang, Y., Jiang, J. (2008). Bibliographical review on reconfigurable fault-tolerant control systems. *Annual Reviews in Contro*l, 32(2), 229–252. https://doi.org/10.1016/j.arcontrol.2008.03.008

Zhao Y, Wang S, Xiao F. (2013). Pattern recognition-based chillers fault detection method using support vector data description (SVDD). *Applied Energy*, 112,1041-8. https://doi.org/10.1016/j.apenergy.2012.12.043