Title

Development and Implementation of Fault-Correction Algorithms in Fault Detection and Diagnostics Tools

Journal

Energies, 13(10)

ISSN

1996-1073

Authors

Lin, Guanjing

Pritoni, Marco

Chen, Yimin

et al.

Publication Date

2020

DOI

10.3390/en13102598

Peer reviewed

energies

MDPI

Article

Development and Implementation of Fault-Correction Algorithms in Fault Detection and Diagnostics Tools

Guanjing Lin, Marco Pritoni, Yimin Chen and Jessica Granderson *

Lawrence Berkeley National Laboratory, Berkeley, CA 94706, USA; gjlin@lbl.gov (G.L.); mpritoni@lbl.gov (M.P.); yiminchen@lbl.gov (Y.C.)

* Correspondence: JGranderson@lbl.gov; Tel.: +1-510-486-6792

Received: 29 April 2020; Accepted: 19 May 2020; Published: 20 May 2020

Abstract: A fault detection and diagnostics (FDD) tool is a type of energy management and information system 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. Building operators on the leading edge of technology adoption use FDD tools to enable median whole-building portfolio savings of 8%. Although FDD tools can inform operators of operational faults, currently an action is always required to correct the faults to generate energy savings. A subset of faults, however, such as biased sensors, can be addressed automatically, eliminating the need for staff intervention. Automating this fault "correction" can significantly increase the savings generated by FDD tools and reduce the reliance on human intervention. Doing so is expected to advance the usability and technical and economic performance of FDD technologies. This paper presents the development of nine innovative fault auto-correction algorithms for Heating, Ventilation, and Air Conditioning pi(HVAC) systems. When the auto-correction routine is triggered, it overwrites control setpoints or other variables to implement the intended changes. It also discusses the implementation of the auto-correction algorithms in commercial FDD software products, the integration of these strategies with building automation systems and their preliminary testing.

Keywords: fault correction; fault detection and diagnostics; building operation; energy efficiency; field testing

1. Introduction

Commercial buildings constitute 18% of the U.S. primary energy consumption [1] and account for $149 billion in annual energy expenditures [2]. Much of this consumption is due to operational waste, representing a tremendous potential for savings. The literature indicates that median whole-building savings of 16% are achieved by commissioning existing buildings [3] and that 5–30% of commercial building energy use is wasted due to problems associated with controls [4–9].

Commercially available fault detection and diagnostics (FDD) tools provide a means of monitoring-based commissioning, through which instances of operational inefficiency can be continuously identified, isolated, and surfaced for resolution by operations and maintenance staff. Today's FDD technology has been documented to enable whole building savings of 8% on average, across users [10]. These technologies integrate with building automation systems (BASs) or can be implemented as retrofit add-ons to existing equipment, and continuously analyze operational data streams across many system types and configurations. This is in contrast to the historically typical variants of FDD that are delivered as original equipment manufacturer-embedded equipment features or handheld FDD devices that rely upon temporary field measurements.

Figure 1 represents an idealized architecture of a BAS, adapted from American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) Guideline 13 [11]. Field devices

(and controllers) connect to the sensors and actuators in the field. Network controllers typically provide supervisory control capabilities, scheduling, alarms, trending, local data storage, and user interfaces, in addition to some security features. Modern versions of these controllers have the ability to communicate via a BACnet (a data communication protocol for building automation and control network) IP over an IP network. When such functionality is not available, a common integration strategy employs "integration gateways" (e.g., Niagara JACE) that translate from proprietary protocols to standard protocols, such as BACnet IP. For larger installations and campuses, the controllers or gateways are also connected to a BAS server that provides configuration and management, long-term data storage (i.e., databases) and visualization tools. FDD tools can be installed in the local IP network, run from the cloud, or have a combination of cloud and local components. Integration with the BAS is typically implemented through a one-way interface using one of these three FDD–BAS integration pathways:

1. The FDD tool collects data from the central server database (common for large campus-wide installations) via a database application programming interface (API) (e.g., structured query language).
2. The FDD tool collects data from a central server, controller, or gateway using vendor-specific API (e.g., Automated Logic web services).
3. The FDD tool collects data directly via the BACnet IP network shared with other controllers and gateways.

Through these interfaces, system-level operational data are made available to the FDD software. Meter data are often also included. Data are continuously analyzed and detected faults are presented to operational staff through a graphical user interface. Since the BAS is the primary source of data, the FDD is most commonly focused on Heating, Ventilation, and Air Conditioning (HVAC) equipment. However, today's technologies offer extensive libraries of FDD logic and algorithms, and therefore can be applied to lighting and other building end-use systems for which data are available [12].

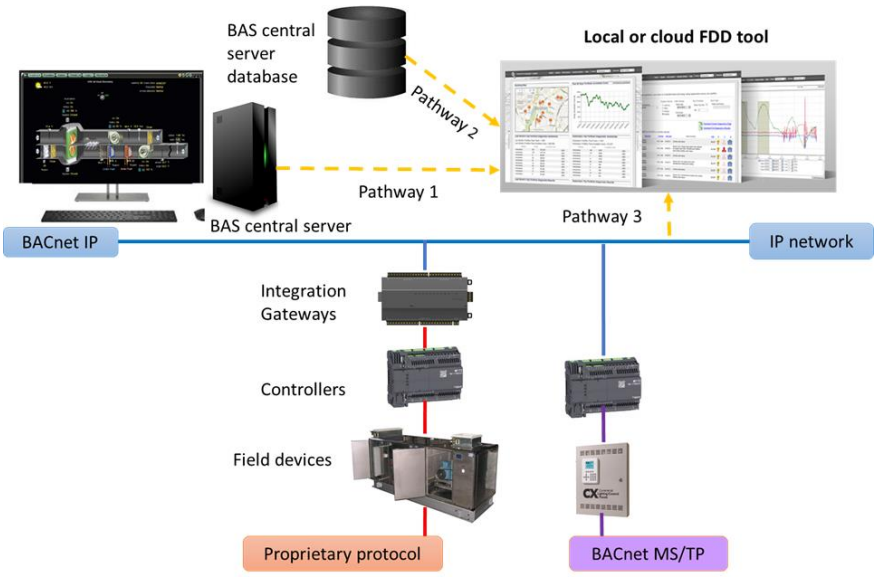


Figure 1. Schematic illustration of the integration of building automation system (BAS) data into fault detection and diagnostics (FDD) products. (BACnet MS/TP - BACnet Master Slave Token Passing protocol).

Although FDD tools are being used to enable cost-effective energy savings, there remains an opportunity to advance the state of the technology. In practice, the need for human intervention to fix faults once they are identified often results in delay or inaction, resulting in additional operations and maintenance (O&M) costs or lost opportunities. Traditionally, FDD generates recommendations

and follow-up actions which are implemented by service technicians or other staff. An emerging capability comprises the integration of FDD outputs with facility management "work order" or CMMSs (computerized maintenance management systems). While this makes it possible to automatically generate work orders from the FDD system, human intervention is still required to implement the corrective action. Therefore, this work seeks to develop automated fault-correction approaches and integrate them with commercial FDD technology offerings, thereby closing the loop between the passive diagnostics and active control. This is possible by converting the one-way BAS interface into a two-way interface, as is done with supervisory predictive control technologies that are emerging in the market.

It is not possible to automate the correction of mechanical faults such as failed actuators; however, there is nonetheless a compelling set of operational problems that are detectable in today's FDD offerings and are correctable through the software-based manipulation of the BAS parameters that can be exposed to external applications via BACnet. For example, Fernandez et al. [6] assessed control problems in commercial buildings, as well as their prevalence and whole-building energy impact for key commercial building sectors. Among the most common faults that relate to biased sensors, improper control parameter settings and inefficient schedules have significant impact and high prevalence rates. Automating the correction of these types of faults can increase the savings realized through the use of FDD tools and reduce the extent to which savings are dependent upon human intervention.

The academic and technical literature has extensively covered the development of automated FDD applied to HVAC and lighting systems [13,14]; however, very little has been published on the automatic correction of the identified faults via the actual control system. One set of relevant papers stem from the vast literature on rule-based FDD algorithms. Fernandez et al. [15,16] developed passive and proactive fault auto-correction algorithms for various HVAC components and systems. The methods proposed to correct some faults which include biased air-handler unit (AHU) mixed air (MA), outside air (OA), and return air (RA) temperature and humidity sensors; damper control hunting; minimum outdoor air damper too open/closed; and manual overrides in large HVAC systems. Using the same approach, Brambley et al. [17] extended Fernandez et al. [15,16] by adding correction routines for the biased AHU supply air (SA) temperature and flow rate sensors and the biased variable-air-volume (VAV) box discharge air temperature and flow rate sensors. This project implemented and tested a subset of these algorithms (sensor bias and minimum outdoor air damper position) in a laboratory experiment. This research stopped short of validating the developed solutions in physical buildings or integrating them with existing BAS and commercial FDD products.

Related to the concept of fault correction is a body of work in the building control literature that focuses on fault tolerant control. The purpose of a fault tolerant controller is to the maintain proper operation of a system despite the presence of faults [18,19]. These approaches have been widely adopted in other industries for safety-critical systems such as nuclear power plants, spacecraft and aircraft. In the context of buildings, Padilla et al. [20] developed a model-based strategy which aims to replace defective sensors in AHUs [20] with "virtual sensors." The signal generated from these "virtual sensors" can be used in the AHU control system when the actual physical sensors behave abnormally. Supply air temperature and pressure sensor faults are effectively corrected by using the proposed algorithms. Wang et al. [21] developed a supervisory control scheme that adapts to the presence of a measurement error in an outdoor air flow rate. The method uses neural network models to estimate the correct behavior of the faulty sensor and to maintain indoor air quality while minimizing energy use. Hao et al. [22] employed principal component analysis to develop fault-tolerant control and data recovery in the HVAC monitoring system. Bengea et al. [23] developed a fault-tolerant optimal control strategy for an HVAC system integrating FDD and model predictive control. The output of the FDD algorithm is used to continuously update the model's predictive control algorithm parameters. The approaches described in these papers offer innovations to the state of the art, yet they are not readily implemented in today's buildings control systems. This is because they comprise strategies

that are not supported by traditional BAS capabilities. Similarly, while the literature focuses on the development of these advanced controllers, it does not explore their integration with existing FDD technologies. An additional practical challenge is that a large volume of non-faulty data under various operational conditions is typically needed to train the models employed in these solutions.

This paper complemented and extended previous work in three ways. (1) It developed a comprehensive set of fault auto-correction algorithms designed to be integrated with commercial FDD tools. These algorithms target incorrectly programmed schedules, manual control "lock out," sensor bias, control hunting, rogue zone, and suboptimal setpoints/setpoints setback. Typically, commercial FDD tools are developed as a software layer on top of the existing BAS. There exists a natural separation of roles in this arrangement, in which the BAS actively controls the building and the FDD tool observes its operation and provides insights and recommendations to the building manager. The new auto-correction algorithms afford the FDD technology a certain degree of control capability. (2) It conducted preliminary testing and performance validation during which two auto-correction routines were deployed in a commercial FDD tool and tested on two AHUs in a real building. The enhanced FDD tool was able to correct faults successfully. (3) It presented the challenges of the integration of developed auto-correction algorithms into commercial FDD tools along with the solutions through work with three industry partners. New insights were gained by implementing the pseudo-code developed by the research team in real systems and real buildings. Sections 2–4 present the auto-correction algorithms, preliminary testing and the implementation changes and solutions, respectively. Section 5 concludes the paper and describes future work.

2. Fault Auto-Correction Algorithms

To identify the faults that are auto-correctable, we reviewed the existing literature and discussed the topic with 10 subject matter experts who had years of experience in FDD research and application. These experts included a set of FDD technology and service providers from the industry who were participating in this R&D effort as implementation partners. These providers maintained a large footprint in the FDD market and have explicitly included fault correction in their product development roadmaps. It is not possible to automate the correction of mechanical faults such as failed actuators, valve leakage and damper stuck, as they require physical repair or replacement. However, there is nonetheless a compelling set of operational problems that are detectable in today's FDD offerings, and correctable through a software-based manipulation of the BAS parameters that can be exposed to external applications via BACnet. Considering both the possibility of the automated correction and the feasibility of implementation in an FDD platform, a set of nine HVAC system faults were isolated, as shown in Table 1.

Table 1. Summary of the auto-correctable faults of focus in this study.

Fault	Fault Description
1. Schedules are incorrectly programmed	Heating, Ventilation, and Air Conditioning (HVAC) equipment does not turn on/off according to its intended schedule due to an error in the control programming.
2. Override manual control	Operator unintentionally neglects to release what was intended to be a short-term override of setpoints or other control commands (e.g., fan variable frequency drive (VFD) speed, cooling coil valve control command).
3. AHU OA or SA temperature sensor bias	Air-handler unit's (AHU's) outside air (OA) or supply air (SA) temperature sensor measurements have constant bias over time, representing an offset from a correct or true value.
4. Control hunting	The damper, valve, pump, or fan hunting fault due to an improper proportional gain.
5. Rogue zone	A rogue zone continuously sends cooling/heating requests whenever its schedule is on, due to the zone-level equipment problems like a leaky reheat valve, a dysfunctional supply air damper, or an insufficient capacity variable-air-volume (VAV) terminal.

Table 1. *Cont.*

Fault	Fault Description
6. Improve economizer high-lockout temperature setpoint	In the AHU with fixed dry-bulb economizer control, the economizer shall be disabled whenever the outdoor air conditions exceed the economizer high-lockout temperature setpoint. If the setpoint is set too low in the control logic, it will result in missed opportunity to use outdoor air to reduce the mechanical cooling load in mild conditions.
7. Improve zone temperature setpoint setback	Each zone has separate occupied and unoccupied cooling and heating setpoints. If the zoom temperature cooling setpoint is too low or the heating setpoint is too high, the space will be overcooled or overheated, causing unintended energy consumption.
8. Improve AHU static pressure setpoint reset	Non-optimized AHU static air pressure setpoint.
9. Improve AHU SA temperature setpoint reset	Non-optimized AHU supply air temperature setpoint.

For each of the faults in Table 1, correction algorithms were developed (for faults 1, 3 and 5–9) or adapted from the existing literature (for faults 2 and 4). The auto-correctable faults in Table 1 were divided into two categories: faults 1–5 were in the "Fault" category, which indicated problems that violated the intended operation of the equipment (e.g., sensor bias); faults 6–9 were in the "Opportunity" category, which indicated problems that represented potential to improve the current operation of the equipment (e.g., improve a setpoint reset). This distinction was made to differentiate between the intent of the restoring operation to what it was originally intended to be and that of optimal control.

As the objective of this study was to develop automated fault-correction algorithms that could be integrated with commercial FDD and BAS products, the auto-correction algorithms described in this section were decoupled from the fault detection and diagnostics algorithms embedded in the FDD tools. This permitted the applicability of the developed correction algorithms across a variety of FDD technologies that employed different FDD rules and algorithms. Furthermore, it was assumed that the FDD tools were able to detect the faults of focus, as they represented some of the more commonly encountered faults in commercial buildings.

Figure 2 shows the flow chart of the general auto-correction process. In this process, after the FDD algorithm generates a fault flag of a specific fault, the fault auto-correction algorithm is initiated to correct this fault with the approval from the building operator. Control_variable_being_overwritten is the key element in the auto-correction process. The algorithm overwrites this variable (Control_variable_being_ overwritten_current) with a new value (Control_variable_being_overwritten _new). The control_variable_being_overwritten_current is the one identified in the FDD algorithm to be associated with the problematic value (fault) or potential to improve (opportunity). The control_variable_ being_overwritten_new is the same variable that has the correct value (fault) or optimized value (opportunity). All of the auto-correction algorithms developed in this work followed this structure, with different control variables overwritten, and different ways to determine the correct or improved value of the variable.

Each auto-correction algorithm is presented and discussed in the following.

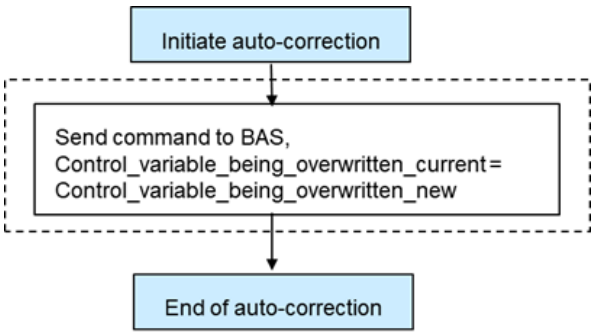


Figure 2. Flow chart of the general auto-correction process.

2.1. Schedules Are Incorrectly Programmed

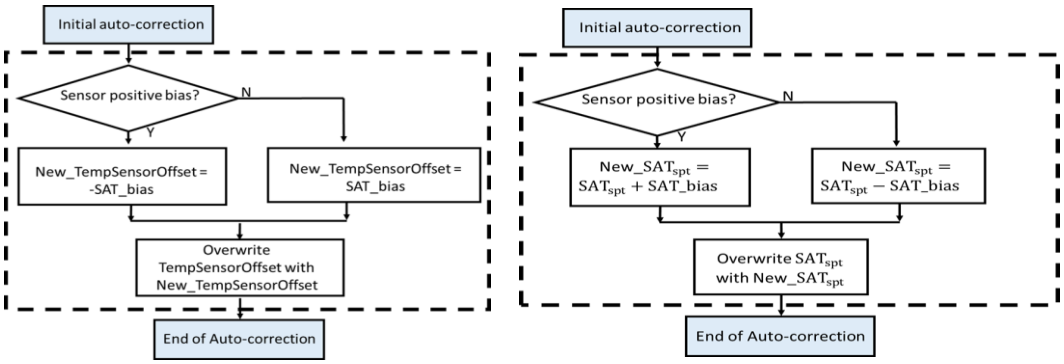
In this auto-correction algorithm, the control_variable_being_overridden is Equip_Schedule, which is the HVAC equipment on/off times that are programmed in the BAS for weekends, weekdays and holidays. The control_variables_being_overwritten_current are the current schedules read from BAS, which are concluded to be incorrectly programmed (e.g., the AHU starts at 4 a.m. and stops at 8 p.m. on weekdays). The control_variable_being_overwritten_new is Intended_Schedule, which is HVAC equipment on/off times as they are supposed to be (e.g., the AHU starts at 6 a.m. and stops at 7 p.m. on weekdays). The auto-correction algorithm overwrites the Equip_Schedule and replaces it with the Intended_Schedule to enable the system to start and stop as it is supposed to. The Intended_Schedule is prior knowledge that is specified by the building operator or another resource.

2.2. Override Manual Control

As documented in [15], the control_variable_being_overwritten in the auto-correction algorithm is Manual_Override. This variable indicates the equipment (e.g., fan speed, valve control command, damper control command) manual control status or equivalent flag: 1—equipment is in manual control, 0—equipment is in automatic control. When there is the override fault, Manual_Override = 1 in the faulty case. The correction algorithm changes the manual control variable back to automatic (Manual_Override = 0).

2.3. AHU Supply Air or Outside Air Temperature Sensor Bias

Two approaches can be used to correct the AHU supply air or outside air temperature sensor bias fault. In the first approach, the control_variable_being_overwritten is a data point (TempSensorOffset) in the control loop that controls the offset of the controller's temperature input. In the second approach, the control_variable_being_overwritten is the control setpoint associated with the temperature. These two approaches are presented here in detail for the supply air temperature (SAT) sensor bias fault. Figure 3a illustrates the auto-correction workflow for the first method. It is assumed that when the FDD algorithm flags the sensor bias fault it also determines the bias value, which is the difference between the sensor reading value and the actual value. This bias value (SAT_bias) is fed into the auto-correction algorithm. After judging the bias direction, the bias value is directly written to the TempSensorOffset. The adjusted TempSensorOffset is added from the SAT reading value and the conversion results (SAT+TempSensorOffset) enter the controller as input (e.g., adjust the incorrect SAT value with a new offset to provide the correct reading which is fed into the cooling/heating coil valve controller). Figure 3b illustrates the auto-correction workflow for the second method. In this method, the new SAT setpoint (SAT_spt) value can be calculated by adding or subtracting the bias value accordingly, and then be written to the BAS. The auto-correction of the outside air temperature uses the similar two approaches as above. The control_variables_being_overwritten in the second approach are the AHU economizer high (low) lockout temperature setpoints, which are the outside air temperatures above (below) which the outside air damper will return to its minimum position.



(**a**) (**b**)

Figure 3. Flowchart of the supply ai temperature (SAT) sensor bias fault auto-correction algorithm. (**a**) Approach 1: overwrite the SAT temperature value, and (**b**) Approach 2: overwrite the SAT setpoint.

2.4. Damper/Valve/Fan/Pump Control Hunting Due to Improper Proportion Gain

In contrast to the other algorithms, the auto-correction of control hunting due to improper proportion gain employs a trial and error procedure [15]. The control_variable_being_overwritten is the proportional–integral–derivative (PID) controller parameter proportion gain (Kp). In the auto-correction process, the Kp is continually adjusted to find out the appropriate value that eliminates the hunting behavior. When the FDD algorithm flags an improper Kp causing the hunting fault, the auto-correction algorithm is initiated. First, a maximum auto-correction duration threshold (T_AC_thresh) is set to avoid an endless auto-correction process. Note that a setting time of an actuator during the control response may be varied due to different actuator control characteristics. For example, for a VAV terminal unit damper, the settling time is typically in the order of one or two minutes, but the settling time for a cooling coil valve may be several minutes. Then, the current value of the Kp is compared to a Kp_threshold, in this case 0.2. This test is meant to avoid an unacceptably long settling time under pure integral control. If the Kp value is above the Kp_threshold, the Kp is decreased by 10% [15]. Then, the algorithm starts a proactive test scenario to see if the hunting issue still persists, by changing the setpoint (T_set) of the damper/valve/fan/pump to trigger the component's movement. If the component is still hunting, the procedure is repeated; otherwise, the procedure is terminated. If the Kp reaches the Kp_threshold and there is still a hunting fault, then it is flagged as an error and the Kp is reset to the original value (Figure 4).

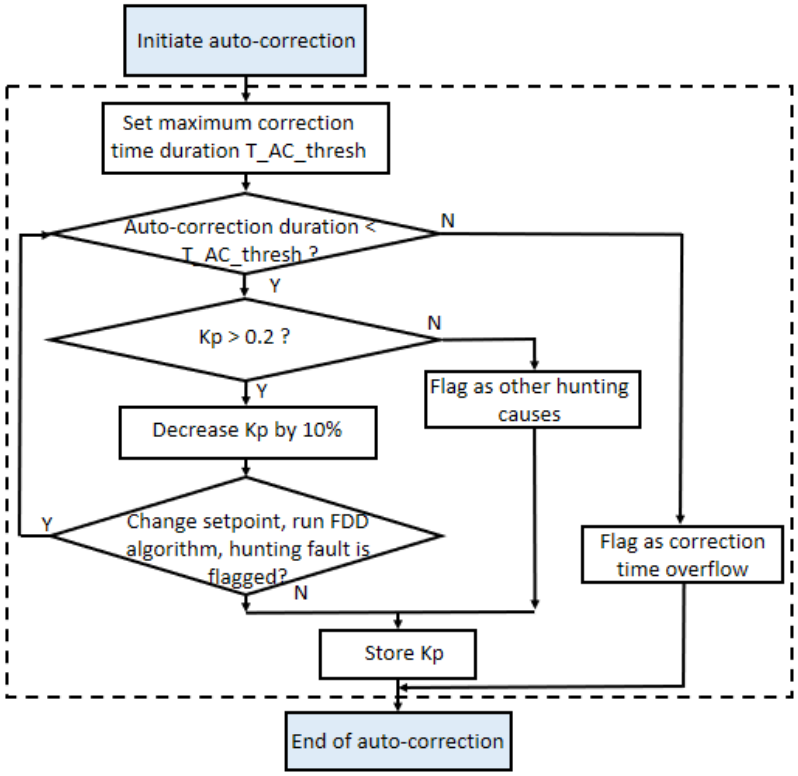


Figure 4. Flowchart of Algorithm 2.4: control hunting due to improper proportion gain (T_AC_thresh - a maximum auto-correction duration threshold, Kp - controller parameter proportion gain).

2.5. Rogue Zone

ASHRAE Guideline 36 [24] defines high-performance control sequences for AHU–VAV systems. The "Trim and Respond" logic (see Sections 2.8 and 2.9) is adopted to reset the supply air temperature and static pressure setpoints at an AHU. The adjustment of these setpoints depends on the number of cooling "requests" generated by downstream zones that are served by the same AHU. For each time step, the change value of setpoint (SPchange) is determined by Equations (1) and (2) below:

$$SP_{change} = SP_{res} \times (R - I) \quad (1)$$

$$R = \sum IM_i Request_i \quad (2)$$

where SPres is a unit respond amount (e.g., 0.06 inches for static pressure setpoint), R is the total number of cooling requests from the downstream zones, I is the defined number of ignored requests, i is the indicator of the downstream zone, IM is the importance multiplier that is used in the control sequence to decide if the cooling requests from the zone level should be used to control the upstream AHU, and *Request* is the cooling request from the zone. Therefore, if there is a rogue zone that continuously sends cooling requests whenever its schedule is on, due to the zone-level equipment problems, the parameter R will always include this request, and it keeps the setpoints in the control loop to its high end. Excluding rogue zones from the corresponding reset control strategies improves operation and increases energy savings. After the zone-level equipment problems that lead to the rogue zone are fixed, the rogue zone is no longer rogue, and all the control variables that are overwritten during the auto-correction process change back to their original value.

Two correction strategies were developed to eliminate the rogue zone impacts (i.e., to ignore the cooling request from the rogue zone). The first is to overwrite I in Equation (1). The auto-correction algorithm increases I by n for each currently identified rogue zone. The value of n is the same as the number of cooling requests determined in the control sequence of that rogue zone. The second is to

overwrite the *IM* of the rogue zone in Equation (2). When the FDD tool flags the rogue zone fault, the *IM* of the rogue zone is overwritten to be zero. Therefore, the cooling requests from the rogue zone can be removed.

2.6. Improve Economizer High-Lockout Temperature Setpoint

The previous five algorithms focused on correcting faults to restore the intended operation. This algorithm and the next three serve to provide more optimal control. They implement improved control setpoints or sequences when the FDD tool identifies the opportunity to do so.

After the opportunity to improve the economizer lockout temperature setpoint is identified, the setpoint is overwritten to the recommended value, following the flowchart in Figure 2. The recommended value can be determined based on the high-lockout limit recommended in the energy code [25]. For example, the recommended lockout setpoint is 23.9 °C, 21.1 °C and 18.3 °C, respectively, in the dry climate zone, the cold–humid climate zone, and the hot–humid climate zone.

2.7. Improve Zone Temperature Setpoint Setback

Similar to the algorithm in Section 2.6, this auto-correction algorithm overwrites the zone temperature cooling or heating setpoint during the occupied or unoccupied hours to the recommended values wherever there is an opportunity.

2.8. Improve AHU Static Pressure Setpoint Reset

The auto-correction algorithms for this and the next opportunities are most closely related to optimal controls. Both algorithms correct the fault "continuously" as it continuously adjusts the control variables to optimize the equipment operation (e.g., resets). They are relevant for AHUs without sophisticated reset strategies, such as no reset or simple resets based on return air temperature or outside air temperature.

The auto-correction algorithm uses the ASHRAE Guideline 36 [24] "Trim and Respond" logic for the static pressure setpoint. To optimize the operation of the AHU and minimize discomfort, the static pressure setpoint (SSP_spt) is continually reset using the Trim and Respond logic between a minimum and maximum setpoint (SP_{min} and SP_{max}). When the supply air fan is off, the setpoint is the initial setpoint (SP_0). The reset logic is active while the supply air fan is proven on, starting a delay timer (T_d) after the initial device start command. When active, for every time step T, when the cooling request from the downstream zones (R) is less than or equal to a defined number of ignored requests (I), the setpoint is trimmed by a trim amount (SP_{trim}), but no less than SP_{min}. If R is more than I, the setpoint changes by a respond amount, (i.e., $SP_{res} * (R - I)$), but no more than the maximum response per time interval ($SP_{res\text{-}max}$).

2.9. Improve AHU SAT Setpoint Reset

Similar to the algorithm to improve the static pressure setpoint reset, this auto-correction algorithm uses the ASHRAE Guideline 36 [24] "Trim and Respond" logic to reset the SAT setpoint continuously between a minimum and maximum setpoint. The control_variable_being_overwritten is the SAT setpoint.

3. Results: Preliminary Testing

Three commercial FDD providers participating in this research selected a subset of the algorithms that were created by the authors and integrated them into their development product environments for field testing. The partners chose the relevant algorithms for a variety of reasons, including: the expected ease of implementation, the reduction of operational cost, savings potential, and the ability to solve problems common to their customers. The implementation process varied depending on the platform, but generally consisted of the following phases: (1) confirm/add two-way communication

functionality between the FDD and the BAS, (2) build an auto-correction interface to communicate with the building operator, (3) translate the algorithms into the FDD programming environment, (4) modify the BAS programming of the specific building to integrate the new control actions sent by the FDD tool, and (5) commission and test the new system. Further details are presented in Lin et al. [26]. This section illustrates the test results of two auto-correction algorithms: "Rogue zone" and "Improve AHU supply air temperature setpoint reset" for one implementation partner. Section 4 summarizes the challenges that were faced by three partners during the implementation process, as well as the solutions that were used by one or more project partners to mitigate them.

In the preliminary testing, the two routines were deployed in a commercial FDD product (SkySpark® by SkyFoundry) and tested on two AHUs in a building in Berkeley, California, US. between March 3 and April 5, 2020. The goal of this preliminary test was to determine whether the enhanced FDD solutions were able to correct faults without adverse operational effects.

3.1. Description of the Testing Site and Equipment

Table 2 summarizes the test site and equipment information. AHU01 and AHU02 are structurally identical. Figure 5 shows the BAS graphics (i.e., native dashboard) for one of the two AHUs.

Table 2. Test site information.

Building Type	Size (m^2)	Building Schedule	HVAC Configuration	BAS Brand and Model
Mixed laboratory and office space	8919	Labs: 24/7 operation, Offices: 4 a.m.–9 p.m., Monday–Sunday	3 chillers and 2 AHUs (AHU01 and AHU02), covering about 90% of the floor area, and the connected zones (n = 83 and n = 80, respectively)	Johnson Controls (JCI) Metasys

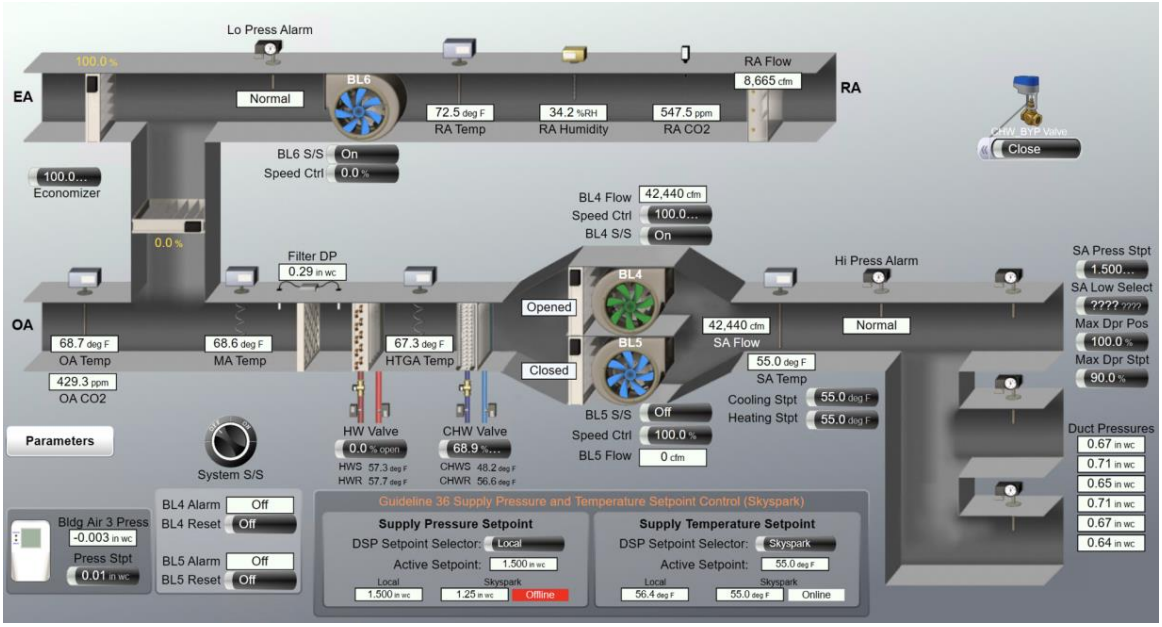


Figure 5. BAS graphics for the AHU02 at the test site. AHU01 has a similar structure.

Both AHU01 and AHU02 were controlled by a control sequence implemented in the native BAS control language and hosted on local controllers. Each AHU was controlled independently. The supply air temperature cooling and heating setpoint was reset based on the algorithm highlighted below in plain English:

- If the AHU is enabled (based on schedules, normally 24/7):
    - Calculate the average cooling demand output for all the zones served by the AHU (cooling demand output is the output calculated by the PI[D] loop based on the proportional, integral [and derivative] component of the difference between zone cooling setpoint and zone temperature).;
    - Constrain the results between min = 3% and max = 12%;
    - Linearly map the average output to a calculated cooling setpoint between 18.3 °C and 12.8 °C. The value of 3% average cooling output is mapped to 18.3 °C, 12% is mapped to 12.8 °C, and all the values in between are calculated linearly;
    - The heating setpoint is fixed to 12.8 °C, except for when the cooling setpoint reaches 12.8 °C. In that case, the heating setpoint becomes 12.2 °C.
- The economizer damper and the chilled water valve are controlled to maintain the cooling supply air temperature setpoint. The heating hot water valve is controlled to maintain the heating supply air temperature setpoint. As a result, when the outside air temperature is between the heating and the cooling setpoints, the air handling unit typically does not cool or heat the air.

The existing SAT control strategy is relatively efficient, compared to common practice in the industry (fixed setpoint or resets based on outdoor temperature or return temperature alone and no deadband). However, the current strategy presents two limitations: (1) it responds to outlier zones or rogue zones, although minimally, as the reset is based on an average cooling demand outputs from all the zones; and (2) its calibration parameters (e.g., min and max average zone feedback of 3% and 12%, respectively) were established via trial and error and personal judgement. Given the limited capabilities of the BAS zone controllers (i.e., field devices in Figure 1), the reset strategy was entirely calculated within the AHU controllers.

The FDD tool connected to the BAS is a commercial product managed by a consultant and the facility manager of the site. The tool allows for custom programming and bi-directional communication to the BAS via the BACnet network. In contrast to the BAS, the FDD tool coding language is a modern scripting language with the ability to use high-level functions that allow the portability of the code between the buildings and equipment. The two auto-correction algorithms were coded using this platform and tested on the two AHUs. In the FDD tool, a zone was identified as a rogue zone when one or more disqualifying conditions were detected for that zone and the zone was sending a request to the AHU. The zone requests are calculated based on zone PID loop output >95%. Disqualifying conditions for cooling requests include:

- Leaky reheat valve (VAV box discharge air temperature (DAT) > AHU SAT + 2.8 °C);
- Supply airflow setpoint not met (<90% or >110% of setpoint and delta > 1.4 cubic meter per minute);
- Zone cooling setpoint too low (lower than 22.2 °C unless exempt).

3.2. Auto-Correction Code in the FDD Tool

3.2.1. Code for "Rogue Zones"

The code adopts the first correction strategy in Section 2.5 and overwrites the number of ignored cooling requests from the identified rogue zones. The number of requests and ignored requests are calculated as in Equations (3)–(5):

$$R' = \max(R - I_{total}, 0) \quad (3)$$

$$I_{total} = I_{default} + I_{rogue\_zones} \quad (4)$$

$$I_{rouge\_zones} = \sum_i I_{rouge\_zone\_i} \quad (5)$$

where R' is the number of net cooling requests from the downstream zones of an AHU; R is the number of total cooling requests from the downstream zones; $I_{default}$ is the default number of ignored cooling requests (set by the user); I_{rouge_zones} is the number of ignored cooling requests from the all rogue zones; I_{total} is the sum of the previous two variables; and $I_{rouge\_zone\_i}$ is the number of cooling requests from the i^{th} identified rogue zone. R' is calculated by subtracting the sum of all the rogue zones ignored based on the conditions described above (I_{rogue_zones}) and a default minimum of the ignored zones ($I_{default}$) from all the requests (R). If the equation leads to a negative result, R' becomes zero. R' is used in the SAT reset calculation below.

3.2.2. Code for "Improve AHU Supply Air Temperature Setpoint Reset"

The supply air temperature cooling setpoint (SAT_spt) is continually reset using "Trim and Respond" logic between a minimum and maximum setpoint (SATmin = 12.8 °C and SATmax = 18.3 °C). When the supply air fan is turned on, the initial setpoint is set to SAT_0 = 18.3 °C and the reset logic is active immediately. When active, for every time step t = 5 min, the net cooling request from the downstream zones (R') is calculated using Equations (3)–(5) above. If the R' is above zero, SAT_spt is decreased by a defined respond amount (SATres = 0.06 °C for each request) until the SAT_spt reaches SATmin; if R' is equal to zero, the SAT_spt is increased by a fixed amount (SATtrim= 0.12 °C) until the SAT_spt reaches SATmax.

3.3. Test Results

3.3.1. Test Results of "Rogue Zone" Algorithm

The "rogue zone" auto-correction algorithm worked as expected on AHU01 and AHU02 during the testing period. Figure 6 shows the results of the values for the requests and ignored requests calculated on 4 March. The two heat maps in Figure 6 depict each zone (in the vertical axis) plotted against the time of the day, for a single day. For each zone, the darker areas show when the requests (red) and ignored requests (blue) happened during the day. R, I_{rouge_zones}, and R' were reevaluated every five minutes. Taking 11:00 a.m. as an example, the vertical line marks the values of R_i, $I_{rogue\_zone\_i}$, R, I_{total}, and R' at 11:00 a.m.: eight zones (i.e., Rm 4107, Rm 411, Rm 4113, Rm 4203B, Rm 5113, Rm 5115, Rm 5116, and Rm 5204A) sent requests ($R = 8$), three zones (i.e., Rm 4107, Rm 5113, and Rm 5116) were identified as rogue zones ($I_{rogue\_zones} = 3$), and two additional zones were ignored by default ($I_{default} = 2$), summing up to a total of five zones ignored (I_{total} = 5). R' is therefore equal to 3, based on Equation (3). This test confirmed that the system correctly calculated and implemented the modified requests, ignoring the rogue zones.

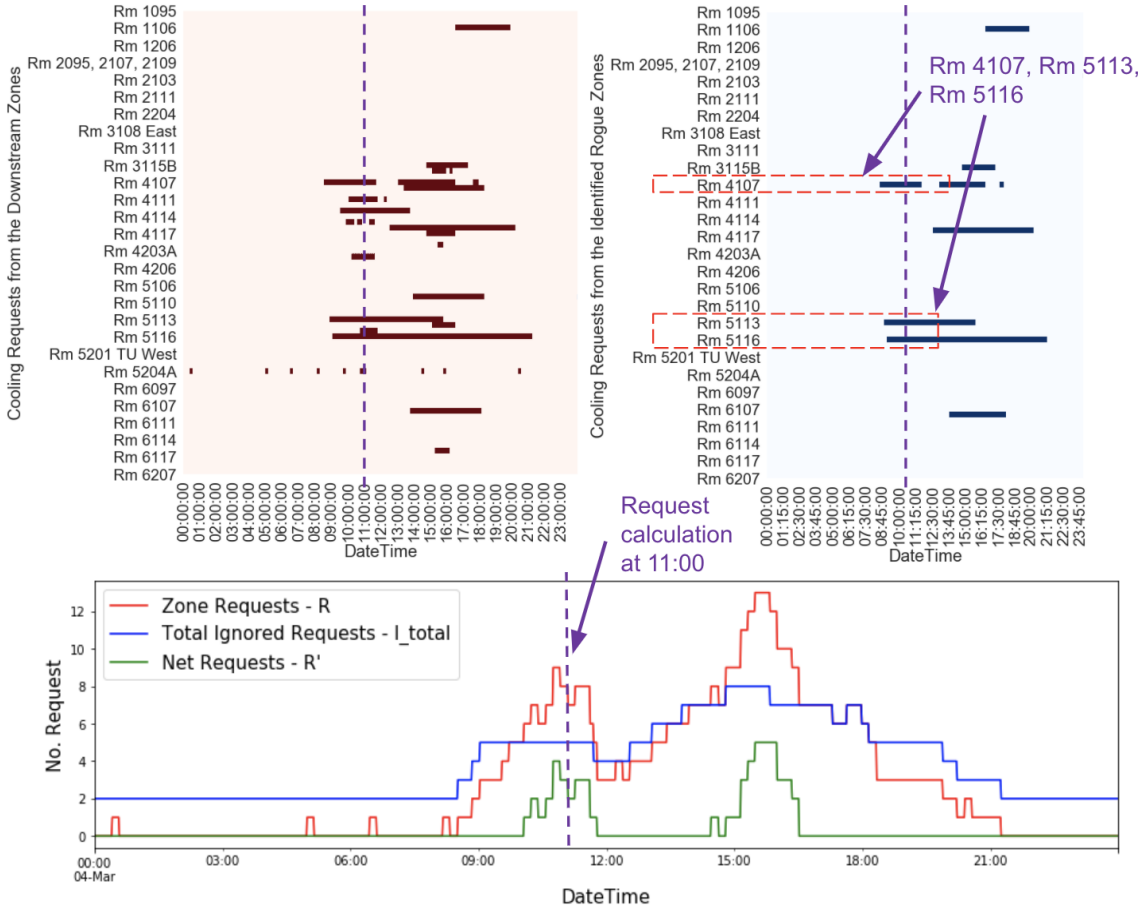


Figure 6. Zone requests per zone (upper left), the ignored requests per zone (upper right), the sum of the requests, the total of ignored requests and the net requests for all the zones of AHU01 on 4 March 2020. The vertical line marks 11 a.m. on all the plots.

3.3.2. Test Results of "Improve AHU SAT Setpoint Reset" Algorithm

The auto-correction algorithm "Improve AHU SAT setpoint reset" successfully changed the SAT setpoint of AHU01 and AHU02 in the BAS. As shown in Figure 7, the SAT setpoint changes followed the routine described in Section 3.2. The supply fan was on for the whole time, since this AHU serves laboratory areas. When R' was 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 R' remained at zero and the routine slowly increased the SAT setpoint by 0.12 °C every five minutes until it reached SAT_{max} (18.3 °C). The SAT setpoint remained at SAT_{max} until R' was larger than zero at 14:50 p.m. Then, the SAT setpoint again slowly decreased when R' was larger than zero and slowly increased when R' was zero.

Because both the original and corrected logic used feedback loops, a direct comparison of the two was not possible without modeling the dynamic behavior of the system or collecting enough data to perform a system-level evaluation. However, since the original controller was still active (for backup purposes) we could qualitatively compare the time at which each algorithm would start reducing the SAT setpoint in the morning. Figure 8 shows this comparison for two consecutive days in AHU01. The red line represents the corrected setpoint that was calculated by the algorithm and the blue line was the actual temperature that tracked the setpoint. The green line depicts the original logic. As highlighted by the text, the original logic would try to reduce the temperature much earlier than the corrective algorithm. This behavior was consistent across the testing period for both AHUs. For AHU01, during 14 days out of the 34 days, the old logic started earlier, while during the other 20 days the system did not require cooling. Conversely, for AHU02, during 13 days the old logic

started earlier, during one day it started later, and the other days it did not require cooling (the test was conducted during a mild spring). Overall, the preliminary test was successful. It showed the uninterrupted operation of the two algorithms in two AHUs for more than a month. The SAT tracked the new setpoint throughout the whole testing period. The new control sequence did not cause any occupant complaints, and it worked more efficiently than the previous one, although a precise savings estimate was beyond the scope of the test.

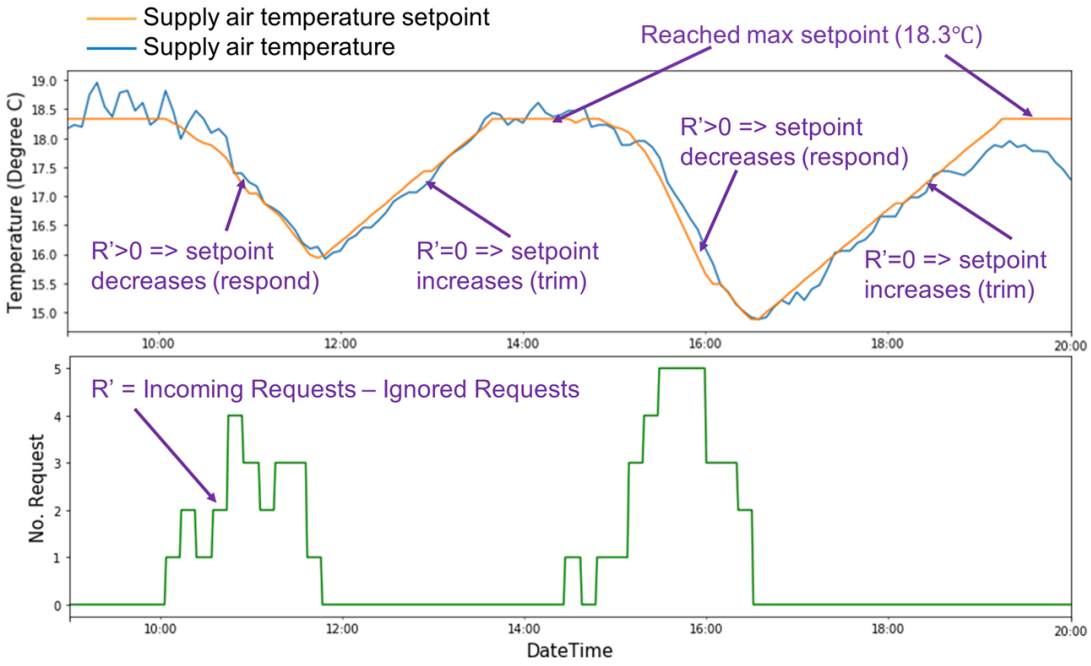


Figure 7. The SAT setpoint of AHU01 after the execution of the auto-correction algorithm (4 March 2020).

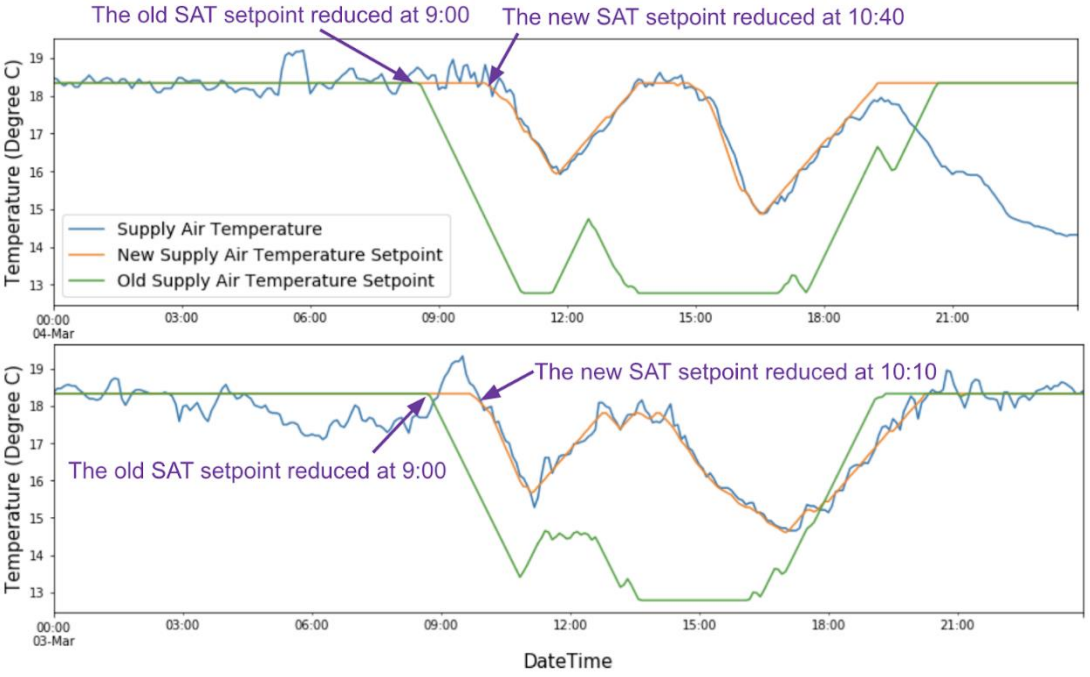


Figure 8. Comparison of the new and the old setpoint control strategies in AHU01 during 3–4 March 2020.

4. Discussion: Implementation Challenges and Solutions

The commercial partners faced a number of challenges when implementing the auto-correction algorithms into the FDD tools. We organized these issues under four areas: (1) developing a secure two-way communication between the FDD tool and the BAS; (2) incorporating operator approval; (3) managing the customizations necessary to the specific BAS/site installation; and (4) managing the potential conflict between the auto-correction and the BAS control actions. This section describes these challenges, as well as the solutions that the partners came up with to mitigate them.

The differences in the solutions described below also stemmed from different FDD software architectures, as well as different BAS network setups across the implementation partners. The FDD products developed by the two partners were based on software platforms that ran from the cloud with centralized fault libraries and analytics engines. These companies used additional hardware and software installed on site to collect data and send it to the cloud. A third implementation partner was the distributor of a third-party software and developed custom algorithms for various customers. This software ran on the local BAS network and had direct access to it, allowing access to external users via a virtual private network connection.

4.1. Develop a Secure Two-Way Communication Between the FDD and the BAS

Opening a two-way communication between the FDD system and the BAS was a challenge seen across all project partners implementing fault auto-correction into their FDD product environment. FDD tools typically read operational data from the BAS, run analytics and flag faults on the software interface. Often, they do not have capabilities to write commands directly onto the BAS. As indicated in Figure 1, the FDD tool commonly collects operational data using three pathways: (1) from the BAS server database, (2) from a central BAS server via API and (3) directly via the BACnet IP network. The first pathway prevents the FDD tool from writing back to the control system, therefore it cannot be used to implement auto-correction procedures. The second one requires BAS-specific interfaces; thus, implementers tend to avoid it. For this reason, when expanding the one-way interface to two-way communication, all the partners selected the third pathway to write back directly via the BACnet IP network.

The project partners mitigated the two-way communication challenge by upgrading their FDD system infrastructure. Figure 9 illustrates the solutions for the two cloud-based FDD systems. The solid line shows the original infrastructure, and the red dashed line shows the upgrade. In the first cloud-based FDD case (Figure 9a), the BACnet stack (a software library allows users to add a native BACnet interface to talk to the devices or applications in the BACnet network) of the FDD data acquisition device was updated to include a "write" function. The local data acquisition device was also updated to make API requests to the cloud FDD platform to retrieve the auto-correction command information. This enabled the FDD system to send the auto-corrective command to the local device and then to the writable properties used to control the BAS. In a cloud-based FDD system of another partner (Figure 9b), the current BACnet library already had writing capabilities. To enable secure communication with the cloud, the system architecture was changed. The standard data acquisition device was paired with a new field device (an auto-correction execution device) specifically designed to execute the new routines and log the interaction with the BAS. The cloud FDD engine initiated the auto-corrective command onto the auto-correction execution device. The device then executed the commands onto the BAS BACnet network and reported back the results to the cloud FDD engine. The BAS data were still acquired by the existing FDD data acquisition field device and delivered to the cloud FDD engine. The third on-premise FDD system was already capable of writing commands via BACnet. It only needed to change a setting in the BAS to authorize the changes.

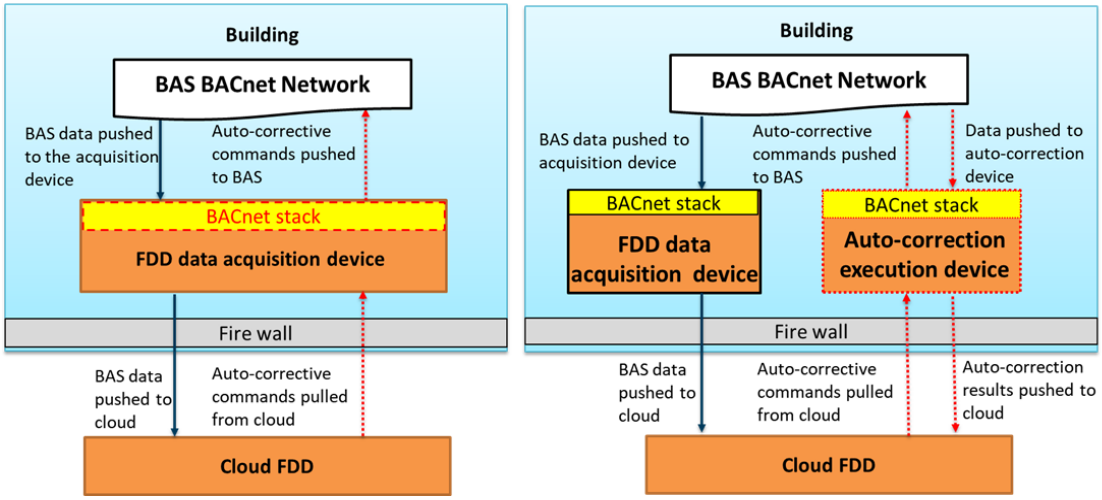


Figure 9. Two-way communication infrastructure for (**a**) the cloud-based FDD system 1, and (**b**) the cloud-based FDD system 2.

4.2. Incorporate Operator Approval

The second challenge faced by the partners was incorporating operator approval. The new auto-correction feature affords the FDD technology a certain degree of control capability. The building operators may be hesitant to trust this new capability and feel a lack of control. To mitigate this challenge, one project partner updated the existing interface to make sure the users were allowed to actively start, interrupt and track the auto-correction activities. Auto-correction enable and disable functionality was added to the user interface (UI), and the name of the control variables, their current value, and the new proposed values were provided to increase the operators' awareness. All the user and system activities of auto-correction are stored in a history log that is available to the user. Figure 10 shows a simplified mockup of the new UI displaying the auto-correction enable/disable functionality, action history and other details. Another partner also developed new interfaces for auto-correction authentication and acknowledgement.

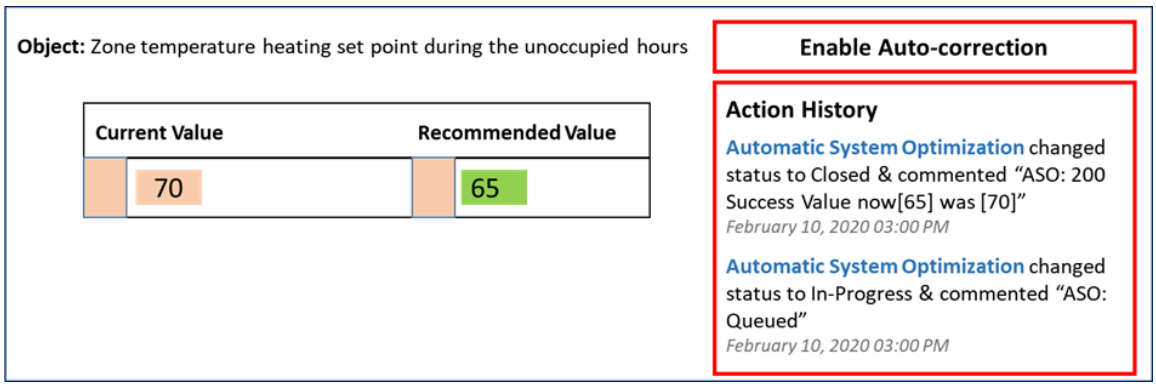


Figure 10. Mockup of the new user interface (UI) developed by one partner, displaying the auto-correction enable/disable functionality, the action history and other details (ASO – Automatic System Optimization).

4.3. Manage BAS and Site-Specific Customizations

The traditional separation of the roles between the FDD and the BAS allowed the FDD tools to develop general algorithms that were independent from some of the details about BAS and the implementation of specific control programs. For instance, an algorithm that detects opportunities

to save energy by shortening the AHU schedules did not need to know how these schedules were implemented in the BAS, it just needs to analyze the data produced by them. However, auto-correcting the same schedule meant overriding the operation of the BAS, therefore the developers must know many more details about the specific implementation of the control logic to avoid unintended consequences. The third challenge confronted was the lack of standardization in the BAS control logic, variables and interfaces. The implementation partners reported issues in (1) deciphering the BAS control sequences and identifying the exact control variables to override, (2) gaining access to these variables and (3) gathering data with frequency and timeliness appropriate to the application. An example of the first issue is the implementation of the "override manual control" algorithm described in Section 2.2. Depending upon the BAS, the override can be accomplished via an "override" variable Manual_override (whose value is 1—equipment is in manual control, 0—equipment is in automatic control) or by the setting of the priority level of the BACnet points (e.g., 8—manual operator override, 16—default automatically operation) [27].

Accessing the proper control variable was another part of the challenge. The auto-correction algorithms may require the FDD tools to be able to access the control variables that are not commonly exposed to the outside by the BAS. An example is the PID-tuning parameters required by the "control hunting" algorithm. One implementation partner reported being unable to retrieve these points via BACnet for a site, since the BAS vendor used a proprietary solution. ˉThe third issue emerged when a partner implemented the algorithms described in Section 3. These routines need real-time data updated every few minutes, since the algorithms are reevaluated continuously, while the existing BAS was storing it at 15-minute intervals and transmitting it to the FDD tool once a day (to save memory and bandwidth).

To address the challenges, all the partners had to spend significant time to understand and modify the BAS programming and setup, in addition to its interface with the FDD tool. The parameters of the BAS controller, gateway or server were changed to expose the necessary variables, making sure they could be modified when needed. Sampling frequency and data transfer rate were increased to implement some of the algorithms. A partner created a virtual point in the actual codes to accommodate the settings of override in various BASs. The virtual point is a string semicolon delimited list of point IDs that are mapped to any points that need to be changed from override in the BAS. A partner reported that particular care had to be put into matching appropriate data types (e.g., binary, analog with different precisions, arrays) used by the BACnet protocol, to avoid communication errors. All these customizations varied by BAS vendor, hardware vintage and site configuration.

4.4. Manage Control Conflicts between the BAS and the FDD Tool

The last challenge reported pertained to the conflicts between the BAS and the FDD control actions. Algorithms that make one-time changes to the BAS operation (e.g., the incorrectly programmed schedule in Section 2.1) may be overridden by operators or the BAS logic at a later date. It is unclear whether or not the auto-correction procedure should periodically update these variables. Moreover, algorithms that continuously change variables may also conflict with the existing BAS sequence of operation. An example is improving the AHU static pressure setpoint reset (Section 2.8) on a BAS that already has a reset strategy. There is a need to understand which one takes precedence and if the existing control sequences should be turned off.

To address the first issue, one implementation partner used an existing feature of the FDD platform to separately track the active schedule and the most efficient schedule and let the operator decide which one to activate. In addition, it logged all the changes to that schedule to offer more information to the user. For the second issue, another partner set up a fallback mechanism in the BAS, for use with the new FDD auto-correction algorithm that continuously modified the control setpoint. A watchdog was added in the BAS programming to make sure the FDD tool was online. If the FDD tool went offline, the BAS reverted back to the setpoint generated by the original control logic in the case of loss of communication with the FDD tool.

4.5. Other Considerations

The development and deployment of these algorithms stimulated an interesting discussion among the partners and advisors of the project about the role of the FDD and the BAS. Typically, the commercial FDD tools are developed as a software layer on top of the existing BAS. There exists a natural separation of roles in this arrangement, in which the BAS actively controls the building and the FDD tool observes its operation and provides insights and recommendations to the building manager. However, some consider FDD tools as a new generation of BAS that can take over some of its functionalities when it is necessary. At the beginning of the project, one facility manager expressed the desire to implement Guideline 36 sequences [24] on a building being controlled by an obsolete control system. To implement the sequences on that system, significant hardware upgrades and BAS programming labor would be required. The code cannot be easily reused between controllers due to the limitations of the control language, therefore this operation was not scalable and its implementation on multiple systems was hampered. As an alternative, he suggested to host the sequences in the significantly more modern FDD tool (algorithms 2.5, 2.8 and 2.9) and use the existing BAS as a simple tool to collect data and provide direct control over the lower-level hardware. This strategy was eventually implemented and was described in Section 3. Other partners disagreed, objecting that extensive real-time control of the building is outside the scope of FDD tools. Business models of the companies developing the tools may play a role in the way these new functions will be eventually incorporated in the FDD products at the end of this project.

5. Conclusions and Future Work

This paper presented nine algorithms for HVAC systems that were designed to automatically correct faults or improve operations relative to incorrectly programmed schedules, overriding manual control, sensor bias, control hunting, rogue zones and less aggressive setpoints or setpoints setback. It also showed preliminary tests confirming the efficacy of a subset of these algorithms, as tested in a large commercial building. Finally, it discussed challenges faced during the integration of these auto-correction algorithms into three commercial FDD tools and the solutions to these challenges that were adopted by the project partners. The main challenges included: (1) developing a secure two-way communication between the FDD tool and the BAS; (2) incorporating operator approval; (3) managing the customizations necessary to the specific BAS/site installation; and (4) managing the potential conflict between the auto-correction and the BAS control actions. The suggested solutions will help future auto-correction developers address similar challenges.

With respect to automated fault auto-correction, future work will focus on more field testing of the FDD integrated correction algorithms in a cohort of existing buildings. 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 cohort and the characterization of challenges and best practices. A second area of future work will entail the design and execution of a techno-economic analysis to quantify the broader market opportunity to inform ongoing commercialization efforts.

The state of today's FDD technology can be advanced through research focused on enhanced diagnostic (as opposed to detection) approaches and methods for fault prioritization. Complementary work to characterize fault prevalence based on empirical data from the field could also prove valuable in guiding future FDD technology development and implementation efforts. There is also an overarching need to navigate issues related to data management, integration, cybersecurity, and interoperability.

Author Contributions: Conceptualization, J.G., Formal analysis, G.L., M.P. and Y.C.; Methodology, G.L., M.P., Y.C. and J.G.; Writing—original draft, G.L., M.P., Y.C. and J.G.; Writing Review & Editing, G.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded 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.

Acknowledgments: 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 for his guidance and support of the research. We also thank the fault detection and diagnostics technology and service providers who participated in this study.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Energy Information Administration. *Monthly Energy Review April 2019*; EIA: Washington, DC, USA, 2019.
2. Energy Information Administration. *Commercial Buildings Energy Consumption Survey (CBECS)*; EIA: Washington, DC, USA, 2016.
3. Mills, E. Building commissioning: A golden opportunity for reducing energy costs and greenhouse gas emissions in the United States. *Energy Effic.* **2011**, *4*, 145–173. [CrossRef]
4. Katipamula, S.; Brambley, M.R. Methods for fault detection, diagnostics, and prognostics for building systems—A review, part I. *Hvac R Res.* **2005**, *11*, 3–25. [CrossRef]
5. Roth, K.W.; Westphalen, D.; Feng, M.Y.; Llana, P.; Quartararo, L. *Energy Impact of Commercial Building Controls and Performance Diagnostics: Market Characterization, Energy Impact of Building Faults and Energy Savings Potential*; US Department of Energy: Washington, DC, USA, 2005.
6. Fernandez, N.E.; Katipamula, S.; Wang, W.; Xie, Y.; Zhao, M.; Corbin, C.D. *Impacts of Commercial Building Controls on Energy Savings and Peak Load Reduction*; Pacific Northwest National Lab: Richland, WA, USA, 2017.
7. Deshmukh, S.; Glicksman, L.; Norford, L. Case study results: Fault detection in air-handling units in buildings. *Adv. Build. Energy Res.* **2018**, 1–17. [CrossRef]
8. Fernandes, S.; Granderson, J.; Singla, R.; Touzani, S. Corporate delivery of a global smart buildings program. *Energy Eng.* **2018**, *115*, 7–25. [CrossRef]
9. Wall, J.; Ying, G. *Evaluation of Next-Generation Automated Fault Detection & Diagnostics (FDD) Tools for Commercial Building Energy Efficiency—Final Report Part. I: FDD Case Studies in Australia, RP1026*; Low Carbon Living CRC: Boca Raton, FL, USA, 2018.
10. Lin, G.; Kramer, H.; Granderson, J. Building fault detection and diagnostics: Achieved savings, and methods to evaluate algorithm performance. *Build. Environ.* **2020**, *168*, 106505. [CrossRef]
11. ASHRAE. *Guideline 13–2015—Specifying Building Automation Systems*; ASHRAE: Akron, OH, USA, 2015.
12. Granderson, J.; Singla, R.; Mayhorn, E.; Ehrlich, P.; Vrabie, D.; Frank, S. *Characterization and Survey of Automated Fault Detection and Diagnostics Tools*; Report Number LBNL-2001075; Lawrence Berkeley National Laboratory: Washington, DC, USA, 2017.
13. Kim, K.; Rao, P.; Burnworth, J.A. Self-tuning of the PID controller for a digital excitation control system. *IEEE Trans. Ind. Appl.* **2010**, *46*, 1518–1524.
14. Shi, Z.; O'Brien, W. Development and implementation of automated fault detection and diagnostics for building systems: A review. *Autom. Constr.* **2019**, *104*, 215–229. [CrossRef]
15. Fernandez, N.; Brambley, M.; Katipamula, S. *Self-Correcting HVAC Controls: Algorithms for Sensors and Dampers in Air-Handling Units, PNNL-19104*; Pacific Northwest National Laboratory: Richland, WA, USA, 2009.
16. Fernandez, N.; Brambley, M.; Katipamula, S.; Cho, H.; Goddard, J.; Dinh, L. *Self Correcting HVAC Controls Project Final Report PNNL-19074*; Pacific Northwest National Laboratory: Richland, WA, USA, 2009.
17. Brambley, M.; Fernandez, N.; Wang, W.; Cort, K.A.; Cho, H.; Ngo, H.; Goddard, J.K. *Fial 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, 2011; Volume 20.
18. Isermann, R. *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*; Springer Science & Business Media: New York, NY, USA, 2006.
19. Zhang, Y.; Jiang, J. Bibliographical review on reconfigurable fault-tolerant control systems. *Annu. Rev. Control* **2008**, *32*, 229–252. [CrossRef]
20. Padilla, M.; Choinière, D.; Candanedo, J.A. A model-based strategy for self-correction of sensor faults in variable air volume air handling units. *Sci. Technol. Built Environ.* **2015**, *21*, 1018–1032. [CrossRef]
21. Wang, S.; Chen, Y. Fault-tolerant control for outdoor ventilation air flow rate in buildings based on neural network. *Build. Environ.* **2002**, *37*, 691–704. [CrossRef]
22. Hao, X.; Zhang, G.; Chen, Y. Fault-tolerant control and data recovery in HVAC monitoring system. *Energy Build.* **2005**, *37*, 175–180. [CrossRef]

23. Bengea, S.C.; Li, P.; Sarkar, S.; Vichik, S.; Adetola, V.; Kang, K.; Lovett, T.; Leonardi, F.; Kelman, A.D. Fault-tolerant optimal control of a building HVAC system. *Sci. Technol. Built Environ.* **2015**, *21*, 734–751. [CrossRef]
24. ASHRAE. *Guideline 36–2018. High. Performance Sequences of Operation for HVAC Systems*; ASHRAE: Akron, OH, USA, 2018.
25. ASHRAE. *ASHRAE/IES Standard 90.1–2016. Energy Standard for Buildings Except Low-Rise Residential Buildings*; ASHRAE: Akron, OH, USA, 2016.
26. Lin, G.; Pritoni, M.; Chen, Y.; Granderson, J. Can We Fix It Automatically? Development of Fault Auto-Correction Algorithms for HVAC and Lighting Systems. *ACEEE* **2020**, in press.
27. BACnet®Primer. What is BACnet? Phoenix Controls: Acton, MA, USA. 2009. Available online: https://www.phoenixcontrols.com/CatalogDocuments/Products/Network%20Integration/BACnet%20Primer%20(MKT-0233).pdf (accessed on 19 May 2020).