Title
Design of a maintenance and operations recommender

Authors
Federspiel, C.
Villafana, L.

Publication Date
2003-06-30

Peer reviewed

Reprinted by permission from *ASHRAE Transactions* 2003, Volume 109, Part 2.

Reprinted by permission from ASHRAE Transactions, Vol. 109, Part 2, pp. 677-683.

KC-03-8-4

Design of a Maintenance and Operations Recommender

Clifford C. Federspiel
Associate Member ASHRAE

Luis Villafana

ABSTRACT

We describe the design of a maintenance and operations recommender. The recommender uses information from computerized maintenance management systems (CMMS) and energy management and control systems (EMCS) to recommend what maintenance personnel should do in response to a maintenance service request or other event requiring a maintenance or control system action. The recommender integrates text information from a CMMS database and sensor information from an EMCS to provide recommendations. Text is processed using the Extended Boolean model, which is a simple text processing method commonly used to retrieve information from large databases. The recommender compares text problem descriptors and sensor data descriptors to estimate the similarity between previous maintenance actions and the maintenance action that will be taken. Actions with a high predicted similarity are more highly recommended than those with a low predicted similarity. The recommender uses reported maintenance actions to learn to improve its recommendations. It compares predicted similarity indexes with similarity indexes computed by comparing maintenance actions. The difference between predicted and computed similarity is used as an error signal to adjust weights that relate similarity indexes of individual problem descriptors. We use a simple example to demonstrate the steps that the recommender uses to make recommendations and to learn.

INTRODUCTION

Modern buildings use computerized maintenance management systems (CMMS) and computer-based energy management and control systems (EMCS). These systems contain large databases of information about historical building operations. Additionally EMCS systems contain real-time information about various subsystems important to building operations, including heating, ventilating, and air-conditioning (HVAC) systems, life-safety systems, lighting systems, and power systems.

Two common uses of CMMS systems and EMCS systems are monitoring and accounting. CMMS systems are used to monitor the frequency of maintenance activities and the time required to perform them. This capability is particularly useful when maintenance services are provided by third parties. EMCS systems are used to monitor energy-intensive equipment such as HVAC equipment. They monitor key system variables such as temperatures, flow rates, and pressures and also derived performance metrics such as chiller efficiency so that when alarms or problems are reported the maintenance personnel can look at these variables to diagnose the problems. There have been some efforts at integrating CMMS operations and EMCS operations. For example, some energy and maintenance systems will automatically initiate a work order in a CMMS in response to an alarm in an EMCS. Piette et al. (2002) describe GEMNet, which is an integrated energy and maintenance information technology infrastructure. GEMNet relies on standard database protocols and the building control system protocol called BACnet (ASHRAE 2001) for transferring information between CMMS and EMCS systems.

In this paper we propose integrating CMMS data with EMCS data for the purpose of recommending to building engineers what they should do in response to a service request or problem report from an occupant. Recommender systems are commonly used to retrieve useful documents from large databases and from the internet (Resnick and Varian 1997). In the

Clifford C. Federspiel is with the Center for the Built Environment and **Luis Villafana** is with the School of Information Management and Systems, both at University of California, Berkeley, Calif.

document retrieval application, the recommender system recommends documents that match a weighted query. The recommender system either uses feedback from the user or watches the user's search patterns to determine the best set of weights for the queries. Hayes and Pepper (1989) describe a system that provides maintenance recommendations based on a decision-tree approach and a knowledge of faults. The system provides sequential recommendations of tests that the maintenance technician should perform in order to diagnose a problem.

There are three features of building maintenance and operations that make the design of a maintenance and operations recommender somewhat unique. First is that the problem involves the integration of text with sensor data. Document retrieval recommenders deal entirely with text, while a maintenance recommender must be able to process values from sensors and integrate them with text data. Another feature of the maintenance and operations recommender is that the text data consist of short descriptions of problems or actions taken, whereas the text data in a document retrieval recommender consist of large numbers of large documents. The third feature of a maintenance and operations recommender is that feedback is readily available. In document retrieval recommender systems, it is sometimes necessary to force the user to rate the recommendations in order for the system to learn. Building engineers routinely report actions they take in response to reported problems. These reported actions form a feedback loop that can be exploited for learning so that it should not be necessary to ask the building engineers or the occupants who reported the problems to rate the recommendations.

METHODS

The maintenance recommender works by making comparisons, by predicting the outcome of comparisons, and by learning from observations. This section describes methods currently used by the recommender to perform these functions. Alternative methods are described here and in the "Discussion" section.

We assume that for every service request there is a set of *N* descriptors that could be sensor data, time, location, or a text description of a problem or action taken. The data could be represented by Table 1. In Table 1, the current problem is number 0. In Table 1 time runs in the opposite direction of the problem number, so problems reported further back in time have a larger problem number.

Text Processing

Some of the information in CMMS systems that will be used by the recommender is text, so we need a way to process text. Specifically, we need a way to compare text descriptions of problems and text descriptions of actions taken to solve problems.

We use concepts from the field of information retrieval (IR) that have been developed for comparing the relevance of text documents for making these text comparisons. Three common IR models used for assessing the relevance of documents are: (1) probabilistic models, (2) vector-based models, and (3) extended Boolean models. Probabilistic models and vector models are best suited to retrieving information from large collections of large documents. In a CMMS database, the "documents" consist of short text descriptions of problems or actions taken in response to problems. Collections in the CMMS context are the sets of problem descriptions and actions. Even long descriptions in a CMMS database are short by IR standards. We use the extended Boolean model because the documents are short, often containing just one instance of a key word. For details on IR models, see Baeza-Yates and Ribeiro-Neto (1999) or Manning and Schutze (1999).

TABLE 1
Tabular Description of the Data

Problem Number	Descriptor 1	Descriptor 2	Descriptor N	Action
0	$D_{0,1}$	$D_{0,2}$	$D_{0,N}$	A_0
1	$D_{1,1}$	$D_{1,2}$	$D_{1,N}$	A_1
M	$D_{M,1}$	$D_{M,2}$	$D_{M,N}$	A_M

For the classical Boolean model, matches to keywords are binary. Either the document matches a query or it doesn't. There is no way to handle partial matches or to determine the size of a mismatch. The extended Boolean model overcomes this partial matching limitation of the Boolean model by interpreting partial matches as Euclidean distances in a vector space of index terms. This allows the extended Boolean model to produce rankings of how closely the query matches the documents.

To compute the rankings, terms in the document are weighted, and the weights are combined to form a similarity index. There are a number of ways to compute term weights. The simplest is to choose term weights of either zero or one. We used term weights that are the product of the normalized term frequency and the inverse document frequency. Mathematically, the term weights are:

$$w_{x,j} = f_{x,j}\frac{idf_x}{max_i idf_i} \quad (1)$$

where x refers to a term in the document, j refers to a document, $f_{x,j}$ is the relative frequency of term x in document j, idf_x is the inverse document frequency, and $max_i idf_i$ is the maximum inverse document frequency for all of the terms in the collection. The terms are derived from the text in the current problem. The normalized term frequency is computed as follows:

$$f_{x,j} = \frac{v_{x,j}}{max_i v_{i,j}} \quad (2)$$

where $v_{x,j}$ is the number of times that term x appears in document j, and $max_i v_{i,j}$ is the maximum number of times that any of the terms in document j appear in document j. The inverse

document frequency is computed as follows:

$$idf_x = \log\left(\frac{N}{n_x}\right) \quad (3)$$

where N is the number of documents in the collection and n_x is the number of documents in the collection that contain term x. When the term weights are computed according to Equations 1 through 3, terms that appear often in the relevant document but infrequently in the collection will have a large weight.

Similarity indexes are computed as the norm of the vector of term weights. We use a 1-norm. Using this norm, the similarity index is the average of the term weights.

$$s_{0,j} = \frac{1}{M}\sum_{k=1}^{M} w_{k,j} \quad (4)$$

In addition to being computationally simpler to compute, the 1-norm has the feature that the similarity calculation for a disjunctive query is the same as the calculation for a conjuctive query (both are the average of the term weights). In other words, the 1-norm implies that the partial match for an OR query is the same as the partial match for an AND query.

Sensor Processing

Some of the data available to the recommender are from sensors. We use the following formula to compute the similarity of sensor readings:

$$s_{j,k} = 1 - min\left(max\left(\frac{|D_{0,k} - D_{j,k}|}{Q_{95} - Q_5}, 0\right), 1\right) \quad (5)$$

where $s_{j,k}$ is the similarity between the values $D_{0,k}$ and $D_{j,k}$ from a particular sensor, where j is the previous problem to which the current problem is being compared (i.e., the row of Table 1), and k is the descriptor index for the sensor (i.e., the column of Table 1). The magnitude of the difference is normalized by the 90% interquantile difference for the entire set of values corresponding to index k. We use a large interquantile difference instead of the range because the range of a sample from any distribution with infinitely long tails will grow with the number of samples even if there were no outliers. For example, the range of samples from a normal distribution will grow as more and more samples are acquired from the normal distribution because the tails of the normal distribution extend to infinity in both the positive and negative directions. The size of the interquantile difference could be other than 90%, but it should be as large as possible while eliminating outliers. The min( ) and max( ) functions are used to prevent similarities less than zero or greater than 1.

Equation 5 is used to compare the relevant system state of two reported problems. Since problems occur in different places and with different systems, it should be applied to values from the same kind of sensor, not just the same sensor itself. For example, Equation 5 would be used to compare the space temperature values for a hot complaint from two different spaces. This implies that sensors of the same kind exist in order for two problem descriptions to be compared. It would not be possible using this method to compare all of the sensors corresponding to a hot complaint from a space heated by a hydronic system with all of the sensors corresponding to a hot complaint from a space heated by a forced-air system. For example, the forced-air system might have an airflow sensor, while the hydronic system would not.

We assume that the database or the recommender has been configured so that the recommender knows whether or not a descriptor is sensor data or text. We also assume that the fields in the database corresponding to a particular kind of sensor data do not change. This could easily be accomplished by making a column in a table in the database always correspond to a kind of sensor (e.g., duct static pressure). The recommender does not figure out data types by itself.

Predicting Action Similarity

To make recommendations, the recommender predicts the similarity that will eventually be computed between the action for the current problem and past actions based on the computed similarity indexes for each descriptor. After a problem has been reported but before the problem has been solved, the similarity between the action taken for the current problem and the actions of past problems can't be used because the current action is yet to be taken. The recommender uses a linear combination of the descriptor similarity indexes to predict action similarity. Mathematically, this prediction is as follows:

$$\hat{S}_j = \sum_{k=1}^{N} w_k s_{j,k} \quad (6)$$

where j is the problem to which the current problem is being compared, and k is the index of the descriptor. Using matrix notation, Equation 6 can be expressed as follows:

$$\underline{\hat{S}} = \mathbf{s}\underline{w} \quad (7)$$

where the underbar denotes a vector quantity and the bold font denotes a matrix. For a database with just four problems (including the current problem) and two descriptors, the quantities in Equation 7 are as follows:

$$\underline{\hat{S}} = \begin{bmatrix} \hat{S}_1 \\ \hat{S}_2 \\ \hat{S}_3 \end{bmatrix}; \qquad \mathbf{s} = \begin{bmatrix} s_{1,1} & s_{1,2} \\ s_{2,1} & s_{2,2} \\ s_{3,1} & s_{3,2} \end{bmatrix}; \qquad \boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

The linear model makes computing the predicted action similarity easy and fast, and it also makes it easy to learn the best set of weights.

The action corresponding to the highest predicted action similarity is the most highly recommended action to take. The

concept is to display the actions to the maintenance engineer

TABLE 2
Descriptors and Actions for Four Example Problems

Problem No.	Temperature	Description	Action
0	78	**Hot** in **cashier's area**	**Lowered min SAT from** 65 to 60 **degrees**.
1	75.5	**Hot and stuffy**	**Lowered SAT.**
2	87.5	**Hot** in **secure room**	**Change duct dampers**. **Remove ceiling tiles**.
3	70.0	**Cold** in **employee**'s **lounge**	**Stat located over computer** & **monitor**. **Computer** & **monitor must** be **moved**.

in a list that is ranked by predicted action similarity from highest to lowest. The underlying assumption is that similar descriptions correspond to similar actions.

Learning

The recommender learns to improve its predictions by adjusting the descriptor weights in Equation 6 so that the difference between predicted and computed similarity of actions is as small as possible. This weight adjustment can be accomplished by solving the following constrained least squares problem:

$$\min_{w} \left\| S_j - \hat{S}_j \right\|^2 \tag{8}$$

$$\sum_{k=1}^{N} w_k = 1 \tag{9}$$

$$0 \le w_k \le 1 \tag{10}$$

S_j is the computed similarity between the action reported for the current problem (number 0 in Table 1) and the action reported for problem number j. It is computed by applying the extended Boolean model (Equations 1-4) to the Actions (column 5 of Table 1).

If there were no constraints, then this would be a standard least squares problem, and the optimal descriptor weights could be computed by a number of methods including recursive methods. Since the problem involves constraints, the optimal descriptor weights are computed using quadratic programming. The extra computation required to solve a quadratic programming problem is not an issue because the system doesn't have to learn immediately after a problem is solved. It would be acceptable for the quadratic programming solution to be computed overnight using the new information acquired from the previous day of maintenance activities.

If we used a nonlinear relationship between descriptor similarities and action similarity (e.g., a neural network) it would still be possible for the system to learn, but it might be computationally more difficult.

This learning strategy assumes that the action taken was the correct action to take most of the time, and that errors in the actions are random. Since the name of the person performing the maintenance is usually recorded in a CMMS database, it is possible for the learning procedure to only use actions taken by maintenance personnel who are known to be "experts." The learning strategy requires that actions be recorded. If they are not recorded, then the recommender cannot learn.

TABLE 3
Similarity Indexes for the Example

Problem No.	Temperature	Description	$\hat{S}_{init}$	$\hat{S}_{opt}$	S
1	0.857	0.069	0.463	0.126	0.2
2	0.457	0.069	0.263	0.097	0
3	0.543	0.0	0.271	0.039	0

RESULTS

In this section we show examples of how the methods of the previous section can be applied to real data. Consider the data for the four problems shown in Table 2. These four problems were taken from a database containing thousands of service requests. They were chosen for illustrative purposes. The first descriptor is space temperature, and the second descriptor is the text description of the problem. In this example, Problem Number 0 is the current problem. Key terms are derived from the text in the current problem (**hot**, **cashier**, and **area** for the Description column, and **lowered**, **min**, **SAT**, **from**, and **degree** for the Action column) and all similarity calculations are with respect to the current problem. Words used in the calculations are shown in bold font in Table 2. We did not include nondescriptive terms, such as pronouns, articles, and numbers, in the calculation (&, be, in, to, 60, 65).

For this example, the computed similarity indexes are shown in Table 3. The similarity indexes for the first two Descriptions (column 3 of Table 3) are low because the only match is the word "hot" and because "hot" occurs more frequently than other words, reducing its term weight. Column 4 (S (equal weights)) shows the predicted action similarity computed assuming that the weights in Equation 2 are equal (both equal to 0.5 for this example). Column 5 ($\hat{S}$ (after learning)) shows the predicted action similarity after choosing the optimal weights by solving the constrained least squares prob-

lem described above. The computed action similarities for Problems 2 and 3 are zero because none of the terms derived from the Action of Problem 0 appears in the Actions of Problems 2 or 3.

The vectors and matrices corresponding to Equation 3 are as follows:

$$\boldsymbol{s} = \begin{bmatrix} 0.857 & 0.069 \\ 0.457 & 0.069 \\ 0.543 & 0.0 \end{bmatrix}; \qquad \underline{w}_{init} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}; \qquad \hat{\underline{S}}_{init} = \begin{bmatrix} 0.463 \\ 0.263 \\ 0.271 \end{bmatrix};$$

$$\underline{w}_{opt} = \begin{bmatrix} 0.071 \\ 0.928 \end{bmatrix}; \qquad \hat{\underline{S}}_{opt} = \begin{bmatrix} 0.126 \\ 0.097 \\ 0.039 \end{bmatrix}$$

The vector $\underline{w}_{init}$ is the descriptor weight vector used prior to learning (i.e., prior to solving the constrained least squares problem), and $\underline{S}_{init}$ is the vector of predicted action similarities corresponding to $\underline{w}_{init}$. The vector $\underline{w}_{opt}$ is the optimal descriptor weights vector.

Prior to learning, the recommender would give the highest ranking to Action 1 and the lowest to Action 2 (i.e., it would recommend the action of Problem 1 most and the action of Problem 2 least). After learning, the recommender would still give the highest ranking to Action 1 but would now give the lowest ranking to Action 3, which was the action taken in response to a cold complaint. Less important to the act of recommending an action is the fact that the predicted action similarities prior to learning are too large. After learning they are approximately the right size.

DISCUSSION

We have described the design of a system for providing recommended actions for maintenance personnel responding to problems reported by building occupants. The recommender uses information stored in a CMMS database, integrates CMMS data with sensor data from EMCS systems, and learns to improve its performance with time.

Making recommendations is a seven-step process. The first step is to record the information describing the problem in a database. This information could include a text description provided by the occupant and other information such as time, location, and sensor data from a control system.

The second step is to compare the description of the current problem with the descriptions of all other problems previously recorded in the database. This is done by making within-field comparisons of like data types. Text descriptions are compared using the extended Boolean model. Sensor data are compared using the normalized absolute difference. These comparisons result in descriptor similarity indexes.

The third step is to combine the descriptor similarity indexes into a predicted action similarity index. The predicted action similarity is a forecast of the computed similarity index between the action that will be taken to solve the current problem and actions taken to solve previously recorded problems.

The fourth step is to sort the past actions by the predicted similarity. The primary purpose of computing the predicted action similarity is to rank past actions. Past actions with a high predicted similarity are more likely to be the right action for the current problem than past actions with a low predicted similarity, so they should appear at the top of the list.

The fifth step is to solve the problem and record the action taken in the database.

The sixth step is the first part of the learning process. After the action for the current problem has been recorded, the similarity indexes between the current action and all past actions are computed. These are the computed action similarity indexes. The purpose of computing these indexes is to provide a feedback signal for the learning process.

The seventh step is to adjust the descriptor weights by solving the constrained least squares problem described in the "Learning" subsection. The "error signal" for this learning process is the difference between the vector of computed similarity indexes and the vector of predicted similarity indexes.

There are a number of important issues regarding a system such as the one we have designed, including data quality, user interface design, acquisition of sensor data, automation of actions, and diagnosis versus recommendation. Regarding data quality, we have found that data quality in CMMS systems is low. It is common for data to be missing. Text descriptions of problems or of actions taken often do not contain much information. This results in few key words, sometimes just one to three words in total. The fact that common words are used in these descriptions (e.g., "hot") compounds the problem when applying IR methods to the text in CMMS systems. Short descriptions also amplify the need for a dictionary. If "warm" had been used instead of "hot" in problem 1 or 2, then the Description similarity index would have been 0 even though "hot" and "warm" are similar. The recommender needs to be able to determine that certain words are similar if free-text descriptions are to be used. Determining that some words are similar to others will require the use of a dictionary.

An alternative to using a dictionary is to eliminate or reduce the use of free-text descriptions. If the user interface used to enter the descriptions and actions into the CMMS system used a catalog, then keywords could be assigned to problem descriptions and actions by clicking on them. The person entering the problem description or action could step through a sequence of lists of key words designed to capture information about various aspects of problems (HVAC vs. lighting, recurring vs. persistent, etc.), which could be used to document the actions taken (investigate, changed, reported).

We assume that sensor data are readily and automatically available at the time a problem is reported. Today this is generally not the case. To get sensor data into the CMMS database today, most systems would require the maintenance engineer to go to the control system, manually record relevant sensor values, then manually enter them into the CMMS system. In the future this will be done automatically. In a companion

paper (Federspiel and Villafana 2003) we describe a web-based user interface that occupants use to report problems. They report the location of the problem using a code on the nearest thermostat. The user interface uses this code to query a database for the most recent space temperature sensor value from the problem location. In that particular case, the temperature sensor values are available in a database because the controls are BACnet compliant. A program that can communicate with BACnet-compliant devices polls all of the space temperature sensors (and other sensors) periodically and stores them in a circular buffer.

Some of the actions taken by maintenance engineers could be automated. Examples of such actions are "raised setpoint" or "started pump." If the recommender is sufficiently accurate, then it might be possible to have the recommender automatically initiate these kinds of actions when the predicted action similarity is high. However, it is difficult to extract actionable commands from free text. We think that it will be necessary to use a catalog for describing actions if some of the actions are to eventually be taken automatically.

The recommender system uses feedback from recorded actions to learn. This does not require that the actions taken always be correct, but the recommender will learn faster if they are correct, and the reliability of its recommendations will be better if the actions are correct. One way to ensure that the recommender gets high-quality actions is to only make recommendations and learn from actions taken by experts. It is common for the name of the person performing maintenance to be recorded in a CMMS database. Ad hoc criteria such as years of experience, years working at this particular site, or a quality rating from a manager could be used to determine whether or not the recommender uses the actions taken by a particular maintenance engineer. Another mechanism for ensuring the quality of recorded actions would be to survey the person who reported the problem to see whether or not the problem was solved and how well it was solved. The results of the survey could be used to filter out incorrect actions and to place more weight on actions that result in high occupant satisfaction. However, care would have to be taken to ensure that factors that could result in low satisfaction, such as a slow response, do not reduce the occupants' assessments of whether or not the actions solved the problem.

We have designed a system that provides recommendations, not diagnoses. There are two good reasons for this. The first is that most of the information available in CMMS systems describes actions. It is much less common for maintenance engineers to describe the cause of problems. The second reason is that it may be possible to solve a problem without knowing the cause. It is certainly more important to solve problems than to know why problems occurred. In some cases the path to the root cause may be so long that deciding exactly what caused the problem is an arbitrary act of stopping somewhere along this path. However, if causes were routinely reported in a CMMS database, then the same procedure we have described for recommending actions could be used for ranking likely causes.

The recommender currently uses a linear model to relate the descriptor similarity indexes to the predicted similarity of actions. We could use a nonlinear model such as a neural network to relate these variables. Doing so would probably become beneficial as the number of descriptors, particularly the number of sensors readings, increases. For some problem types, the values from some sensors will be highly relevant, while the values from others will be irrelevant. We could design a neural network to operate on individual key words so that it learns which sensors are important for certain problem descriptions.

CONCLUSIONS

We have shown how CMMS data and EMCS data can be integrated to recommended actions to maintenance and operations personnel in response to a problem reported by occupants. The recommender has the following features:

- Integrates text information from a CMMS with sensor data from an EMCS.
- Learns to improve its recommendations without requiring anyone to rate past recommendations.

The first feature allows the recommender to use information from building occupants in a systematic way, effectively utilizing occupants as virtual sensors. The second feature allows the recommender to adapt to the building operations without the undesirable need to rely on explicit ratings of its performance.

ACKNOWLEDGMENTS

This work was supported by the California Energy Commission's Public Interest Energy Research program and the Assistant Secretary for Energy Efficiency and Renewable Energy of the U.S. Department of Energy under contract to Lawrence Berkeley National Laboratory. This material is based upon work supported by the National Science Foundation under Grant No. 0122599.

REFERENCES

ASHRAE. 2001. *ANSI/ASHRAE Standard 135-2001, A Data Communication Protocol for Building Automation and Control Networks*. Atlanta: American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.

Baeza-Yates, R., and B. Ribeiro-Neto. 1999. *Modern Information Retrieval*. Reading, Mass.: Addison-Wesley.

Federspiel, C.C., and L. Villafana. 2003. Design of an energy and maintenance system user interface for building occupants. *ASHRAE Transactions* 109(2).

Hayes, P., and J. Pepper. 1989. Towards an integrated maintenance advisor. *Hypertext '89 Proceedings*, 119-127.

Manning, C.D., and H. Schutze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, Mass.: MIT Press.

Piette, M.A., M. Levi, D. McBride, S. May, and S. Kinney. 2002. GEMNet status and accomplishments: gsa's energy and maintenance network. *Proceedings of the 2002 ACEEE Summer Study on Energy Efficiency in Buildings.*

Resnick, P., and H.R. Varian. 1997. Recommender systems. *Communications of the ACM* 40(3), 56-58.

DISCUSSION

Jim Coogan: The data say "no action" is the most common action taken. I'm trying to imagine how the software might present that advice to the maintenance department as a suggested response to a complaint. How would you do that?

Clifford Federspiel: The recommender could tell the maintenance engineer that it has determined that taking no action is likely to be the best action, and then provide reasons why no action occurs. The list of possible reasons could include HVAC system running at full capacity and conditions now acceptable to the occupant.