Improving Special Purpose Machine User-Interfaces by Machine-Learning Algorithms

Valentin Plenk

Institute of Information Systems
Hof University of Applied Sciences
95028 Hof, Germany
Email: valentin.plenk@iisys.de

Abstract—**This paper proposes to make complex production machines more user-friendly. Improved machines help the operator in case of an error message or a process event by displaying recommendations, such as "in the last 10 occurences of this event the operators performed the following keystrokes". The messages are generated from statistical data on former user-interaction and previous process-events. The data represents the knowledge of all the machine operators. The data is gathered by logging user-interaction and process-events during regular operation of the production machine. This approach allows to store the operators' expert knowledge in the production machine without human intervention.**

Keywords–machine-learning; human machine interfaces; special-purpose machines; production machines

I. Introduction

State of the art appliances (e.g., photocopiers) are equipped with user interfaces that help the operator fix problems (e.g., a paper jam). The implementation of the software driving the interface contains structured knowledge about error scenarios and step by step instructions on how to deal with them.

While this approach leads to very well usable appliances, its proliferation is hampered by the engineering effort required for the definition of the error scenarios. This effort is only economically reasonable, if it can be refinanced over a large number of appliances. In the context of production machines, particularily special purpose machines, where the usual lot size is in a range below 10 similar machines per annum [1], a different approach is needed.

To deal with that problem, the author proposes to use machine-learning algorithms to generate situation-specific user guidance information from former user interactions and previous process events. Figure 6 shows a screenshot of the prototype displaying recommendations for fixing an error.

Similar applications of machine learning algorithms are commonly used to enhance user interfaces in smartphones or other IT-systems [2], [3]. In the production-machine sector, however, the application of machine learning algorithms is apparently limited to applications dealing with pattern detection in process data or distingushing different datasets [4]–[7]. In these papers the authors use the output of the algorithms to detect errors or problems with production quality. This information is then presented to the production-machine operator, requesting him to deal with the situation. While this approach undoubtedly helps to make processes more stable, it also demands ever more expertise of the operators.

In Section II, we describe how we model the production-process to make machine-learning applicable and give an overview of the system structure and the test environment. In Section III, we detail how we generate the events for the model and present the mechanisms generating the recommendations. In Section IV we evaluate the recommendations generated by our algorithm. Section V briefly summarizes our findings so far and then gives an outlook on our work plan for the future.

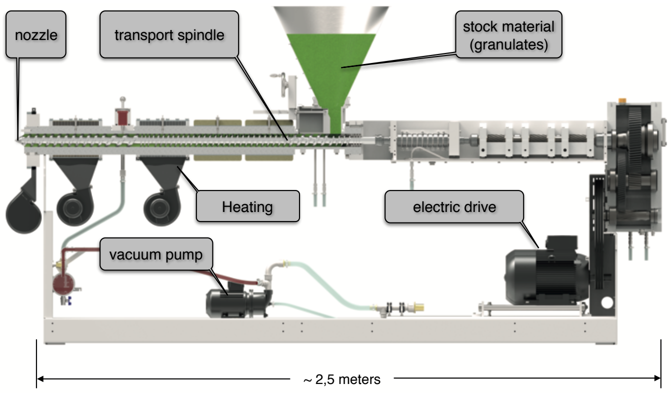


Figure 1. Working principle of an extruder
(courtesy: Hans Weber Maschinenfabrik, Kronach, Germany)

II. Proposed Approach and Test Environment

Machine-learning algorithms work on event sequences. Therefore we need to model the production process as a sequence of events. Such an event represents one out of a limited number of conditions and the time when the condition became true. Some events are generated by the process, i.e., error messages, and some events are generated by the operator, i.e., commands.

By logging these events over a significant amount of time, we can build a map of event sequences and their frequency.

Once the production machine encounters an error condition and prompts the operator for interaction, we use this map to find user interactions that were made in prior occurrences of the error condition and display a list of these interactions as recommended actions (Figure 6).

We develop the system on a simulated production machine. The machine in question is a plastics extruder (see Figure 1). The basic purpose of this machine is to transport and melt plastic granulates by means of a threaded spindle towards a nozzle. The main process parameters are the speed of the spindle, the temperature of the extruded material and the pressure of the material at the nozzle.

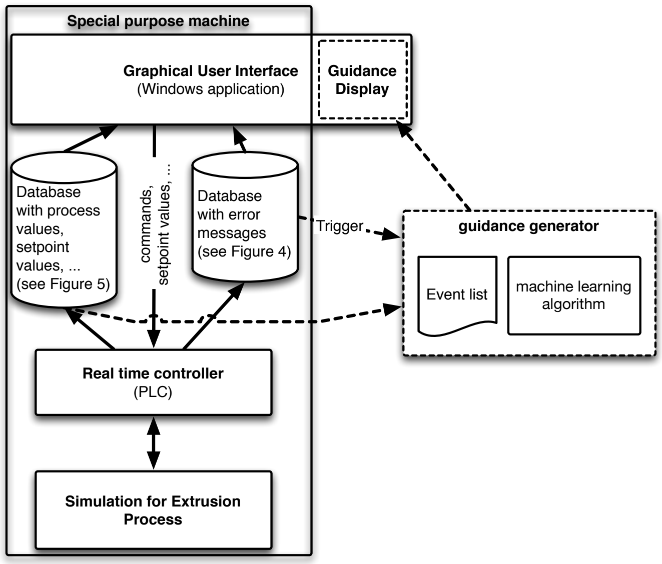

Special purpose machine
Graphical User Interface (Windows application)
Guidance Display
Database with process values, setpoint values, ... (see Figure 5)
commands, setpoint values, ...
Database with error messages (see Figure 4)
Trigger
guidance generator
Event list
machine learning algorithm
Real time controller (PLC)
Simulation for Extrusion Process

Figure 2. Structure of the system
(solid lines: standard control system; dotted lines: additions for recommendation mechanism)

Figure 2 shows the structure of the machine control system. The user interface is running on windows and communicates with the real time controller, a PLC, via two databases.

One database contains a log of all error-messages (Figure 4).

The other database contains a log of all PLC variables (Figure 5). Each row holds the values of all the variables at a given time. Each column corresponds to a PLC variable. There are variables representing process values, e.g., temperature values (`Loop1_PV`) or the pressure at the nozzle (`MP1_Value`), and variables representing parameters entered by the operator, e.g., temperature setpoints (`Loop1_SP`). The actual database has approximately 100 columns.

For our development, the actual control system is coupled to a simulation model of the extruder. An operator is running the simulated machine, thereby generating and fixing errors.

III. Generation of Events and User Guidance

Figure 2 shows that we added a guidance generator. The generator is a separate application running in parallel to the graphical user interface. This application is polling the database with the PLC variables shown in Figure 5 and processes it into an event list.

This processing is based on a description of the columns: which columns contain parameters and which columns contain process values.

For parameter columns the algorithm compares two consecutive rows and generates an event for each column that differs. These events represent operator interactions. They are coded as string values, e.g., `Loop2_SP_170`, containing the name of the PLC variable and the value of the new setpoint.

For process variables we use a different approach to represent the continuously variable values: First we calculate the deviation from the setpoint of the variable

$$Var_{rel} = \left| \frac{Var_{Process}}{Var_{Setpoint}} \cdot 100\% \right|$$

Then we sort Var_{rel} into one of the classes shown in Table I and generate an event every time the process value enters a new class. These events are also coded as string values, e.g., `Loop_1_2_PV_X`.

TABLE I. CLASSES FOR PROCESS VARIABLES

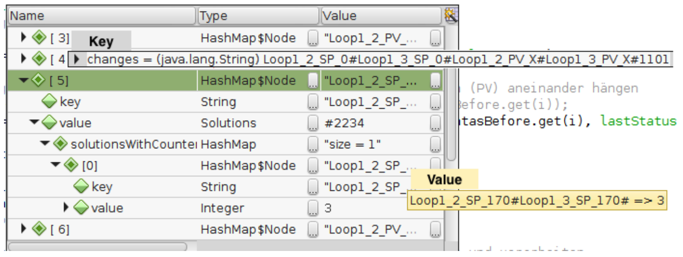
Var_{rel}	$\leq 1.77\%$	$\leq 3.16\%$	$\leq 5.62\%$	$\leq 10.0\%$	$\leq 17.78\%$
Class	A	B	C	D	E
Var_{rel}	$\leq 31.62\%$	$\leq 56.23\%$	$\leq 100.0\%$	$\leq 177.8\%$	$> 177.8\%$
Class	F	G	H	I	X

When the machine needs operator assistance, i.e., it has added a new row with an error message in the error message database (Figure 4), our guidance generation algorithm is triggered.

The algorithm then scans all past events in the event list to generate an event sequence map of both process variables and operator interactions that has led to the current error message. The length of the generated event sequences has to be limited. For now, we use all events that occurred in the five minutes before the error message.

The event sequences are stored in a content addressable memory, i.e., a Java map, where the events before the error message are used as the key, whereas the value represents a list of all operator interaction events between the time the error condition became true and the time the error condition became false again. For each of these sequences, the map also contains the frequency of the respective sequence in the past.

Figure 3 shows an abbreviated example for the map. The key `Loop1_2_SP_0# Loop1_3_SP_0# Loop_1_2_PV_X# Loop1_3_PV_X#1101` consists of a concatenated sequence of process variable, operator interaction and error events. The value `Loop1_2_SP_170#Loop1_3_SP_170# ==> 3` consists of a concatenation of operator interaction events and their frequency in the past.



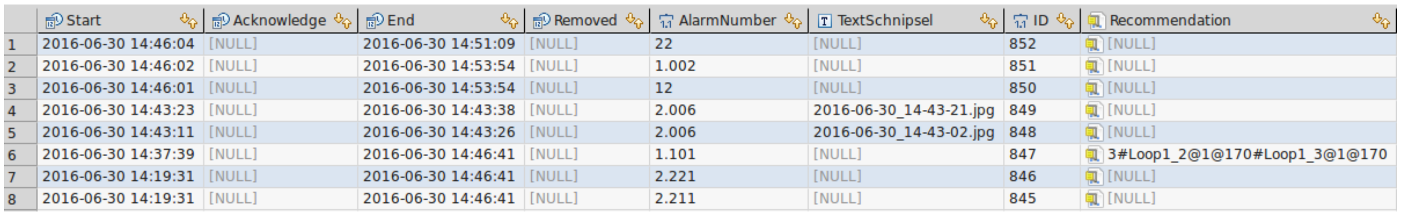
```
Name                        Type            Value
▸ [ 3]   Key                HashMap$Node    "Loop1_2_PV_...
▸ [ 4  ▸ changes = (java.lang.String) Loop1_2_SP_0#Loop1_3_SP_0#Loop1_2_PV_X#Loop1_3_PV_X#1101
▾ [ 5]                      HashMap$Node    "Loop1_2_SP_...
    key                     String          "Loop1_2_SP_...
  ▾ value                   Solutions       #2234
    ▾ solutionsWithCountei  HashMap         "size = 1"
      ▾ [0]                 HashMap$Node    "Loop1_2_SP_...
          key               String          "Loop1_2_SP_
        ▸ value             Integer         3
▸ [ 6]                      HashMap$Node    "Loop1_2_PV_...

Value
Loop1_2_SP_170#Loop1_3_SP_170# => 3

(PV) aneinander hängen
efore.get(i));
tasBefore.get(i), lastStatus
```

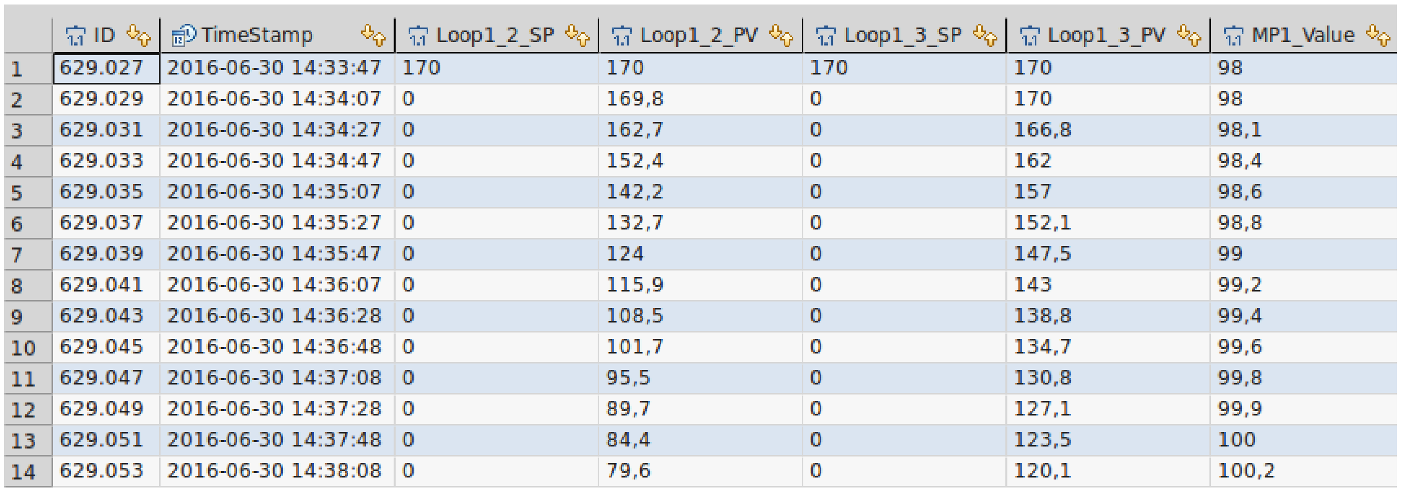
Figure 3. One entry in the event-sequence map of the guidance generation algorithm

The three most frequent interaction event sequences are stored in the error database. Once the operator requests guidance by selecting an error message in the error list on the right of the user interface, shown in Figure 6, the modified user interface of the machine reads this field, decodes the coded string and displays the guidance box shown on the left of Figure 6.



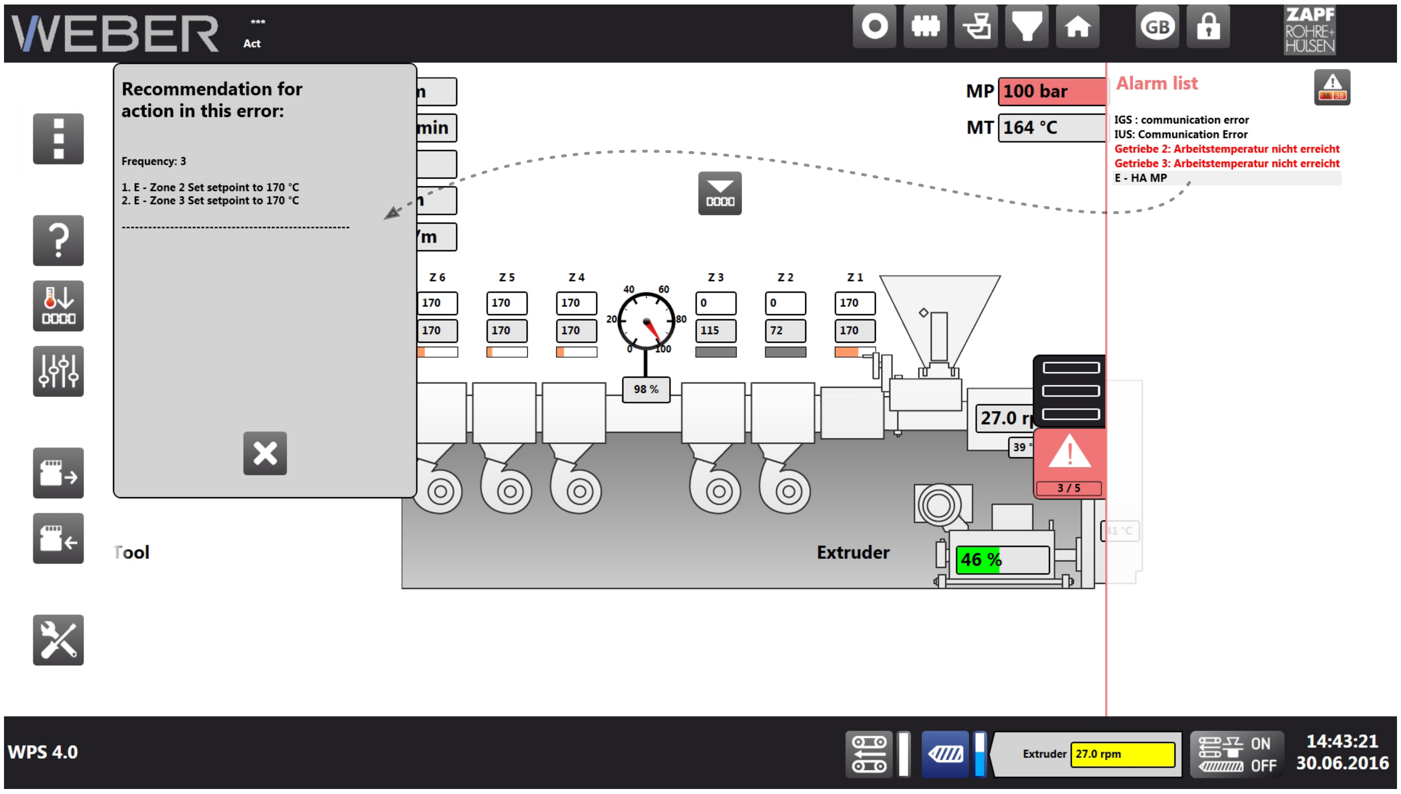
	Start	Acknowledge	End	Removed	AlarmNumber	TextSchnipsel	ID	Recommendation
1	2016-06-30 14:46:04	[NULL]	2016-06-30 14:51:09	[NULL]	22	[NULL]	852	[NULL]
2	2016-06-30 14:46:02	[NULL]	2016-06-30 14:53:54	[NULL]	1.002	[NULL]	851	[NULL]
3	2016-06-30 14:46:01	[NULL]	2016-06-30 14:53:54	[NULL]	12	[NULL]	850	[NULL]
4	2016-06-30 14:43:23	[NULL]	2016-06-30 14:43:38	[NULL]	2.006	2016-06-30_14-43-21.jpg	849	[NULL]
5	2016-06-30 14:43:11	[NULL]	2016-06-30 14:43:26	[NULL]	2.006	2016-06-30_14-43-02.jpg	848	[NULL]
6	2016-06-30 14:37:39	[NULL]	2016-06-30 14:46:41	[NULL]	1.101	[NULL]	847	3#Loop1_2@1@170#Loop1_3@1@170
7	2016-06-30 14:19:31	[NULL]	2016-06-30 14:46:41	[NULL]	2.221	[NULL]	846	[NULL]
8	2016-06-30 14:19:31	[NULL]	2016-06-30 14:46:41	[NULL]	2.211	[NULL]	845	[NULL]

Figure 4. Part of the error message database
(line 6 shows the error and the recommendation displayed in Figure 6)



	ID	TimeStamp	Loop1_2_SP	Loop1_2_PV	Loop1_3_SP	Loop1_3_PV	MP1_Value
1	629.027	2016-06-30 14:33:47	170	170	170	170	98
2	629.029	2016-06-30 14:34:07	0	169,8	0	170	98
3	629.031	2016-06-30 14:34:27	0	162,7	0	166,8	98,1
4	629.033	2016-06-30 14:34:47	0	152,4	0	162	98,4
5	629.035	2016-06-30 14:35:07	0	142,2	0	157	98,6
6	629.037	2016-06-30 14:35:27	0	132,7	0	152,1	98,8
7	629.039	2016-06-30 14:35:47	0	124	0	147,5	99
8	629.041	2016-06-30 14:36:07	0	115,9	0	143	99,2
9	629.043	2016-06-30 14:36:28	0	108,5	0	138,8	99,4
10	629.045	2016-06-30 14:36:48	0	101,7	0	134,7	99,6
11	629.047	2016-06-30 14:37:08	0	95,5	0	130,8	99,8
12	629.049	2016-06-30 14:37:28	0	89,7	0	127,1	99,9
13	629.051	2016-06-30 14:37:48	0	84,4	0	123,5	100
14	629.053	2016-06-30 14:38:08	0	79,6	0	120,1	100,2

Figure 5. Part of the PLC variable database
(the error shown in Figure 4 occurs in line 13)



Recommendation for action in this error:

Frequency: 3

1. E - Zone 2 Set setpoint to 170 °C
2. E - Zone 3 Set setpoint to 170 °C

Alarm list

IGS : communication error
IUS: Communication Error
Getriebe 2: Arbeitstemperatur nicht erreicht
Getriebe 3: Arbeitstemperatur nicht erreicht
E - HA MP

Figure 6. An example user guidance screen
(list of error messages on the right side; selection of last error message in list displays recommendation for action on the left side)

IV. Quality of recommendations

We simulated production and user interaction on the actual user interface and PLC connected to the simulation model. The resulting PLC variable database partly shown in Figure 5 now contains 713.677 rows with approximately 100 process values. The error message database partly shown in Figure 4 contains 1050 error messages. 163 of these messages were caused by the error event `1101`. During our operation of the simulated machine we cleared 140 of these error events by performing operations via the user interface. We cleared the remaining error events by resetting the simulation model.

In these 163 occurrences our algorithm identified 40 different event-sequences that led to the error event `1101`. Table III shows a part of that event-sequence map. The key is shown at the top of the row, sometimes spanning multiple lines. The number to the left of the lines following the key represents the number of cases (frequency) in which the user performed the operation sequence shown in that line (value). In many cases the operation-sequences in the map are identical, but tied to different event-sequences.

Table II shows the 9 different operation sequences from Table III. These operation sequences were derived solely from logged process data.

TABLE II. OPERATION SEQUENCES FROM THE EVENT SEQUENCE MAP SHOWN IN TABLE III

Operation sequence	Frequency
AOut1_1_27#	93
Loop1_2_SP_170# Loop1_3_SP_170#	38
Loop1_2_SP_170# Loop1_3_SP_170# Loop1_4_SP_170#	3
AOut1_1_27# AOut1_1_28# AOut1_1_27#	2
Loop1_2_SP_250# Loop1_3_SP_250#	2
AOut1_1_29#	1
AOut1_2_2# AOut1_2_3# AOut1_2_5# AOut1_2_7# AOut1_2_8# AOut1_2_9#	1
Loop1_2_SP_200# Loop1_3_SP_200# Loop1_2_SP_250# Loop1_3_SP_250#	1
Loop1_3_SP_170# Loop1_4_SP_170#	1

On that basis, we can provide the operator with different recommendations for clearing the error event `1101`. This is significantly better than tying a predefined help message to the error.

Table II also shows that some operation sequences were performed much more frequently than others. The first two sequences cover 93 % of all occurences of the error event.

V. Conclusion and future work

One important result of our work so far is a structure for interfacing a special-purpose machine with a guidance generation mechanism. The interface requirements are fairly easy to meet: (a) there must be a way to read all (relevant) PLC-variables, (b) for each variable, we need to know whether it represents a process variable or a process parameter, and (c) we need a way to trigger the guidance generation. This allows for a wide range of machines to be interfaced.

The way we convert changing process values and changed process parameters into discrete events allows us to apply machine-learning algorithms to the problem.

Our first, quite simple approach, loosely based on Markov-Chains, already shows a promising potential: We can generate recommendations solely from logged process data. These findings justify deeper research. We have therefore applied for funding to continue our work.

For the near future, we plan to collect real data from customers using the production machines in their companies. We plan to equip these production machines with a "black box" that contains the guidance generator and interfaces with the machines. The display of the hints can either be done on the machine user interface or on a separate user interface in the "black box".

Once we have collected a sufficient number of interactions, we will test a variety of machine learning algorithms. We see two issues with the application of the algorithms:

First: We use the event sequence before the trigger as a search criterion. Which / how many events should be included in the search key?

Second: Because of the slow response time of the process / the machine there can be delays in the order of magnitude of several minutes between a user interaction and a significant change in the process. This makes it possible that some user interaction events occur between the time an error condition has become true and false again (after user intervention). In that case, it will be difficult to distinguish between "helpful" and "intermediate / time filling" interactions. Which events should be included in the recommendation?

Current user feedback shows that it will also be necessary to structure the results of the learning algorithms in a way that allows a proficient user to specifiy "nonsensical" interactions and to remove them from the knowledge base. This requirement might be hard to meet, since statistics based learning algorithms do not provide a way to tie outputs to one of the many input sets.

As a long-term goal, we think about generating our own trigger events by changing the way we search the event database. By comparing the current event sequence to event sequences stored before an error event, we can recommend an interaction before the event occurs that triggered it in the past. By coupling that approach with a function in the "black box" allowing the operator to indicate why he interacted with the machine, the operator can mark event sequences that require intervention. On the basis of these marked sequences, we can then recommend interactions for problems that the original machine software cannot detect.

References

[1] V. Plenk, "A benchmark for embedded software processes used by special-purpose machine manufacturers," in Proceedings of the third international Conference on Software Engineering Advances. Los Alamonitos: CPS, 2008, pp. 166 – 171.

[2] H. J. C. et al., "Learning styles diagnosis based on user interface behaviors for the customization of learning interfaces in an intelligent tutoring system," in Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS 2006). Jhongli, Taiwan: Springer, June 26-30 2006, pp. 513 – 524.

[3] B. D. Davison and H. Hirsh, "Predicting sequences of user actions," in AAAI Technical Report WS-98-07, 1998, pp. 5 – 12.

[4] P. K. Wonga, J. Zhonga, Z. Yanga, and C. M. Vong, "Sparse bayesian extreme learning committee machine for engine simultaneous fault diagnosis," in Neurocomputing, 2016, pp. 331 – 343.

[5] K. Zidek, A. Hosovsky, and J. Dubjak, "Diagnostics of surface errors by embedded vision system and its classification by machine learning algorithms," in Key Engineering Materials, 2015, pp. 459 – 466.

[6] J. Luo, C.-M. Vong, and P.-K. Wong, "Sparse bayesian extreme learning machine for multi-classification," in IIEEE Transactions on Neural Networks and Learning Systems, 2014, pp. 836 – 843.

[7] A. Ziaja, I. Antoniadou, T. Barszcz, W. J. Staszewski, and K. Worden, "Fault detection in rolling element bearings using wavelet-based variance analysis and novelty detection," Journal of Vibration and Control, vol. 22, no. 2, February 2016, pp. 396–411.

[8] J. Lunze, Ereignisdiskrete Systeme. München: Oldenbourg, 2006.

TABLE III. PART OF THE EVENT-SEQUENCE MAP FOR ERROR `1101`
(To save space, we do not show all of the less frequent keys.)

Key	Freq.	Value
AOut1_1_27# MP1_99# AIn1_1_27# AIn1_2_46# AIn2_1_539# AOut1_1_28# MP1_103# AIn1_1_28# AIn1_2_47# AIn2_1_560# 1101		
	4	AOut1_1_27#
AIn1_2_46# MP1_100# MP1_101# 1101		
	3	Loop1_2_SP_170# Loop1_3_SP_170#
AOut1_1_28# MP1_101# AIn1_1_28# AIn1_2_46# AIn2_1_56# 1101		
	4	AOut1_1_27#
	1	Loop1_2_SP_250# Loop1_3_SP_250#
Loop1_2_SP_0# Loop1_3_SP_0# Loop1_2_PV_X# Loop1_3_PV_X# MP1_99# AIn1_2_46# MP1_100# 1101		
	23	Loop1_2_SP_170# Loop1_3_SP_170#
Loop1_2_SP_0# Loop1_3_SP_0# Loop1_2_PV_X# Loop1_3_PV_X# MP1_100# MP1_101# 1101		
	2	Loop1_2_SP_170# Loop1_3_SP_170#
AOut1_1_28# MP1_103# AIn1_1_28# AIn1_2_47# AIn2_1_560# 1101		
	66	AOut1_1_27#
	1	AOut1_1_29#
	1	AOut1_1_27# AOut1_1_28# AOut1_1_27#
MP1_100# AIn2_1_544# AOut1_1_28# MP1_103# AIn1_1_28# AIn1_2_47# AIn2_1_560# 1101		
	1	AOut1_1_27#
Loop1_2_SP_0# Loop1_2_PV_X# Loop1_3_SP_0# Loop1_3_PV_X# MP1_99# AIn1_2_46# MP1_100# 1101		
	2	Loop1_2_SP_170# Loop1_3_SP_170#
MP1_100# Loop1_2_PV_B# Loop1_2_PV_C# Loop1_3_PV_A# Loop1_2_SP_0# Loop1_3_SP_0# Loop1_2_PV_X# Loop1_3_PV_X# MP1_101# 1101		
	1	Loop1_2_SP_170# Loop1_3_SP_170#
Loop1_2_PV_A# MP1_100# Loop1_2_PV_B# Loop1_3_PV_A# Loop1_2_PV_C# Loop1_3_PV_B# Loop1_2_PV_D# Loop1_4_PV_A# Loop1_3_PV_C# Loop1_4_PV_B# Loop1_2_PV_E# Loop1_2_SP_0# Loop1_3_SP_0# Loop1_4_SP_0# Loop1_2_PV_X# Loop1_3_PV_X# Loop1_4_PV_X# MP1_101# 1101		
	1	Loop1_2_SP_170# Loop1_3_SP_170# Loop1_4_SP_170#
AIn2_1_555# AOut1_1_28# MP1_101# AIn1_1_28# AIn1_2_46# AIn2_1_560# AOut1_1_27# MP1_99# AIn1_1_27# AIn1_2_45# AIn2_1_539# MP1_98# AOut1_1_28# MP1_101# AIn1_1_28# AIn1_2_46# AIn2_1_560# 1101		
	1	AOut1_1_27#
Loop1_2_PV_A# MP1_100# Loop1_2_PV_B# Loop1_3_PV_A# Loop1_2_SP_0# Loop1_3_SP_0# Loop1_2_PV_X# Loop1_3_PV_X# MP1_101# 1101		
	1	Loop1_2_SP_170# Loop1_3_SP_170#
AIn1_4_97# AOut1_1_8# MP1_29# AIn1_1_8# AIn1_2_13# AIn2_1_16# AOut1_1_16# MP1_63# AIn1_1_16# AIn1_2_29# AIn2_1_32# AOut1_1_24# MP1_96# AIn1_1_24# AIn1_2_44# AIn2_1_49# AOut1_1_27# MP1_109# AIn1_1_27# AIn1_2_50# AIn2_1_53# 1101		
	1	AOut1_2_2# AOut1_2_3# AOut1_2_5# AOut1_2_7# AOut1_2_8# AOut1_2_9#
Loop1_2_PV_A# AIn2_1_555# AOut1_1_28# MP1_101# AIn1_1_28# AIn1_2_46# AIn2_1_560# 1101		
	1	AOut1_1_27# AOut1_1_28# AOut1_1_27#
AOut1_1_28# MP1_102# AIn1_1_28# AIn1_2_47# AIn2_1_560# MP1_103# 1101		
	4	AOut1_1_27#
AOut1_1_28# MP1_101# AIn1_1_28# AIn1_2_46# AIn2_1_560# 1101		
	1	Loop1_2_SP_250# Loop1_3_SP_250#
	1	Loop1_2_SP_200# Loop1_3_SP_200# Loop1_2_SP_250# Loop1_3_SP_250#
AOut1_1_28# MP1_103# AIn1_1_28# AIn1_2_47# AIn2_1_560# AOut1_1_27# MP1_99# AIn1_1_27# AIn1_2_46# AIn2_1_539# AOut1_1_28# MP1_103# AIn1_1_28# AIn1_2_47# AIn2_1_560# 1101		
	1	AOut1_1_27#
MP1_100# AIn2_1_541# AOut1_1_28# MP1_103# AIn1_1_28# AIn1_2_47# AIn2_1_560# 1101		
	2	AOut1_1_27#
AOut1_1_27# MP1_100# AIn1_1_27# AIn1_2_46# AIn2_1_539# MP1_99# AOut1_1_28# MP1_103# AIn1_1_28# AIn1_2_47# AIn2_1_560# 1101		
	2	AOut1_1_27#
MP1_100# Loop1_2_PV_B# Loop1_2_PV_C# Loop1_3_PV_A# Loop1_3_PV_B# Loop1_2_PV_D# Loop1_3_PV_C# Loop1_2_PV_E# Loop1_2_SP_0# Loop1_3_SP_0# Loop1_2_PV_X# Loop1_3_PV_X# MP1_101# 1101		
	1	Loop1_2_SP_170# Loop1_3_SP_170#
AOut1_1_28# MP1_102# AIn1_1_28# AIn1_2_47# AIn2_1_560# 1101		
	1	AOut1_1_27#
Loop1_2_SP_60# Loop1_3_SP_60# Loop1_2_PV_I# Loop1_3_PV_I# Loop1_2_PV_H# Loop1_3_PV_H# MP1_99# AIn1_2_46# Loop1_2_PV_G# Loop1_3_PV_G# MP1_100# 1101		
	1	Loop1_2_SP_170# Loop1_3_SP_170#