Analysis and Generation of Logical Signals for Discrete Events Behavioral Modeling

Rogério Campos-Rebelo, Anikó Costa, Luis Gomes

Analysis and Generation of Logical Signals for Discrete Events Behavioral Modeling

Rogério Campos-Rebelo[1,2], Anikó Costa[1,2], Luís Gomes[1,2]

[1] Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia, Portugal.
[2] UNINOVA – Centro de Tecnologias e Sistemas, Portugal.
{rcr, akc, lugo}@uninova.pt

Abstract. This paper presents a proposal for structuring logical signals for discrete events behavioral modeling. Graphical formalisms will be used to illustrate applicability of the proposed techniques. Input logical signals are generated based on the analysis of physical signals coming from the environment and other input logical signals. Their analysis is introduced in the flow of the input signal analysis, defined as pre-processing when using a modeling formalism. Their analysis is done in the first step of pre-processing, allowing to put these signals at the same level of the physical ones, enabling its use in the rest of the signals' analysis, and also allowing the generation of events and conditions associated with these signals. Likewise, logical signals are also defined in the flow of post-processing, allowing the definition of output logical signals through analysis and composition of the output signals generated by the model execution. The output logical signals are analyzed in the last step of post-processing in order to allow the use of the signals affected by the events in its analysis. The usage of these signals allows improvements in terms of expressiveness and compactness of the resulted models, both directly due to its use as well as by the increase of the possibilities of analysis provided to the following parts of the development flow.

Keywords: Signal Analysis, Modelling Formalisms, Embedded Systems.

1 Introduction

During the last decades, computer systems have experienced an increase in their complexity. The main reasons are related with the development of hardware that supports new functionalities, increasing memory and processing capacity, as well as software frameworks supporting their development. Becoming possible to create more complex systems, it allows the creation of more robust and enjoyable systems with more capabilities. Furthermore, their development becomes more complex and susceptible to failures, both in increase of quantity and variety. To handle these complexity problems, several approaches have been proposed in order to decompose the system development. The Model Based Development approach, as the Model Driven Architecture (MDA) defined by the Object Management Group (OMG) [1], is an example, which uses specific languages for system specification.

There is a wide variety of modeling languages. Among them, graphical languages are distinguished by providing an easier reading by the human user. Some examples

of these type of languages are Statecharts [2], state-machines variants [3], and Petri nets [4][5][6]. They allow a graphical view of the behavioral model, providing a visual representation of the system behavior easily understandable (and analyzed) by a human user. Creation of quite large models makes them confuse and difficult to interpret. Therefore, decomposition of modeling techniques had been proposed, allowing creating less complex models.

Some of these techniques allow transferring the signal analysis for pre- and post-processing, removing them from an explicit representation within the model.

The growing development of the Service Oriented Architecture (SOA) and the Event-Driven Architecture (EDA) [7] combined with the reduction of cost in sensor technology have led, in the last years, to a big development in the use of events in the analysis of signals and system' behavior [8]. Use of event modeling allows the development of more complex systems expressed through a much more compact model, supporting the development of much more user friendly interfaces.

One of the most well-known examples is the Complex Event Processing (CEP). They were first presented by David Luckham in his book The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems [9]. They are used in different areas, as an application based in RFID presented in [10]. Together with CEP was defined RAPIDE [11] [12], a computer language for definition and execution of models of system architectures with CEP.

This paper emphasizes introduction to the pre- and post-processing flows allowing putting the techniques of logical signals analysis within them. This addition allows the modeler to use these techniques to define new signals from which the analysis through events can take advantage.

2 Relationship to Cloud-based Engineering Systems

Cloud-based engineering service-oriented systems are systems that allow the user to configure the system through services that are available online. In these systems, the software and hardware as services are reconfigured. Event-Driven Systems are similar, where the events are provided like services.

The work proposed in this paper presents the addition of logical signals to the signals analysis flow, in order to improve the detection of behaviors. The use of this flow with events makes it suitable for creation of event-driven systems, allowing an easier analysis of the signals. The use of the events like services in the model makes this proposal interesting to create cloud-based systems. We argue that the use of model-based development strategies within cloud-based systems will benefit from adopting the proposed signal analysis techniques.

3 Related Work

The analysis of input and output signals on pre- and post-processing is a solution that has been developed to solve the problem of increased complexity of the models. For

this purpose, the entire analysis of signals from the environment is made outside the model, communicating with the model through the results of this analysis.

In terms of inputs, two forms are considered to analyze a signal:

Condition - the value of the signal is analyzed to verify that a certain equation holds. A condition checks the signal value at a specific point in time.

Event - the event analyzes the signal evolution in a time window (at least two points in time, associated with two execution steps) in order to verify if the signal evolution obeys to a specific defined behavior.

Previously, signal analysis was also presented in [13], considering not only the evolution of the signal values, but also variations of the signal at consecutive steps, as any difference level of variation of the signal (Δ). The first level of variation (Δ^1) is defined as a difference of the signal value in two consecutive instants. The second level of variation (Δ^2) is defined as a difference of Δ^1 in two consecutive instants. With these it is possible to define higher order of differences.

Table 1 - Events and Conditions Definition

Conditions		Closed Events		Open Events	
$>$	$\Delta^m X_n > k$	$\leq$ to $>$	$\Delta^m X^{n-1} < k \wedge \Delta^m X_n > k$	$<$ to $>$	$\Delta^m X^{n-1} < k \wedge \Delta^m X_n > k$
$\geq$	$\Delta^m X_n \geq k$	$<$ to $\geq$	$\Delta^m X_{n-1} < k \wedge \Delta^m X_n > k$	$>$ to $<$	$\Delta^m X_{n-1} < k \wedge \Delta^m X_n > k$
$<$	$\Delta^m X_n < k$	$>$ to $\leq$	$\Delta^m X^{n-1} < k \wedge \Delta^m X_n > k$	$<$ to $=$	$\Delta^m X^{n-1} < k \wedge \Delta^m X_n > k$
$\leq$	$\Delta^m X_n \leq k$	$\geq$ to $<$	$\Delta^m X_{n-1} < k \wedge \Delta^m X_n > k$	$>$ to $=$	$\Delta^m X_{n-1} < k \wedge \Delta^m X_n > k$
$=$	$\Delta^m X_n = k$	$\neq$ to $=$	$\Delta^m X^{n-1} < k \wedge \Delta^m X_n > k$	$=$ to $>$	$\Delta^m X^{n-1} < k \wedge \Delta^m X_n > k$
$\neq$	$\Delta^m X_n \neq k$	$=$ to $\neq$	$\Delta^m X_{n-1} < k \wedge \Delta^m X_n > k$	$=$ to $<$	$\Delta^m X_{n-1} < k \wedge \Delta^m X_n > k$

In Table 1, the definitions are presented to an m-level variation. Given this set of assumptions, it is possible to define several ways to analyze a signal. The work presented in [14] defines a set of elementary events, allowing the definition of a set of behaviors that the signal can have.

In Table 1, these types of events, defined as in [14] as open and closed events, are presented, including the conditions that allow their determination. The various conditions are also presented, which allows analysis of a given signal value.

In a similar way, output signals can be affected by two different methods. They can be updated by an action directly over the signal or by creation of an event associated to the output signal that will determine the signal's behavior:

Action - it is an assignment action of the signal, at a certain moment, with a specific value, as in Signal_ID = k.

Event - the event is associated with an output signal; when the event is active, generates a defined behavior with which the signal will be affected.

The output events are defined in [15]. In this work, the behavior associated with the event, with which the signal will be affected are defined taking into account three variables: Evolution Function, Window, and Final Value.

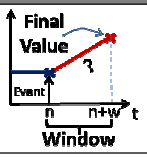
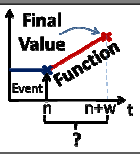
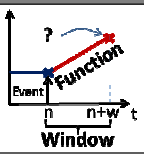
The **Evolution Function (f(n))** can be defined as any f(n) function. This function will guide the evolution of the signal, giving, for each execution step, the contribution value of the event to the associated signal. Generically the contribution of the event to the associated signal to each step n+p is equal to the value of f(p).

The **Window (w)** is an integer value that defines the number of steps following the occurrence of the event where the event contributes to the value of the associated output signal. For example, if the value of the window is 2, the event will contribute for the signal associated value in the execution steps n+1 and n+2.

The **Final Value (fv)** is the value that the associated signal should have at the end of the event contribution, which means at the end of the window.

To define the evolution behavior, it is necessary to define only two of these three variables (Evolution Function, Window, and Final Value). Therefore, three different types of output events, based on variables which are left free are defined.

Table 2. Output Events definitions.

	Timed Assignment	Guided Assignment	Evolution
Evolution Example			

In Table 2 the three types of output events are presented. For each one, a graphical representation of a generic evolution is presented.

The **Timed Assignment** event is based on the definition of the window value and the value for the signal in the last step of the window. In this case, knowing previously the final value of the signal, the function needs to do the approximation to this value in the exact number of steps defined in the window.

The **Guided Assignment** event, as an assignment event defines the value for the signal in the last step of the contribution. The evolution function is defined as well with the final value of the signal. In this case, the window is not a priori defined. The window value is obtained as the number of steps needed to reach the final value guided by the defined function.

In an **Evolution** event, unlike the assignment events, the final value of the associated signal is not a priori known. The signal value evolves depending on the defined function associated to the event during the defined time window. The final value will be obtained accordingly, from the computation of the evolution function during the time window.

4 Signal Analysis

Input and output events, such as the conditions and the actions presented in the previous chapter, are defined as a way to analyze the signals in order to find or generate behaviors in the chosen input or output signal, respectively. As result of such definition the analysis of the input signal and the assignment of the output signals are

transferred to pre- and post-processing, respectively. Consequently, the presented events are a result of the flow of pre- or post-processing.

This paper proposes to include, in the flow of pre- and post-processing, the analysis of logical signals, generated from the analysis of the physical signals and other logical signals.

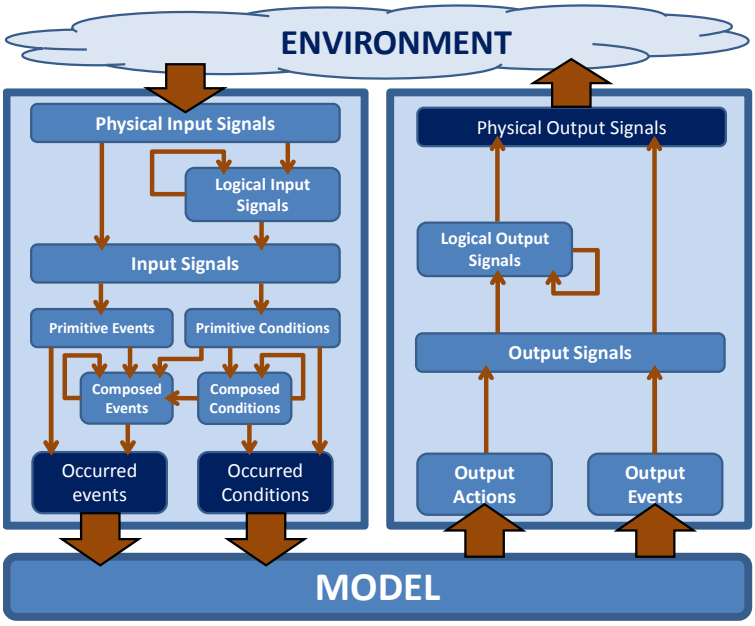


Fig. 1 – Pre- and post-processing flow

These logical signals, after defined and calculated their values, will become available for analysis through events and conditions, such as the physical signals. Although they are placed at the same level for the subsequent analysis, the value of the logical signals may represent quite different amounts.

With this addition the pre- and post-processing of the system, in terms of signals, is defined by the diagram in Fig. 1.

In the pre-processing part (Fig. 1-left) the values of the physical signals from the environment are received at each execution step. Then, based on their values, logical signals are defined and analyzed. They can be used in the analysis of other logical signals. After the analysis of the logical signals, these, together with the physical signals, are provided as a set of input signals for further analysis. In the next step the primitive conditions and events associated to the signals are calculated. Finally, based on primitive events and conditions, the composition of events and conditions are computed. Similarly as for logical signals, the analysis of some composed events and conditions can depend on the analysis of other events or conditions. The results of those analyses are introduced to the system model.

In the post-processing part (Fig. 1-right), the execution of the model generates events and actions associated to the output signals. These actions and events update the output signals by changing their values. Similarly to input signals, the analysis of a logical output signal can depend on the analysis of other logical output signals. After all the logic signals analyzed, the values of the signals generated by the events and actions in conjunction with logic signals are provided as outputs to the environment.

4.1 Physical Signals Analysis

Internal representation of a physical input signal is composed by the value received from the environment in the current time step and a series of values previously received during a time window. Associated with this window of values various levels of differences between values are also calculated. This list of values, as shown in Fig. 2, saves a set of consecutive values that the signal had in the past execution steps and the difference between them.

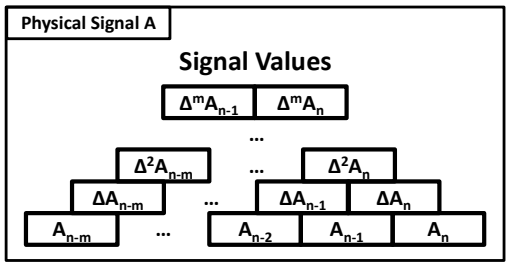


Fig. 2 – Physical Signal Composition

This window with the history of values of the signal allows the analyses of events associated with the signal, and analyze logical signals. To calculate the size of this window is necessary to analyze all events and logical signals associated with that signal.

The number of deltas (steps) to be analyzed is given by the maximum of the largest deltas analyzed in each event associated with the signal.

Regarding the number of values to store in terms of Delta zero (d0), the value depends on delta (Δ). The maximum value is given by the results in the following equation for each event associated with signal:

$$for\ \Delta^n,\ d0 = n + 1$$

For instance, for an event that analyzes the Δ^2 of the signal, the signal needs to hold 3 values (from A_n to A_{n-2})

In terms of logical signals is necessary to find the oldest value being analyzed in the logical signal to determine how many values to hold.

Exemplifying, if the maximum value analyzed in a logical signal is $\Delta^3 A_{n-3}$, it will be needed to analyze values until the third delta and in the delta zero from A_n to A_{n-6}. The value is obtained by the following analysis: $\Delta^3 A_{n-3} = \Delta^2 A_{n-4} - \Delta^2 A_{n-3}$; $\Delta^2 A_{n-4} = \Delta A_{n-5} - \Delta A_{n-4}$; $\Delta A_{n-5} = A_{n-6} - A_{n-5}$.

4.2 Logical Signals Analysis

A logical signal can represent different things, for example the conversions of the value of a physical signal in radians to degrees, the average value, or even complex values resulting from complex functions, like the FFT of a signal.

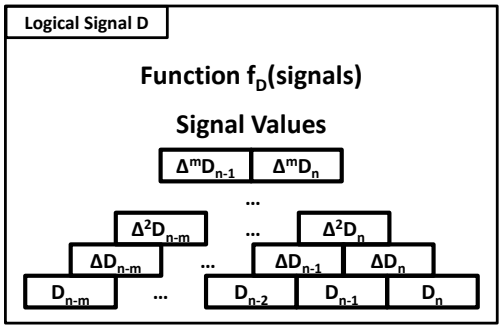


Fig. 3 – Logical Signal Composition

In terms of its composition, the logical signal inherits all the characteristics of the associated physical signal. It also has a function (or set of functions) associated (Fig. 3). The values of the logical signal are the results of this function.

In the case of a logical signal, the values from *n* to *n-m* may not represent time steps. Although the representation is equivalent, it depends of the following processing steps to know the meaning of the values.

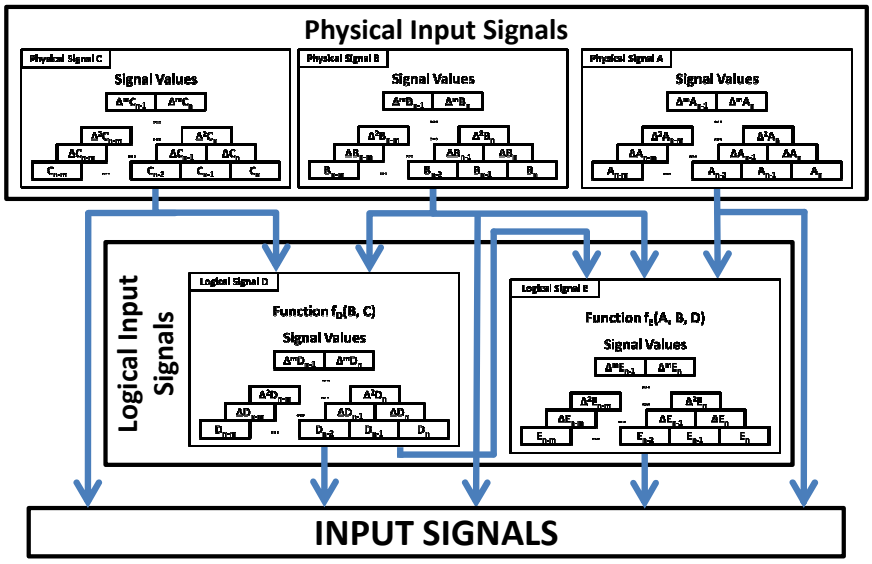


Fig. 4 - Analysis of Input Logical Signals.

Logical signals are present in both, pre- (Logical Input Signals) and post-processing (Logical Output Signals). Logical Input Signals are analyzed in the pre-processing step immediately following the acquisition of physical input signals. Their analysis depends on the values of the physical signals as well as other logical signals.

Fig. 4 presents the analysis process of the Logical Input Signals.

Once received and analyzed all physical input signals, the logical input signals that only depend on physical signals are computed. Next, all logical input signals also depending on recently computed logical signals are computed. This process is repeated until all the logical signals have been computed.

At the end of the analysis, physical and logical signals become available to the rest of the analysis at the same level. This means that for the continuation of the flow does not matter if these signals come from the environment or have been generated.

On the other hand, the output signal analysis flow is accomplished through post-processing, and the analysis of events and actions affecting the output signals is made in a similar way as for input signals.

After having analyzed all the events and executed all actions, a set of output signals is available. These signals are provided to the environment, but they are also used in the analysis of logical output signals.

The analysis of logical output signals is similar to the analysis of input logical signals. The flow is equivalent to the Fig. 4 where Input physical signals are replaced by Output Signals and Input Signals are replaced by physical output signals.

Thus, the analysis of the logical output signals is performed in the final steps of post-processing immediately before the last step where the output physical signals are available for the environment. The steps of analysis between the Physical Output Signals and Output Signals are similar to those presented previously between Physical Input Signals and Input Signals.

5 Discussion

This work intends to take advantage from techniques of generation of logical signals in order to improve the performance of the flow presented before, contributing to the compactness of the representation of dependencies on signals, where the analysis of the signals is moved to pre- and post-processing of the model.

The addition of logical signals is intended to get smaller and more readable models, removing complexity from the model to pre- and post-processing, in order to exploit the benefits of graphical modeling formalism and its readability by humans.

The definition of the logical signals in this flow, being analyzed through functions, allows a wide number of applications. It is possible to define functions that only convert a physical signal value into some unit of interest. For example, if there are two temperature sensors, one measuring in Celsius and other one in Fahrenheit, in order to allow the further analysis depending on the measured temperature by both sensors, it is necessary to create at least one logical signal that convert one of the temperature unit into the other one. For example, one can create a logical signal *Tc* (equivalent value in Celsius) associated to the physical signal *Tf* (temperature value in Fahrenheit). The function associated to the logical signal Tc is the following:

$$Tc = (Tf - 32) * (5/9)$$

This conversion allows in the next processing steps the analysis of the two signals under the same conditions.

Other example is the average value of the signal in the last 5 time steps *Ta,* or the difference between the present value and the value 5 steps before *Td* obtained by associate the following functions:

$$Ta = (T0 + T1 + T2 + T3 + T4) / 5$$
$$Td = T5 – T0$$

It is also possible to present values resulting from the analysis of more than one signal, obtaining values of other quantities. For example, consider a vehicle inside a factory, controlled from a docking station where it must return to be recharged. It is possible to calculate the distance to the station (Ds), in order to know if it is still inside the control area, using the values that vehicle traveled in latitude (Lt) and longitude (Lg) associating the following function to a logical signal:

$$Ds = sqrt(Lt^2 + Lg^2)$$

In a more complex way, it is possible to calculate values in a transformed space (not analyzed in the time domain), obtaining analysis such as spectral distribution when calculating the FFT of a given signal or even the average value of a set of signals.

6 Conclusions and Further Work

Moving the analysis of Logical Signals into the pre- and post-processing phases of the modeling flow provides a set of new capabilities for the analysis of signals to be used by the model.

This addition allows acquiring extra information from the signals coming from the environment, so allowing a more detailed analysis and defining more input conditions and output actions.

Moreover, this addition can also allow obtaining certain values without the need to add new hardware that would measure them, since this value could be determined by analysis of other signals. In this way, the number of variables available for use by events and conditions previously defined improves.

Comparing with models produced without these logical signal techniques, these additions will potentially contribute to a significant reduction of the models. In some cases they may even allow a huge compression of models as, for example, complex behaviors can be encapsulated in an event that depends on logical signals.

Acknowledgments. The first author was supported by a Grant from project PTDC/EEI-AUT/2641/2012, financed by Portuguese Agency" FCT - Fundação para a Ciência e a Tecnologia". This work was partially financed by Portuguese Agency"

FCT - Fundação para a Ciência e a Tecnologia" in the framework of projects PEst-OE/EEI/UI0066/2011 and PTDC/EEI-AUT/2641/2012.

References

1. Object management group [Internet]. Available from: http://www.omg.org/
2. Harel D. Statecharts: A visual formalism for complex systems. Sci Comput Program. Elsevier; 1987;8(3):231–74.
3. Börger E, Stärk RF. Abstract State Machines: A Method for High-level System Design and Analysis [Internet]. Springer Berlin Heidelberg; 2003. Available from: http://books.google.pt/books?id=Am43BAC06L8C
4. Jensen K. High-Level Petri Nets. In: Pagnoni A, Rozenberg G, editors. Appl Theory Petri Nets SE - 12 [Internet]. Springer Berlin Heidelberg; 1983. p. 166–80. Available from: http://dx.doi.org/10.1007/978-3-642-69028-0_12
5. Reisig W, Berlin H. Distributed Algorithms: Modeling and Analysis with Petri Nets W. Reisig Institut fur Informatik Humboldt-Universitat. 1998;38–43.
6. Zurawski R, MengChu Z. Petri nets and industrial applications: A tutorial. Ind Electron IEEE Trans. 1994;41(6):567–83.
7. Bruns R, Dunkel J. Event-Driven Architecture [Internet]. Springer London, Limited; 2010. Available from: http://books.google.pt/books?id=BtsfBAAAQBAJ
8. Eckert M, Bry F. Complex Event Processing (CEP). Informatik-Spektrum [Internet]. Springer-Verlag; 2009;32(2):163–7. http://dx.doi.org/10.1007/s00287-009-0329-6
9. Luckham D. The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. In: Bassiliades N, Governatori G, Paschke A, editors. Rule Represent Interchang Reason Web SE - 2 [Internet]. Springer Berlin Heidelberg; 2008. p. 3. Available from: http://dx.doi.org/10.1007/978-3-540-88808-6_2
10. Zang C, Fan Y. Complex event processing in enterprise information systems based on RFID. Enterp Inf Syst [Internet]. Taylor & Francis; 2007 Feb 1;1(1):3–23. Available from: http://dx.doi.org/10.1080/17517570601092127
11. Luckham DC. Rapide: A language and toolset for simulation of distributed systems by partial orderings of events [Internet]. Computer Systems Laboratory, Stanford University; 1996. Available from: http://books.google.pt/books?id=AWamHAAACAAJ
12. Luckham DC, Kenney JJ, Augustin LM, Vera J, Bryan D, Mann W. Specification and analysis of system architecture using Rapide. Softw Eng IEEE Trans. IEEE; 1995;21(4):336–54.
13. Campos-Rebelo R, Costa A, Gomes L. Enhanced Event Modeling for Human-System Interactions Using IOPT Petri Nets. Human-Computer Syst Interact Backgrounds Appl 3. Springer; 2014. p. 39–50.
14. Campos-Rebelo R, Costa A, Gomes L. Elementary Events for Modeling of Human-System Interactions with Petri Net Models. In: Camarinha-Matos L, Barrento N, Mendonça R, editors. Springer Berlin Heidelberg; 2014. p. 219–26. Available from: http://dx.doi.org/10.1007/978-3-642-54734-8_25
15. Campos-Rebelo R, Costa A, Gomes L. Output events for human-system interaction modeling. Hum. Syst. Interact. (HSI), 2014 7th Int. Conf. 2014. p. 261–6.