Improving e-Assessment Platform with Flexible Scoring Mechanism based on Event Log Analysis

Younes Djaghloul[1], Raynald Jadoul[1], Vincent Porro[2], Thibaud Latour[1], Patrick Plichart[1]

[1]Research Centre Henri Tudor - Luxembourg, [2]SOGETI - Luxembourg

Key words: *TAO, e-assessment, log analysis, pattern matching, complex scoring*

Abstract:

This paper describes the new scoring system and user interactions logging of the TAO e-assessment platform [1]. Its principles and high-end features will be demonstrated. In our approach we improve the TAO platform with two main components: a suitable and expressive language to describe the variables and the score engine that parses queries to calculate the variable values. Our approach manages the complex pattern matching in the case of problem solving assessment by adapting other approaches proposed in the field of Complex Event Processing.

1 Introduction

To score accurately and meaningfully is a fundamental dimension of the assessment process, but the definition of a score may vary while its computation may be very challenging. The definition of the scoring rule usually precedes the test-taking phase. It consists in specifying a finite set of variables whose values are to be computed and set at the very moment of the test by a direct comparison between a test taker's actual answer and an expected answer. However, within various contexts of the computer-based assessment, for example, in problem solving test units based on rich environment manipulation (e.g., laboratories, mail clients, or office-like applications), this simplistic scoring method conclusively fails to capture the essence of what has to be evaluated, i.e. the competency level of the test taker. In complex test units, instead of a simple answer, a test taker is rather expected to develop a strategy and the provided response results in a collection of actions generating events. This is in line with the definition of the "Problem Solving" component of the OECD PIAAC study [2].
Our approach improves the existing solutions on three main aspects: a flexible definition of score variables in pre and post-test; a clear and efficient mechanism to calculate score values based on collected events; the support of scoring based on the analysis and processing of complex patterns met in the sequences of collected.
The remaining of this paper is sectioned as follow: section 2 gives some related work in the domain of e-assessment and the Complex Event Processing. The third section presents an overview of the proposed scoring mechanism. In section 4 the main components of the proposed architecture are proposed. The paper is ended by a conclusion.

2 Related works

There is a shortage of experimental studies examining scoring through the use of events monitoring in computer based assessment. Despite the fact that new rich item types like

serious games or general immersive 3D simulations represent venue for conducting assessment research. Psychometric reliability challenge of such scoring, the lack of available open source solutions and scalability issues restrict this approach. Most of existing CBA solutions that are open source (LimeSurvey [3], HotPotatoes [4]) and commercial solutions (Perceptiontm Question Mark [5], FastTest Web [6], WebQuiz XP [7])provide classical scoring algorithms based on answers given to simple interactions medium like selection in a multiple choice question , input in a free text entry, dropdown list selection. There is no systematic recording of all possible events, but only a few pre-designed variables computation (time spent on items, number of visits). They also provide at results management level the possibility to build statistical information based on the proficiency of the subjects population. On the other hand, serious games like mōsbē (MOdding and Simulation By Everyone [8]), OLIVE (On-Line Interactive Virtual Environment) allowing to conduct a 3D replay or perform an after-action review [9] are much more advanced from the point of view of events tracking. Some of them provide frameworks or facilities to support designers to track identified behaviours and report subject proficiency.
Unfortunately serious games engines are very often rather expensive and still require strong IT skills to implement tracking rules.
Similar work also has been achieved in other fields as audit trail file analysis achieved in surveys systems and used to evaluate interview performance or to pre-test survey instrument and the Complex Event Processing. Blaise systems can produce general log file including keystroke, entering a field or leaving a field events. It is possible then to compute statistics from it but no supporting tools allowing the designing of specific behaviours using pattern of events to be tracked across the log file without IT skills [10].
The domain of Complex Event Processing gives a precious help in event management. CEP is well investigated by the scientific community and it aims to provide an efficient way to use events as inputted data with the respect of the characteristics of event models.
In [11] one presents an interesting glossary with more than 39 terms belong to this domain. The work of [12] is focused on exploitation of complex patterns on a sequence of events in database. The main contribution of this work is a syntactic extension of SQL named SQL-TS. This extension supports the creation of complex queries with sequence management. In CEDR project, a specific language is proposed that permits the detection of declared patterns in a stream of events [13]. Different kinds of sequence strategies can be declared in the language. The possible strategies are ‘SEQUENCE’ for exacts sequence, ‘ANY’ if any symbol in the pattern is matched and ‘ALL’ to verify the existence of all symbols in the pattern.
The SASE+ project proposes a rich and power full language to create expressive query on event traces [14]. It proposes four strategies of pattern sequence and allows also to express the negation in patterns. For the pattern matching task, the SASE+ engine is based on nondeterministic finite state machine.
Our approach is inspired from the techniques developed in the Complex Event Processing area in order to provide an efficient mechanism of scoring and complex pattern matching in the context of e-assessment.

3 Overview of the Proposed Logging and Scoring Mechanism

In our proposition, we aim to offer an efficient approach serving a flexible scoring mechanism that allows flexibility in the definition of score variables and the computation of the score. Figure 1 presents an overview of the scoring task in TAO platform.

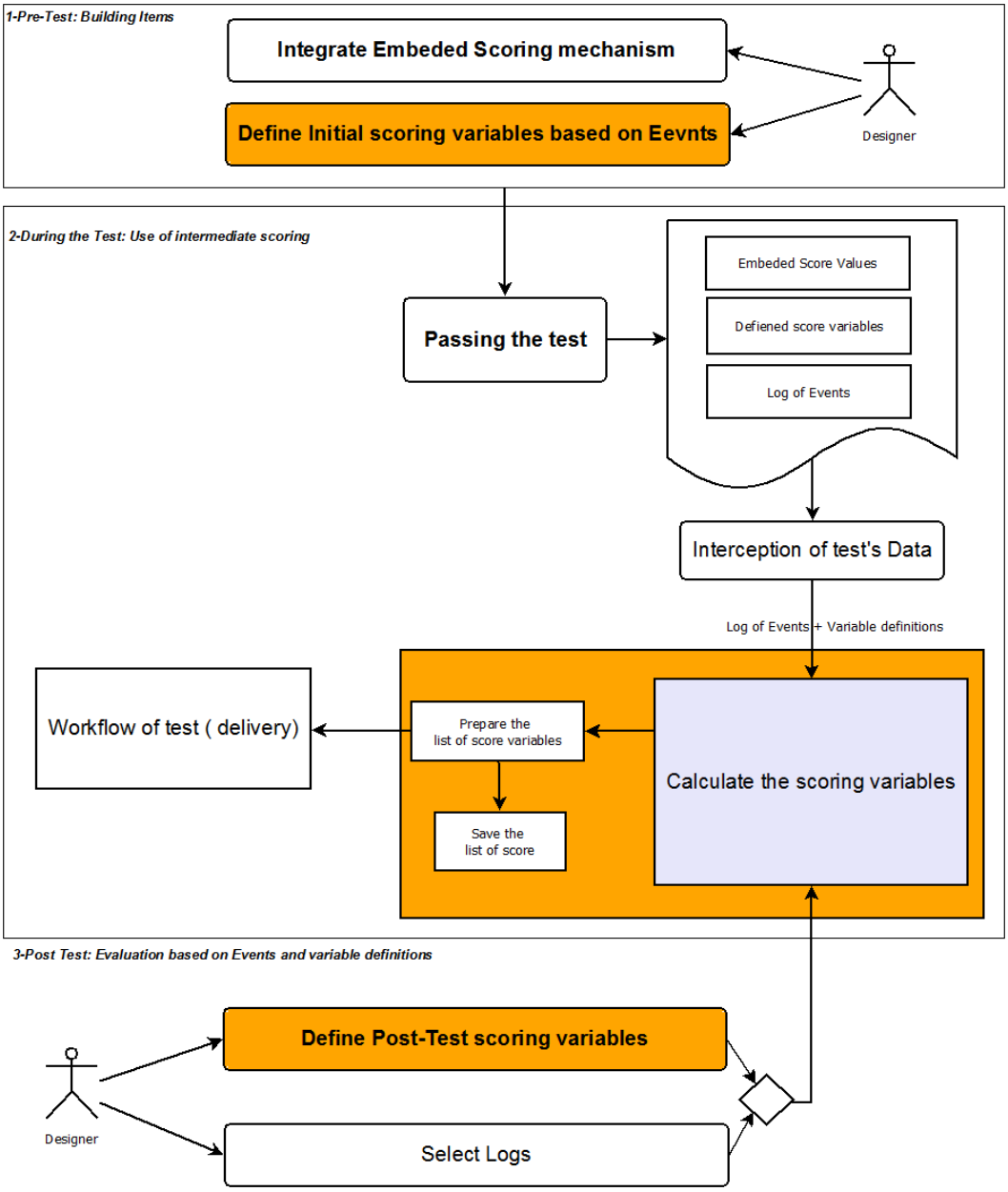

Figure 1. Scoring task in TAO

We can notice a high flexibility in the variable definition. Usually, the test designer defines the score variables in the item building phase according to the objectives of the test. However, in some cases one needs to define other variables after the item building phase. In TAO, the definition of variables can be done during the item building phase or after passing the test itself. This important feature is based on the capacity to use traces of the test maker as main input in the computation of the score. Without these traces, one cannot calculate these types of variables defined after the test was finished.
The second thing that we notice is the scoring computation is done at various stages of the assessment process.

- During the test itself (e.g., for adaptive assessment)
- Right after the test was finished (e.g., to provide grades to the test taker).
- A long time after one or many sessions of the test. This depends on a temporal requirement of the test. In the three cases, especially the last one, an adequate events log model is fundamental to achieve this task.

4 Components of the architecture

In order to accomplish the event based scoring, two main components are proposed: the query language to define the score variables, and the score engine to compute them.

4.1 The score engine and its toolbox of functions

The objective of the scoring engine is to set the values of the score variables; as input, it uses collected events and variables' definitions. The outputs are stored in a persistence layer. In our case, the persistence resides in a *Result Ontology* of the TAO platform. Figure 2 presents the main components of the *Scoring Engine*.

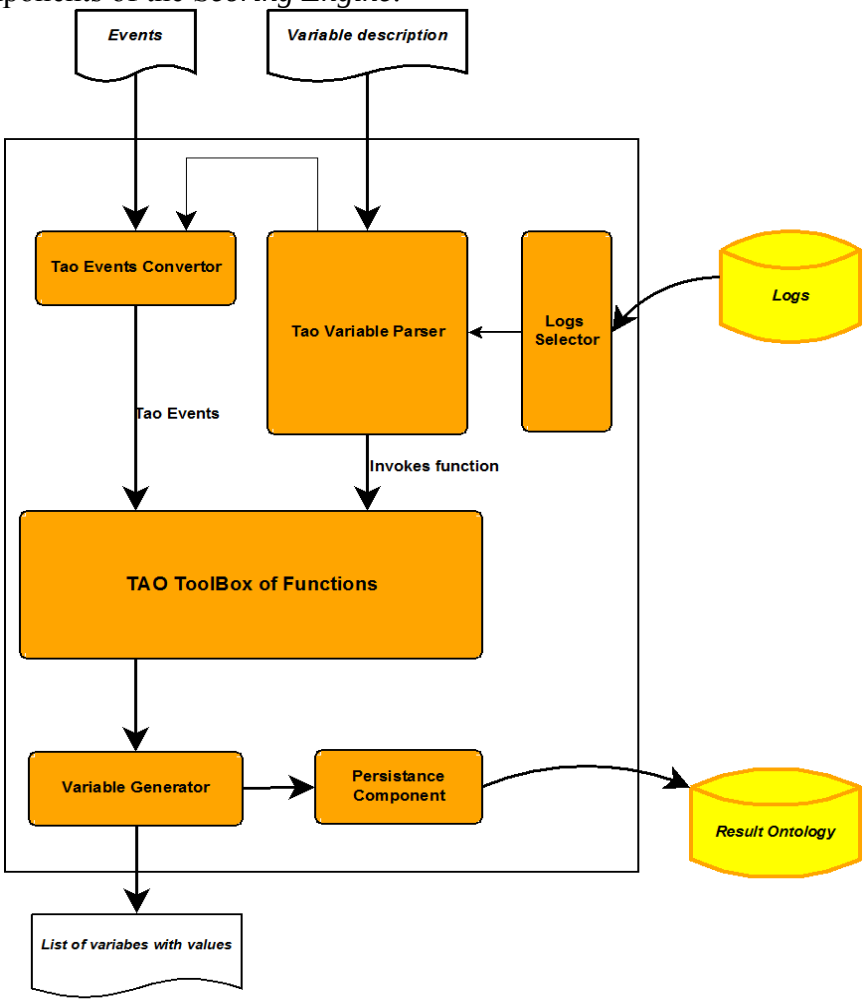

Figure 2. Architecture of the scoring engine

4.1.1 Tao Events Convertor

The *TEC* component converts the received events from their initial models to the *Tao Event Model*. This operation guarantees the unification of the trace model and allows a unique computation mechanism by the other components of the scoring engine.

4.1.2 Tao Variable Parser

This component parses the query formulated with the proposed language and invokes the appropriate score functions from the toolbox (i.e., a flexible and open set of functions dedicated to scoring). In some variable definitions, the source of event traces can be a log file

from a specific location. In this case, the *Logs Selector* component downloads this log and sends it to the *Parser*. This log will be converted into *TAO Event Model* by the *TEC* component.

4.1.3 Toolbox of functions

The Toolbox contains all functions needed in the assessment. One can enumerate three kinds of functions:

1. *Simple Matching and Scoring functions* calculate the score according to a direct comparison between the value of an event element and the correct response.
2. *Complex Pattern Matching and Scoring functions* provide a complex mechanism of matching for some kinds of assessment, especially in problem solving tests. The score depends on the sequence of actions that the test maker accomplished as a response. One can express a complex sequence pattern using several strategies and options inspired from the existing approaches in the field of *Complex Events Processing*, e.g., the mechanism of pattern matching and finite automata.
3. *External Scoring functions* are not implemented in the system. They are used just for calling external scoring functions implemented in other systems. This bridge is important to allow a collaboration scoring in an open assessment approach.

4.1.4 Variable Generator

The *Variable Generator* intercepts the variables' values and creates a specific list of variables that can be used in two manners:

- As output of the scoring engine;
- As input for the *Variable Persistence Component* in order to set the variables' values in the *Result Ontology*

4.2 The proposed language

Although predefined definitions of score variables are available at the creation of a test, one may define new variables to be calculated and set by the event-based scoring mechanism. In order to define these variables and to allow the computation of their values, a novel language is proposed. This language relies on a clear XML-based syntax to express complex queries whereas the score processing itself is supported by a rich toolbox of dedicated functions, e.g., countOfEventAction(), existEvent(), eventPatternMatching(), etc.
In the following we present a brief description of the main important elements of this language. Each query *Q* is a tuple of five parts:
Q (Variable_Description, Sources, Window, Pattern_Matching, Score_Value_If_Match)

1. *Variable_Description (VariableDesignation, VariableURI)*. The designation is used as label of the variable. *VariableURI* is used to establish a link with the variable in the *Result Ontology*. After the computation, the *Persistence Component* sets the appropriate ontology variable with the calculated value.
2. *Sources* indicate the sources of events. If the value of this element is *CurrentXML*, the engine uses the trace already sent by the caller. Otherwise, one puts *URLs* of logs to be downloaded by the *Logs Selector* Component.
3. *Window (FilterType, WindowBegin, WindowEnd)*. The *Window* plays the role of a filter on the imputed traces. The *FilterType* can be either '*Time'* for timestamp filtering or '*EventPostion'* to filter according to the position of events in the trace. The window interval is between *WindowBegin* and *WindowEnd*.

4. *PatternMatching (SymbolDeclarations, PatternOfSymbols, PatternStrategy)*. In this part the designer proposes the matching criterias for the variable. According to the matching result one sets the value of the variable. The pattern matching can be simple (just a comparison) or very complex one to match a sequence of actions. This is an important feature in *Problem Solving* assessment. The *PatternMatching* part is divided into three parts:
    a. *SymbolDeclarations* integrates a set of symbols that will be used in pattern matching. For each symbol declaration one indicates the *Symbol_ID* (usually a letter) and the *SymbolQuery*. The query permits to associate each event in the trace with the appropriate *Symbol_ID*. The syntax of the query is very close to SQL Where clause and provides powerful possibilities of filtering.
    b. *Pattern.* In this part the designer puts a sequence of symbols that constitutes the pattern to match. In the case of a simple matching only one symbol is used. However, in the case of a complex problem solving, the *Pattern* can be very complex according to the sequence of actions the designer wants to match. The syntax of a *Pattern* is similar to the regular expression one.
    c. *PatternStrategy* is used to define the strategy of sequence of the proposed *Pattern*. These strategies are inspired from the *Complex Event Processing* area and the possible values are: *Strict*, *Order*, *Set*.
5. *Value_If_Match* indicates the value of the variable if the pattern is matched. One can put a direct value or the name of a scoring function from the *Toolbox of functions*. The returned value from this function will be used.

4.3 Pattern Matching and Scoring Process

Figure 3 presents an overview of the proposed *Pattern Matching and Scoring Process*.

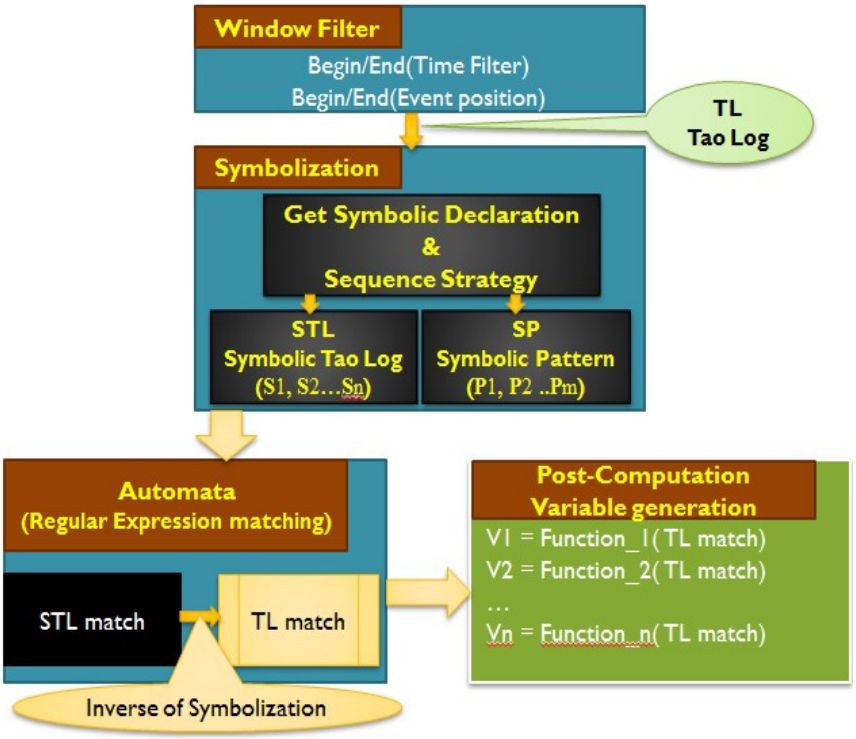

Figure 3. Pattern Matching Process

The *Window Filter* step provides a set of traces according to the defined interval. After that, the *Symbolization* step creates the *Symbolic Tao Log* according to the symbol declarations in the variable descriptions. Another activity in this step consists of the creation of a new pattern based on the Sequence Strategy.

In the third step (*Regular Expression Matching*), one uses the powerful mechanism of pattern matching of the regular expression in order to extract the subset log. The second activity permits to do the conversion from the *Symbolic Tao Event*s to its equivalent subset of the initial log. The last step will used this subset as a parameter of the chosen scoring function.

5 Conclusion

If the quality of the data extracted from the results of a test – whatever the nature of the test – is an essential feature of a fully-fledged e-assessment platform then, via our contribution, the TAO platform will rise up to an unprecedented level of flexibility and power in scoring phase. A clear process is proposed, and appropriate techniques and facilities have been elaborated. For the benefit of the educational community, this mechanism – already used for the OECD PIAAC study [2] – has been integrated in the "result module" of the open-source assessment platform named TAO – *Testing Assisté par Ordinateur* or Computer-Based Assessment [1].

References:

[1] TAO platform is located and documented at: http://www.tao.lu
[2] OCDE PIAAC (Programme for the International Assessment of Adult Competencies)
[3] Lime Survey is located at : http://www.limesurvey.org/
[4] Hot Potatoes is located at: http://hotpot.uvic.ca/
[5] Perception Question mark is located at: http://www.questionmark.com
[6] Fast Test Web is located at: http://www.fasttestweb.com
[7] Web Quiz XP is located at: http://www.smartlite.it
[8] Mōsbē is located at : http://www.mosbe.com/index.shtml
[9] Forterra System is located at: http://www.forterrainc.com/
[10] H. Sue Elle, M. Theresa. "*Reporting on Item Times and Keystrokes from Blaise Audit Trails*". 7th International Blaise Users Conference (Washington DC, 2001), pp 1-16, 2001.
[11] E.Luckam, R. Sculte. "*Event Processing Glossary - Version 1.1*", pp 1-19, 2008.
[12] R. Sadri, C. Zaniolo. "Expressing and Optimizing Sequence Queries in Database Systems". ACM Transactions on Database Systems (TODS). Volume 29, Issue 2, pp: 282 – 318. 2004.
[13] R.S. Barga, H. Caituiro-Monge. "*Event Correlation and Pattern Detection in CEDR*". Current Trends in Database Technology – EDBT 2006, pp 919-930, 2006.
[14] Jagrati Agrawal, Yanlei Diao, Daniel Gyllstrom, and Neil Immerman. "*Efficient Pattern Matching over Event Streams*". In Proceedings of SIGMOD 2008.

Author(s):

Younes, Djaghloul; Raynald, Jadoul; Patrick, Plichart; Thibaud, Latour.
Research Centre Henri Tudor, Luxembourg.
Email: {younes.djaghloul, raynald.jadoul, patrick.plichart, thibaud.latour}@tudor.lu

Vincent, Porro; SOGETI, Luxembourg; Email: vincent.porro@sogeti.lu