Francisco J. Almeida-Martínez
Jaime Urquiza-Fuentes

Preliminary evaluation of the syntax error recovery visualization using VAST.

Number 2011-08

Serie de Informes Técnicos DLSI1-URJC
ISSN 1988-8074
Departamento de Lenguajes y Sistemas Informáticos I
Universidad Rey Juan Carlos

Index

Preliminary evaluation of the syntax error recovery visualization using VAST

Francisco J. Almeida-Martínez and Jaime Urquiza-Fuentes

Universidad Rey Juan Carlos
Departamento de Lenguajes y Sistemas Informáticos I
c/ Tulipn s/n, 28933 Madrid, Spain
francisco.almeida@urjc.es, jaime.urquiza@urjc.es

Abstract. This document describes the preliminary evaluation of the syntax's error recovery visualization using VAST. This evaluation was performed after implementing this new functionality into VAST. As we can read in the document, results are satisfactory, so we obtain tips of how to continue the development of VAST.

1 Introduction

After carrying out different observational, usability and educational effectiveness evaluations, we added a new functionality to VAST: **syntax error visualization**. We implemented the basic functionality of this aspect and solved some errors which were detected in the tool. We decided to test two different strategies of syntax error recovery: panic and insertion modes. The features of the tool are similar to the version used in previous evaluations.

2 Description of the evaluation

In this section we explain the experiment. We describe the participants, the experimental design, the tasks done during the evaluation and the protocol used in the whole experiment.

2.1 Subjects

In this evaluation participated **6 old students** of the *Language Processor* subject of the Rey Juan Carlos University during the 2009-2010 course. The participation was totaly voluntary. The gender distribution was not balanced, so the 16.7% (1/6) were women. We have to take into account that the participants passed the subject in different years.

2.2 Experimental design and tasks

This evaluation was designed as an observational study plus an usability and educational effectiveness one, however it was impossible to join all the partipants in one unique session to check the educational effectiveness. As we were interested in checking the participants' knowledge about the syntax error recovery we performed a pretest (see appendix 5). The dependent variables of the experiment were the students' opinion about four different aspect of VAST: ease of use, learning support to the syntax error recovery using visualizations, quality of the tool and global satisfaction.

The tasks performed by students had to be documented in the evaluation test, using text answers. The tasks consisted in two exercises about syntax error recovery, one for insertion strategy and another one for the panic mode (see appendix 5).

Insertion recovery. This recovery method is one of the strategies used by the generator tool ANTLR. In this case we gave the participants two visualizations built by a particular parser. We asked them for loading the visualizations into VAST and using its views (tree, input stream, grammar and stack) determine how the parser had recovered from the existing errors.

Panic mode recovery. This method is used by the generator tool Cup. In this case, we gave the students the corresponding parser (annotated and generated). Given a particular syntax error recovery implemented in the parser's specification, students had to propose different input streams to make the parser perform different actions.

2.3 Protocol

Three weeks before the session we sent the pretest of knowledge by email. As it was impossible to join all the participants in one session students did not get used to the tool. Due to this, at the beginning of the session the instructor explained how to work with VAST and its main features during 15 minutes. During the experiment students had to do the two exercises and answer the questionarie (see appendix 5). One week after the evaluation we performed an individual interview to certain participants. In table 1 we show the protocol followed in this evaluation.

Grupo nico
Pretest of knowledge
VAST presentation + Evaluation
Answer to questionaries
Postest of knowledge
Interviews

Table 1. Protocol followed in the evaluation

3 Results

In this section we describe the results of the experiment which are divided in: instructor's observations, answers to questionaries and interviews.

3.1 Instructor's observations

During the expermient the instructor observed how the participants used the tool, detected errors and found difficulties to execute the task required.

In the evaluation students had to load certains visualizations and from them they had to explain how the parser had recovered from the corresponding syntax errors. In all cases, participants used the animation of the syntax's tree building process and the stack to determine the actions performed by the parser.

In general, all participants finished correctly the evaluation and dedicated much time to explore the VAST's menus. There was a special interest on the technical details about the parser importation feature, although it was not the evaluation objective.

3.2 Answers to questionnaires

As in previous evaluations, all the opinion question had to be answered using a likert scale from 1 to 5, being 1 the worst mark and 5 the best one. Besides, we included open question in order the participants could express their opinion. In table 2 it is shown the average marks of the dependent variables of the experiment. In table 3 we show the specific marks for each part of the tool according to *Ease of use* and *Quality*.

General aspects	**Average**	**%[4-5]**
General ease of use	4.3	83.33 %
Comprenhension of insertion strategy	3.5	50%
Comprenhension of panic mode	4	66.67%
General quality	3.8	83.33%
General satisfaction	4.2	66.67%

Table 2. Marks of the general aspects

The open questions were divided in aspects which VAST did not have but they would be useful, aspect not needed, positive and negative. According to the first one (useful but not included), students mentioned the posibility to add *tooltips* over the reproduction's toolbar. Another aspect was to indicate when the parser could not recover from a specific syntax error using a notification window or something similar. According to the positive comments, students pointed out the global view, so it makes easier to navigate through the tree specially when it is very big. Also they pointed out the different views, multiplatform, ease of use,

speed of execution and the learning support to the syntax error recovery. Besides, one participant refered to the remark of the input stream as an extremely positive aspect. Finally, according to the negative comments, students mentioned the lack of information about the buttons in the toolbar.

Aspect	**Ease of use**	**% [4-5]**	**Quality**	**% [4-5]**
Main menu	4.3	83.33%	4.5	100%
Icons	3.3	50%	3.3	33.33%
Anim. controls	3.8	50%	4	66.67%
Global view	4.3	66.67%	3.8	66.67%
Subtree	3.7	66.67%	3.3	50%
Zoom	4.5	100%	4.7	100%
Input stream	4.2	83.33%	4	66.67%
Stack view	3.5	66.67%	3.3	50%
Configuration	3.8	50%	3.8	66.67%
Config. manage	3.8	50%	3.7	50%
Edit input stream	4	66.67%	3.7	66.67%

Table 3. Specific marks

3.3 Interviews

Some aspects of the open questions were ambiguous and confusing. Due to this it was neccesary to perform some interviews to some students.

Analyzing the marks of the syntax error learning support we observed that only for one student the syntax error recovery using insertion strategy was better represented than the panic mode. In this case we asked the students the reasons. This was because of the visualization of the panic mode was very similar to the insertion one.

Another comment which was detected in the interview was due to the low mark of the stack visualization. For this participant the stack's visualization was not a stack.

Finally, another students showed that it would be necessary to introduce interactive comments in the visualizations.

4 Conclusions

The results of this evaluations have been satisfactory. Although it has been a preliminary evaluation of the syntax error recovery, we have obtained clues of how continue the development of VAST in both ways: usability and quality of the tool. From these results, we plan three future line of works:

– Finish the development of the syntax error visualization: although we carried out this evaluation, there were incomplete aspects of implementation. In

the panic mode the visualization was extremely dependent from the parser. According to the insertion strategy it did not work properly in all cases due to an internal bug of ANTLR parser generator.
- Improve the existing features: we refers to work in those aspect which are useful but VAST does not have.
- Add new features to VAST: according to the students opinion, we should include some features as the user help, *tooltips*, error messages, etc.

5 Acknowledgment

This project is supported by project TIN2008- 04103/TSI of the Spanish Ministry of Science and Innovation.

A Questionarie used to the pretest

1. How would you define the panic syntax error recovery?
2. How would you define the syntax error recovery using error productions?
3. How does the insertion syntax error recovery works?
4. Given the following grammar:

```
S::=id=FS';

S'::=+FS' |-FS'|LAMBDA

F::=id|cte|(S)
```

   Using the notation ($Stack, Input$) write cases of an LL(1) non recursive parser indicating the input stream and stack before and after each of the following actions:
   (a) A derivation
   (b) The recognition of a token from the input stream
   (c) The recovery of an error using the insertion strategy
5. Using the previous grammar:

```
S::=id=FS';
S'::=+FS' |-FS'|LAMBDA
F::=id|cte|(S)
```

   We have implemented a LL(1) parser with insertion syntax error recovery, and with the following input stream $id(cte + cte;$. Draw the resulting syntax tree indicating the part processed, where the error has been detected and the point of recovery.

6. Given the following grammar:

```
S::=id=E;
E::= E+T|E-T|T
T::= id|cte|(E)
```

   Write cases of a LR (1) parser indicating the state of the input stream the stack before and after each of the following operations:

   (a) A reduction
   (b) The recognition of a token in the input stream
   (c) The recovery from an error using the panic mode

7. Using the same grammar of the previous exercise:

```
S::=id=E;
E::= E+T|E-T|T
T::= id|cte|(E)
```

   We have implemented a LR (1) parser with panic error recovery where the synchronization tokens are the follows of the antecedent which is being processing. With the input stream: $id + (cte - -) - (idcte;$, draw the syntax tree indicating the part recognized, the part ignored, the point of the error and the point of recovery.

B Exercises of the evaluation

1. Given the following grammar to represent arithmetics expressions:

```
S::=FN
N::=+FN | -FN | * FN | /FN | lambda
F::= id | cte | ( S )
```

   (a) Load the visualization in the file **1-1.xml** (*File–Load XML*). This file contains the representation of a syntax tree for a valid input stream: **(10+3+(40-50)+6)**
   (b) Load the visualization in the file **1-2.xml**. This file contains the syntax tree for an erroneus input stream: **(10+3++(40-50)+6)**. How does the parser recover from this error?
   (c) Load the visualization in the file **1-3.xml**. This file contains the representation of a syntax tree for a erroneous input stream: **(10+-3+(40-50)+6))**. How does the parser recover from this error?

2. Given the following grammar to represent sentences **if**:

```
S::=E
E::=if(COND) {SENT} | if error ) {SENT}
COND::= T OP T
SENT::= SENT EXP | EXP
EXP ::= eval (EXP2) | ident = EXP | eval (error)| EXP2
EXP2::= EXP2 and L | EXP2 or L | L
L::= L nand T | L xor T | T
T::= ident | cte
OP ::= <| >| =
```

Given the syntax error recovery in panic mode where the recovery used the **error** symbol :

(a) Generate an erroneus input stream which makes the parser ignore the minimun number of tokens but at least one.

(b) Generate an erroneus input stream which makes the parser ignore the minimun number of tokens.

(c) Generate an erroneus input stream which makes the parser ignore the maximum number of tokens.