Evaluating the Effect of Feedback on Syntactic Errors for Novice Modellers

Ralf Laue
Chair of Applied Telematics/e-Business,* University of Leipzig
Klostergasse 3, 04109 Leipzig, Germany
laue@ebus.informatik.uni-leipzig.de

Stefan Kühne
Business Information Systems, University of Leipzig
Johannisgasse 26, 04103 Leipzig, Germany
kuehne@informatik.uni-leipzig.de

Andreas Gadatsch
University of Applied Sciences Bonn-Rhein-Sieg
Grantham-Allee 20, 53757 Sankt Augustin, Germany
Andreas.Gadatsch@fh-bonn-rhein-sieg.de

Abstract: In this paper, we present the results of a controlled human experiment where students in a business process modelling course had to model a business process from a case study as part of their coursework. One group could take advantage of the continuous validation feature that is implemented in the *bflow* modelling tool, i.e. they were provided with alerts about modelling errors. A control group had to create a model for the same case study without using continuous validation. The results of the experiment indicate that the presence of continuous validation indeed has had a positive effect on the number of syntactic errors in business process models.

1 Introduction and Related Work

Modern development environments like Eclipse or Visual Studio provide immediate feedback on syntactic and semantic errors by performing continuous parsing and compilation. This continuous compilation feature helps to detect coding errors very quickly and can reduce the time between making an error and fixing it.

An extension is continuous testing [Saf04] where regression tests are run in the background, providing the developer an instant feedback about test failures.

It has been shown experimentally that continuous compilation and continuous testing have a statistically significant effect on the success in completing a programming task [SE04, Saf04].

*The Chair of Applied Telematics/e-Business is endowed by Deutsche Telekom AG.

The idea of continuously validating the correctness has been adapted to the domain of graphical modelling as well. *ARGO/UML*[RR00] allows to run so-called *design critics* in background and can give suggestions for improvements. *MetaModelAgent* [BI06], which is an extension of *IBM Rational Software Architect*, contains a continuous model verification feature. With each alteration of the model, it is verified against several guidelines. Violations are reported, and suggestions for a correction are provided. *UML/Analyzer* [Egy06] provides feedback on model consistency.

The aforementioned tools allow to check a model for meta-model compliance, violations of modelling style conventions and intra-model consistency.

Literature on experiments testing the usefulness of continuous validation for graphical models is rare. Dranidis [Dra07] showed that the consistency checking feature of the educational UML modelling tool *StudentUML* helps students in identifying and correcting their modelling mistakes.

Lange et al. [LBCD06] conducted an experiment in which they compared three groups of modellers: One group did not use modelling conventions, a second one used modelling conventions and a third group used a tool that checked adherence to modelling conventions. They found that the group using tool-supported modelling conventions had less defects, although the results were not statistically significant.

In this paper, we describe an experiment with a small group of business administration students. We gave the students the task to build a business process model from a case study. The language that had to be used for this purpose were Event Driven Process Chains. A short introduction into this language is given in Sect. 2. The modelling tool used for the experiment, *bflow Toolbox*, comes with a continuous validation feature which will be described in Sect. 3. We describe our experiment to evaluate the effect of using this feature in Sect. 4. The results of a group that used this feature and the results of a control group which modelled without continuous validation are given in Sect. 5 and statistically compared in Sect. 6. Afterwards, we discuss the findings in Sect. 7.

2 Event-Driven Process Chains

For our experiment, we used the modelling language Event Driven Process Chains (EPC).

EPCs consist of functions (activities which need to be executed, depicted as rounded boxes), events (pre- and postconditions before / after a function is executed, depicted as hexagons) and connectors (which can split or join the flow of control between the elements). Arcs between these elements represent the control flow. The connectors are used to model parallel and alternative executions. There are two kinds of connectors: Splits have one incoming and at least two outgoing arcs, joins have at least two incoming arcs and one outgoing arc. AND-connectors (depicted as ⊛) are used to model parallel execution. When an AND-split is executed, the elements on all outgoing arcs have to be executed in parallel. The corresponding AND-join connector waits until all parallel control flows that have been started are finished. XOR-connectors (depicted as ⊗) can be used to model alternative execution: A XOR-split has multiple outgoing arcs, but only one of

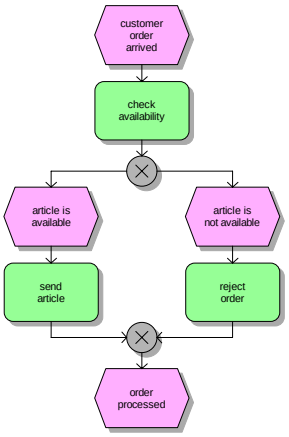


Figure 1: Simple Business Process modelled as EPC

them will be processed. A corresponding XOR-join waits for the completion of the control flow on the selected arc. Finally, OR-connectors (depicted as ⓥ) are used to model parallel execution along one or more control flow arcs. An OR-split starts the processing of one or more of its outgoing arcs. The corresponding OR-join waits until all control flows that have been started by the OR-split are finished.

The EPC elements described above are sufficient for modelling simple business processes like the following one: "When a request from a customer arrives, the availability of the product has to be checked. If it is available, the item will be sent; otherwise the customer will get a negative reply." Fig. 1 shows this business process modelled as EPC diagram.

A detailed definition of the syntax of the EPC language can be found in [NR02, van99]. For the purpose of this article, we just mention the most important syntax requirements of EPC models: Functions have exactly one incoming and exactly one outgoing arc. Events have at most one outgoing and at most one incoming arc. Events without an incoming arc are called start events (like "customer order arrived" in Fig. 1), and events without an outgoing arc are called end events (like "order processed" in Fig. 1). Connectors can be either splits or joins as described above. Events and functions alternate along the flow of control, i.e. an event has to be followed by a function and vice versa. Finally, an event must not be followed by an XOR- or OR-split. The intention of this last rule is that the activity that lead to the (X)OR-decision should be modelled explicitly as a function.

3 Automatic Checks in bflow

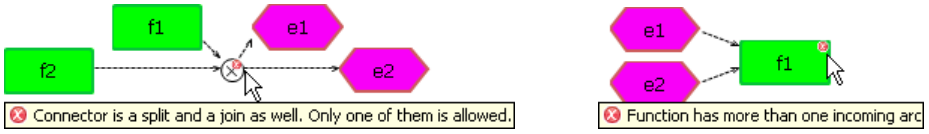
For the experiment described in this paper, we used a localized German version of the open source modelling tool *bflow Toolbox 0.0.4*[1]. Currently, this Eclipse-based tool supports

[1] `www.bflow.org`

three graphical modelling languages, including EPCs. The version number 0.0.4 indicates that the development status of the tool is that of an alpha version. In many points, the usability of the tools still needs to be improved.

The *bflow Toolbox* comes with a feature called continuous validation. Checks for correct syntax, control-flow correctness (for example absence of deadlocks) and adherence to modelling conventions can be run in background. The user gets a feedback about errors and warnings. This feedback is shown by placing a marker at a modelling element which is related to the problem and by adding a textual information to the "Problems" view of the Eclipse IDE. By placing the cursor on the marker, the user can get additional information as shown in Fig. 2. This way, a problem is shown immediately without forcing the users' attention away from the primary modelling task, as recommended in [McF02].

A detailed description of the continuous validation feature can be found in [KKGL08].



(a) (b)

Figure 2: Feedback on two syntax errors

4 Experimental Design

The experiment was run at the the University of Applied Sciences Bonn-Rhein-Sieg, Germany. The participants were 15 business administration students. The students have selected business information systems as one of two fields of specialisation. So it can be assumed that they have an high affinity in busines process modeling like practicioners in real business process management projects. The students were familiar with modelling using Event Driven Process Chains from lectures they followed in the previous academic years. During those lectures, they had to draw models using pen and paper, but they did not have to model using an editor.

At the beginning of the experiment, we introduced the *bflow* editor and shortly recapitulated syntax and semantics of Event Driven Process Chains. We told the students that we will give them a case study for which they will be asked to create a business process model. We told them that the case study is similar to the ones they will be given at the written examination. Also, we told the students that the purpose of the experiment was to get feedback on the usability of the *bflow* editor and to collect suggestions for possible improvements (which was indeed the case, but is out of the scope of this paper).

We asked the students to fill a questionnaire, where we asked the following questions:

1. What is your age?
2. What is your sex?

3. What is your field of study, and how long have you been studying so far?
4. Is German your native language?
5. How familiar do you feel with modelling using Event Driven Process Chains? (allowing a choice between five answers from “very good” to “unfamiliar”)
6. How many business process models you have modelled so far using a graphical editor? (allowing the choice between “none”, “less than 5” and “more than 5”)
7. How familiar you are with Eclipse, regardless whether used as a programming environment or an editor for modelling? (allowing a choice between five answers from “very good” to “unfamiliar”)

Each of the 15 students filled the questionnaire. We exploited the answers to the questions 2, 4, 5, 6 and 7 for grouping the students into two groups: Group A had to create a model without continuous feedback on errors, group B could make use of this feature.

Among the 15 students, there were 13 male students whose native language was German and two female students whose native language was not German. As the ability to model correctly depended on understanding the case study (which was given in German and included domain-specific phrases from the field of mechanical engineering), the two latter students were placed in different groups.

We made the group assignments such that the groups were as similar as possible with regard to the other questions.

The students was given a case study - a hypothetical mechanical engineering business process. The case study has been constructed for the purpose of the experiment.

The students had to work individually. If there was a general question about the handling of the editor, the question has been answered by us. We did, however, not answer any questions about the case study.

While group A used the *bflow Toolbox* without the continuous validation feature, group B was provided with the same tool where continuous validation was running in the background.

The allotted time for solving the task was 45 minutes for each group. Students, who completed the model earlier, were told to write down the time needed.

We saved the models created during the experiment for being analysed later. Unfortunately, the result of one student of each group could not been saved because of a technical problem, i.e. we were able to collect 13 models only. However, this did not affect the considerations to make both groups as similar as possible with regard to the answers given on the questionnaire: In each group of the 13 students whose models have been collected was one student who has used Eclipse “a little bit” before, the others have never done so. Also, each group contained two students who rated their knowledge of the EPC modelling method as “moderate”, two students who rated it as “imperfect” and two students who rated it as “rather poor”. With one exception, the students have never used a tool for creating a business process model. The student who has already used a tool was also the only one who rated his knowledge of the EPC method as “good”. With the exception of this student, who was randomly assigned to group B, the groups were absolutely equal with respect to the answers to the questions on the questionnaire. The arithmetic mean of the

ages in group A was 25.2 years, and the arithmetic mean of the ages in group B was 26.9 years.

5 Quantitative Results

All students spent the given time of 45 minutes on modelling, no one finished the task earlier.

Because of the simplicity of the case study, no model contained one of the semantic errors (deadlocks etc.) that can be detected using the continuous validation feature. The syntactic errors that were made are shown in Tab. 1. Altogether, there were 24 such errors in 6 models in group A and 6 such errors in 7 models in group B.

	without validation						with validation						
model	1	2	3	4	5	6	7	8	9	10	11	12	13
event or function has more than one incoming or outgoing arcs	0	0	3	1	1	1	0	0	0	0	0	0	2
model starts with a function instead of an event	0	0	1	0	0	1	0	0	0	0	0	0	0
model ends with a function instead of an event	1	0	2	1	5	2	0	0	0	0	0	0	1
connector is both split and join	0	0	0	0	0	0	0	1	0	0	0	0	0
isolated element	0	0	2	0	0	0	1	0	0	0	0	0	0
event is followed by an (X)OR split	2	0	0	0	0	1	0	0	0	0	0	0	1
Sum	3	0	8	2	6	5	1	1	0	0	0	0	4

Table 1: Syntactic Errors Found in the Models

For the purpose of our comparison, we did not count two kinds of syntactic errors: One error we disregarded for the purpose of this experiment was the use of the wrong type of arc. *bflow* offers a dotted arc for modelling the control flow and a continuous arc for modelling the information flow. It is a weakness of the current version of *bflow* that selecting the wrong type of arc can happen very easily. The second error we did not count was when a function was followed by a function instead of an event. Strictly, this would be against the EPC syntax. However, the students have been told in the course before (in accordance with most textbooks on the subject) that in some cases such a construct is allowed. Without excluding both types of errors, there would have been 53 errors in 6 models in group A and 12 errors in the 7 models in group B.

We also evaluated the overall quality of the models under terms of regular exams. This was done by asking five university lecturers who are very familiar with business process modelling to give subjective marks for each model. They could choose from a range

	without validation						with validation						
model	1	2	3	4	5	6	7	8	9	10	11	12	13
manual quality rating	3.6	1.6	4.0	2.4	3.6	3.6	4.2	3.2	4.6	2.0	2.5	2.3	3.0

Table 2: Marks given to the models

between mark 1 (“very good”) to mark 5 (“fail”). The averages of the marks given are shown in Tab. 2.

The mean grade for group A was 3.13; the mean grade for group B was 2.97.

6 Statistical Analysis

For comparing the number of errors made in each of the two groups, we performed the one-tailed Mann Whitney U test. The null hypothesis was that the results of both groups were drawn from the same population, i.e. the presence of the validation feature does not have an effect on the number of syntactic errors.

The group sizes were 6 and 7; and the value of U was 6. From this follows that the probability for the results from both groups belonging to the same population is p=0.02275. Using a significance level of 0.025 for the one-tailed test, we can conclude that the results from both groups are significantly different, i.e. the continuous validation feature has had a positive effect on the number of errors.

7 Discussion and Further Research Directions

An obvious weakness of our experiment was the small number of students taking part. Nevertheless, the results shown in Tab. 1 give raise to the assumption that the continuous validation feature indeed has a positive effect on the presence of errors. It has to be noted that the errors investigated in our experiment contributed only to a very small part of the model quality. The marks given to students in group B were only slightly better than the marks given to students in group A; the difference is in no way statistically significant.

The fact that only one student has used a business process modelling tool before limits the scope of the result. Most likely, more experienced modellers would not profit from finding rather trivial syntactic errors as much. Our result indicates that continuous validation for such syntactic errors is valuable at least for novice modellers.

Because of the time limitation, we could not test the effect of continuous validation on errors in larger and more complicated models. As such models are more likely to contain control-flow related errors (like deadlocks), we would expect that the ability of the validation in *bflow* to detect such errors as well would make it more helpful even for more experienced users.

Further experiments are needed with more participants and if possible with more complex models. It is our intention to repeat the experiment with a larger group of students. However, before doing so, we will make some improvements to the tool. For example, it seems to be less useful to create an alert when one draws two incoming arcs into an event or function. A much more straightforward action to prevent this kind of errors would be to forbid such constructs by restricting the meta-model.

References

[BI06] Fredrik Bäckström and Anders Ivarsson. Verification and Correction of UML Models. 2006. Online; zuletzt abgerufen am 3. Juli 2009.

[Dra07] Dimitris Dranidis. Evaluation of StudentUML: an Educational Tool for Consistent Modelling with UML. In *Proceedings of the Informatics Education Europe II Conference IEEII 2007*, 2007.

[Egy06] Alexander Egyed. Instant consistency checking for the UML. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 381–390, New York, USA, 2006. ACM.

[KKGL08] Stefan Kühne, Heiko Kern, Volker Gruhn, and Ralf Laue. Business Process Modelling with Continuous Validation. In *Business Process Management Workshops, BPM 2008 International Workshops, Milano, Italy, September 2008, Revised Papers*, volume 17 of *Lecture Notes in Business Information Processing*, pages 212–223. Springer, 2008.

[LBCD06] Christian F. J. Lange, Bart Du Bois, Michel R. V. Chaudron, and Serge Demeyer. An Experimental Investigation of UML Modeling Conventions. In *MoDELS*, volume 4199 of *Lecture Notes in Computer Science*, pages 27–41. Springer, 2006.

[McF02] Daniel C. McFarlane. Comparison of Four Primary Methods for Coordinating the Interruption of People in Human-Computer Interaction. *Human-Computer Interaction*, 17:63–139, 2002.

[NR02] Markus Nüttgens and Frank J. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In *Promise 2002 - Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen*, pages 64–77, 2002.

[RR00] Jason E. Robbins and David F. Redmiles. Cognitive support, UML adherence, and XMI interchange in Argo/UML. *Information & Software Technology*, 42(2):79–89, 2000.

[Saf04] David Saff. *Automatic continuous testing to speed software development*. PhD thesis, Massachusetts Institute of Technology, 2004.

[SE04] David Saff and Michael D. Ernst. An experimental evaluation of continuous testing during development. In *ISSTA 2004, Proceedings of the 2004 International Symposium on Software Testing and Analysis*, pages 76–85, Boston, USA, July 12–14, 2004.

[van99] Wil M. P. van der Aalst. Formalization and verification of event-driven process chains. *Information & Software Technology*, 41(10):639–650, 1999.