Evaluation of the E3 Process modelling language and tool for the purpose of model creation

M. Letizia Jaccheri\textsuperscript{1} and Tor Stålhane\textsuperscript{1}

Norwegian University of Science and Technology,
IDI/NTNU 7491 Trondheim, Norway
\{letizia, Tor.Stalhane\}@idi.ntnu.no

Abstract. In this paper, we report from an experiment which compared the E3 PML with respect to the standard modelling language IDEF0 for the purpose of model construction. The experiment has been run as part of a software process improvement course in which forty students participated.

Our hypothesis was that E3 will lead to less problems than IDEF0 when constructing software process models. Here, we show how our experiment has validated the hypothesis.

1 Introduction

The research community has developed several software process modelling languages (PMLs) and tools in the last twenty years. Moreover, some attempts have been made to test these languages and tools, also in an industrial context [2] [3].

In the last years we have developed the E3 system which offers a PML and a tool [7]. We have also done some attempts to test E3 both in industrial and in academics settings. This paper describes an experiment\textsuperscript{1} we have performed in order to test if the E3 system is easy to use for creating software process models.

Since E3 is designed as a special purpose tool for modelling software development processes, we expected that users would encounter less modelling problems when using the E3 PML than when using a general purpose modelling language.

In order to evaluate E3 PML, we decide to compare it to a standard modelling language and its associated tool support. As a standard modelling language we choose IDEF0 which is based on SADT and is well known in academia and widely used industry.

We designed a modelling exercise in the context of a software process improvement course [6] where 40 students, divided into 10 groups modelled a software process by using one of the two PMLs. Five groups modelled with E3 and five groups with IDEF0. As a quality measure, we choose the number of problems

\textsuperscript{1} This work was partially supported by the SPIQ project. SPIQ is a Norwegian project on Software Process Improvement (1997-1999). SPIQ means Software Process Improvement for better Quality.
encountered using the PMLs. We provide both a statistical analysis of problems and a discussion of these problems.

The rest of the paper is structured as follows. Section 2 introduces E3 and IDEF0 (with the associated supporting tool KBSI AI Win). It also presents previous evaluation of the E3 system. Section 3 presents our experiment. Our conclusions are given in section 4.

2 Background

2.1 The E3 system

The E3 PML E3 [7] PML is a formal object-oriented modeling language developed at Politecnico di Torino, specially aiming at process elicitation and process analysis. The representation is graphical. The E3 p-draw drawing tool supports creation and management of E3 process models. E3 PML offers three modeling levels: meta-classes (which is used by system administrators), class or template (to denote general process models), and instance (to express instantiated models or plans). E3 provides five kernel classes:

- **Task**: a class whose instances represent the activities carried out in the process.
- **Resource**: a class whose instances model actual resources with their responsibilities and skills.
- **Data**: a class whose instances model process artefacts, such as source code fragments and invoices.
- **Tool**: a class whose instances represent a specific version of an automated tool or of a written procedure.
- **E3Object**: it is the superclass of the latter four.

E3 also provides kernel associations:

- **Association** (C1, C2): expresses a generic relationship between an object of class C1 and a set of objects of class C2.
- **Aggregation** (C1, C2): expresses the "is-composed-of" relationship between an object of class C1 and a set of objects of class C2.
- **Subtask** (T1, T2) is used to represent the decomposition of an activity into its subtasks. T1 and T2 must be subclasses of class Task.
- **Preorder** (T1, T2): denotes the precedence relationship among tasks. T1 and T2 must be subclasses of class Task.
- **Responsible** (R, T) denotes the "is-responsible-for" relationship between a resource and a task. T must be a subclass of class Task; R must be a subclass of class Role.
- **Input** (T, D): identifies the inputs for a task T. T must be a subclass of class Task; D must be a subclass of class Data.
- **Output** (T, D): identifies the outputs of a task T. T must be a subclass of class Task; D must be a subclass of class Data. The transitive closure of the Input and Output associations defines the data flow in the process model.
- **Use** (T, TL): identifies the tools used within a task. T must be a subclass of class Task; TL must be a subclass of class Tool.
E3 p-draw The E3 p-draw is a drawing tool that supports creation and management of E3 process models, and provides a mechanisms that enables model inspection according to selected views. E3 p-draw is a beta release and not a commercial product. The tool offers drawing views to develop process models. Figure 1 shows an example of such a view which contains the definition of one of the models developed during the experiment.

![Diagram of E3 p-draw](image)

**Fig. 1.** An edit view displaying one of the five E3 model developed during the experiment.

E3 p-draw also offers several types of derived views or filters:

**Simple** This filter is applied to a class (object) to show all associations (links) in which the class (object) is involved, with the exception of the Aggregation and Subtask associations (links). This filter makes it possible to illustrate the entities that are in some way related to a specific entity at the same level of abstraction. See figure 2 for an example.

**Composite** When applied to a class C (object O), this filter shows all the classes (objects) that are components of C (O) through the Aggregation and Subtask associations (links). Moreover, it applies the simple filter to each of these classes (objects).
Recursive composite When applied to a class (object), this filter performs the recursive application of a composite filter to show all the classes (objects) that are directly or indirectly associated (linked) to the selected class (object) through Aggregation or Subtask associations (links).

2.2 IDEF0 and KBSI AI Win

IDEF0 The IDEF0 function modeling method is designed to model the decisions, actions, and activities of an organization or system. IDEF0 was derived from the Structured Analysis and Design Technique (SADT). IDEF0 does not distinguish between class and instance level. As an analysis tool, IDEF0 assists the modeler in identifying the functions performed and what is needed to perform them. Only input output associations can be defined. The IDEF0-models are based on a simple syntax. Each activity is described by a verb-phrase label placed in a box. Inputs are shown as arrows entering the left side of the activity box while the outputs are shown as exiting arrows on the right side of the box. Controls (denoting control flow) are displayed as arrows entering the top of the box and mechanisms (denoting resource allocation) are displayed as arrows entering from the bottom of the box. The processes can be decomposed by expanding activities.

KBSI AI Win Knowledge Based Systems, Inc. (KBSI) [5] has developed several tools for the Integration Definition for Function Modeling (IDEF0) family. AI Win is one of the tools that uses IDEF0 as the modeling language. Figure 3 shows
a snapshot of an AI Win view containing one the IDEF0 models developed during our experiment.

![Diagram]

**Fig. 3.** An AI Win view displaying one of the five IDEF0 model developed during the experiment.

### 2.3 Earlier attempts of testing E3

**Testing the ideas** The design of E3 focused on software process description (understanding and static analysis) and the use of object oriented modeling. In 1993, we started by using object-oriented techniques (Coad-Yourdon) to model the software process described in a quality manual we had obtained from IvecO - a division of Fiat. From this first modeling phase and interaction with process owners, we deduced the requirements for the E3 system and we had the chance to test our ideas.

For several years, we have been using the IvecO process as a reference for internal benchmarking of subsequent implementations of the E3 system.

**Testing E3 for the purpose of understanding** The E3 system has been used for four years in a software engineering course being taught at Politecnico di Torino to describe the software process used for all student project activities.
3 Our experiment

This section reports our experiment. For the definition, planning, and analysis of the experiment, we have mainly used [?].

3.1 Problem Statement

This work has been performed in the context of a case study in which 40 students (organized into 10 groups) from a software quality and process improvement course interacted with a Norwegian telecom company. During the case study the students came in contact with three actors: the quality manager, the manager of the process group, and a project leader. Among other tasks, the students were asked to model a process fragment, and to report about the problems they encountered. Five groups used E3 and five used IDEF0.

All students were majoring in computer science, with some exceptions: one PhD student and three business administration students. The groups were hence quite homogenous in age and knowledge. The input which was the basis for the case study is shown in Figure 4. The educational objective was mainly an exercise in process modelling and gaining experience with a formal process modelling language.

3.2 Experiment Planning

Since E3 is designed as a special purpose tool for modelling software development processes, we expected that users would encounter less modelling problems when using the E3 PML than when using a general purpose modelling language. Our hypothesis is thus that:

\textit{H1: For the purpose of creating software process models, the E3 PML is easier to use than a standard modelling language and tool.}

This can be rephrased as:
Fig. 4. A view of the input process description.

For the purpose of creating software process models, the average number of modelling problems ($\mu_1$) that students encounter when using E3 (PML and tool) is less than the average number of modelling problems they encounter when using IDEF0 ($\mu_2$). We state the null hypothesis as:

$$H_0: \mu_1 = \mu_2$$

We have one factor (PML and tool) with two treatments (E3 and IDEF0). The student groups are the subject of the experiment and the dependent variable is the number of modeling problems encountered.

3.3 Experiment operation

As mentioned before, we have decided to use students as subjects of our experiment. The reason for our choice is that it is difficult to convince industry persons to use the tools we propose. Moreover, it is almost impossible to convince them to perform the same task several times.

We motivated students by providing them with a real process (see figure 4) instead of just a toy example. The process was presented by a person from industry who also was aware of the experiment and communicated to the students the importance of performing such experiments. Concerning experiment execution and data collection, we had at least two choices:

- Observing the students while developing the process models. This could be supplied by video registration.
- Ask the students to list the problems encountered as part of the report they had to deliver after the exercise.
We chose to ask the students to write a list of the encountered problems. The reason for rejecting a direct observation of the student work, was that it would have been time consuming. Also, the students did not always solve their exercises at the university and not always during normal working hours.

Most of the groups had some problems interpreting the process model for the company. Four out of ten groups reported that they had problems interpreting it. These groups are not the same as the ones that reported problems interpreting the model. This indicates that the input description was not good enough. The underlying information from company was vague. More interaction after a first modelling session may have helped. An alternative would have been to enable more interaction between the students and the company by letting the students present the results directly to the company.

### 3.4 Presentation of the data

The entire set of collected data is presented in table 1.

If we look at the reported problems in 1 we notice that most of the IDEF0 problems are related to complexity while E3 problems can mainly be related to usability. It seems that it is more complicate to map a software process to a model using a general purpose modelling language than using a process specific language and tool, like E3. Table 2 displays the raw data on which we have performed statistical analysis.

### 3.5 Experiment analysis

We choose a non-parametric analysis in order not to make strong assumptions about the data. We started by ranking our data so that the biggest number of problems get the lowest rank. The result is shown in table 3.

Table 3 is organized so that the less problems a user has with a PML, the higher rank he will get. We observe that the sum of ranks (called $W_s$) for E3 is 33 while the sum of rank numbers for IDEF0 is 22. From this, it appears that E3 PML is better than IDEF0.

The data analysis is performed by Wilcoxon analysis.

\[
E(W_s) = \frac{n}{2} (n + 1)
\]

\[
Var(W_s) = \frac{nm}{12} (N + 1) - \frac{nm}{12N(N - 1)} \sum_{i=1}^{n} d_i (d_i - 1)
\]

Here $n$ is the number of elements in the group which is the base for $W_s$ and $m$ is the number of elements in the other group. $N = m + n$. $e$ is the number of groups with identical values while $d_i$ is the number of identical values in group $i$.

For the data set displayed in table 2, we have $n = m$. This gives $E(W_s) = 27.5$. 
Table 1. Collected data

<table>
<thead>
<tr>
<th>Group</th>
<th>Problems with PML</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (IDEF0)</td>
<td>None</td>
</tr>
<tr>
<td>2 (E3)</td>
<td>None</td>
</tr>
</tbody>
</table>
| 3 (IDEF0) | 1. What was complicate with the modelling activity was to decide whether a given influence on a process should be interpreted as control or input to the process.  
2. We had problems to specify resources to activities and subactivities precisely.  
3. We have used the general concept resources as we did not have available more precise concepts.  
4. We found that the constraint that one must have between 3 and 3 subactivities in a IDEF0 model limiting in a case in which we wanted to have two sub-activities.  
5. Sometimes there can be very many arrows between the different boxes even if we only have 6 boxes. Such big quantity of arrows makes the models more difficult to follow and to manage. |
| 4 (E3) | None |
| 5 (IDEF0) | 1. Some activities has many inputs and this makes the model over-complex. |
| 6 (E3) | None |
| 7 (IDEF0) | None |
| 8 (E3) | 1. The problem is the overview. Although with a rather simple process like this one, it is difficult to maintain control.  
2. The fact that one must model both orienthial and vertical relationships in addition to document flow contributes to this. |
| 9 (IDEF0) | It is difficult to decompose activities.  
2. We had problems to distinguish between constraints and input.  
3. The model soon becomes over-complex, especially when one has many inputs and outputs |
| 10 (E3) | None |

Table 2. Raw Data: Number of logical problems encountered by the 10 groups.

<table>
<thead>
<tr>
<th>PML</th>
<th>Number of Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDEF0</td>
<td>0</td>
</tr>
<tr>
<td>E3</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3. Ranked data.

<table>
<thead>
<tr>
<th></th>
<th>IDEF0</th>
<th></th>
<th>E3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>n. of problems</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rank</td>
<td>7.5</td>
<td>1</td>
<td>4</td>
<td>7.5</td>
</tr>
<tr>
<td>n. of problems</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>rank</td>
<td>7.5</td>
<td>7.5</td>
<td>7.5</td>
<td>3</td>
</tr>
</tbody>
</table>
The standard deviation \( \sigma_{W_s} \) is then 4.25. Moreover, we will use the following approximation:

\[
\frac{W_s - E(W_s)}{\sigma_{W_s}} \sim N(0,1)
\]

We can use this approximation to test if the observed range sum \( W_s \) is so much higher than its expected value that we can say that it is not a coincidence.

When inserting the computed values of \( W_s, E(W_s), \) and \( \sigma_{W_s}, \) we find a test value of 1.29 giving a 10% significance. In other words, there is a 10% probability that \( W_s > 33 \) even if there is not any difference between IDEF0 and E3.

4 Discussion and conclusion

As a conclusion from our data, we are 90% sure that there will be less modelling problems when using E3 PML than when using IDEF0 for the purpose of creating software process models.

Concerning statistical analysis, there are three risks stemming from the proposed conclusion:

- We have counted problems with the assumption that all the problems are equal. It could be the case that the two problems observed with E3 are extremely serious while the nine problems observed with IDEF0 all are small. After a study of the reports delivered by the students, we decided that problem counting is acceptable.
- We use a normal distribution approximation in the statistical test although we have few data (only five observations).
- Six of ten observations are identical. This also makes the normal distribution approximation unsure. There will, however, always be a big number of persons who will not have problems when using any of the languages. In this way it is not possible to get an experiment result without a big number of 0-observations.

We thus have to treat our conclusion "there will be less problems when using E3 PML than when using IDEF0 for the purpose of creating software process models" with some reservation. We need more data to increase our confidence in the conclusion. In addition, we should extend the data collection by letting the subjects:

- Register the time they used to model the process. If the time is fixed we should evaluate how much of their task they finish.
- Register the seriousness of each problem, for example on a three value scala (low, medium, high).

We have learned two lessons: first process modelling tool evaluation. Is it at all useful to compare two tools? This kind of comparison is useful for an
organization which wants to choose between two tools. But does it make sense to compare them in an experiment setting?

Second, we have used course students for research evaluation. Is it ethically correct? I.e., can it affect education in a negative way? How can we be sure that we do not influence students for example by giving them better instructions in the tool that we want to prove to be better? Last, how can we make sure that students do not cheat when reporting?

References