ABSTRACT
A number of methods, languages, tools and techniques are widely used in Software Engineering projects even before being formally evaluated in practice. In critical phases of software life cycle, such as requirements engineering, this practice may lead to deceptions. In this paper, an evaluation of the Systems Modeling Language (SysML) diagrams and constructions when applied to requirements engineering activities is proposed. This evaluation was performed based on the Technology Acceptance Model (TAM). Three criteria were used to evaluate the acceptance of SysML as a language for requirements engineering: perceived usefulness, perceived ease of use, and perceived usage. The evaluation was performed in practice in a company that develops software-intensive systems.

Keywords
TAM, SysML, Software Requirements.

1. INTRODUCTION
Application of unevaluated methods, languages, tools and techniques in Software Engineering industry projects’ is common. This practice can lead to a number of related issues, such as lack of confidence on technologies and complete project failures. Unfortunately, for some reasons such as aggressive marketing and decisions taken without adequate knowledge, untested methods and techniques have been routinely used in practice throughout the entire spectrum of requirements, design, implementation and evaluation activities of software development [9] [30].

Requirements Engineering (RE) is considered by some experts the most problematic and difficult phase within the development of software [15] [4] [25]. In specific domains, such as automotive software, RE is considered one of the crucial issues [6]. According to [5], knowing what to build, which includes requirements elicitation and technical specification, is the most difficult phase in the design of software. Dealing with ever-changing requirements is considered the real problem of Software Engineering [3]. Some studies [28] [10] found that the main factors for problems with software projects (cost overruns, delays, user dissatisfaction) are related to requirements issues, such as lack of user input, incomplete requirements specifications, uncontrolled requirements changing, and unclear objectives.

In this paper we want to evaluate the use of the Systems Modeling Language (SysML) [18] in activities of documenting and analyzing requirements. SysML is chosen mainly because it is a language that provides a specific diagram to model requirements. The SysML Requirements diagram covers important features of RE that are partially addressed or not addressed at all by other techniques [27]. Examples are facilities for traceability between requirements and the modeling of other types of requirements besides the functional ones. Another reason is that SysML is largely based on UML [17], a well-known modeling language and considered the de facto modeling language in industry. SysML has been applied to a number of projects [31] [14] in various fields, such as large telescopes [13], car manufacturing [2], nuclear automation [20], modeling the dynamics of hydraulic pump systems [12], and road traffic management systems [27].

Also, in this paper, SysML diagrams and constructions tailored for activities concerning RE are evaluated in practice in a company that develops software-intensive systems. The novelty of this paper lies on two aspects. First, the paper presents an attempt to evaluate part of SysML, more specifically diagrams and constructions used in requirements engineering activities. Second, the evaluation is performed in practice using TAM [9], a well-known model to evaluate technologies.

2. SysML Elements for Requirements Engineering
SysML is considered both a subset and an extension of UML [18] (Fig. 1). As a subset, UML diagrams considered too specific for software (Objects and Deployment diagrams) or redundant with other diagrams (Communication and Time Diagrams) were not included in SysML. Some diagrams are derived from UML without significant changes (Sequence, State-Machine, Use Case, and Package Diagrams), some are derived with changes (Activity, Block Definition, Internal Block Diagrams) and two new diagrams were introduced (Requirements and Parametric Diagrams). As a matter of fact, SysML is compatible with UML, which can facilitate the integration of the disciplines of Software and
System Engineering.

Two diagrams of the SysML language, the Requirements and the Use Case diagrams, and the SysML Tables, are recognized as useful for being used in activities of RE. These constructions are briefly presented in this Section.

2.1 SysML Requirements Diagram

The SysML Requirements diagram is a stereotype of the UML Class diagram, as depicted in Fig. 2. A SysML Requirement can appear on other UML/SysML diagrams to show its relationship to design. With the SysML Requirements diagram, visualization techniques are applied from the early phases of system development.

The SysML Requirements diagram allows several ways to represent requirements relationships. These include relationships for defining requirements hierarchy, deriving requirements, satisfying requirements, verifying requirements and refining requirements [18]. The relationships can improve the specification of systems, as they can be used to model and analyze requirements.

2.2 SysML Use Case Diagram

The SysML Use Case diagram is derived without important extensions from the UML 2.0 Use Case diagram. The main difference is the wider focus, as the idea is to model complex systems that involve not only software, but also other systems, personnel, and hardware.

The Use Case diagram shows system functionalities that are performed through the interaction of the system with its actors. The idea is to represent what the system will perform, not how. The diagrams are composed of actors, use cases and their relationships. Actors may correspond to users, other systems or any external entity to the system.

One important limitation of Use Cases diagrams is that their focus is on specifying only functional requirements. Non-functional requirements, such as performance, and external requirements, such as interfaces, which are fundamental in software-intensive systems, are not well-represented by Use Case diagrams.

2.3 SysML Requirements Table

SysML allows the representation of requirements, their properties and relationships in a tabular format. One proposed table shows the hierarchical tree of requirements from a master one. The fields proposed for Table 1 are the requirement’s ID, name and type. There is a table for each requirement that has child requirements related by the relationship hierarchy.

Table 1: A SysML Hierarchy Requirements table

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
</table>

3. EVALUATION OF SYSML DIAGRAMS AND TABLES USING TAM

The research presented in this paper is about Software Engineering, which is a discipline that is most necessary to develop large, complex software. This is most likely to happen in practice, with real world problems. Thus, the choice was to test the hypothesis at an industrial environment.

3.1 Industry as Laboratory
The industry as laboratory approach uses the actual industrial setting as test environment [21]. Within this approach, the research team builds an intimate relationship with an industrial product creation team, which is mutually beneficial [16]. The research team gets inspiration from real industrial challenges, and at the same time it gets a means to verify research results in industrial settings. The industrial partner gets inspiration from results, and is continuously challenged by unbiased and critical people. The testing was performed at a company specializing in developing software-intensive systems in the field of road traffic management.

In practice, the introduction of a new technology in industry is often done by a course. A common format for a course is that an expert from outside the company lecture to the employees. This saves time, as the best practices are already introduced through an experienced person.

### 3.2 Course on Model-Driven Software Engineering

Proposed course was followed by ten employees (all members of the development department), varying from developers to managers. This course had a broad content, including modules about Object-Oriented Modeling with UML, Software Architecture, and Formal Methods, and the content of the module about SysML was as follows:

1. Introduction to SysML
2. SysML Use Cases
3. SysML Requirements diagram
4. SysML Tables
5. SysML Block diagrams

This module of the course had two main objectives. The first one was to present the SysML diagrams relevant for the experiment. The emphasis was on the SysML diagrams used for RE, including extensions to the basic SysML Requirements diagram as proposed in [26]. The second objective was to discuss the advantages of combining UML and SysML diagrams for describing software architectural views.

During the module the participants had the opportunity to learn the theory combined with examples in a variety of domains. A number of exercises were proposed during the course. The purpose of these exercises was twofold: (i) employees could practice what they have just learnt, and (ii) as an evaluation of the difficulties they had with the modeling languages. The exercises were tailored as entry-level ones, with low level of complexity and considered feasible to be done individually. One example of exercise was to design a SysML requirements diagram based on a list of user requirements.

### 3.3 Putting SysML to Work

After the course, a simple project was proposed by the company to each participant, in which they had to use the knowledge gained during the course to create a design for the project. The project was suitable to be developed individually.

After the course and the use of SysML for “toy projects”, the company started to seriously use SysML as a complement to UML in a number of industrial projects. The company is currently trying to find the best way to incorporate SysML in the company’s development process.

### 3.4 Evaluation using TAM

Many models have been proposed to explain and predict the use of a system [29]. The most widely employed model of technology adoption and use is the Technology Acceptance Model (TAM) [9] [1]. The model suggests that when users are presented with a new technology, a number of factors influence their decision about how and when they will use it. It posits that individuals’ behavioral intention to use a technology is determined by two beliefs: perceived usefulness, defined as the degree to which a person believes that using the technology will enhance his or her job performance, and perceived ease of use, defined as the degree to which a person believes that using a particular system will be free of effort. A third variable is considered in this paper: perceived usage [1], defined as the degree to which the user will actually use the technology.

In this paper we are interested in the individuals acceptance process, which is a good starting point to understand how a new technology might be accepted in a community. Therefore, extensions to TAM in which the social context is taken into account [29], although useful, are not investigated here. In addition, the purpose is to evaluate initial acceptance of technology, which makes TAM a suitable model.

TAM was initially proposed to evaluate information systems tools [9] [1], but it has also been used in other contexts, such as to evaluate the adoption of the BPMN notation to teach business process at the university level [24] and the intention to continue using a process modeling technique [22].

Evaluation was performed using two techniques: surveys, based on the TAM theory, and interviews. All the participants that returned the questionnaire had followed the Model-Driven Software Engineering course and had had some practice after making a design for a simple, but realistic, project. Thus, the questionnaire was not proposed as soon as the course ended. Participants answered a questionnaire composed of 11 statements in which they made their opinions explicit. A 5-point Likert scale was proposed to measure perceived attitudes of the employees by providing a range of responses to each statement. The scale ranged from (1) strongly disagree, (2) disagree, (3) neutral, (4) agree, and (5) strongly agree. For the negative statements, it was inferred from low scores that the participants are positive about a statement in the questionnaire, while for the positive statements, a high score was inferred that the participants were positive about the statement.

The response categories in Likert scales have rank order, but the intervals between values cannot be presumed equal [11]. As a result, Likert scales fall within the ordinal level of measurement. When treated as ordinal data, Likert responses can be analyzed using non-parametric tests, such as the Mann-Whitney test, the Wilcoxon signed rank test, and the Kruskal-Wallis test [8]. Non-parametric statistical methods were used to analyze the findings because the subjects involved in the study were few and not chosen randomly from a large population. The objective was to find the representative power of the results, and not necessarily the statistical significance. It is worthwhile to note that non-parametric techniques do not solve the problem of potential dependency between the answers of the participants.
4. RESULTS

4.1 Proposed Questionnaire

A questionnaire was answered by ten employees. The answers regarding each question are shown in tables 2, 3, and 4, in which “m” represents the mean, “s” represents the standard deviation, and “pos” indicates the number of positive answers, given in modulus because of negative statements. We arbitrarily considered as positive the answers “Agree” or “Strongly Agree” (values 4 or 5). For negative sentences we considered 1 and 2 for a positive response.

The answers relative to statements 1 to 4 indicate that the SysML Requirements diagram has potential of usefulness in general. Most respondents considered that the diagram is useful to make the models more expressive and that it improves traceability between requirements and other design models, which is fundamental for designing software-intensive systems.

Even being a UML profile and sharing UML concepts, constructs and diagrams, details related specifically to SysML were not considered easy to understand and use. Moreover, the relationship and the differences between UML and SysML diagrams for RE are not clear. The results of statements 5 to 8 indicate that even for developers that are knowledgeable about UML, a specific training in SysML is highly recommendable. This is reflected in statements 9 to 11, about the usage of SysML. Although there was an interest in knowing more, there was also concern about the acceptance in real projects and the amount of time needed to learn the language.

4.2 Interviews

Individual interviews were conducted with some of the participants. The purpose was to better understand the responses and to ask for more details of the process of software development at the company. The focus was on interviewing, when possible, those employees whose answers deviate from the group. For instance, in statement 4, although SysML Requirements diagram was recognized as useful to make models more expressive, and also to facilitate traceability, still most individuals were neutral regarding the improvement of RE activities by using SysML. The main reason for this is related to the introduction of a new modeling language to a well-established modeling process. Thus, training is highly recommended.

A major concern mentioned during the interviews was related to the customer and the requirements. That confirms the many publications cited in this paper [3] [4] [25] that consider RE a critical phase in software development. For instance, according to one developer, the customer comes with very specific requirements, making it difficult to change or to adapt. In order to facilitate the implementation, developers may interpret the requirements, which is considered “dangerous” by one manager. Incorrect interpretations may lead to different implementation of certain functionalities and displease the customer.

SysML was considered easy to use and almost directly applicable, as it is an extension of UML. Another perceived advantage of SysML is the flexibility provided by the language and the possibility of creating specific profiles with important company-specific additions. Specifically about the SysML Requirements diagram and Tables, developers and managers saw potential in SysML compared to their old requirements specification method, mainly because it clearly shows relations between requirements, and between requirements and design. SysML Requirements diagrams and Tables were already applied to two projects at the company.

5. ANALYSIS OF FINDINGS

According to many authors [19] [23] [7], gap between new academic methods, techniques and processes and their application in industry is common in Systems and Software Engineering. The challenge is not only to develop better theories, but also to effectively introduce and use these theories in practice. One way to address this gap is to use industry as a testing environment. Thus, researchers learn what real world problems exist, and if the proposed solutions really contribute to solving problems.

SysML Requirements diagram may become the language of choice for requirements specification. As SysML is a language, not a methodology, it can be added without many problems into the current development process in an organization. The language can be adapted and integrated into the existing methodology and processes. In addition, SysML is based on UML, which is widely known and used, both in academia and industry. As a matter of fact, SysML is considered easily introduced to teams that are already using UML. The major concern is to find the correct relationship between UML and SysML diagrams. During the course, it became clear that the employees could understand the proposed SysML diagrams of the examples.

Finally, although training was mentioned as highly recommended, one has to keep in mind that developers are always very busy and have little time to stop their work and learn new technologies. Formal training, in which groups of employees attend classes, is often not what they prefer. In practice, the learning process is frequently done “on the job”, by interacting with more experienced employees. As a result, informal training, in which one or more employees explain not at once, but daily during the effective execution of real tasks, has more potential to be successful.

6. CONCLUSIONS

Introduction of new techniques, methods, tools or procedures is challenging in Software Engineering. Not only people are taken out of their comfort zone, but it also takes an effort to adapt legacy systems to the new approaches. As a result, risks may increase, as well as developers dissatisfaction and financial losses. In order to avoid these problems, intensive training is necessary. Common approach for training in many companies is to have a formal course to be held at a specific date and time, followed by everyone at the same time, lectured by an expert. The main issue in this case is the assumption that every person has the same ability to follow the course, which is not true in most cases.

The approach followed in this paper has a close relationship with industry, in what is often called “Industry as Laboratory”. This approach offers advantages for both academia and industry. Working in cooperation with partners from industry bring real problems and cases to researchers, and show them what Software Engineering looks like in practice. The effect is that researchers have real challenges to confirm/refute their theories. In addition, their results are stronger when tested in a real industry environment, not only with toy projects. The advantage for industry is that
Table 2: Perceived usefulness of SysML - statements 1 to 4

<table>
<thead>
<tr>
<th>Statement</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>m</th>
<th>s</th>
<th>pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - Using the SysML Requirements Diagram will make my models more expressive.</td>
<td>2</td>
<td>8</td>
<td>3.8</td>
<td></td>
<td>0.42</td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>2 - Using the SysML Requirements Diagram relationships will make it easier to trace requirements.</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>3.9</td>
<td>0.56</td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3 - Using the SysML Requirements Diagram extensions will improve the requirements engineering phase.</td>
<td>4</td>
<td>6</td>
<td>3.6</td>
<td>0.51</td>
<td></td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>4 - Using the SysML Requirements Diagram to model all types of requirement will improve the requirements engineering phase.</td>
<td>7</td>
<td>3</td>
<td>3.3</td>
<td>0.48</td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Perceived ease of use of SysML - statements 5 to 8

<table>
<thead>
<tr>
<th>Statement</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>m</th>
<th>s</th>
<th>pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 - It is easy for me to know the difference between UML and SysML Use Cases.</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>2.9</td>
<td>0.73</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>6 - It is easy for me to know the difference between each SysML Requirements diagram relationship.</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>3.1</td>
<td>0.87</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>7 - I find it easy to relate SysML Use Cases and SysML Requirements Diagrams.</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>0.47</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8 - It is easy for me to model requirements using the SysML Requirements diagram.</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3.1</td>
<td>0.87</td>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Perceived usage of SysML - statements 9 to 11

<table>
<thead>
<tr>
<th>Statement</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>m</th>
<th>s</th>
<th>pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 - I worry about the acceptance of SysML in a real project.</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2.8</td>
<td>0.78</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>10 - I worry about the amount of time needed to learn SysML.</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>2.4</td>
<td>0.7</td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>11 - I am interested in knowing more about SysML.</td>
<td>4</td>
<td>0</td>
<td>3.6</td>
<td>0.51</td>
<td></td>
<td></td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Researchers are most likely capable to think "outside the box", proposing non-conventional solutions that may have high impact on problem solving. The industrial partners get inspiration from results, and are continuously challenged by unbiased and critical people.

SysML language has been applied in many projects, as indicated by a number of publications cited in this paper. These first results are promising, but there is still a long way to go to make SysML acceptable and used as widely for Systems Engineering as UML is for Software Engineering. The challenge is even greater than was the acceptance of UML. With SysML, not only the Software Engineering community has to approve, but also many other engineering communities. As a matter of fact, the first results are promising, but additional evaluation by the research community is needed.

7. REFERENCES


T. Rozman, R. V. Horvat, and I. Rozman. Modeling the Standard Compliant Software Processes in