ABSTRACT
Several researches have been done about formal and semi-formal requirements specification in the last years. The most common approach to specify software requirements has been based on natural language and semi-formal diagrams (like UML). The lack of rigor in the requirements specification and the consequent lack of precision in the requirements definition have alarmed requirements engineering researchers. However, the requirements engineering community is not sure if the formal specification methods is the better choice to overcome the ambiguity and imprecision problems in the requirements documents. This position paper briefly presents our efforts to address and to investigate such issue. In the empirical studies conducted by our group we have used UML as representative of semi-formal requirements specification and Z as representative of formal requirements specification.

Categories and Subject Descriptors
D.2.1 [Requirements/Specifications]: Elicitation Methods and Methodologies.

General Terms
Documentation, Reliability, Experimentation.

Keywords

1. INTRODUCTION
The modeling of several aspects of software systems has become an important activity during the software development, mainly in the early phases when special attention is given to the requirements definition [1]. Requirements usually are elicited, modeled, discussed and finally specified, in such a way that it is produced a document that specify in details what the software must do [2]. This document is used as a guide for the next phases of the software development. The most of software specification documents have specifications described in natural language and cognitive notations (semi-formal specification using diagrams), this second modality has been used largely by software development community. However, the semi-formal specification with diagrams, typically ERD (Entity-Relationship Diagram), DFD (Data-Flow Diagram) and more recently UML (Unified Modeling Language) diagrams, frequently shows ambiguity and imprecision that can make mistakes during the software implementation.

Several formal methods for software specification have been proposed in the last decades, and one important goal of such methods is to offer precise notations and semantics for the software specification, in such a way that ambiguity and imprecision can be avoided, consequently decreasing the mistakes in the software implementation. However the formal methods have been used little within the software development community, compared with semi-formal methods. Some reasons for that are:

• Software developers have just a tiny contact with formal methods during their academic formation (undergraduate courses in computing);
• Formal notations are difficult to understand at the first moment, because usually they require previous mathematics knowledge;
• There are no "mature" software tools that generate effective source code for use during software implementation, keeping a gap between formal specification and program implementation.

2. EXPERIMENTAL DESIGN
Currently we are investigating the co-relations between formal and semi-formal software requirements specification techniques, as well as the benefits that can be reach using both approaches during the software requirements process [2]. As representative of semi-formal requirements specification techniques we have adopted UML diagrams [4] (specially use cases diagram and classes diagram), and as representative of formal requirements specification techniques we have adopted Z specification language [3] [5]. The sub-sections bellow briefly will comment the experimental design conducted in the context of a health plan information system.

2.1 Context
The experimental design was conducted in the real environment of a health plan institution. The main goal of the experimentation was to produce a software requirements specification for an information system called "Home Care", using both semi-formal and formal requirements specification techniques. We produced this requirements specification trying to answer two questions (among others):

1) What kinds of co-relations are possible to identify between semi-formal (UML) and formal (Z) requirements specification techniques ?
2.2 Participants
Five people participated in the experimental study: two requirements engineers (one senior and one junior), two health plan clerks, and one section supervisor of the health plan institution. The requirements engineers had basics knowledge about Z language (six months of training) and an intermediate to advanced knowledge about UML (one year and a half of training and five years of practice). About six interview sections were conducted to obtain the information about the “Home Care System”.

2.3 Materials
The materials used in the experimentation were as follows:
- Forms used in the “Home Care System”, for example: form to record medicines and clinical materials used in the patients home; form to schedule the paramedics visits in the patients home; etc.
- Forms to record the information collected during the interviews (these forms were created by the requirements engineers).
- Use cases and classes diagrams produced as semi-formal software requirements specification.
- Schemes and axioms produces in Z language as formal software requirements specification.
- Rational Rose™ software to create and edit the use cases and classes diagrams.
- Z-Eaves software to create, edit and validate the schemes in Z language.

2.3 Experimental Procedure
The software requirements specification began with a semi-formal modeling of the information around the system in analysis. Using the information obtained with the interviews the first model produced was the use cases diagram. After the users validated the initial requirements expressed by use cases diagram, an application data model was built using the UML classes diagram. Three classes diagrams were produces, one for each subsystem of the “Home Care”. After the classes diagrams were done, a validation process was conducted with the system's stakeholders. The semi-formal modeling could be understood by all stakeholders, allowing them to validate the documents produced without any problem.

The next step was to build the formal specification. This was done starting from the information captured in the semi-formal specification models (use cases diagram and classes diagrams). As commented before, the notation adopted by the formal specification was Z language. During this process the strategy adopted was: (1) to specify each class from the classes diagrams as a type in Z, using the schema notation (without the predicative part); (2) to specify each relationship between classes as functions or relations in Z; and (3) to specify each use case from the use case diagram as a schema in Z, showing in the predicative part the schema the constraints and operations necessary for the effective work of the use cases.

3. SUMMARY AND FINAL REMARKS
The experiment conducted has shown that for the early software requirements specification the semi-formal models are more appropriated than formal models because they help to abstract unnecessary details for the early phase of the requirements process, facilitating the communication among the system's stakeholders. However, for the late phase of requirements process a detailed requirements specification is necessary, and for such intention the semi-formal models are insufficient, because they don't have a well founded semantic for a precise specification. This is the point where formal specification can be very useful, the empirical study conducted supports such assertion.

The software requirements specification quality was improved with Z specification, to conduct a formal specification contributed to find mistakes and misunderstood requirements in the specification produced only with the semi-formal approach. At the end of the experiment we detected that the following quality attributes of requirements specification were improved with formal approach: precision, completeness, correctness and non-ambiguity. Integrating the semi-formal specification (UML) with the formal specification (Z) helped to produce a better requirements specification document, where the models completed each other, building a robust requirements document of what must be implemented in the software to support “Home Care System”.

The questions put in the introduction of this article was partially answered with the experiment reported here. Several co-relations were identified between UML and Z, but of course not in a exhaustive way. The mapping between UML and Z is steel under investigation.

REFERENCES