# SET, a CASE tool to guide the creation of domain and use case models in an introductory Software Engineering course

Francisco J. García Peñalvo [(1)], Sergio Bravo Martín [(1)], Miguel A. Conde González [(1)], Héctor Gonzalo Barbosa León [(2)]

[(1)] Universidad de Salamanca, Facultad de Ciencias
Plaza de los Caídos s/n, 37008 Salamanca (España)
{ser, fgarcia, mconde}@usal.es
[(2)] Universidad de Colima
Av. Universidad #333. Colonia Las Víboras, 28040 Colima (México)
barbosah@usal.es

## Abstract

The idea of using CASE (Computer Aided Software Engineering) tools in education as an interactive learning has been emerging for several topics in Computer Science. The learning process proves to be more effective, rapid and even persistent. This paper presents a CASE tool named Software Engineering Tutor (SET), which main aim is to improve the students' knowledge in Software Engineering field, specifically to guide them in the creation of domain and use case models. Besides, this tool offers a repository of case studies, trying to make an effort to share experiences around the university and professional community. Our experience with this tool during the 2008-2009 academic year, in an introductory Software Engineering course, shows that SET is a useful tool for teachers, in their learning approach, and very instructive in the assimilation of knowledge for students. Moreover, it has become a key element for the continuous assessment process support that was introduced in 2005-06 course.

## Keywords

Software Engineering Tutor, CASE tool, interactive learning, problem-based learning, continuous assessment, case study, computer science education

## 1. Introduction

Software Engineering, which traditionally has appeared like a discipline of Computer Science, is being considered in recent years as an entity separate curriculum, but with deep roots in the Computer Science and Mathematics. It offers methods or techniques for developing and maintaining quality software that solve all kind of problems.

There are many CASE tools that provide support to Software Engineering processes and contribute greatly to increase productivity in software development reducing the cost in terms of time and money. These tools are applied in all aspects of the lifecycle of software development such as planning, analysis, design, project documentation (textual and graphical), automatic code generation, error detection, and so on.

This paper presents a new CASE tool named Software Engineering Tutor (from now on SET), designed to provide an interactive learning support [1] for the early stages of the software lifecycle to new students of disciplines related to Software Engineering. It particularly focuses attention in requirement engineering and analysis stages, with the facility to create use cases and domain models respectively. But the elements that differentiate SET from other case tools are the innovative learning approach and the self-training for software engineer, due to an intelligent modeling wizard that guides the user step by step in building models. SET has been introduced as

an optional support tool for the practical part, and particularly in the problem-based learning workshops planned in the Software Engineering subject of the Diploma in Informatics at the University of Salamanca.

The article is organized as follows: Section 2 reviews the context of the subject and how SET fits into it. Section 3 describes functional characteristics of SET, and particularly the modeling wizard by a step-by-step detailed case study. Section 4 presents the results from the introduction of SET in the learning methodology. Finally, Section 5 presents the conclusions of this case experience.

## 2. Software Engineering subject: context

A set of disciplines related to the profession of the **Software Engineering Body of Knowledge** (SWEBOK, a product of the Software Engineering Coordinating Committee sponsored by the IEEE Computer Society) [2], and in 2004 the public curriculum was published (Computing Curriculum - Software Engineering) [3] by the joint action between IEEE-CS and ACM, which remains one of the five professional profiles in the Computing Curricula 2005 [4], along with the profiles of Computer Engineering, Computer Science, Information Systems and Information Technology.

Computer Science at the University of Salamanca can be studied at two levels. There is a three-year Diploma in Informatics that is possible to continue with a two years more in order to achieve the Superior Engineering Degree in Informatics. In this context, the students have their first contact with Software Engineering at the final year of the three-year degree in the current curriculum dates from 1997 [5].

### 2.1 General concepts of the subject

Software Engineering has great importance in the Informatics Engineering curriculum because it is essential to have a strong knowledge about developing software in all the stages of its lifecycle. In order to understand the different topics is necessary that students will be capable of performing a job for reflection, assimilation and practice of different underlying topics of the program subject. Encouraging these disciplines and considering that in most cases students apply every effort only to pass, it becomes necessary to apply a methodology for continuous assessment [6].

The course comprises 60 hours and tries to focus attention on the following topics:

- Lifecycle and process requirement elicitation and documentation.
- Analysis and design methods by the object-oriented paradigm.
- Modularity, software architecture, and software reuse principles.

The presentation of these topics emphasizes abstraction as the fundamental technique for understanding and solving problems [7]. This is the main problem that students find in this course. Obviously they have no experience and there is not a scientific method to complete the early stages of the lifecycle of software development. Moreover, the problem is accentuated in those Diplomas of Computer Science in which the object-oriented paradigm is taught in early years.

The traditional way to perform the practical part and assimilate the knowledge required by this subject is developing a small engineering project [8]. However, it is necessary to supplement the conceptual and practical parts with a series of problem-based learning workshops [9].

### 2.2 Planning the workshops

The problem-based learning workshop teaching approach is a practical method in which the relationship between the teacher and the students is very close. In this subject, every session is

devoted to solving a modeling problem using a concrete modeling technique, followed by discussion of a proposed solution in a debate moderated by the teacher.

Currently, the workshop planning is organized into two main groups. The first one is devoted to the conceptual models and includes three workshop sessions: one devoted to the **entity-relationship model**, in order to review conceptual data modeling principles, and two workshops to introduce the **object-oriented analysis** using UML (Unified Modeling Language) [10] in order to build conceptual class diagrams. The second group is devoted to introduce the basic principles of **use case diagrams** like a first approximation to requirements engineering.

At this point it is worth mentioning how SET offers support to new students in the stages of requirements engineering and object-oriented analysis, particularly in the construction of domain and use case models covered in the 2nd, 3rd and 4th workshops planned.

The main characteristics of this teaching approach are: modeling software systems with object-oriented techniques, explaining in public the most common mistakes in the supported modeling techniques, interactive learning based on CASE tools to create models and reports, encouraging teamwork and improving the students' communication skills.

### 2.3 Introducing SET in the learning and assessment process

In order to apply a methodology for continuous assessment in the subject [6], the teachers need agile and effective mechanisms to obtain the learning degree of students over the course.

For the past two years, the subject of Software Engineering has added SET, like a new supporting tool in the learning and assessment process and specially to support the students on workshops.

The use of SET on the subject offers the following advantages:

- Supporting the teacher to present different topics about object modeling techniques.
- Guiding the student (or software engineer) in the use case and domain models creation (a conceptual class diagram by UML like notation language).
- Normalizing the practical workshop and voluntary exercise reports.

## 3. Functional description of SET

This section presents the key features of SET (ver. 1.1.0):

- A modeling wizard to support the construction of use case and domain models.
- A standard mechanism to make personalized and normalized reports.
- A central repository of case studies.
- Compatibility with other case tools and desktop applications.
- A user interface based on different views of the model under construction.

### 3.1. The modeling wizard

The original idea of Software Engineering Tutor is mainly to introduce the student (or software engineer) at the requirements engineering and object-oriented analysis stages by the use case and domain models construction. Both models are built graphically using UML diagrams. However, its contribution in the real world of CASE tools lies in its orientation on training and instruction in building such models using an integrated modeling wizard. This wizard makes SET an original tool thanks to a modeling method based on an intelligent tutor that supports the heuristics referenced

at the main bibliography. This guided construction mechanism is the difference between SET and other tools presented from past to present subject editions, like Rational Rose and Visual Paradigm from commercial use, ArgoUML as one of the most representative open source tool, and even regarding the academic community development it is also worth mentioning Left CASE [12] and REM (REquirements Management) [13].

Now a case study is presented to clarify the power of this modeling wizard. This **case study** is compiled from Borland's UML Tutorial published on the official website of the Object Management Group (OMG) and **presents a class diagram to model a customer order from a retail catalog**. The following figures show screenshots of the six steps needed to create domain models using the recommendations of Larman: identification of conceptual classes, associations, attributes, superclasses and subclasses, whole-part relationships and packages [14].

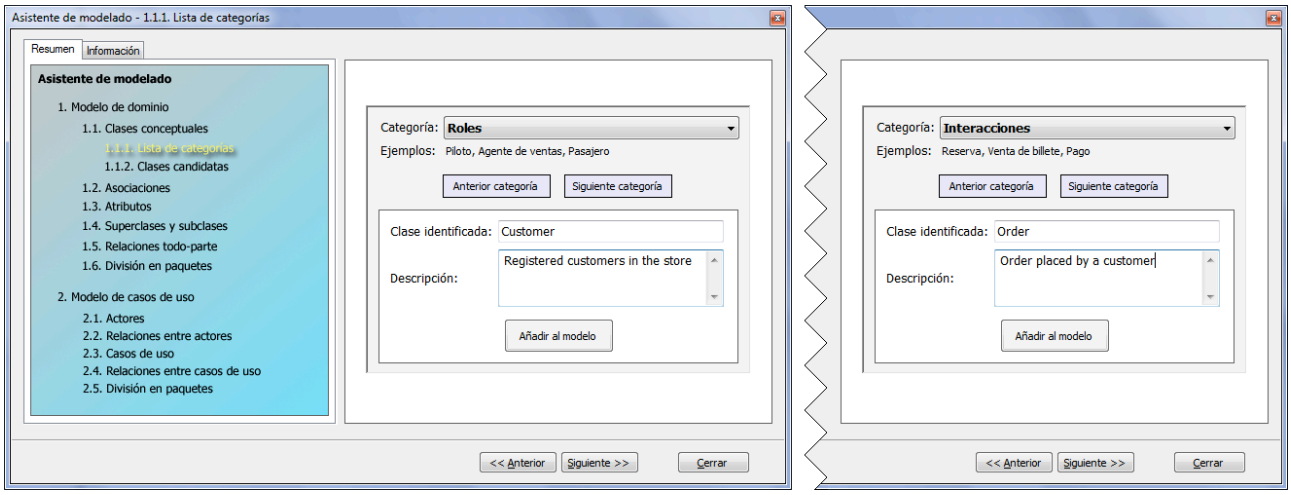


Fig. 1. Identification of the conceptual classes: Customer (left) and Order (right)

The modeling wizard is a dialog box that allows the user to navigate freely through a set of steps that guides the building models process. The wizard starts with the identification of conceptual classes using a strategy based on list of categories available on the subject bibliography [15, 16]. For example, a conceptual class **Customer** fits on *Roles* category and the **Order** on *Interactions* (see Fig. 1).

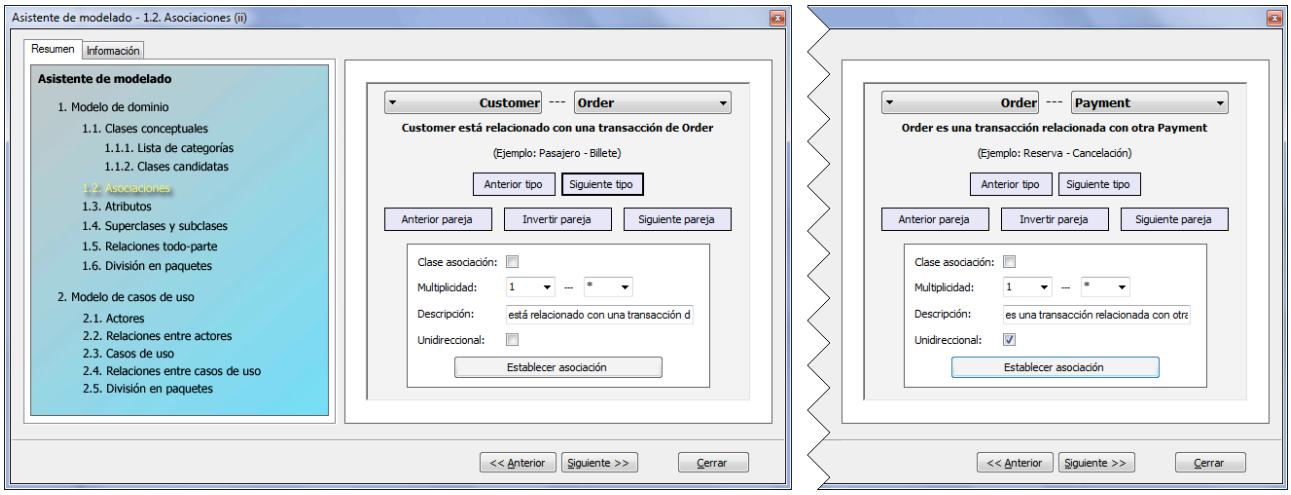


Fig. 2. Identification of associations between conceptual classes: Customer is associated with an Order transaction (left) and Order is a transaction involving a Payment (right)

The conclusion after identifying most relevant conceptual classes is that central class is **Order.** The other classes, **Customer** (that makes the purchase) and **Payment**, are associated with **Order**.

The second step of the wizard is the **identification of associations**. At this point the wizard uses grammatical constructions combining pairs of conceptual classes identified above (see Fig. 2) [14].

Next step of the wizard is shown in Figure 3. It presents the **identification of attributes** passing throw all the conceptual classes. For each attribute is assigned a data type and a visibility.

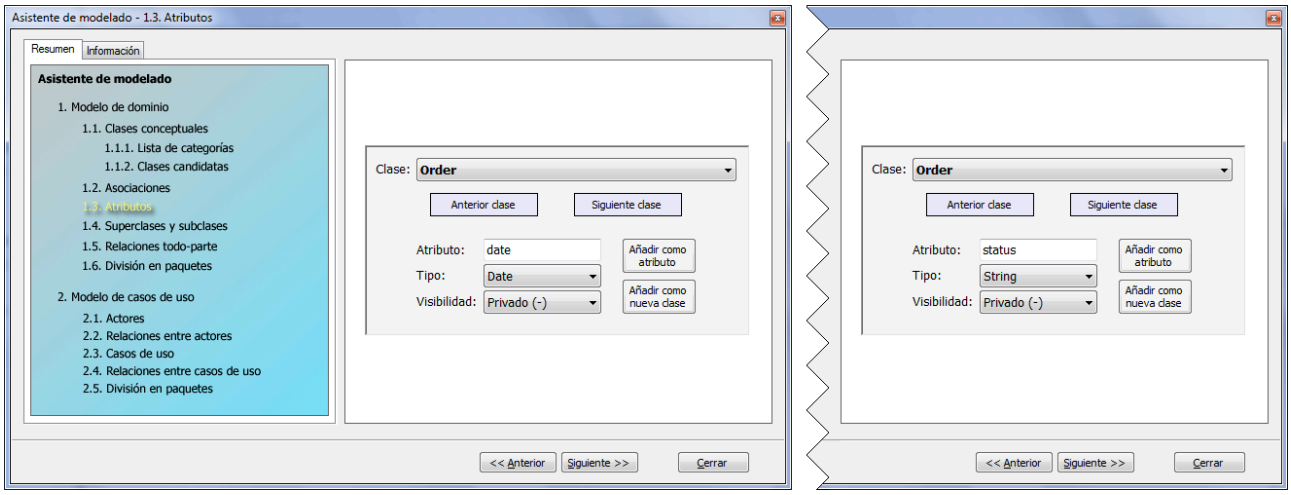


Fig. 3. Identification of the Order class attributes: date (left) and status (right)

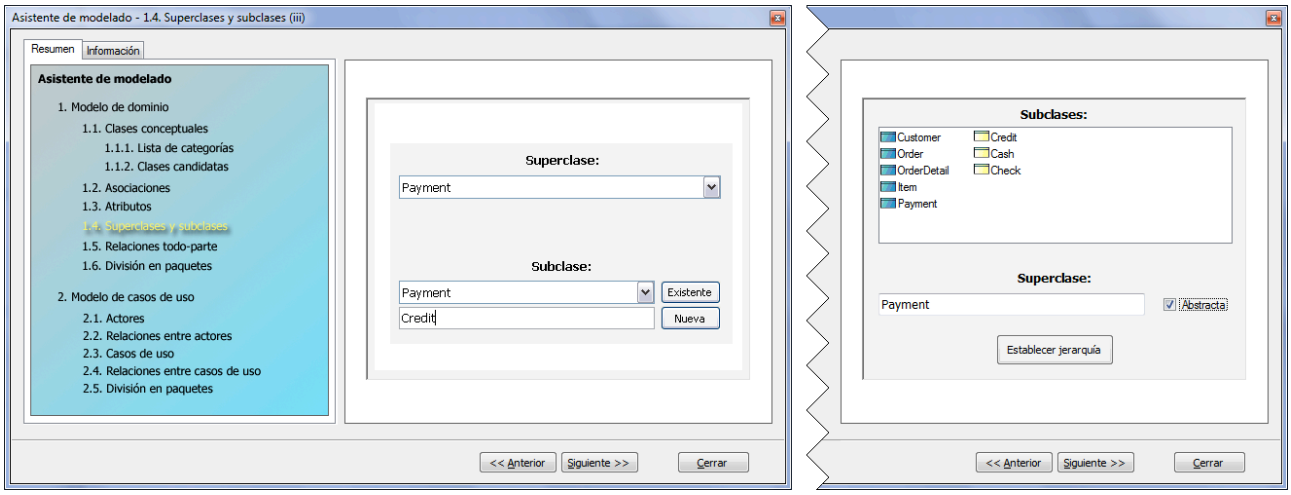


Fig. 4. Identification of superclasses & subclasses: three new subclasses: Credit, Cash and Check (left) and a class hierarchy with abstract Payment superclass (right)

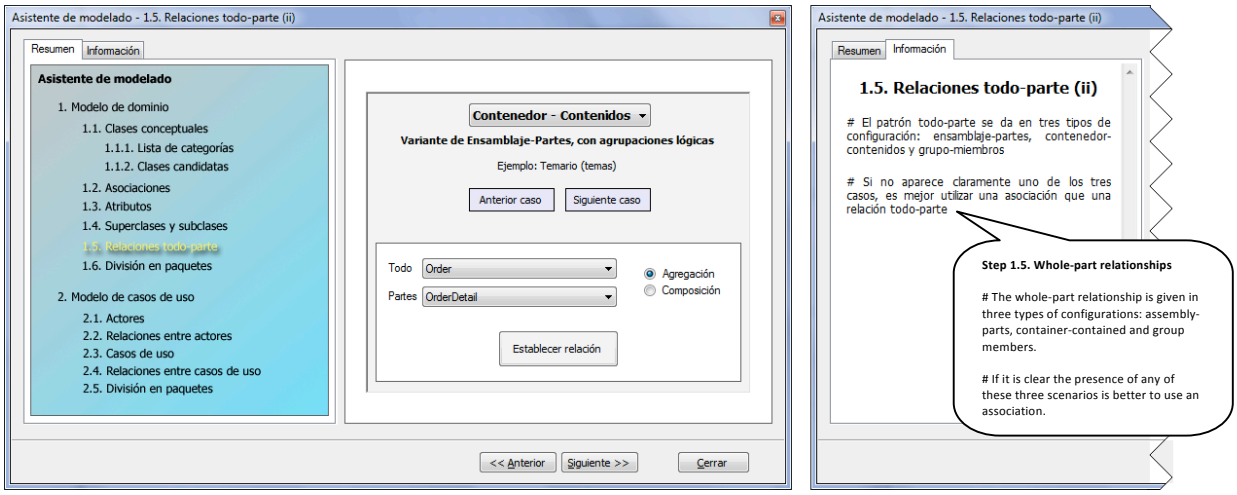


Fig. 5. Identification of a whole-part relationship between Order and OrderDetail (left) and the context-sensitive help (right)

The fourth step of the wizard is the **identification of superclasses and subclasses**, generalization and specialization relationships (see Fig. 4). At this point the conceptual class **Payment** fits well because a **Payment** (superclass) is one of three kinds: **Cash**, **Check**, or **Credit** (subclasses).

At fifth step, the wizard presents the **identification of whole-part relationships**. This pattern uses different configurations. In the case study, the **Order** contains **OrderDetails** (line items), each with its associated **Item**. Figure 5 shows this association in which one class belongs to a collection.

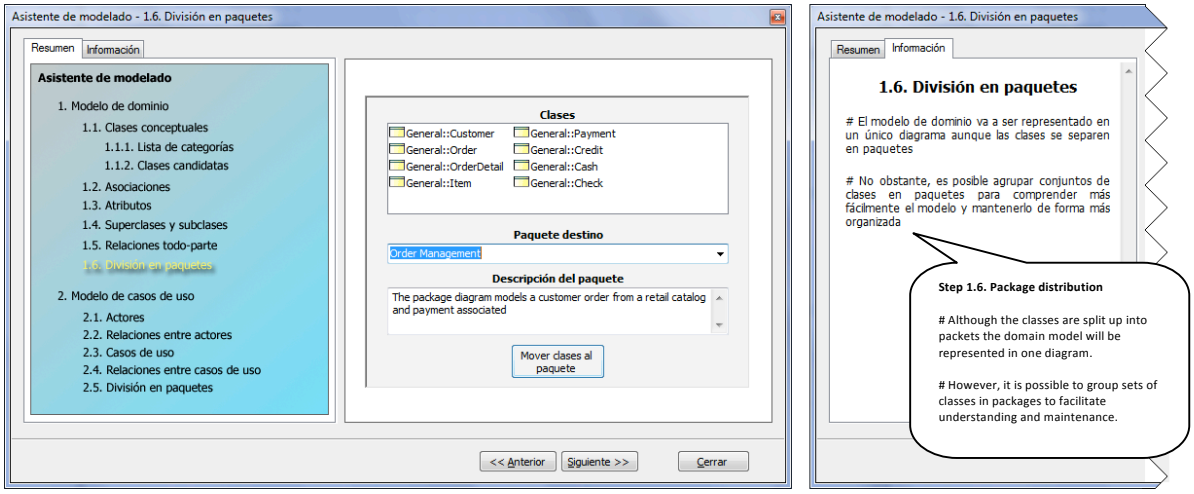


Fig. 6. Distribution of conceptual classes of case study in a package named Order Management

Finally, the wizard ends with one step to **distribute conceptual classes in packages** (see Order Management package in Fig. 6). All steps may extend the information on the current step with detailed instructions (context-sensitive help) as it shows in Figures 5 and 6.

Usually it will be necessary to iterate the wizard more than once. For this reason, it can be invoked at any time from the workspace and advanced to the step as necessary. Once finished with the wizard is likely that the resulting class diagram requires a repositioning of objects in the drawing area in order to facilitate understanding (see Fig. 7).

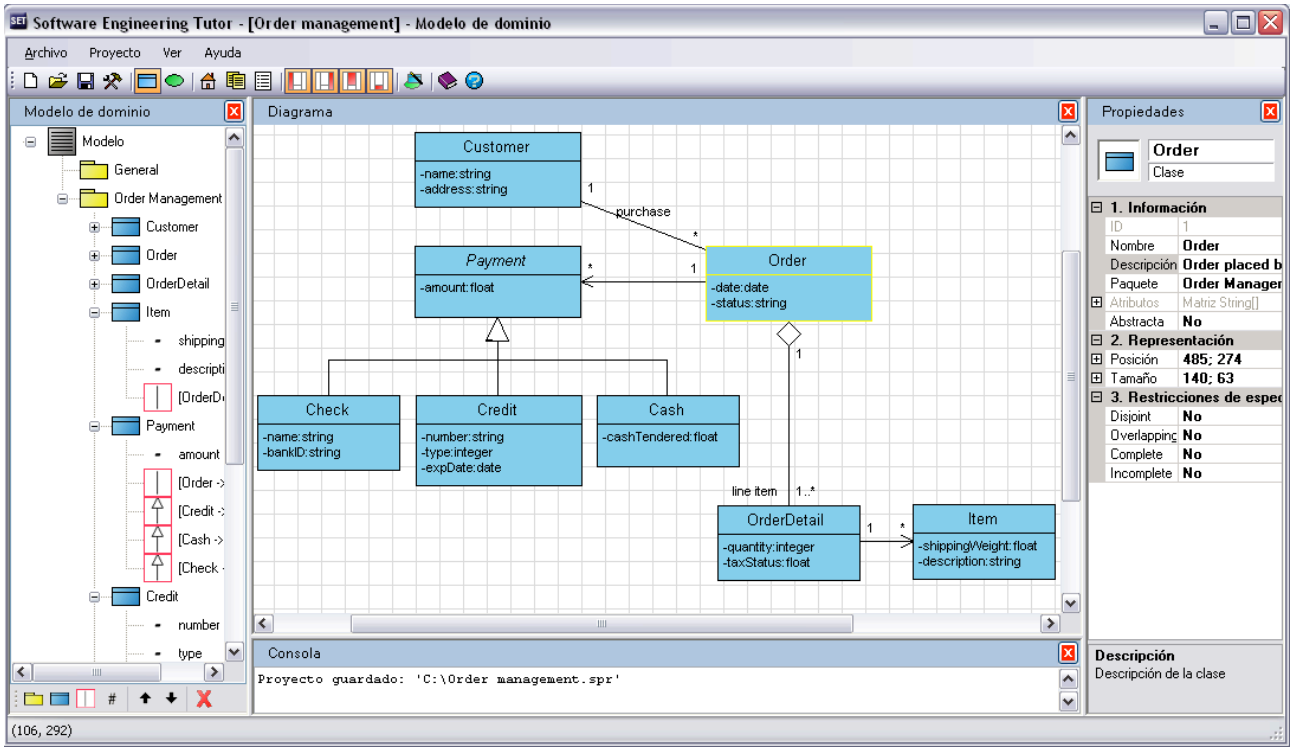


Fig. 7. SET workspace with “Order management” case study opened in

### 3.2. Normalized reports

The tool allows to the user printing reports of a project (previously saved) and exporting them to Portable Document Format (PDF). In addition, it is able to create own report types. This functionality offers to the teachers the possibility to normalize the reports at the practical workshops and even in the volunteer exercises delivered. In this way, the evaluation of the reports is more agile and at the same time is possible to ensure the originality of the work and, for example, the detection of plagiarism.

### 3.3. Central repository

Any tool that provides a teaching approach requires a basic knowledge and useful learning support. In our case, SET provides a central repository with a set of standard templates and solutions to well-known case studies where every registered user can download directly from SET without leaving the workspace.

In order to generalize the use of the tool, the Computer Science Department of the University of Salamanca has created a website on Internet has created a website for evaluation of the tool by the university community (http://set.usal.es).

### 3.4. Compatibility

SET can be exported according to the XMI standard (XML Metadata Interchange) for exchange diagrams [17]. In this way, the tool does not become a close application, but it is supplemented with other CASE tools that provide support in other stages in the software development lifecycle. It could be said that, although this tool provides the user on everything needed for the construction of diagrams related to the domain model and use cases, the main purpose of SET is not to draw diagrams, but tutoring and assisting in the creation of these models.

### 3.5. User interface based on different views

The workspace of the case tool is divided into four views that provide different perspectives of the model under construction (see Fig. 7): model (elements model in a tree view on the left), diagram (current model in diagrammatic form in the center), properties (properties of the selected element of the current model on the right) and a console view (text read-only with a history of all relevant actions from the current model at the bottom).

## 4. Results

During the academic year 2008-09 we have collected numerical data from the assessment process in order to analyze the introduction of SET in the learning approach. These empirical data are information control about the activity from the course workshops and serve to test the operation of the case tool. In addition, the data report provides information on the degree of acceptance by students in the process of assimilation of knowledge by SET.

The numerical data are collected from the second and third workshops of modeling classes and the last workshop centered on requirements engineering because of the purpose and context of SET. The process indicators collected from the workshops activity are the following:

- Number of students who participate, deliver the report and assist to the workshop.
- Number of students who use SET to complete each workshop report.

These indicators are intended to show a degree of expectation, excitement and attendance at the workshops, as well as the degree of acceptance of the tool between the students registered in this introductory course of Software Engineering.

The Table 1 shows the absolute numbers of participating students in workshops that use (and do

not use) SET as supporting tool in each workshop.

Table 1. Comparative in absolute numbers of participating students in the Software Engineering workshops using (and do not using) SET during academic year 2008-09

| | Class modeling workshop (I) | Class modeling workshop (II) | Use case modeling workshop |
|---|---|---|---|
| **Participating students do not use SET** | 38 | 24 | 26 |
| **Participating students use SET** | 116 | 123 | 103 |
| **Students do not participate** | 29 | 36 | 54 |

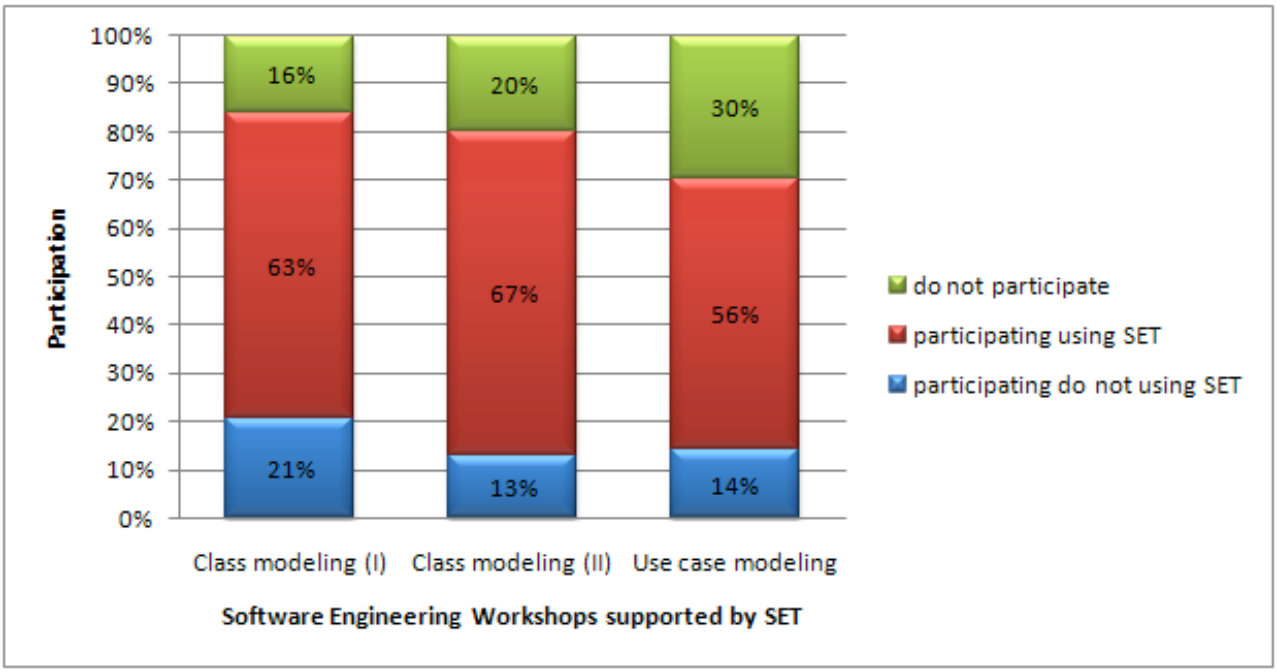


Fig. 8. Participation percentages in Workshops using (and do not using) SET over the total number of students registered in the subject

Figure 8 presents the same numerical data collected by percent rates and shows very high usage rates of SET in the Software Engineering Workshops despite the alternative use of SET. It is significant how the number of students that do not participate in the workshops increases when the course progresses. This is because the results of the exams of theoretical concepts scheduled in the continuous assessment implemented on the subject. A significant number of students that fail a continuous assessment exam do not participate actively on next workshops.

If we consider only the students who participate in workshops the statistical study is obviously much more satisfactory as shows the Figure 9. In this case, the figure shows that the average of students who use SET as the supporting tool in the workshops has significantly increased and also is distanced from those who do not use SET.

Moreover, after further analysis of the resulting statistics, we can conclude that approximately 83% of students who participate in the continuous assessment of Software Engineering course have used SET in at least one of the three workshops planned for this.

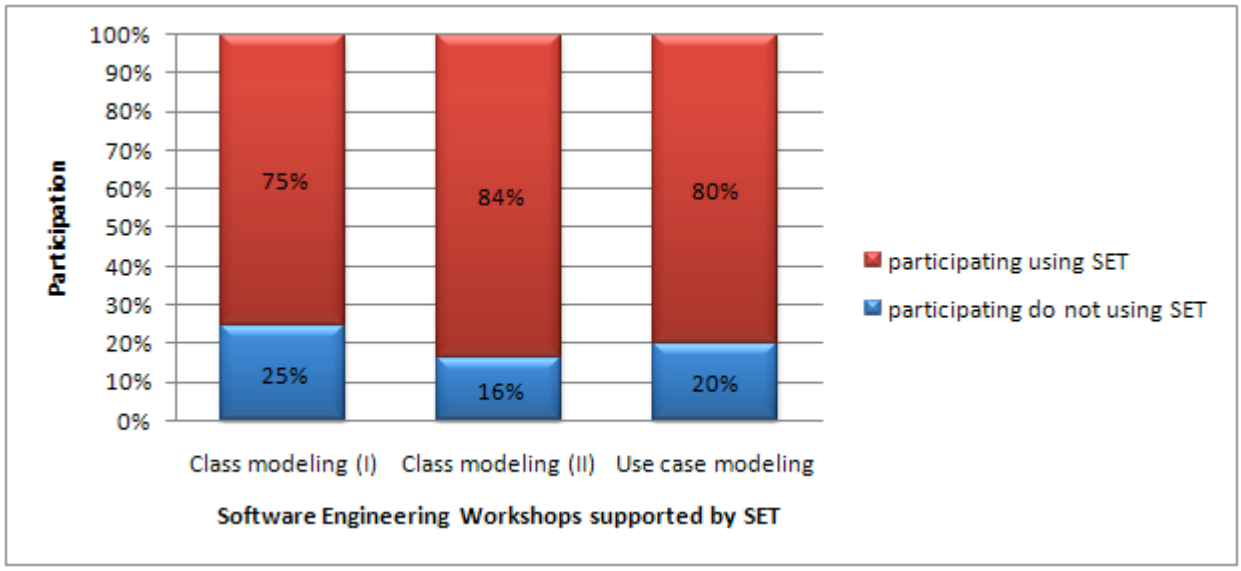


Fig. 9. Participation percentages in Workshops using (and do not using) SET

Undoubtedly, this percentage does not really matter if it does not improve the number of students that pass the course, motivated by the use of SET in the topics related. In order to this approach we collect a set of performance indicators that show the impact of SET about three key points of the evaluation process:

- The results of the exams scheduled by the continuous assessment (number of students that pass the tests of the continuous assessment using SET, and not).
- The results of the first part of the official assessment (number of students that pass the test of the official assessment using SET, and not).
- The results of the question related to using the SET at the second part of the official assessment (number of students that pass the diagram class question using SET, and not).
- Students who pass the subject (using SET, and not)

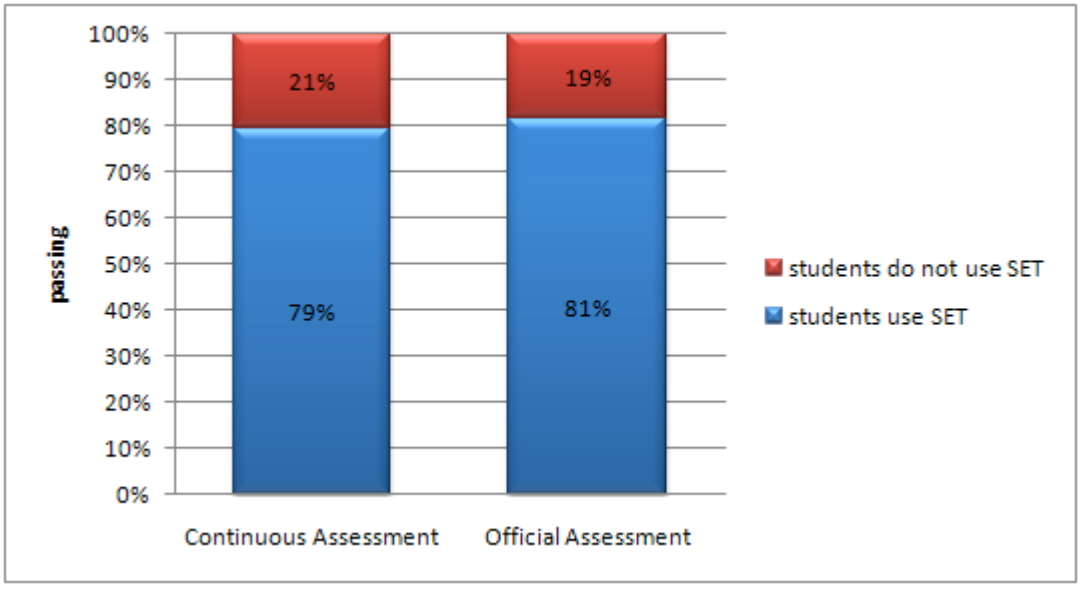


Fig. 10. Students who pass the first part of the theory evaluation

These indicators help us to measure the effectiveness of the introduction of SET in the learning methodology. The set of collected data related to the first part of the theory evaluation is shown in percent rates in Figure 10. This figure shows a remarkably influence from SET in students that

pass the first part of the theory evaluation both by continuous and official assessment. Strangely enough the rates in both types of evaluation are identical. Approximately 80% of students that pass the first part of the theory evaluation have used SET in at least one workshop versus 20% that do not use it.

The Figure 11a referrers to results collected only of one question from the Part II of the theory evaluation. This question is related to a fundamental scope of SET, the construction of conceptual diagram classes or domain models. This is one of the key points where the use of SET should be noted. In fact, Figure 11a shows the 85% of students passing this question have used SET.

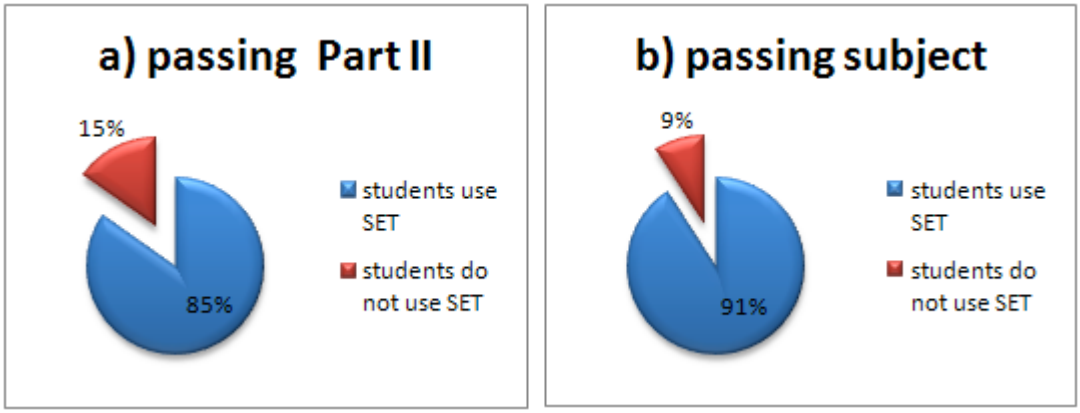


Fig. 11. (a) Students who pass a concrete question from the second part of the theory evaluation, (b) Students who passed the Software Engineering subject in academic year 2008-09

More directly, Figure 11b shows the passing students separating those who have used SET and not. The results are very conclusive. Almost all students who use SET pass the course.

In order to obtain an effective indicator about the execution of the activity we have compared the results between academic year 2007-08 and 2008-09. Figure 12 shows the results of this comparative presented in two parts:

- Students that pass the **test from the Part I** of the theory evaluation.
- Students that pass the **question from the Part II** usually related to the creation of a conceptual class diagram (one scope of SET).

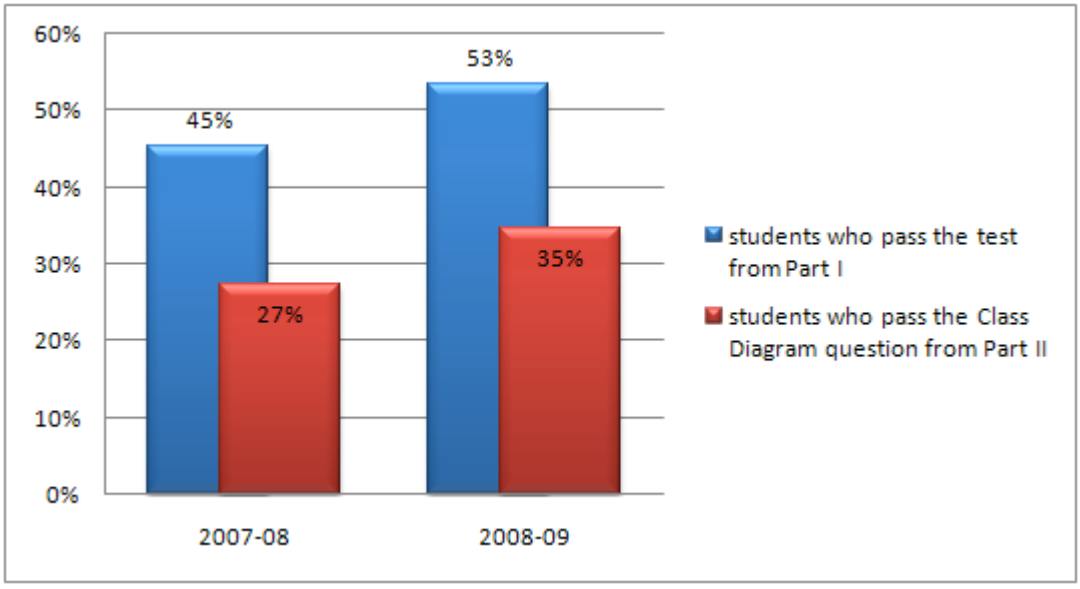


Fig. 12. Comparative of results between academic year 2007-08 and 2008-09

Figure 12 shows a substantial increase in academic year 2008-09 in both parts where using SET

makes a crucial difference. These parts are, at first the test from the Part I of the theory evaluation and second, the question from the Part II related to the creation of a class diagram.

**5. Conclusions**

The use of interactive learning has been emerging for several topics in Informatics and particularly in the field of software engineering using CASE tools. SET aims to be the germ of a new type of CASE tools for the training of future software engineers. The learning process is marked by a complete and proven wizard that guides the construction of domain and use cases models.

There is no doubt that using SET as a supporting tool in the workshops provides many guarantees for students to pass partial, and even totally, the subject by either of two available assessment methods (official and continuous assessment). These guarantees are based on the weight of theoretical and practical concepts supported by SET in the evaluation criteria and in particular those related to the object-oriented paradigm, main purpose of SET.

One of the benefits on the assessment process after incorporating the tool on practical workshops on the Software Engineering subject has been the unification of all delivered documents by the functionality based on the automatic reports generation. Also, the initiative of the central repository of case studies gives to students a complete catalogue of resources to enhance knowledge and it expands the possibilities of using the tool and even sharing case studies made by other members of the academic or professional community. Moreover, due to the distributed nature of this case tool, we find the possibility of working with the client application in offline mode and with other case tools or desktop applications because of the compatibility with standards like UML and XMI.

Despite the success results obtained with the introduction of SET during the academic year 2008-09, we consider that SET must be improved in order to cover the rest topics planned in the subject. Overall, we feel very motivated with the introduction of the tool in the learning methodology, it has been a very rewarding experience.

## Acknowledgment

This work is partially supported by the Regional Ministry of Education of Junta de Castilla y León through the project GR47.

## References

1. Leen-Kiat Soh (2006). *Incorporating an intelligent tutoring system into CS1*, In Proceedings of the 37th SIGCSE technical symposium on Computer science education - SIGCSE'06, pp 486 – 490.
2. Abran, A. (Co-Executive Editor), Moore, J. W. (Co-Executive Editor), Bourque, P. (Editor), Dupuis, R. (Editor), Tripp, L. L. (Chair). *Guide to the Software Engineering Body of Knowledge*: 2004 Edition - SWEBOK. IEEE-CS Press, 2005.
3. The Joint Task Force on Computing Curricula: IEEE Computer Society and Association for Computing Machinery. Software Engineering 2004. *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. A Volume of the Computing Curricula Series. http://sites.computer.org/ccse/SE2004Volume.pdf. August 23, 2004.
4. The Joint Task Force for Computing Curricula 2005. *The Overview Report covering undergraduate degree programs in Computer Engineering, Computer Science, Information Systems, Information Technology, and Software Engineering*. A volume of the Computing

Curricula Series. A cooperative project of The ACM, AIS and IEEE-CS.

5. Boletín Oficial del Estado de 4 de noviembre de 1997 (BOE núm. 264); *Resolución de 15 de Octubre, de la Universidad de Salamanca, por la que se publica el Plan de Estudios de Ingeniero Técnico en Informática de Sistemas*.
6. García, F.J., Conde, M.A., Bravo, S. (2008). *Continuous Assessment in Software Engineering, In Proceedings of the 1st Workshop on Methods and Cases in Computing Education – MCCE'08, pp 41-45.*
7. Wasserman, A. (1996). *Toward a discipline of software engineering*. IEEE Software, Vol. 13.
8. Chamillard, A.T. , Braun, K.A. (2002). *The software engineering capstone: structure and tradeoffs, In Proceedings of the 33rd SIGCSE Technical Symposium of Computer Science Education*—SIGCSE'02, pp 227–231.
9. García, F.J. and Moreno, M.N. (2004). *Software Modeling Techniques for a First Course in Software Engineering: A Workshop-Based Approach*. IEEE Transactions on Education, Vol. 47, No. 2.
10. OMG (2004). *Unified Modeling Language: Superstructure. Version 2.0.* Object Management Group Inc. Document formal/05-07-04 [on line]. Available on: http://www.omg.org/cgi-bin/doc?formal/05-07-04
11. García, F. J., Álvarez, I. (2003). *Left CASE – A Free Software Component-based CASE Tool for Software Engineering Practice Support*. In Proceedings of the 33rd ASEE/IEEE Frontiers in Education Conference. IEEE-CS Press.
12. Durán, A. (2000). *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información*. PhD Disertation. University of Seville.
13. Larman, C. (2002). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*, 2nd Edition, Prentice Hall.
14. Coad, P., Yourdon, E. (1990). *OOA Object-Oriented Analysis*, Prentice-Hall.
15. Shlaer, S., Mellor S. J. (1988). *Object-Oriented Analysis: Modeling the World in Data*. Yourdon Press.
16. OMG (2007). *MOF 2.0/XMI Mapping, v2.1.1.* Object Management Group Inc. Document formal/2007-12-01 [on line]. Available on: http://www.omg.org/docs/formal/07-12-01.pdf.