# An empirical comparison of SaveUML and SaveCCM technologies

**Version 1.0**

Mälardalen University, Department of Computer Science and Engineering

Ana Petričić, Luka Lednicki, Ivica Crnković

March, 2009

# 1. Introduction

## 1.1 Purpose of this document

The purpose of this document is to describe the conduction and results of an experimental evaluation of two different modelling approaches SaveUML and SaveCCM.

## 1.2 Document organization

The document is organized as follows:

- Section 1, *Introduction,* describes contents of this guide, used documentation during developing process etc.
- Section 2, *The modelling approaches*, provides an brief introduction into the respective approaches, SaveCCM modelling and SaveUML modelling
- Section 3, describes experiment organization and objectives
- Section 4, *Temperature regulation system* holds the specification of a simple temperature regulation system that was used in the training phase of the experiment.
- Section 5, *Autonomous truck navigation system*, holds the specification of an autonomous truck navigation system that was used in the modelling phase of the experiment to evaluate the modelling capabilities and feasibility of the two approaches
- Section 6, Questionnaire, presents the contents of an questionnaire that was given to the students to inquire their user experience and to get feedback from the participants of the experiment. Also this section contains the answers given by the participants.

## 1.3 Definitions and acronyms

### *1.3.1 Definitions*

| **Keyword** | **Definitions** |
|---|---|
| **SaveUML** | An modelling approach for modelling embedded systems using an specialized UML profile which represents the SaveCCM language |
| **CASE tool** | Computer Aided Software Engineering tool |

### *1.3.2 Acronyms and abbreviations*

| **Acronym or abbreviation** | **Definitions** |
|---|---|
| **SaveCCM** | SaveComp Component Model. |
| **UML** | Unified Modeling Language |
| **OCL** | Object Constraint Language |
| **DSL** | Domain Specific Language |
| **RSM** | Rational Software Modeller – an IBM toll for UML modelling |
| **TRS** | Temperature regulation system |
| **ATN** | Autonomous truck navigation |
| **SaveIDE** | An Eclipse-based modelling tool for developing systems by using SaveCCM language |

## 1.4 References

[1] Åkerholm, M., Carlson, J., Håkansson, J., Hansson, H., Nolin, M., Nolte, T., and Pettersson, P. 2007. The SaveCCM Language Reference Manual. MRTC report ISSN 1404-3041 ISRN MDH-MRTC-207/2007-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, January, 2007

[2] Sentilles, S., Pettersson, A., Nyström, D., Nolte, T., Pettersson, P., and Crnkovic, I., 2009. Save-IDE - A Tool for Design, Analysis and Implementation of Component-Based Embedded Systems,

Proceedings of the Research Demo Track of the 31st International Conference on Software Engineering (ICSE), Vancouver, May, 2009

## The modelling approaches

In the experiment described in this report we aim to compare two different approaches of modelling safety-critical embedded systems: using a domain-specific language and using a general purpose language. For domain-specific modelling we will use custom tools and technologies built for SaveCCM. General purpose will be handled using our SaveUML technology.

### 1.5 SaveCCM modelling

As the domain-specific tool in this experiment we will use the SaveIDE. SaveIDE is a research tool built on the Eclipse framework. It comprised of a tool for designing SaveCCM models, and several tools for analysis of SaveCCM models. Modelling is done using graphical editors that reflect the SaveCCM graphical notation.
The drawbacks of this approach are that it calls for all stakeholders to be acquainted with the domain-specific language, it becomes hard to connect the domain-specific models with the more generic models of the system, and custom tools have to be developed.

### 1.6 SaveUML modelling

The SaveUML approach tries to connect domain-specific modelling with a general purpose modelling language. The aim of SaveUML is to provide the ability for modelling SaveCCM systems using UML 2.0 and already existing tools that support UML. Therefore, in SaveUML we have provided a formal way for developing models that conform to SaveCCM. We achieved this by defining a UML profile, a common UML extension mechanism. Our SaveUML profile brings the SaveCCM semantics to UML and limits the use of UML elements to a subset that satisfies our modelling needs. By using this profile (e.g. applying stereotypes or tagged values to UML elements), a developer can design SaveCCM models using UML in any UML tool that supports UML 2.0 and UML profiles.
Modeling of the SaveCCM components and their interfaces, and connections between them is supported by a total of 21 UML stereotypes that can be applied to standard UML elements. By using these stereotypes a developer can distinguish between all the modelling elements SaveCCM provides.
As already stated, a special part of the SaveUML profile is a set of 117 OCL constraints that enforce SaveCCM semantics on UML. These constraints can be validated in two ways, by using either live validation or batch validation. Constraints with live validation are checked every time an element, to which a stereotype is applied, is modified. E.g. a constraint that suppresses using any connectors directly on a component (in SaveCCM no connections except the ones explicitly captured by the ports are allowed). If a constraint with live validation is violated an immediate notification arises. Constraints with batch validation are checked when the user runs a validation action. An example is a constraint which requires that all ports owned by a component have an appropriate stereotype applied. To meet this demand, it is necessary to take a two-step process, and therefore this constraint can not have live validation (as the constraint would be violated after the first step). This distinction of constraint categories and their handling facilitates model validation and allows the validation to be achieved already during design time.

## 2. Experiment objectives and organization

### 2.1 Experiment objectives

In cases when usability of an UML profile is satisfying, and the expressiveness of UML extensibility mechanisms is sufficient for particular domain, then the need for building specialized tools is questionable. In these cases, designing an UML profile and using some of existing UML tools could replace a custom built tool. A number of UML CASE tools exist that provide a user friendly interface and functionality which is a behalf on using UML, in our case on using the SaveUML technology instead of genuine SaveCCM in the SaveIDE modelling environment.

There are a few significant advantages that support this approach:

- Using of already existing UML tools, which reduces time and cost spent on developing a domain-specific modelling environment.

- Any knowledge of and experience with standard UML is directly applicable. This is very useful for users accustomed to UML, or if UML is a standard modelling language within a company, so there is no need to switch to SaveCCM.
- The UML profile is compatible with standard UML, thus any tool that supports UML can be used for manipulating models based on a UML profile. This brings the feature of portability to models designed using SaveUML profile among many CASE tools. Contrary to UML models with SaveCCM semantics, SaveCCM models created with the SaveIDE tool can only be managed by SaveIDE.

However, the approach has some drawbacks:

- Having in mind that UML profiles are an upgrade to basic UML, this can lead to an overly complicated model within an already complex UML specification and the modeller might get confused with extraneous UML semantics or modelling elements.
- Using standard UML notation, in which an existing shape corresponding to SaveCCM element is reused, could compromise the readability and clarity of the diagrams.
- A DSL (such as SaveCCM) is usually not used independently, but in combination with other tools, models and DSLs (e.g. resource and time analysis tools in SaveCCM). In such a case the problem is not only expressing one DSL by UML extensions, but about a set of DSLs that should be mapped.

Our aim is to empirically evaluate our approach, with respect to development efficiency, ease of use and user experience. We want to compare the two modelling tools (SaveIDE and RSM along with SaveUML profile), to get feedback from users and to ascertain advantages, but particularly drawbacks of using UML profiles i.e. to reveal which characteristics are the ones that make the worst difference in the usability of these two approaches.

Results of this experiment can help us to improve our technology. In addition results can point to disadvantages of using UML profiles in general, and details that need to be considered when building a UML profile for domain-specific modelling.

In this experiment 18 software engineering master students were given the task to design a model of a real-time system using either RSM (with SaveUML profile) or SaveIDE tool. As one of criteria for measuring development efficiency, we were monitoring efforts spent and quality of designed model in terms of its validity and detailness. After the modelling was finished we analysed models delivered by students, and students were given a questionnaire regarding their experience of working with the given modelling tool.

With this experiment we tried to answer the following questions:

- Is using of SaveUML profile efficient with regard to time and efforts, in comparison to using SaveIDE?
- Do extraneous UML elements and semantics confuse developers and lead to an invalid or incomplete SaveCCM model?
- Which of technologies is more user-friendly and provide better user experience?

The given questions have more explorative character, so the results are shown mostly as descriptive statistics.

## 2.2 Conduction of the experiment

Participants of the experiment were 18 computer science students, who obtained bachelor degree, and almost all have worked with UML and UML tools before the experiment. We conducted our study in a form of minutely described assignments for students with strictly defined deliverable deadlines. Our experiment was not conducted under controlled conditions in laboratory, but it is executed in the field under normal conditions i.e. in a real development situation. Except for the varying factor we wanted to study, which was a modelling technology, we controlled the qualification of the testers and an input for testing i.e. an example of a real-time system.

The actual study consisted of three phases as shown on Figure 1. First we trained students in concerned technologies, then we conducted the experiment, and finally we let students to fill in a questionnaire and we analysed the results.

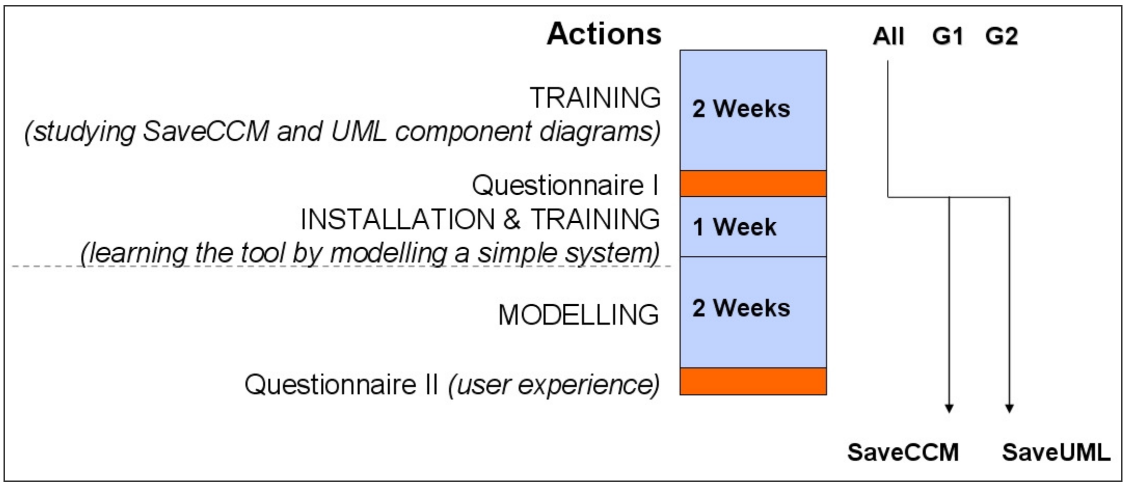

Actions
All
G1
G2
TRAINING
(studying SaveCCM and UML component diagrams)
2 Weeks
Questionnaire I
INSTALLATION & TRAINING
(learning the tool by modelling a simple system)
1 Week
MODELLING
2 Weeks
Questionnaire II (user experience)
SaveCCM
SaveUML

**Figure 1. Experiment activities**

*Training phase*

The training phase lasted for three weeks. First two weeks were reserved for studying SaveCCM and UML (precisely UML component diagram). After two weeks students were given an exam which tested their knowledge. Based on the results of this exam we separated students into two groups, one that will use SaveIDE tool (9 students), and one that will use RSM tool with SaveUML profile (9 students). The disposition was made in a way to have two homogeneous groups with a comparable qualification range.

Third week of training phase was intended for getting familiar with the tools by for modelling a simple Temperature regulation system (TRS) example (presented in section 4.).

*Modelling phase*

For the experiment we prepared a specification of Autonomous Truck Navigation (ATN) system. This autonomous system is intended to navigate the truck to find and follow a straight black line drawn on the surface area. A model of the system is described in section 5. Both system example models (TRS and ATN system) that were used within this experiment are designed in away to contain as much as possible distinct elements in order to force the use of all architectural elements of SaveCCM language.

The specification of the system was provided to students in a textual form, with detail description of system elements and their properties. Their task was to design a model using the given modelling tool.

*Questionnaire*

After the modelling was finished, participants completed a questionnaire regarding their user experience in the given tool. This questionnaire covered a number of subjects such as initial effort participants had to make to learn the technology, complexity of using the tool and clearness of graphical representation of modelling elements. Also there were questions about problems and bugs that encountered during their work, and several questions about different aspects of using SaveUML profile.

## 3. Temperature regulation system description

This section describes the simple Temperature regulation system that participants of the experiment used in the training phase to get practice in modelling with SaveCCM and using the modelling tool (either SaveIDE or RSM tool with SaveUML profile). The system is intended to regulate a temperature of the home, taking into consideration the current indoor temperature and the desired temperature set by the user.

A system is consisted of several parts:

- *A temperature sensor* for measuring the indoor temperature.
- *An air-conditioning unit*, which has two sub-units, *cooling unit* and *heating unit.*
- *A regulation panel*, intended to display the current temperature and to set the desired temperature. On regulation panel there is a display which shows the current indoor temperature, and a temperature chooser that enables to set the desired temperature.
- *A control point,* which is the central part of the system, it receives an information about the current and desired temperature, makes the necessary calculations and activates/deactivates air-condition unit in order to achieve the right temperature.

The system overview is presented on Figure 2.

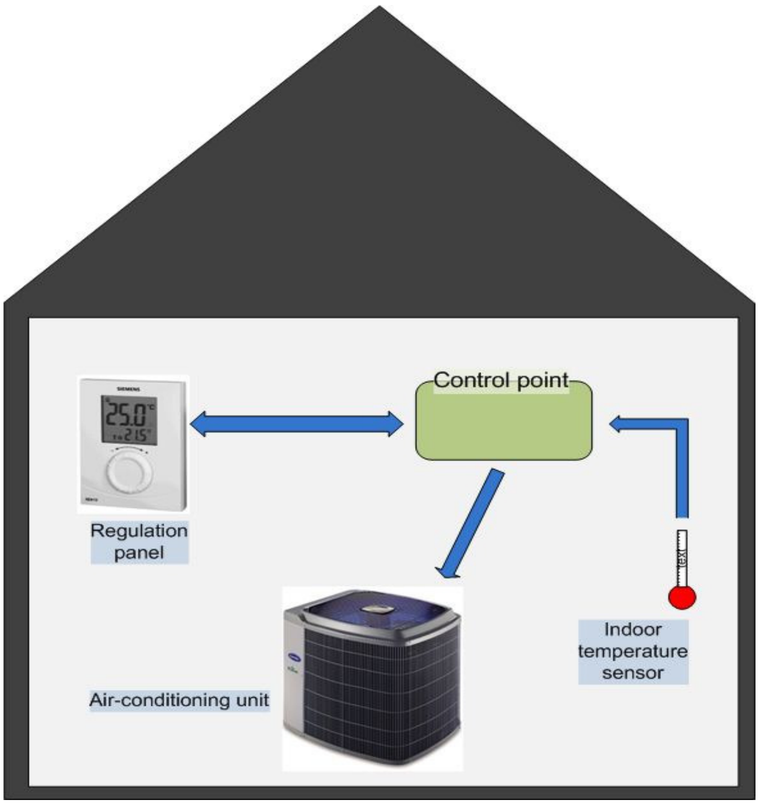


**Figure 2: Temperature regulation system overview**

### 3.1 Temperature sensor

Temperature sensor is the only active element in the system (it generates events that start the system execution). The sensor measures a temperature and deliver its value to the control point, thus it triggers the control point which then performs the necessary actions.

Sensor is specified by a temperature range that it is able to measure; this range is defined through two sensor's attributes, maximal and minimal temperature. Sensor updates temperature value approximately every 1 minute with a maximum time deviation of 30 seconds (1±0.5 min).

### 3.2 Air-conditioning unit

Air-conditioning unit is a device that heats or colds an indoor environment. As it is able to perform both actions, its internal structure is built from two main sub-units, one for heating and another for cooling operation. Therefore, the air-conditioning unit can be in one of three possible modes: *cooling*, *heating* or *idle* mode. For example, in cooling mode, the cooling unit is active until the air-conditioning unit receives a signal to switch to another mode.

### 3.3 Regulation panel

The regulation panel is intended for user interaction with the system. The panel itself is just a frame that encapsulates two independent devices: the *display* and the *temperature chooser*. The display shows current indoor temperature; this value is provided by the control point to the display unit. The temperature chooser is a simple device that allows user to choose the desired temperature, the device is able to provide the value of chosen temperature through its interface.

### 3.4 Control point

As it is already mentioned, this simple system has a centralized architecture, with the control point as its main part. Control unit collects the information, performs processing and controls other devices in the system. After reading the temperature values (indoor temperature and the temperature set by the user), the control point makes decision on necessary action, i.e. if the heating or cooling unit has to be activated.

All processing and deciding functionality of this system part is implemented by a single entry function.

### 3.5 The SaveCCM model of TRS

The model of temperature regulation system modelled in SaveCCM language consists of several elements:

- An Clock component which is intended to trigger the system execution
- SaveCCM components, Sensor, Control, Cooling-unit, Heating-unit, Chooser and Display
- SaveCCM switch element Switch-Mode
- SaveCCM assembly element for modelling the Regulation-Panel
- SaveCCM composite component for modelling the Air-Conditioning unit.

The SaveCCM model of TRS modelled in SaveIDE modelling tool, is presented in figures below, namely the top hierarchy level of the system on Figure 3, the internal structure of Air-Conditioning unit on Figure 4 and the internal structure of Regulation-Panel on Figure 5.

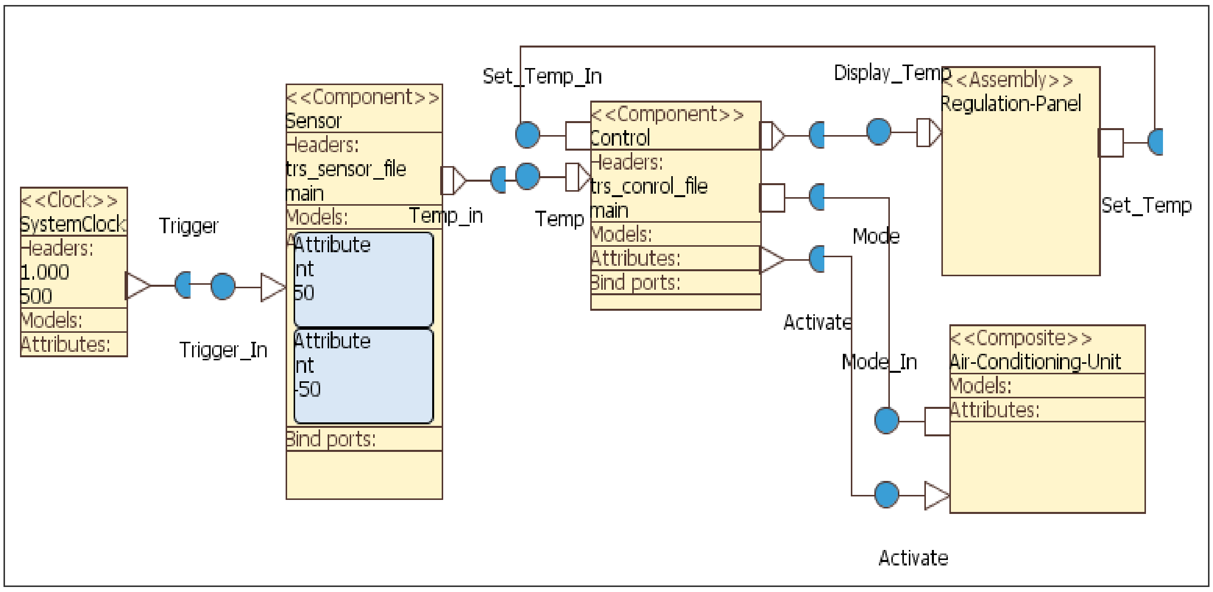


**Figure 3. SaveCCM model of temperature regulation system**

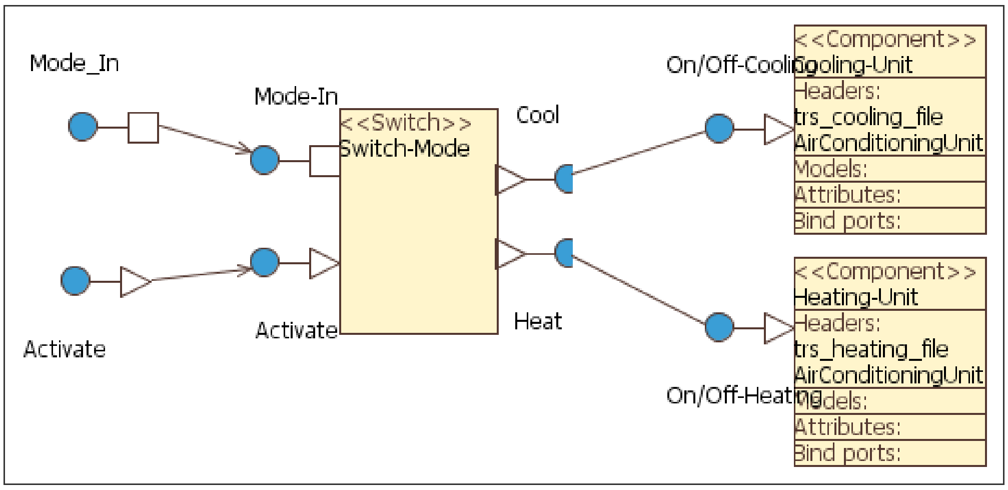


**Figure 4. Internal structure of the Air-Conditioning unit within the TRS system**

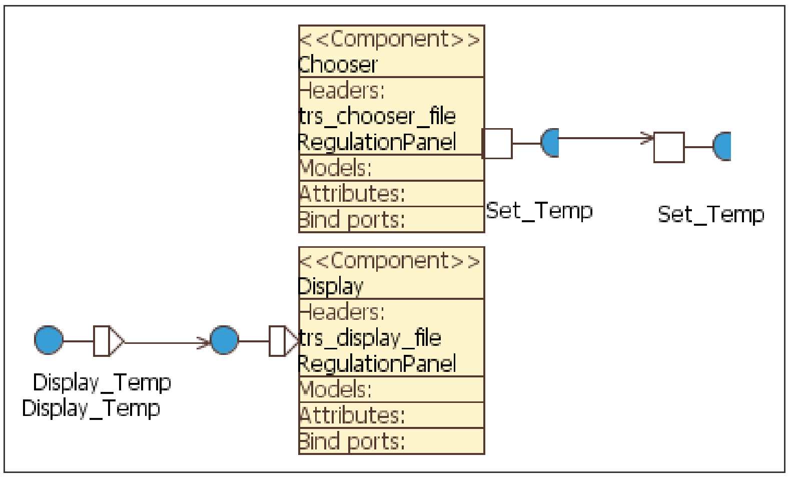
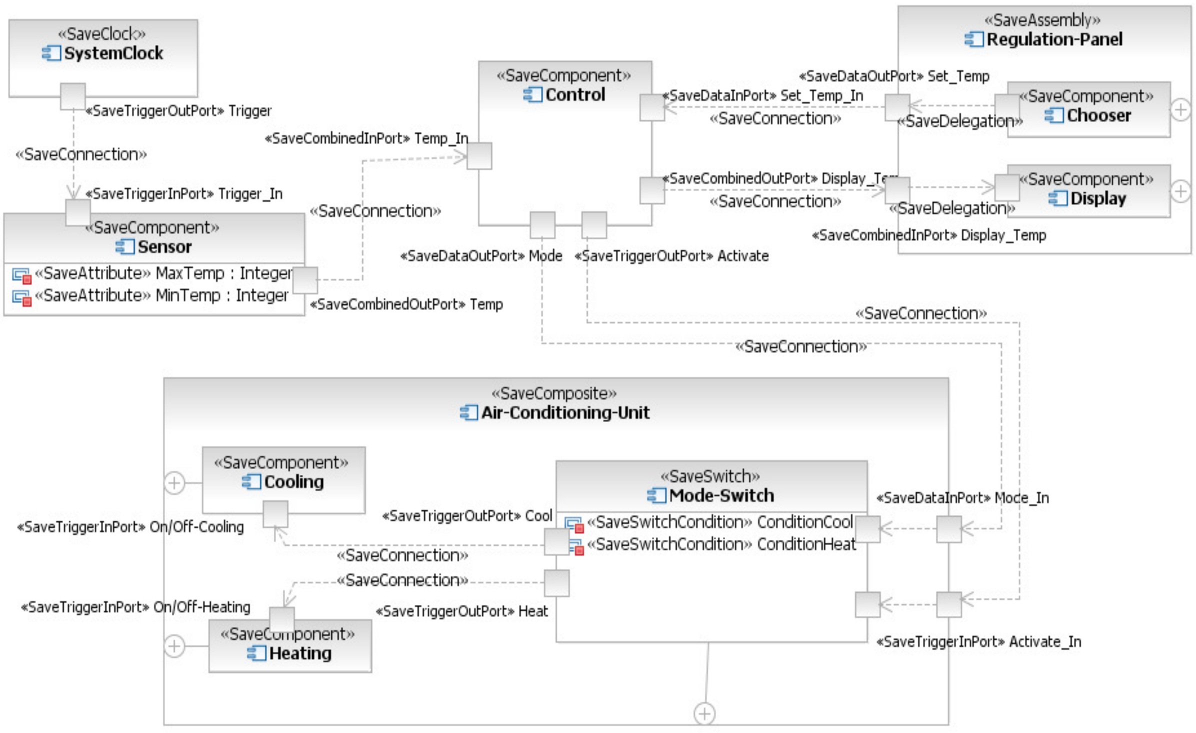


**Figure 5. Internal structure of the Regulation-Panel element within the TRS system**

## 3.6 SaveCCM model of TRS system modelled with SaveUML profile

The system model consists of the same elements ad the one modelled in SaveIDE tool, however it is modelled using the SaveUML profile. The system model from RSM tool is presented on figure below.



«SaveClock»
SystemClock
«SaveTriggerOutPort» Trigger
«SaveConnection»
«SaveTriggerInPort» Trigger_In
«SaveComponent»
Sensor
«SaveAttribute» MaxTemp : Integer
«SaveAttribute» MinTemp : Integer
«SaveCombinedOutPort» Temp
«SaveConnection»
«SaveCombinedInPort» Temp_In
«SaveComponent»
Control
«SaveDataOutPort» Mode
«SaveTriggerOutPort» Activate
«SaveDataInPort» Set_Temp_In
«SaveConnection»
«SaveCombinedOutPort» Display_Te
«SaveConnection»
«SaveAssembly»
Regulation-Panel
«SaveDataOutPort» Set_Temp
«SaveDelegation»
«SaveComponent»
Chooser
«SaveDelegation»
«SaveComponent»
Display
«SaveCombinedInPort» Display_Temp
«SaveConnection»
«SaveConnection»
«SaveComposite»
Air-Conditioning-Unit
«SaveComponent»
Cooling
«SaveTriggerInPort» On/Off-Cooling
«SaveTriggerOutPort» Cool
«SaveConnection»
«SaveConnection»
«SaveTriggerInPort» On/Off-Heating
«SaveTriggerOutPort» Heat
«SaveComponent»
Heating
«SaveSwitch»
Mode-Switch
«SaveSwitchCondition» ConditionCool
«SaveSwitchCondition» ConditionHeat
«SaveDataInPort» Mode_In
«SaveTriggerInPort» Activate_In

**Figure 6. TRS system modelled with SaveUML**

# 4. Autonomous truck navigation system description

This section describes the Autonomous truck navigation system which was used in the modelling phase of the experiment to test the feasibility of our approach. The system was initially demonstrated in [2].

A system is consisted of several parts:

- *A sensor panel,* which holds two light-sensors (left and right sensor) for determining the truck position.
- *A navigation module*, which is an application that navigates the truck according to the data provided by sensors.

- *An actuator* intended to control the speed and direction of the truck by controlling the truck engine and wheels.

This autonomous system in intended to navigate the truck to follow a straight black line with two filled black circles on each end (the line is drawn on the surface area) as shown on Figure 7.The truck has to follow a straight line, when it reaches the end of the line it turns around and starts searching for the line again, as shown in fig. 1. The truck executes the navigation application running in 3 different operational modes, namely:

- *Follow mode* in which the truck simply follows the black line using its light sensors. When the truck detects the end of the line, it changes mode to the *Turn mode*.
- *Turn mode* in which the truck turns without guidance from any line until it reaches a state where it is supposed to be able to find the line again. Upon completion, the truck changes mode to the *Find mode*.
- *Find mode* in which the truck searches for the line using a search pattern. When the line is detected it aligns the truck such that the *Follow mode* can be normally executed.

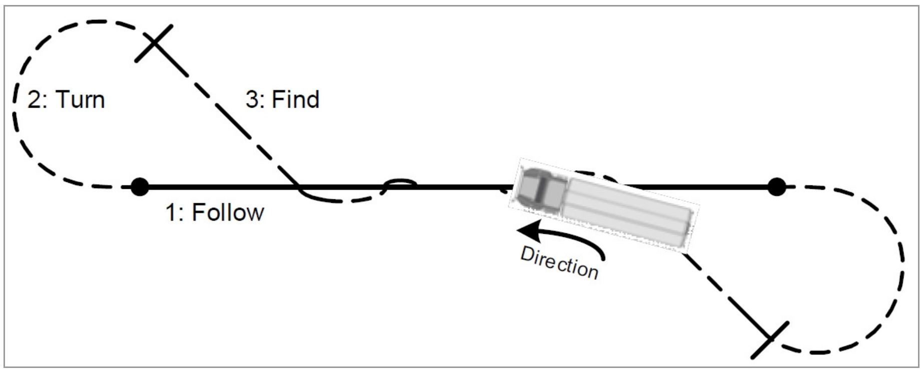


**Figure 7. The track layout for the autonomous truck**

## 4.1 Sensor panel

The senor panel is a frame that encapsulates two light sensors – left sensor and right sensor. These sensors are able to detect the line on the surface, and are used to determine that the truck has a correct position with respect to the black line. They observe the surface and provide data to the navigation module (thus they activate the application execution); each sensor yields high when it detects a line and low otherwise.

The sensors are periodically polled by a system clock with a frequency of *1/T* (within this assignment students had to estimate the eligible value for T, more information on this is provided in section 5.4).

## 4.2 Navigation module

The navigation module is the most important part of the system. It is an software application that collects sensor values (from both, left and right sensor), performs the control-algorithm and reacts to the environment (steer the truck). Internally, navigation module consists of several main functions:

- *Follow* – the function that is able to navigate the truck (by defining its speed and direction) to follow the line, taking into consideration data provided by sensors.
- *Turn* – performs an algorithm that turns the truck back to the line (taking into consideration data provided by sensors).
- *Find* – performs an algorithm of searching for the line. It uses sensor values as input information, and makes the decision on the necessary actions. It controls the truck movement by defining the required speed and direction.
- *ModeChange* – intended to determine and to control the operational mode of the navigation module by taking into consideration data provided by sensors and feedback given by *Follow*, *Turn* and *Find* functions (for example the status code of their execution). It processes the collected data, and makes a decision about the mode.

According to information about the mode provided by ModeChange, navigation module switches to one of the operational modes (thus the appropriate function has to be activated).

## 4.3 Actuator

Actuator is an element that is able to activate/deactivate the truck engine and to control the rotation of wheels of the truck. It does this according to the desired speed and direction provided by to the actuator the navigation module. The functionality of this system part is implemented by a single entry function.

### 4.4 Timing constraints

The autonomous truck navigation (ATN) system is an example of a time critical system, because the truck has to react to the sensor values within a short time period. As a part of this assignment students had to take account of time constraints of the system and execution times (precisely worst case execution time -WCET) of every system part.

In order for the navigation module to guarantee a proper and stable function, the truck has to react to the sensor values within a maximum specified time *T*. As a part of design, students specified the time period – *T* (this is the period of system clock) and estimated its value with respect to the dynamics of the truck (see section 5.5).

Except the period *T*, participants had to specify WCET (by estimating the appropriate values) for every part of navigation module, as well as for the navigation module itself. (they modelled these constraints by using quality attributes).

### 4.5 Truck specification

In order for students to define the right values for time period *T* and WCET of every system part, we provided them the details about truck dynamics. For example, let's consider the case when light-sensors are able to detect the line 1 meter in front of the truck. In this case if the truck is driving straight forward, the ATN system has to react to sensor values before the truck passes over this distance (otherwise it would be a "blind driving"). If the truck is turning around, reaction should be even faster, because the truck route is not straight.

In this assignment, participants had to assume that the truck is approximately 1 meter long and can drive with maximum speed of 10 kilometres per hour. Also each light-sensor placed on this truck is able to detect the line 30 centimetres in front of the truck.

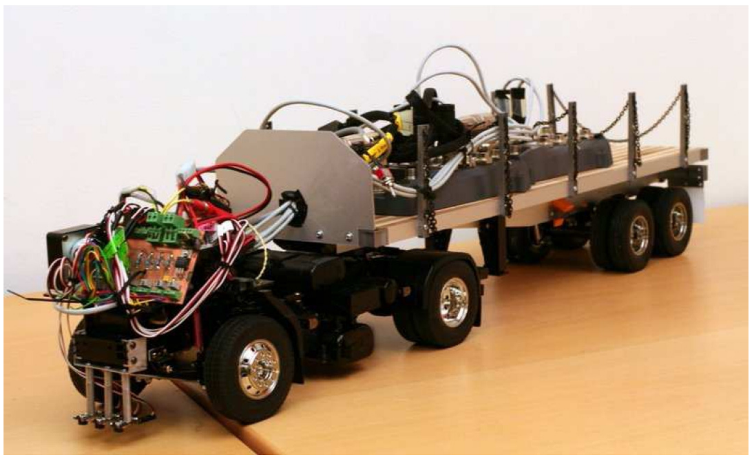


**Figure 8: Picture of the autonomous truck**

### 4.6 The SaveCCM model of ATN system

The model of autonomous truck navigation system modelled in SaveCCM language consists of several elements:

- An Clock component which is intended to trigger the system execution
- SaveCCM components, LeftSensor, RightSensor, Actuator, ModeChange, Follow, Find and Turn
- SaveCCM switch elements ModeSwitch (for activating different execution modes, follow, turn or find) and Data-Switch (for forwarding the output data according to the execution mode)
- SaveCCM Assembly element for modelling the SensorPanel
- SaveCCM Composite component for modelling the NavigationModule.

The SaveCCM model of ATNS modelled in SaveIDE modelling tool is presented in figures below, namely the top hierarchy level of the system Figure 9, the internal structure of NavigationModule Figure 10 and the internal structure of SensorPanel Figure 11.

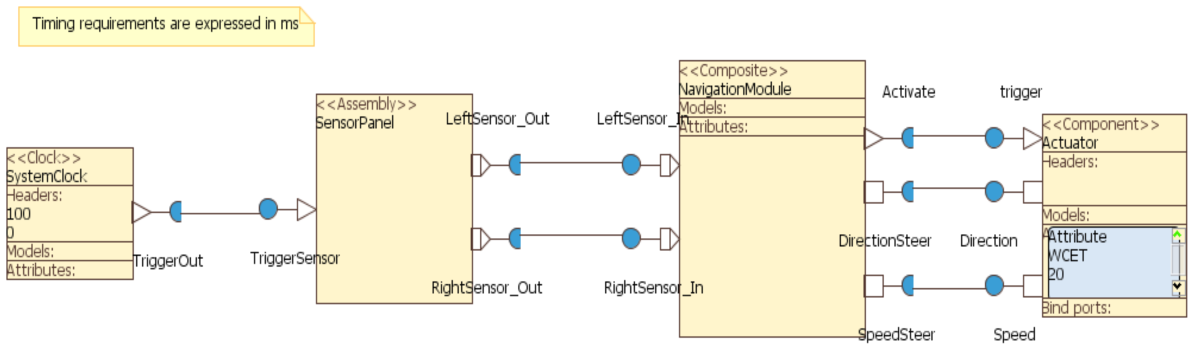


**Figure 9. SaveCCM model of autonomous truck navigation system**

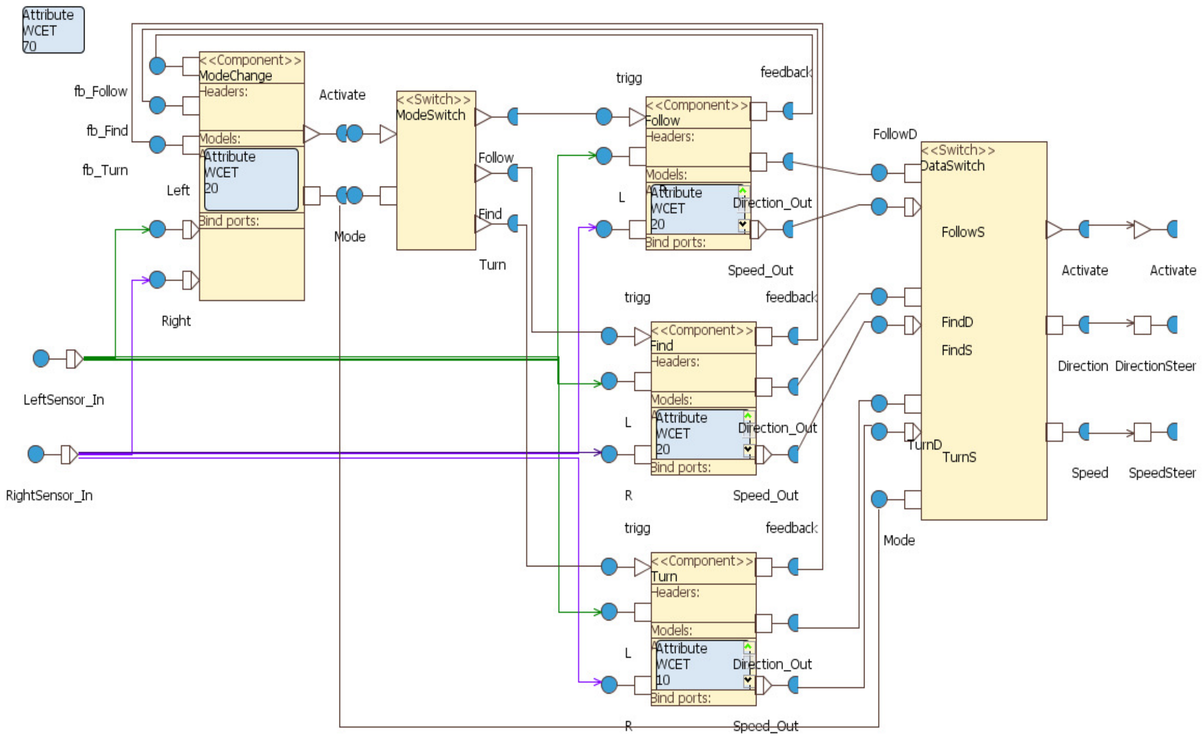


**Figure 10. Internal structure of NavigationModule within ATN system**

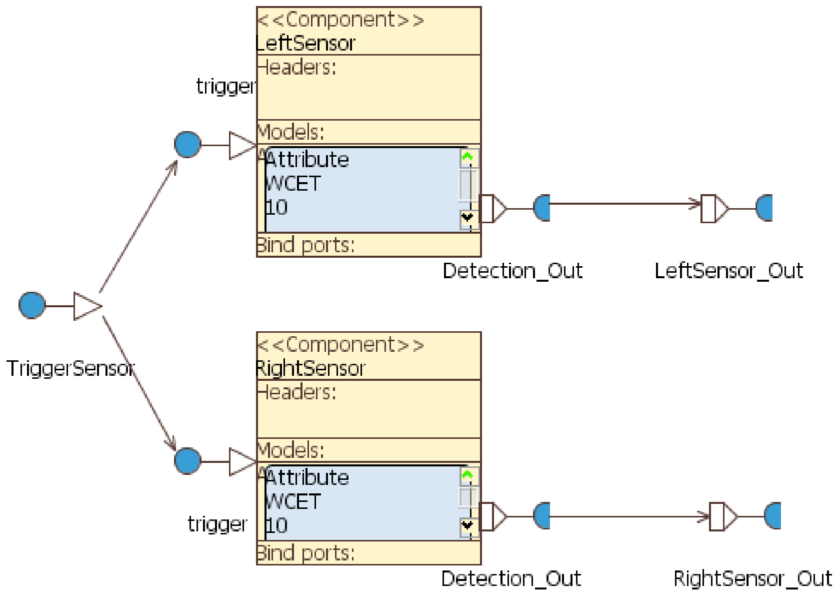


**Figure 11. Internal structure of SensorPanel within ATN system**

## 4.7 Model of the ATNS designed with SaveUML profile

The system model consists of the same elements ad the one modelled in SaveIDE tool, however it is modelled using the SaveUML profile. The system model from RSM tool is presented on figure below.

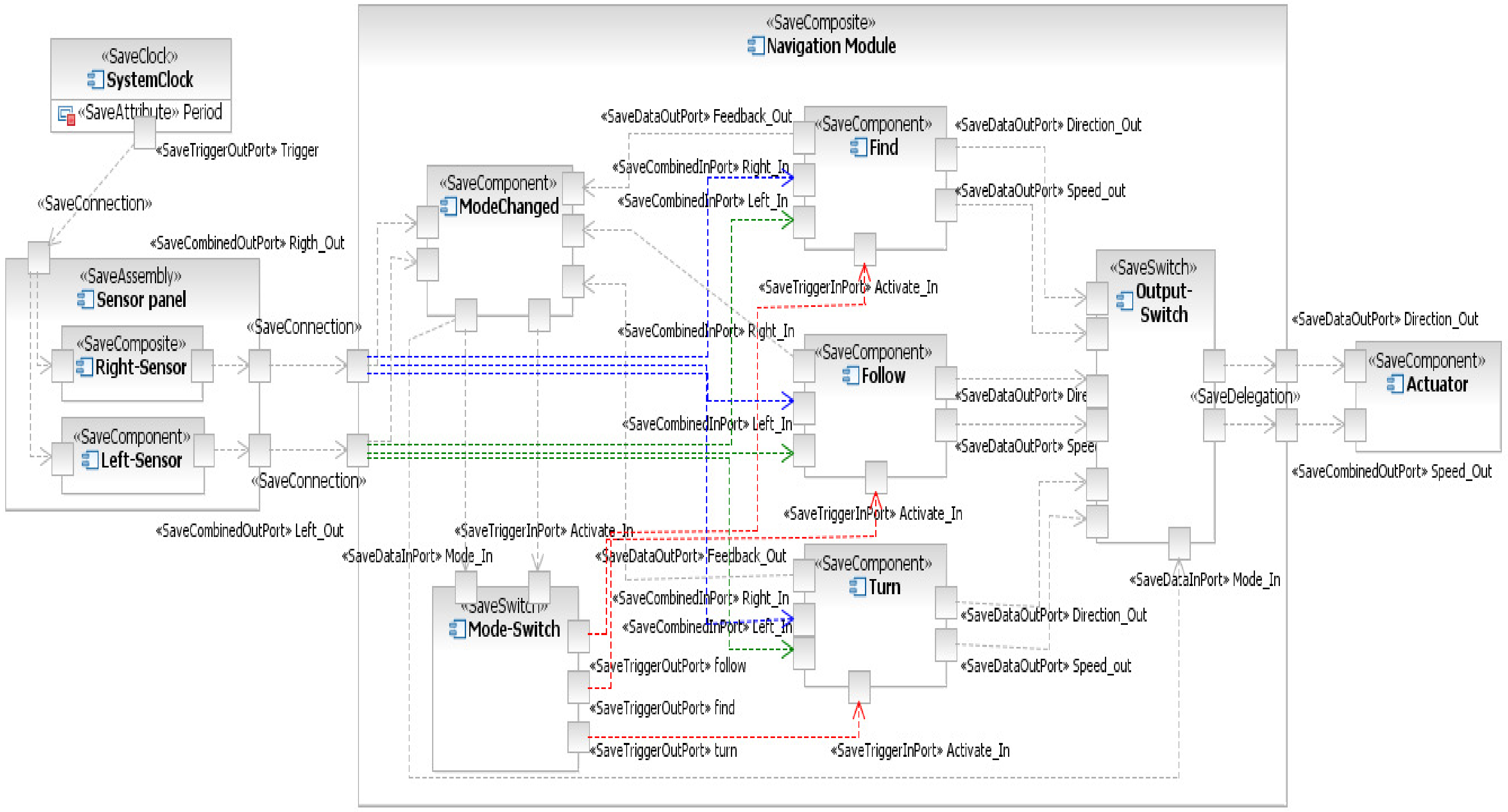

«SaveClock»
SystemClock
«SaveAttribute» Period
«SaveTriggerOutPort» Trigger
«SaveConnection»
«SaveCombinedOutPort» Rigth_Out
«SaveAssembly»
Sensor panel
«SaveComposite»
Right-Sensor
«SaveComponent»
Left-Sensor
«SaveConnection»
«SaveConnection»
«SaveCombinedOutPort» Left_Out
«SaveComposite»
Navigation Module
«SaveComponent»
ModeChanged
«SaveDataOutPort» Feedback_Out
«SaveCombinedInPort» Right_In
«SaveCombinedInPort» Left_In
«SaveComponent»
Find
«SaveDataOutPort» Direction_Out
«SaveDataOutPort» Speed_out
«SaveTriggerInPort» Activate_In
«SaveCombinedInPort» Right_In
«SaveComponent»
Follow
«SaveCombinedInPort» Left_In
«SaveDataOutPort» Dire
«SaveDataOutPort» Spee
«SaveTriggerInPort» Activate_In
«SaveSwitch»
Output-Switch
«SaveDataInPort» Mode_In
«SaveDelegation»
«SaveDataOutPort» Direction_Out
«SaveCombinedOutPort» Speed_Out
«SaveComponent»
Actuator
«SaveTriggerInPort» Activate_In
«SaveDataInPort» Mode_In
«SaveDataOutPort» Feedback_Out
«SaveSwitch»
Mode-Switch
«SaveComponent»
Turn
«SaveCombinedInPort» Right_In
«SaveCombinedInPort» Left_In
«SaveDataOutPort» Direction_Out
«SaveDataOutPort» Speed_out
«SaveTriggerOutPort» follow
«SaveTriggerOutPort» find
«SaveTriggerOutPort» turn
«SaveTriggerInPort» Activate_In

**Figure 12. Model of ATNS designed with SaveUML profil**

## 5. Questionnaire

To get the feedback from the participants we prepared an questionnaire with questions concerning their using experience. The questions are listed in section 6.1 and they are divided into several categories, *background information*, *assignment understanding*, *initial effort to learn the technology*, *complexity and assistance*, *graphical representation*, *bugs*, *SaveUML profile* and *overall experience*.

### 5.1 User experience questionnaire

Background information

1. Have you participated any software engineering project? (Yes/No)
2. Do you have industry experience in software engineering? (Yes/No)
3. Did you meet with UML before this course? (Yes/No)
4. Did you use any UML modelling tool before this course? (Yes/No)
5. If you used some UML modelling tool before, which one?

Assignment understanding

6. Your level of understanding of methodology that was used in this assignment was:
   - Excellent
   - Good
   - Average
   - Adequate
   - Poor

7. Was the description and specification of Temperature regulation system clear enough?
   - Completely clear
   - Partially clear
   - Partially ambiguous
   - Very ambiguous

8. Was the description and specification of Autonomous truck navigation system clear enough?
   - Completely clear
   - Partially clear
   - Partially ambiguous
   - Very ambiguous

USING MODELLING TOOLS

Initial effort to learn the technology:

9. How much learning effort you had to make before you started using the modelling tool? (learning about SaveCCM, UML, UML profile …)
   - A lot of effort
   - Reasonable effort
   - Neither too much or too less effort
   - Not a lot of effort
   - Minor effort

10. How much learning effort you had to make to start using the tool? (getting to know with the tool)
    - A lot of effort
    - Reasonable effort
    - Neither too much or too less effort
    - Not a lot of effort
    - Minor effort

11. Was using the tool easier after you made your first model (TRS system)? (was it easier to use the tool for working on your second model - ATN system)?
    - It was much easier
    - It was easier
    - Neither easier or harder
    - It was harder

Complexity and assistance of the tool:

12. Rate the complexity of using the modelling tool: (intuitive-5, complex-1)
    - Workspace organization (are available functionalities located where you expected)
    - Project management
    - Adding new elements to a model (did it worked as you expected)
    - Defining properties of model elements (e.g. data type of a port)
    - Overall complexity

13. The assistance (automated procedures, offering default values etc.) provided by the modelling tool was:
    - Excellent
    - Good
    - Average
    - Adequate
    - Poor

Graphical representation of component model

14. The graphical representation of modelling elements (for example SaveCCM component, ports etc.) is:
    - Adequate and understandable
    - Sufficient but should be improved
    - Unreadable
    - Confusing
    - Something else: _______

Bugs

15. Taking into account perceived amount of bugs, the tool you used is;
    - Professional level
    - Solid
    - Stable
    - Unstable
    - Very unstable

16. Encountered bugs were:
    - Most of them were irritating
    - Some were irritating
    - All were tolerable
    - Almost unnoticeable
    - I didn't encounter any bugs

17. *(a question for SaveUML group)*
    Using of the SaveUML profile was:
    - Very intuitive and clear
    - Understandable and not difficult
    - Time consuming in comparison with using only UML (without the profile)
    - Difficult to use at first, but easy after some practice
    - Confusing and difficult to use correctly
    - Something else:__________________

18. *(a question for SaveUML group)*
    The functionality of model validation was:
    - Useful
    - Useful, especially the pop-up warnings
    - Useful, but pop-up warnings were irritating
    - Neither useful or useless
    - Partially useless
    - Useless

19. Rate your overall experience of using SaveIDE (if you used SaveIDE) or SaveUML profile (if you used RSM). (grades: 1-5)

20. Describe your overall experience of working with the tool?______________________
21. Rate your overall experience of using SaveUML profile? (1-5)

## 5.2 Answers provided by participants

The answers are shown in chart diagrams below.
Answers to questions 1.-4 (background information) are presented on Figure 13.

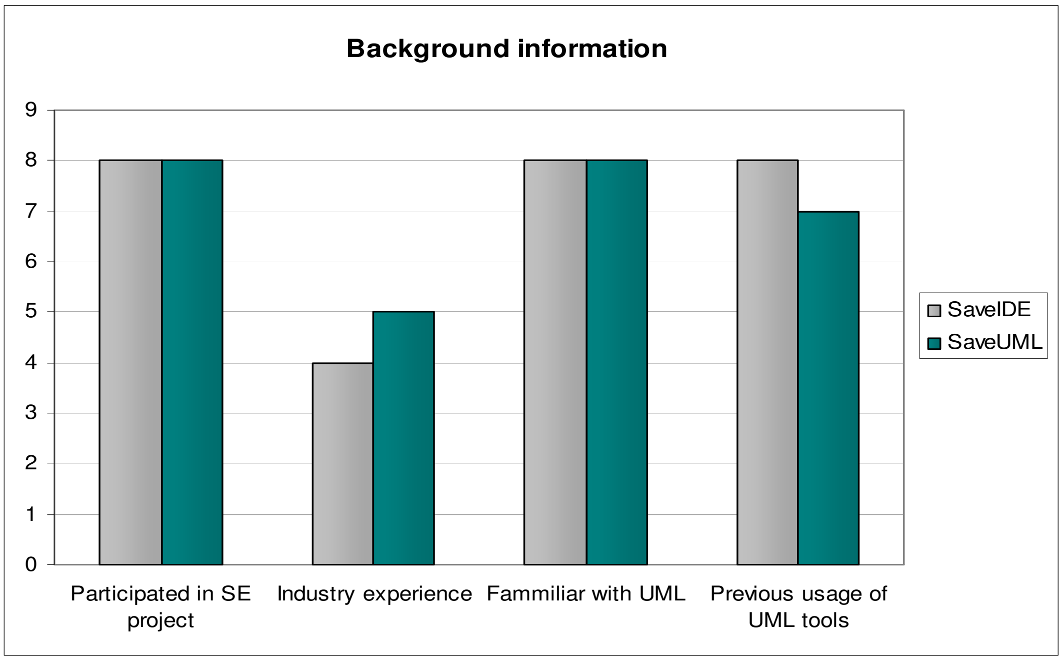


**Figure 13. Background of participants**

Some of the tools that participants used for UML modelling before this experiment are: StarUML, IBM Rational Rose, Visual Paradigm, Microsoft Visio, Omondo, Telelogic Rhapsody, MID Innovator.

Answers to questions 6.-8 (assignment understanding) are presented on figures below.

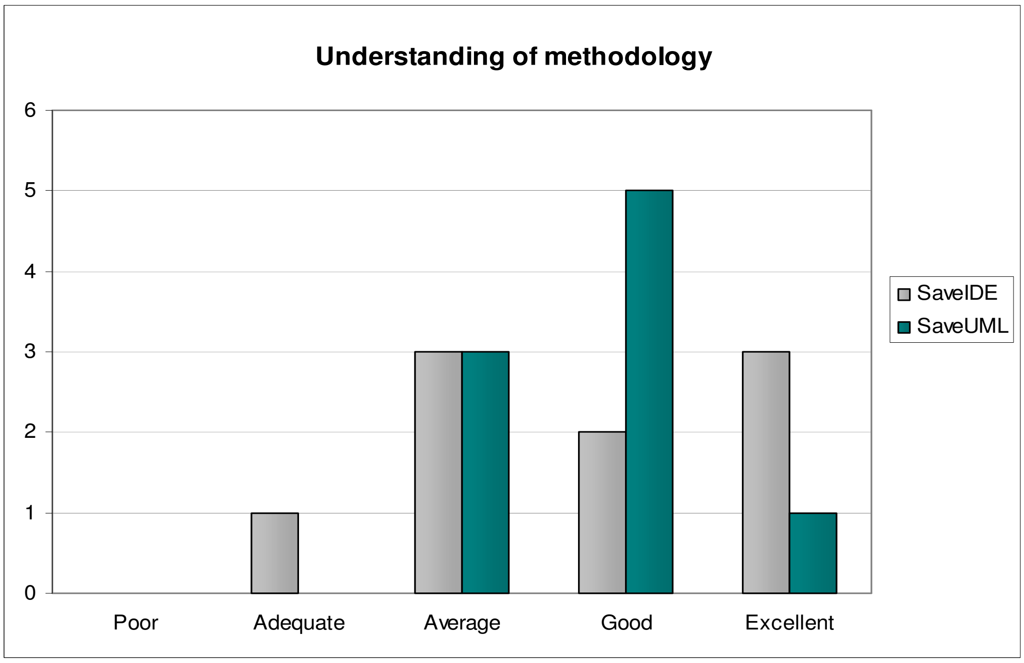


**Figure 14. Assignment understanding – methodology**

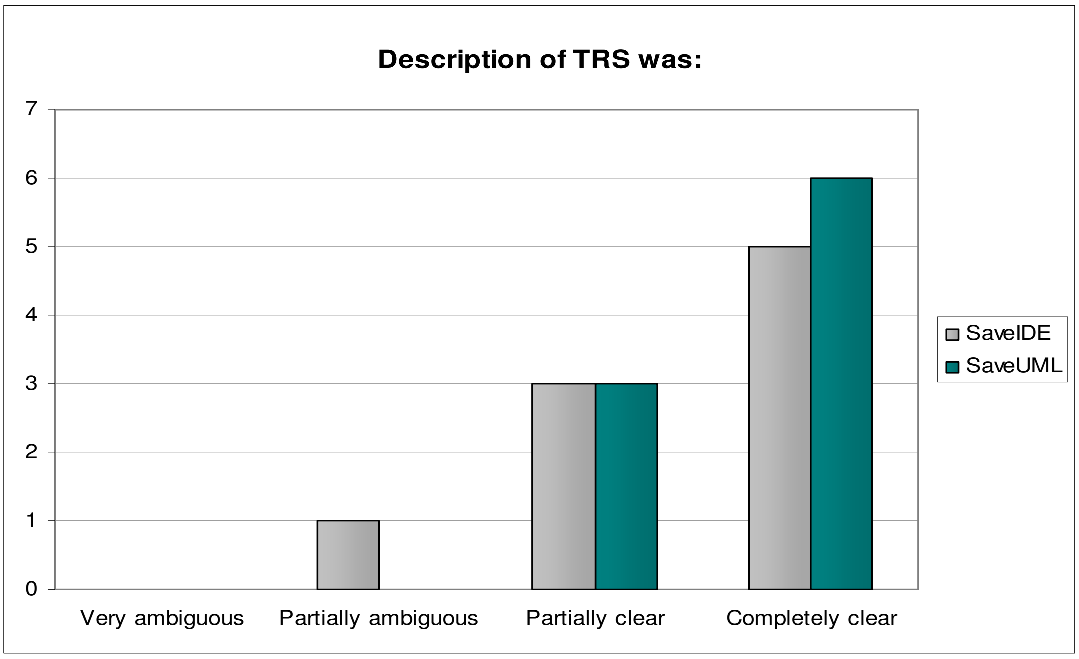


**Figure 15. Assignment understanding – TRS**

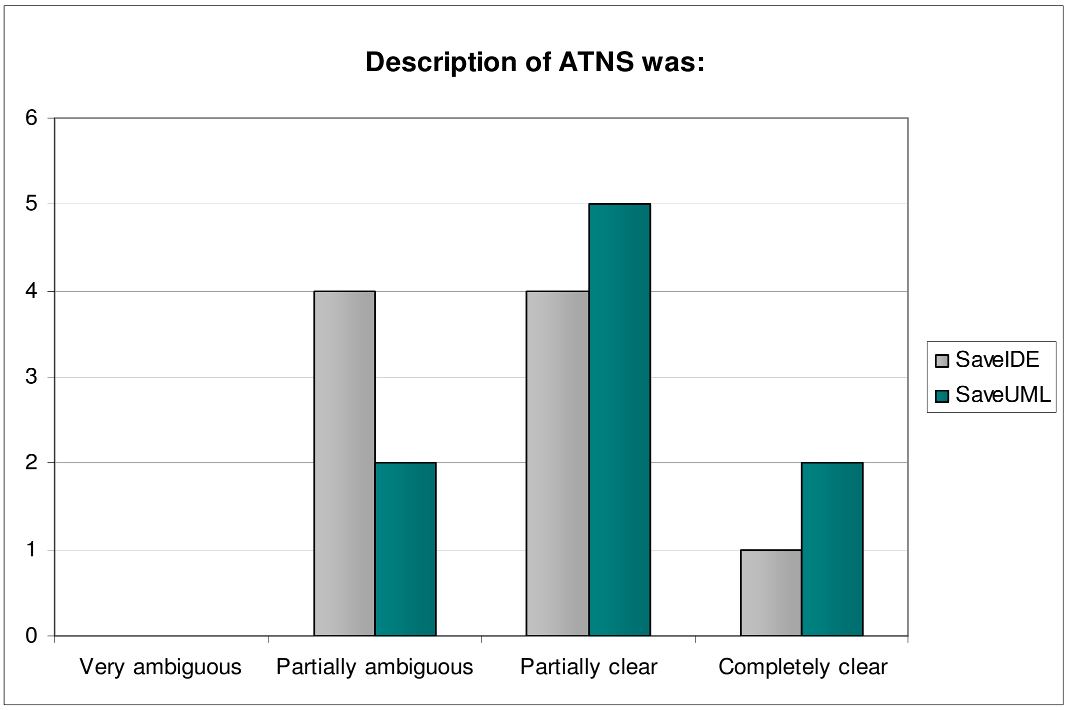


**Figure 16. Assignment understanding – ATNS**

Answers to questions 9.-11 (initial effort to learn the technology) are presented on figures below.

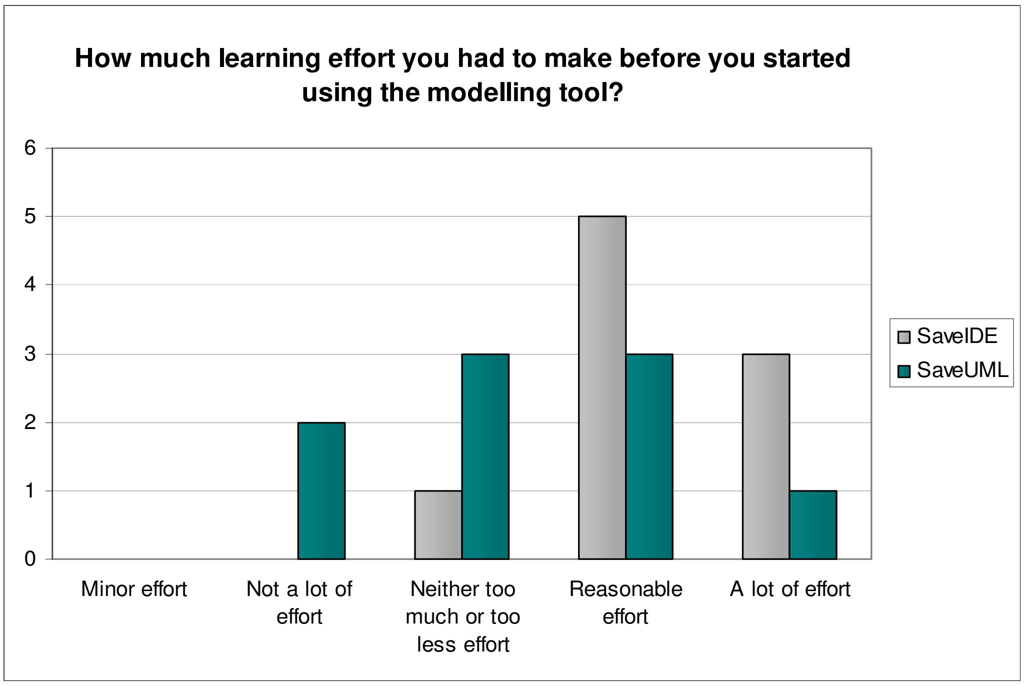


**Figure 17. Initial effort - theoretical part**

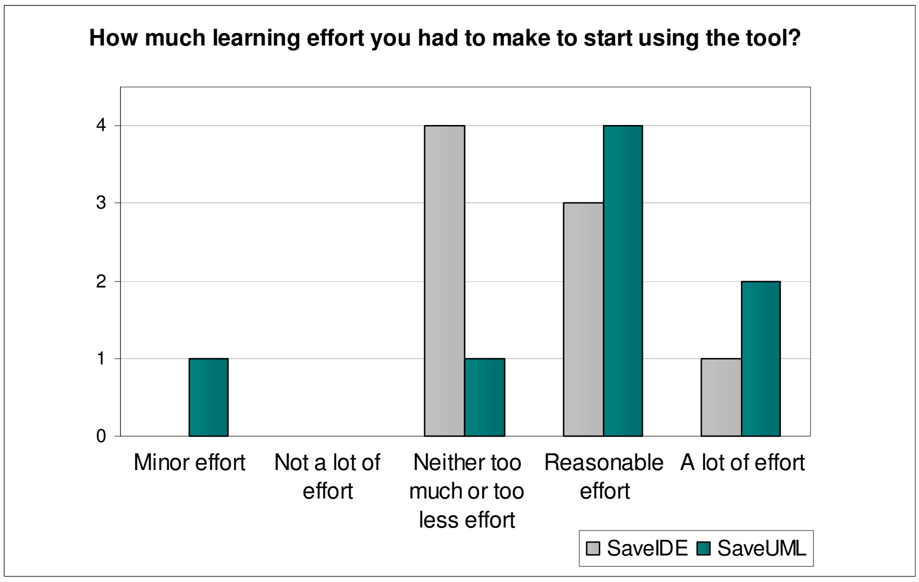


**Figure 18. Initial effort – tool**

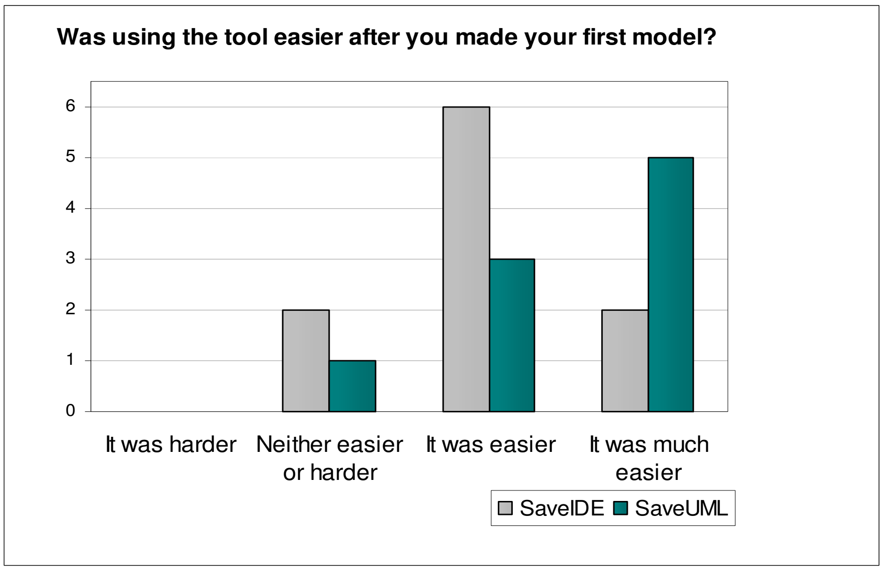


**Figure 19. Effort after some practice**

Answers considering the complexity of using the tool, assistance provided by the tool and graphical representation used for presenting elements of the component model (questions 12., 13. and 14.) are shown on Figure 20, Figure 21 and Figure 22.

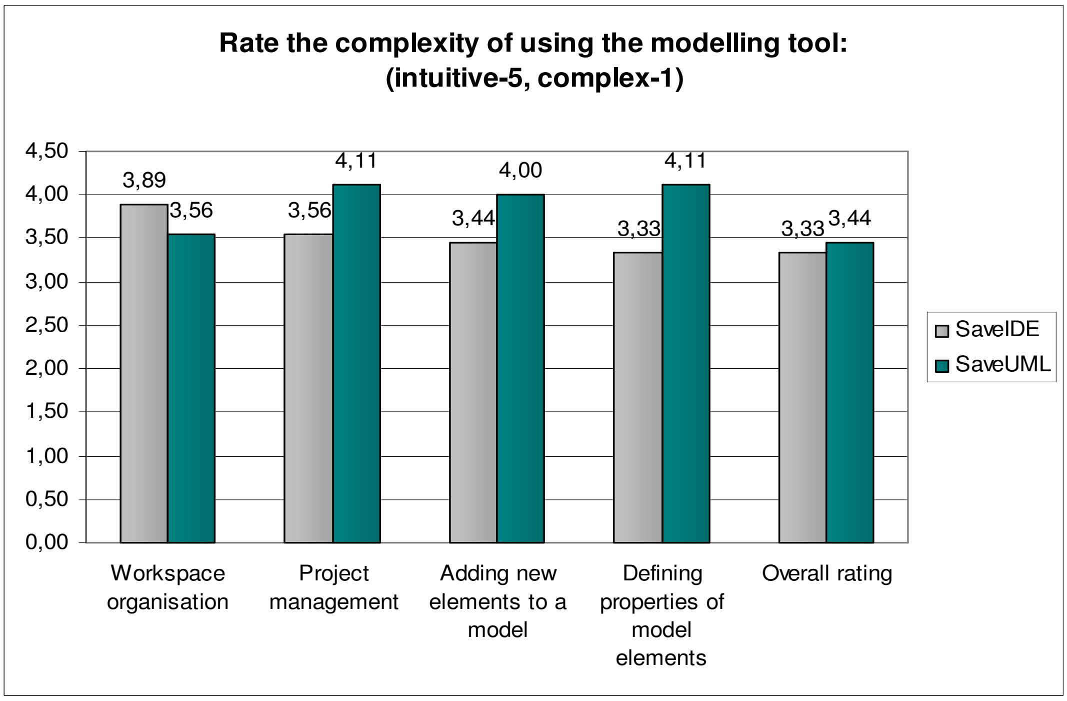


**Figure 20. Complexity of using the tool**

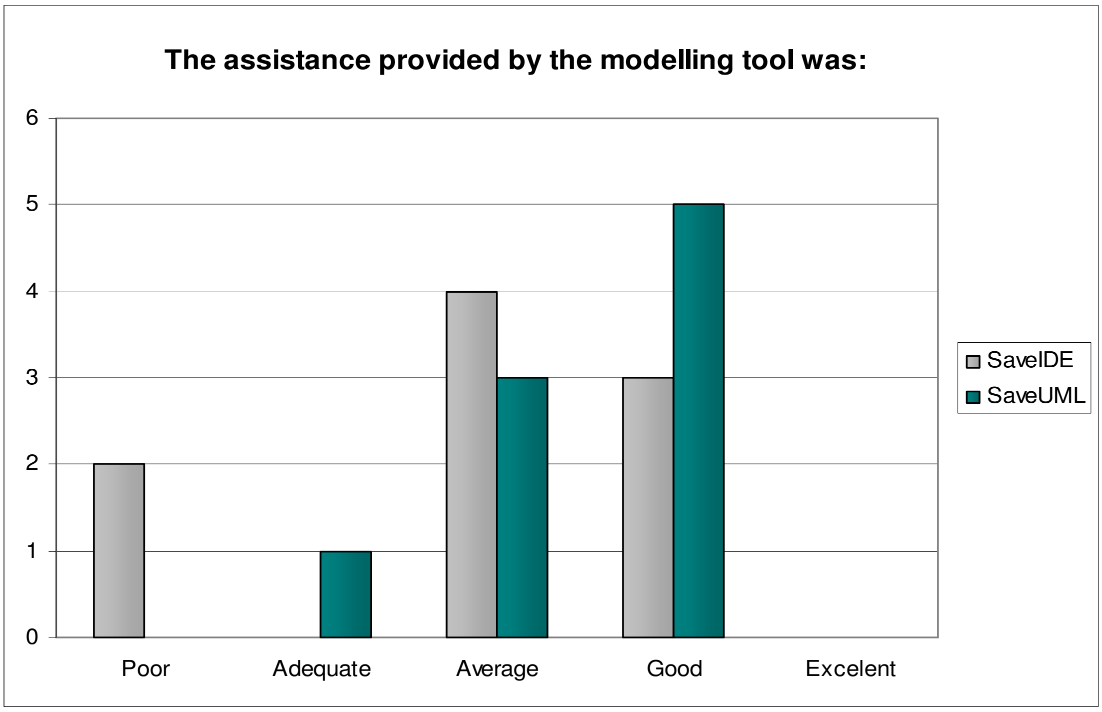


**Figure 21. Tool assistance**

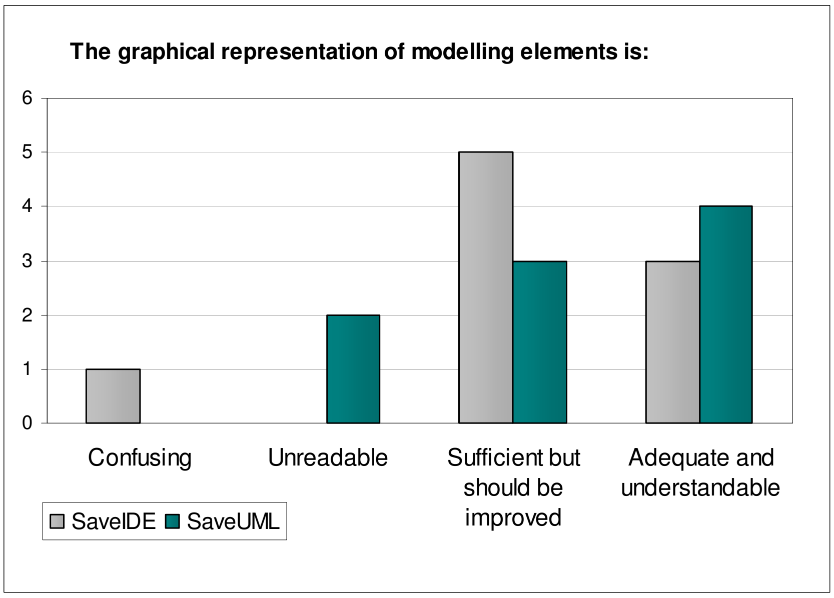


**Figure 22. Graphical representation of the model**

Answers to questions 17.and 18. (regarding the bugs that encountered during the work) are presented on figures below.

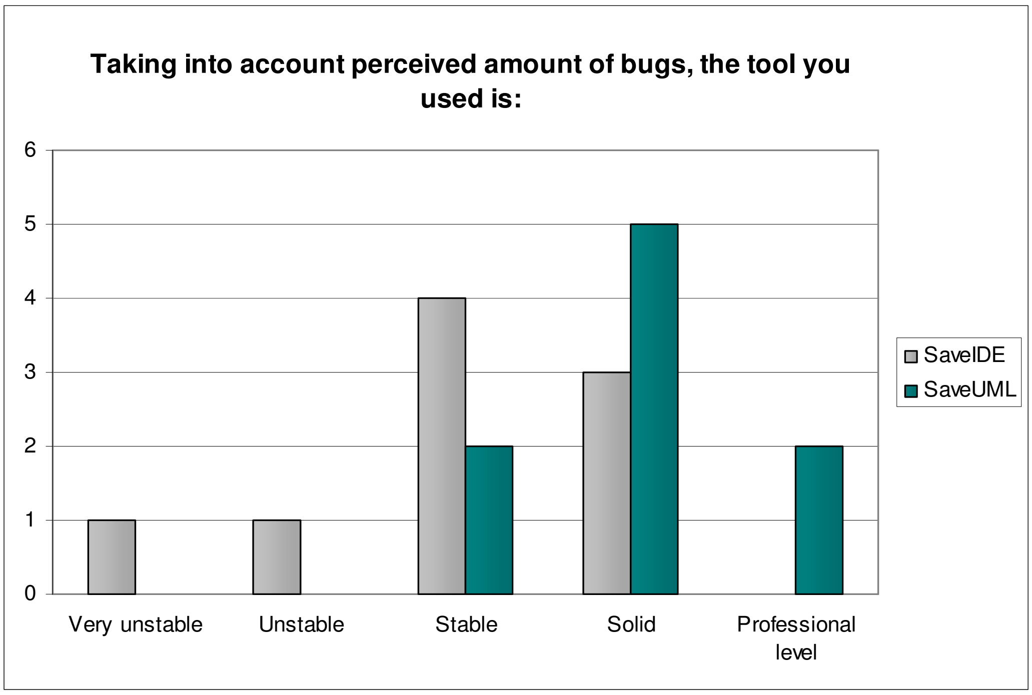


Figure 23. Encountered bugs - tool stability

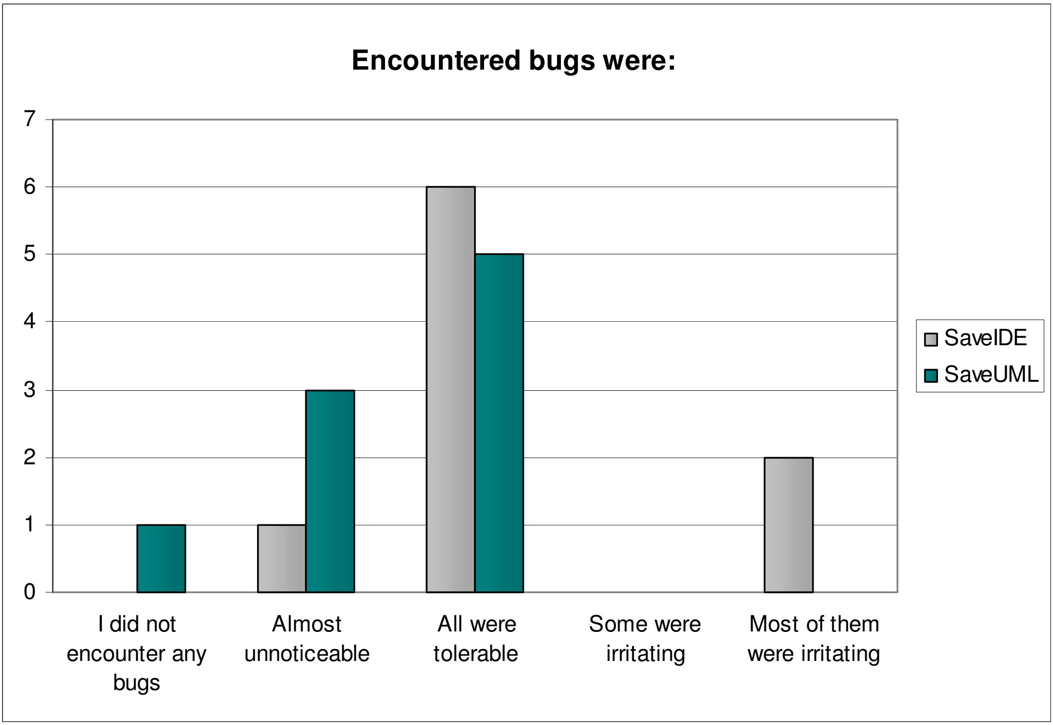


Figure 24. Encountered bugs - user experience

Questions regarding the SaveUML profile were answered only by participants from the SaveUML group (9 students), and their questions are presented on charts below.

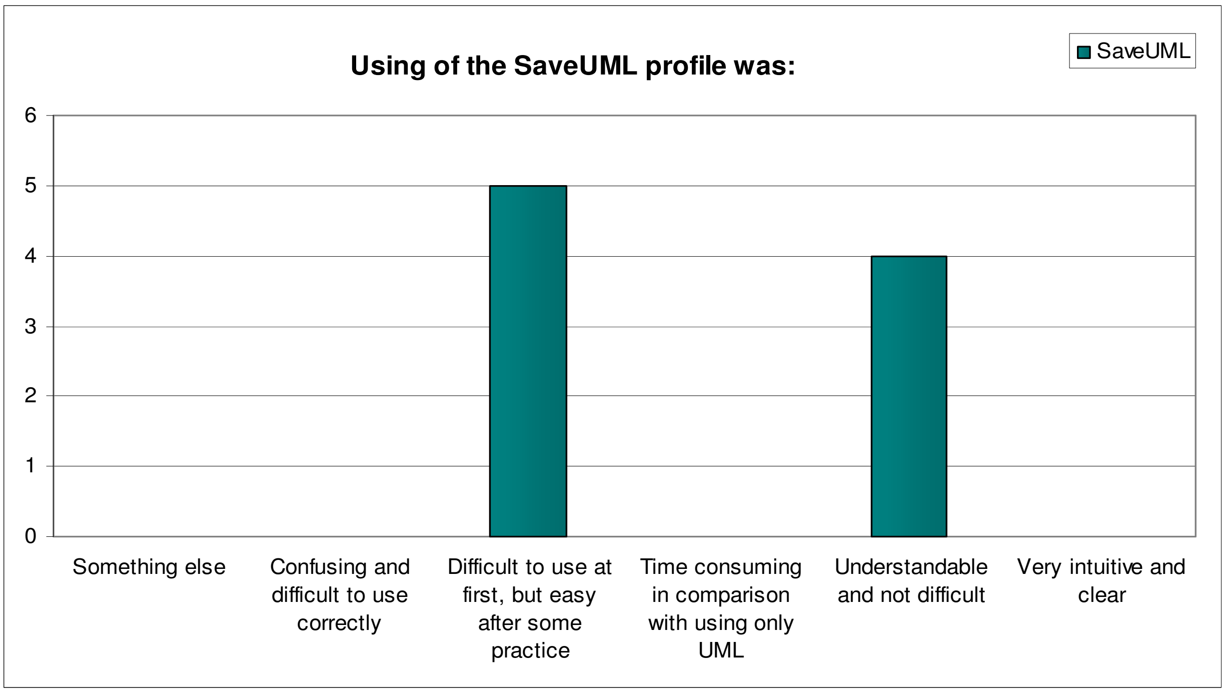


**Figure 25. SaveUML profile - difficulty of using**

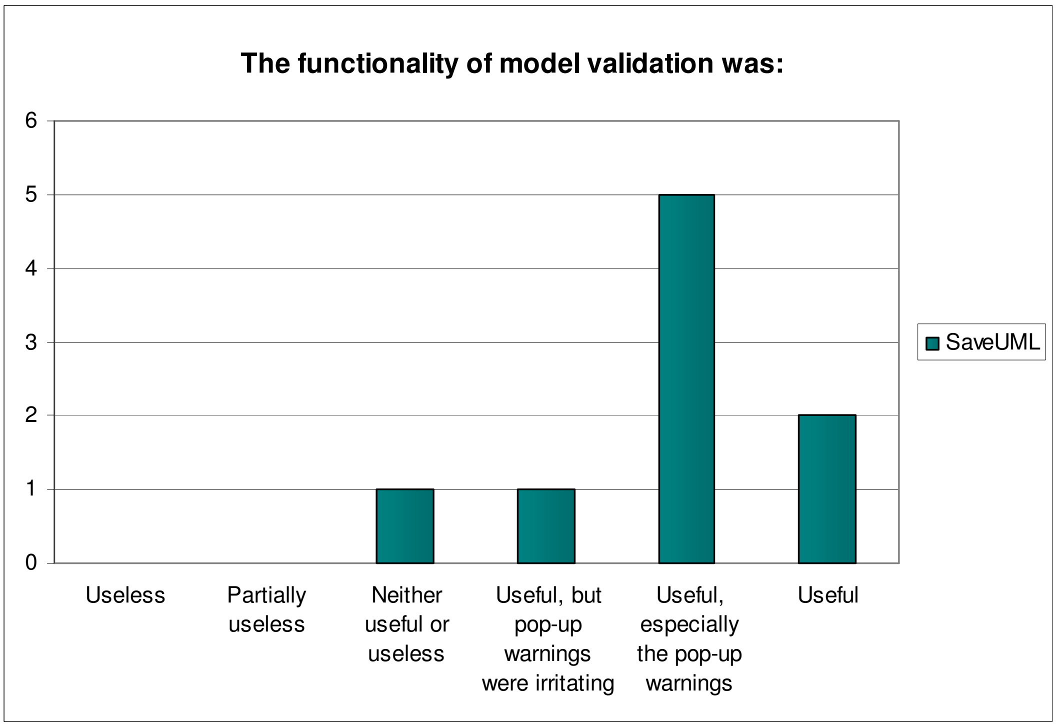


**Figure 26. SaveUML - validation experience**

The overall grade given by participants is a bit higher for SaveUML technology despite the fact that the participants had more working hours, as it can be seen from the Table 1.

**Table 1. Overall experience of using SaveUML and SaveIDE**

| Technology | Working hours | | | Overall grade |
|---|---|---|---|---|
| | Avg. | Min. | Max. | |
| SaveIDE | 6.28 | 3 | 15 | 3.78 |
| SaveUML | 8.22 | 4 | 13 | 4.11 |