A Model-based Approach to Gamify the Learning of Modeling

Valerio Cosentino[1], Sébastien Gérard[2], Jordi Cabot[1,3]

[1] SOM Research Lab, UOC, Barcelona, Spain
[2] CEA LIST, Paris, France
[3] ICREA, Barcelona, Spain
vcosentino@uoc.edu, sebastien.gerard@cea.fr,
jordi.cabot@icrea.cat

Abstract. Conceptual modeling is an essential activity in the development of any information system. But at the same time, it is a difficult one which may discourage some professionals that may be tempted to skip it altogether in favor of more code-oriented tasks. In recent years, gamification has emerged as a new approach to increase learner engagement and has been successfully applied to a wide range of training and educational scenarios. We believe gamification can play a key supportive role in teaching and learning conceptual modeling, an area where gamification has not been really applied so far.
In this sense, this paper presents a model-based approach for gamifying scenarios for learning modeling. Our approach includes a new language for modeling the gamification process itself and an environment where this new language can be embedded in current modeling tools to allow instructors and students design and use gaming scenarios all within a full modeling infrastructure.

1 Introduction

Modeling is a central activity in any engineering task. In the development of information systems, (conceptual) schemas can even be the driving force of the whole development process (what is known as conceptual-schema centric development [1], or more generally, Model Driven Engineering).

Among the plethora of modeling languages proposed so far, the ER and, lately, the Unified Modeling Language (UML) are the most popular ones. When considering the development process as a whole, modelers should also be knowledgeable in more technical languages to tune/refine the implementation of the system derived from those models. This is specially true for the database schema of the system-to-be. Almost all vendors offer powerful code-generation features that would at least cover the creation of the SQL scripts required to initialize such database.

However, learning to use the full power of modeling and technical languages can be a very difficult task due to the related theoretical and practical difficulties [2–4], thus learners may become less and less engaged over time. This is a challenge that every modeling teacher probably agrees with.

In recent years, the gamification of learning [5] is arising as a new approach to strengthen the engagement of learners, and an increasing number of research efforts

report successful experiences in applying it [6]. Nevertheless, previous gamification solutions have targeted specific scenarios and little attention has been paid to the learning of modeling.

In this paper, we propose a generic model-based approach to support gamification. By leveraging on model-driven technologies, designers can easily define a game by describing the achievements, badges and levels as well as the rules to earn them. The game definition is then used to set up the gamification enviroment where the learner actions are evaluated to assign rewards. Furthermore, our approach also provides support to collect and analyse the gamification data, thus easing monitoring activities. Our approach pays special attention to cheating prevention and user privacy, two common pitfalls in gamification.

This work has been done in collaboration with Papyrus[1] representatives, with the purpose of increasing the onboarding of new users for the Papyrus modeling tool in the Eclipse platform. Thus, we have applied our gamification framework to the specific case of learning UML, however to show the generality of the approach, we have also targeted the learning of SQL, as an example of a very different language (more technical, textual) that can be also gamified. The tool is available as a set of open source Eclipse plugins together with the UML and SQL scenarios[2].

The remainder of the paper is organized as follows. Section 2 introduces gamification, Section 3 and 4 present how gamification is modelled and the framework underlying our approach, Section 5 discusses the related work and finally Section 6 ends the paper with conclusion and further work.

2 Background

Gamification aims at incorporating the use of game elements in a non-game context to encourage and engage users by leveraging on their motivations [7].

There is no clear and commonly-accepted taxonomy of the game elements [8]. A recent work [9] has proposed a unified view of the different classifications, which summarizes the gamification dimensions to three, respectively *Components*, *Mechanics* and *Dynamics*. *Components* are the basic building blocks of gamification, they represent the objects that the users see and interact with, such as badges, levels and points. *Mechanics* define the game as a rule-based system, specifying how everything behaves and how the player can interact with the game. *Dynamics* are the top level of gamification elements. They include all aspects of the game that cannot be implemented and managed directly, and are related to the emotional responses of the users (e.g., progression, exploration).

The success of gamifying a given non-game context strongly depends on the gamification design chosen for those dimensions. Several research efforts have focused on identifying the phases composing the gamification design [10, 11]. However, and similarly to the gamification elements taxonomy, there are no commonly-accepted phases, they vary in numbers and terminology used. Nevertheless, we believe they can be summarized in two phases *Understanding* and *Gamifying*, plus an optional one *Monitoring*.

[1] https://eclipse.org/papyrus/

[2] https://github.com/SOM-Research/gamification-modeling-learning

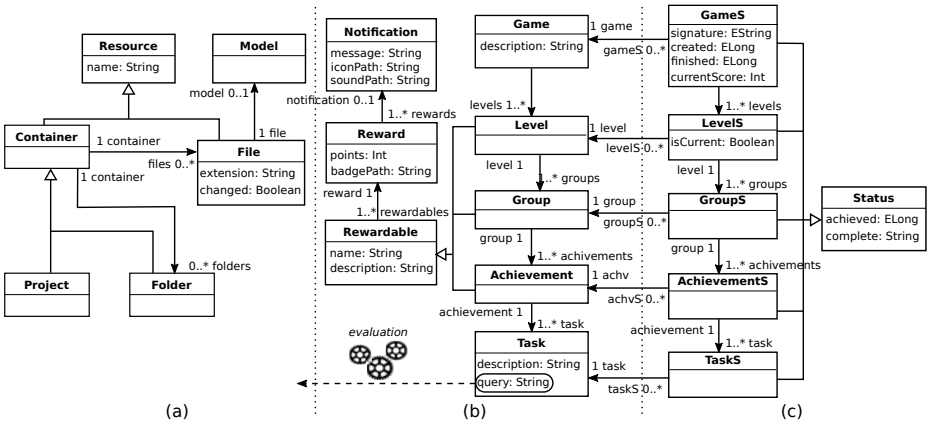

Fig. 1. (a) Project, (b) Game, and (c) Game status metamodels

The first phase includes the identification of the business goal as well as the analysis of the users and their motivations to interact with the gamified system. The second phase focuses on defining the gamification environment. Thus, it is in charge of setting up the game components and mechanics and presenting the gamification information. The third phase focuses on monitoring the gamified environment to understand the emotional responses of the users, prevent unintended outcome (e.g, cheating, decrease of interest) and drive refinements of the business goal.

The literature review suggests that gamification does work, but some caveats exist [11–13]. They may arise when gamification is not properly designed, thus leading to bad user behaviors and unintended consequences. Cheating and privacy issues are two common concerns in gamification application, since they can induce declining effects. The former can appear when users feel a disadvantage due to others cheating to easily gain rewards, while the latter bears the risk that users perceive a high level of surveillance. Thus, it is paramount for a gamification environment to define cheating prevention controls as well as to allow any user the freedom to disclose her data.

The next sections will present how our own gamification approach deals with all these concepts.

3 Modeling Gamification

To model a gamification environment, we propose to use a new Game metamodel (shown in the center of Fig. 1) complemented with a textual syntax (see Fig. 2) to easily specify such an environment.

As a complement to this core metamodel, we also need two auxiliary schemas (left and right parts of Fig. 1). The first one is needed to characterize the object to gamify, in this case, files and projects where the game mechanics will be applied. The second one to store the status of the environment for a given user. Both metamodels can be specialized depending on the specific gamification scenario.

Game and Project metamodels are instantiated at design time to model the game. The Status metamodel is only used at run-time to track the game progress.

3.1 Game Metamodel

The game definition is shown in Fig. 1b. A *game* is defined by a *description* and contains a set of levels. Each *level* (e.g., beginner, advanced and expert) contains groups, where a *group* represents a specific domain area. Examples of areas in the context of modeling could be the mastering of different UML diagrams (e.g., structural and behavioral ones). Groups contain achievements, and each achievement may require several tasks to be accomplished (e.g., create a UML file and a class in the corresponding model).

A task comes with a description and it is linked to an OCL constraint [14] that must evaluate to true for the task to be completed, i.e., the constraint must be true in the model created by the user for the task to be marked as achieved. Levels, groups and achievements are rewardable classes. The *rewardable* class has a *name* and *description* and it is associated to a reward. A *reward* is represented by a number of *points* and a badge (see *badgePath* attribute). Finally, a reward can be optionally linked to a notification. A *notification* is presented to the user when she gets a reward, it comes with a *message*, an icon and a sound (see attributes *iconPath* and *soundPath*).

While a graphical notation to instantiate all these elements in a game model could have been defined, we have opted for a textual syntax we believe is more direct in this scenario. Figure 2 shows a small textual DSL we provide for this purpose, using railroad syntax diagrams. As can be seen, the root rule is *game*, which expands to the rule *level*. *Level* expands to *group*, *group* to *achievement*, etc. This grammar matches the game metamodel described before and therefore both can be used interchangeably.

3.2 Project Metamodel

The schema of the artefacts in a (software) project is shown in Fig. 1a. As can be seen, project, folders and files are identified by a *name* (see class *Resource*). However, project and folders can act like containers for files and folders (see class *Container*), while a file stores also information about its *extension*, whether it changed after the last modification (see attribute *changed*), and optionally a reference to the corresponding model representation (if exists).

3.3 Status Metamodel

Status information is shown in Fig. 1c. As can be seen, all classes mirror a counterpart in the Game Metamodel to add stautus (see attributes *achieved* and *complete*) information to that game element for a specific game instance.

4 Gamification Framework

With the previous metamodels, we can model a gamification project. In this section, we explain how, because of our gamification framework, it is easily possible to execute the game and monitor the users attempting to perform the game tasks.

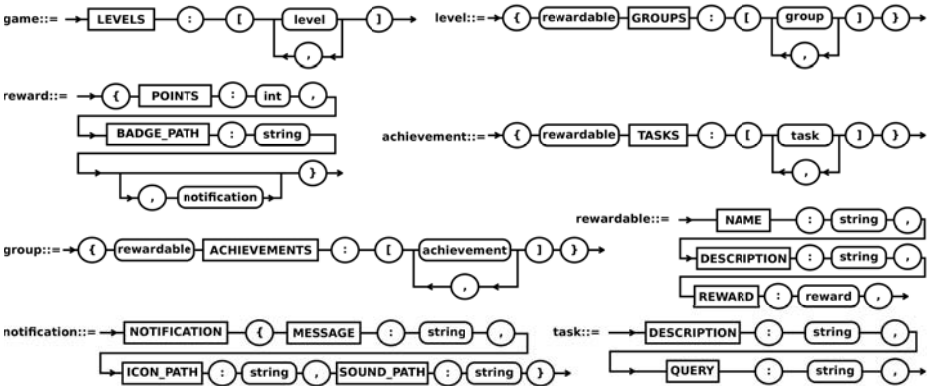

Fig. 2. Game textual syntax

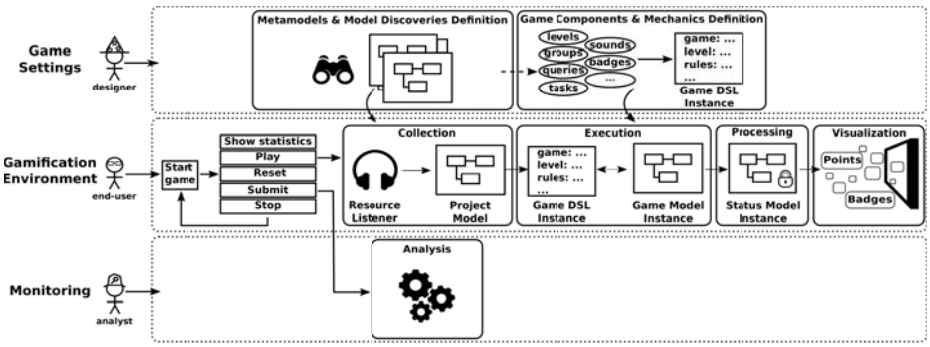

Fig. 3. Gamification framework

Our framework, shown in Fig. 3, is composed of three main components: *game settings*, *gamification environment* and *monitoring*. The first component supports the definition of games, the second one allows the user to interact with the game and progress within it, while the latter enables the definition of general metrics collecting the progress of learners.

4.1 Game settings

In the game settings component, the game designer indicates the metamodels for the relevant file extensions that can appear in the project, i.e., the types of models we are going to be dealing with in the gamification scenario. The corresponding model discoveries (i.e., the components able to parse the source files and generate the corresponding model AST from them) need also to be plugged in at this point.

At this point, the game designer also specifies the game components and mechanics using the textual language described in Figure 2 she wishes to include in the game.

4.2 Gamification environment

The gamification environment allows the learner to interact with the game as defined in the previous secton.

When the game is started for the first time, a Status model is instantiated according to the game definition, and the attributes *created* and *signature* (see Fig. 1c) are initialized at that specific time. An internal signature (based on hash values generated from this timestamp and a textual representation of the game definition) is also stored together with the status model. Every time this model is loaded later in the game, this signature is checked to make sure the user has not altered the tasks proposed to him (e.g., making them easier to accomplish). If the hashes differ, a cheating-detection message is shown to the user.

From this point on, after any action of the learner, the gamification environment is in charge of performing the following steps: (1) collect the project data where the action was performed, (2) execute the game mechanics, (3) process the gamification data generated and (4) visualize it. Each step is described below.

Collection The collection step relies on a listener that reacts to any creation or modification of resources within a project. Once a change is detected, this step generates a model that stores the information about the project where the change occurred. In case the change happened on a file, the file is marked and, if there exists a model discovery for it, its model-based representation is derived, linked to the file and added to the generated project model. The obtained project model is passed then to the execution step.

Execution This step is in charge of executing the tasks associated to each game achievement over the project model received as input. More specifically, for each achievement, the list of the corresponding tasks is retrieved. As discussed before, each task contains the name of an OCL constraint with the precise definition of what needs to be found in the current project for the task to be completed. The output of the constraints are boolean values, which are passed to the next step together with the corresponding tasks.

Listing 1.1 shows the example of an OCL constraint that requires the project (1) contains only one folder and (2) the folder is named 'MyFirstFolder'.

Listing 1.1. Example of OCL constraint

```
context Project =
inv createfolder:
  self.folders->size()=1 and self.folders->forAll(f|
                                        f.name = 'MyFirstFolder')
```

Processing The processing step receives the gamification data, calculates which rewards have been achieved and creates the notifications for the user. For each task completed, the corresponding element is retrieved in the status model and its attributes *complete* and *achieved* are updated. Then the processing step analyses which achievements

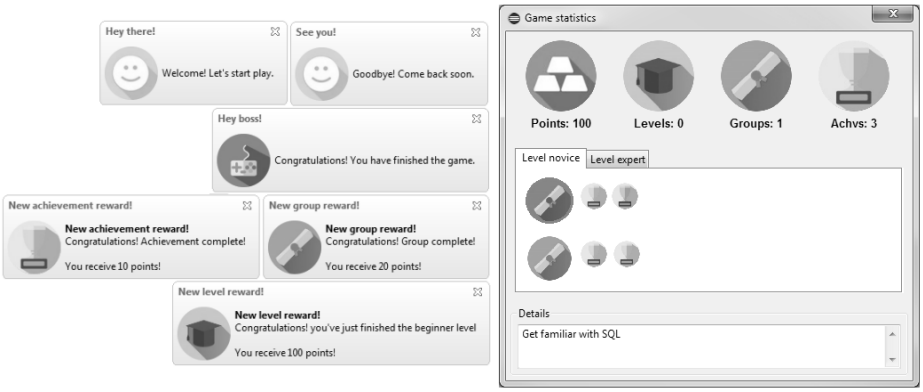

Fig. 4. (a) Game notifications and (b) statistics

have been accomplished (the ones that have all tasks completed), updates the corresponding status information, and in a similar way, it repeats the operations for groups and levels. For each *rewardable* element completed (i.e., *Level*, *Group*, *Achievement*), the corresponding points are added to the current score of the user, and a notification containing a sound, a message, the points and the badge earned is created. Note that, in case all the groups of the current level have been completed, the value of the latter is updated to the next level (see attribute *isCurrent* of *LevelS*). Furthermore, if the user has finished the game, a notification is created to acknowledge the user and the status of the game is updated (see attribute *finished*, in the *GameS*).

The status model is saved every time a reward is achieved in order to avoid information loss in case of application failures. Furthermore, the model is also encrypted before any saving operation to prevent cheating.

Visualization The visualization step shows the game information to the user using notifications and a global summary of her progress. It is worth noting that also this step prevents cheating, since the information is presented in read-only mode within the gamification environment and not externalized (e.g., via a web page). Thus, avoiding malicious modifications of the game status information.

When the user starts, stops or wins the game and for each reward earned, a notification shows up in the gamification environment. Figure 4a shows the default notifications used in the gamification environment. The first and second rows contain notifications that appear when the user starts, stops and finishes the game. The remaining rows include notifications used to inform the user about achievements, groups and levels completed.

The user can also check at any moment the game statistics, which contains both the instructions to get rewards and her current advances. As can be seen in Fig. 4b, general game statistics about the number of levels, points, groups and achievements completed are shown on top. Information concerning the status of each level is presented in the middle using a tabbed document interface, where achievements and groups still not

finished are gray-shaded. Finally, details about groups and achievements are shown at the bottom, when clicking on the corresponding images.

4.3 Monitoring

The monitoring component can be used to support the definition of gamification analytics [15] by collecting and analysing the advances of users. However, in order to protect the user privacy, this component is defined in a non-intrusive way. Thus, the user is free to share at any moment information regarding her progress to the remote server configured to process, store and analyze the game information. Not sharing that information does not affect in any way her game experience, it just limits the collective data on the "teacher" side.

5 Related Work

This section reviews previous works on gamification and learning of modeling. They are discussed below.

5.1 Gamification

Many works in the field of gamification present extensible and generic approaches [16–18], assist designers in the definition of games [19, 18] and provide gamification environments [20, 18]. Nevertheless, they do not cover all the aspects that our framework tries to address (for instance cheating and monitoring are often overlooked in previous works). Furthermore, although a language has been already proposed for modeling gamification concepts [19], it cannot be used in our context. In particular, it is not clear how the user actions within the model representation of a file (e.g., creating UML classes of a given type) can be described and evaluated.

5.2 Learning of modeling

Several works attempt to address the complexity of learning the act of modeling. They can be grouped in two categories, didactic-based and game-based approaches. The former [21–24] provides automatic feedback on the user attempts to solve an assignment, but they do not include game elements. The latter [25, 26] do rely on game elements to support learning. However they adopt Serious Games solutions, thus the learner plays in a fully fledged virtual game environment. As far as we know, no approach is really proposing gamification to learn modeling.

Moreover, our framework tries to overcome different limitations identified in previous works. They are presented and compared with our proposal below.

– **Extensibility.** Modeling is an activity that can be performed using different languages. Some previous works focus on SQL [21, 22, 25, 26], while other ones on Entity-Relationship diagrams [27, 28] and UML [23, 24]. However none of them comments on possible extensions to include other modeling languages. Conversely, our framework can be easily extended to include learning support for other modeling languages thanks to its model-based infrastructure.

– **Genericity.** Two different approaches have been used by previous works to assess the correctness of learners' solutions. The former, and most common, consists in matching the proposed solutions to pre-calculated correct ones [22, 24]. However, since the matching is a computationally expensive operation, the range of problems that can be solved is generally limited. The latter relies on a set of constraints that the learner's answer has to satisfy in order to be accepted as a correct solution [21, 23]. Constraint-based approaches are computationally less expensive and that's also the direction we follow. Nevertheless, previous works use ad-hoc constraint languages that cannot be easily reused in other scenarios. In our framework, learner solutions are evaluated by constraints written in OCL, which is a standard language widely used in MDE. Thus, it is generic enough to be reused in different model-based scenarios.
– **Cheating.** Only two works [21, 22] discuss or define strategies to detect cheating (or plagiarism). Our framework pays special attention to this issue by encrypting the current status of the game in order to avoid malicious modifications.
– **Privacy.** None of the previous work mention measures to protect learner privacy. Our framework allows the learner to decide whether her advances in the game can be collected and analysed.

6 Conclusion

In this paper we have presented a generic model-based approach to support gamification, which can be easily applied to learning modeling scenarios. Our framework provides means to prevent cheating and protects user privacy, and by leveraging on Model-driven technologies, it is also easy to extend and generic.

As further work, we would like to predefine a set of gamification learning paths and exercises to help individual users to benefit from the approach. We are also interested in completing the monitoring component by providing default analytics and extended game session recording information for users, thus enabling a deeper analysis of the evolution of the students of a "class". Finally, we plan to provide a graphic pattern-based language to express the game tasks, to be used instead of OCL for those users that prefer a more graphical (though less expressive) language. This was one of the first feedback we got from the Papyrus community. Getting more validation from them is also part of our future work.

References

1. Olivé, A.: Conceptual schema-centric development: a grand challenge for information systems research. In: CAISE. (2005) 1–15
2. Lu, H., Chan, H.C., Wei, K.K.: A survey on usage of sql. SIGMOD Rec. **22**(4) (1993) 60–65
3. Siau, K., Loo, P.P.: Identifying difficulties in learning uml. ISM **23**(3) (2006) 43–51
4. Erickson, J., Siau, K.: Theoretical and practical complexity of modeling methods. Commun. ACM **50**(8) (2007) 46–51
5. Kapp, K.M.: The gamification of learning and instruction: game-based methods and strategies for training and education. John Wiley & Sons (2012)

6. Dicheva, D., Dichev, C., Agre, G., Angelova, G.: Gamification in education: A systematic mapping study. Educ. Technol. Soc. **18**(3) (2015) 75–88
7. Deterding, S., Sicart, M., Nacke, L., O'Hara, K., Dixon, D.: Gamification. using game-design elements in non-gaming contexts. In: CHI. (2011) 2425–2428
8. Pedreira, O., García, F., Brisaboa, N., Piattini, M.: Gamification in software engineering–a systematic mapping. IST **57** (2015) 157–168
9. Shpakova, A., Dörfler, V., MacBryde, J.: The role(s) of gamification in knowledge management. In: EURAM. (2016) 1–40
10. Mora, A., Riera, D., Gonzalez, C., Arnedo-Moreno, J.: A literature review of gamification design frameworks. In: VS-Games. (2015) 1–8
11. Webb, E.N.: Gamification: when it works, when it doesnt. In: DUXU. (2013) 608–614
12. Hamari, J., Koivisto, J., Sarsa, H.: Does gamification work?–a literature review of empirical studies on gamification. In: HICSS. (2014) 3025–3034
13. Callan, R.C., Bauer, K.N., Landers, R.N.: How to avoid the dark side of gamification: Ten business scenarios and their unintended consequences. In: Gamification in education and business. (2015) 553–568
14. Warmer, J.B., Kleppe, A.G.: The object constraint language: getting your models ready for MDA. (2003)
15. Heilbrunn, B., Herzig, P., Schill, A.: Tools for gamification analytics: A survey. In: UCC. (2014) 603–608
16. Piras, L., Giorgini, P., Mylopoulos, J.: Acceptance requirements and their gamification solutions. In: RE. (2016) 365–370
17. Piras, L., Paja, E., Giorgini, P., Mylopoulos, J., Cuel, R., Ponte, D.: Gamification solutions for software acceptance: A comparative study of requirements engineering and organizational behavior techniques. In: RCIS. (2017) 255–265
18. Sripada, S.K., Reddy, Y.R., Khandelwal, S.: Architecting an extensible framework for gamifying software engineering concepts. In: ISEC. (2016) 119–130
19. Herzig, P., Jugel, K., Momm, C., Ameling, M., Schill, A.: Gaml-a modeling language for gamification. In: UCC. (2013) 494–499
20. Herzig, P., Ameling, M., Schill, A.: A generic platform for enterprise gamification. In: WICSA. (2012) 219–223
21. Sadiq, S., Orlowska, M., Sadiq, W., Lin, J.: Sqlator: an online sql learning workbench. In: SIGCSE Bull. Volume 36. (2004) 223–227
22. Russell, G., Cumming, A.: Improving the student learning experience for sql using automatic marking. In: CELDA. (2004) 281–288
23. Sedrakyan, G., Snoeck, M.: Technology-enhanced support for learning conceptual modeling. In: EIS. (2012) 435–449
24. Py, D., Auxepaules, L., Alonso, M.: Diagram, a learning environment for initiation to object-oriented modeling with uml class diagrams. JILR **24**(4) (2013) 425–446
25. Soflano, M., Connolly, T.M., Hainey, T.: An application of adaptive games-based learning based on learning style to teach sql. Comput. Educ. **86** (2015) 192–211
26. Soflano, M.: Modding in serious games: Teaching structured query language (sql) using neverwinter nights. In: Serious Games and edutainment applications. (2011) 347–368
27. Hall, L., Gordon, A.: A virtual learning environment for entity relationship modelling. In: SIGCSE Bull. Volume 30. (1998) 345–349
28. de los Angeles Constantino-Gonzaléz, M., Suthers, D.D.: A coached collaborative learning environment for entity-relationship modeling. In: ITS. (2000) 324–333