

An Educational Development Tool Based on Principles of Formal Ontology

Rodolfo Guzzi

ISAC CNR Via Gobetti 101
Bologna, Italy
r.guzzi@isac.cnr.it

Stefano Scarpanti, Giovanni Ballista, Walter Di Nicolantonio

Carlo Gavazzi Space at ISAC, CNR
Bologna, Italy
s.scarpanti@isac.cnr.it
g.ballista@isac.cnr.it
w.dinicolantonio@isac.cnr.it

Abstract

Computer science provides with virtual laboratories, places where one can merge real experiments with the formalism of algorithms and mathematics and where, with the advent of multimedia, sounds and movies can also be added. In this paper we present a method, based on principles of formal ontology, allowing one to develop interactive educational tools. The structure of our system starts from general considerations on knowledge itself moving on to formal ontological principles in order to obtain a robust knowledge frame and good awareness of knowledge levels involved during the teaching process. Our system is split into object-knowledge - the knowledge of the phenomenon to be taught - and meta-knowledge - i.e., how to teach it. Using a trip (journey) metaphor, together with the flexibility of semantic graph representations, we define the reference frame and we apply it to a case study related to the Planetary Missions, the subject of our research. This tool is provided with a multimedia interface to show the results of several current missions, but may be implemented with new missions.

Keywords

Formal ontology principles, Edutainment, Knowledge graph, Knowledge representation.

Introduction

With the advent of multimedia technology in computer science, the word edutainment (a combination of the words education and entertainment) has become broader in scope. In multimedia computers, the virtual laboratory, a room where experiments can be carried out, merges real experiments with the formalism of algorithms and mathematics, by means of sounds and movies. Even though at a first glance several words are not needed to justify the use of multimedia support to have entertainment in education, given its strong appeal to users, it is necessary to be aware of the importance of the use of entertainment within the framework of the dynamics of teaching.

Several web sites in which edutainment is used, are poor and not really effective: the use of movies or images is marginal and ornamental. There is no real interactivity and the content is worse than a poorly written book, so that the level of teaching is not sufficient for learning. To better analyze the dynamics of teaching, especially in the web case, it is suitable to take advantage of the concept of control in teaching and its transfer from the teacher to the learner, a sharp way of analysis born in the 90s, even before the web widespread diffusion.

Educational web sites can be analyzed by point of view of transfer of control. Our analysis may start from the following two cases as illustrative of several other cases. Without any intent of criticism we refer about two different approaches selected as representative of the matter under discussion. In NOAA education resources site (www.education.noaa.gov) the relation between users and the computer environment, the transfer of control is parcellized in different parts: those primarily dedicated to teachers, those dedicated to students and the cool sites for everyone. In such approach the web acts as a remote book with beautiful pictures with some quick and brief movie but not with a true interactivity. In a second case the NASA site (quest.arc.nasa.gov) the transfer of control is supported by a minimal interactivity mode with related articles and guides on space exploration. In both cases two actors are identified by the transfer of control: the person learning (that is supposed to learn) and the teaching environment (or better the environment in which teaching should be carried out). Then when a teacher is not physically present it is evident that the best way to learn is by a certain degree of computer interactivity. Initially, the learner is passive because she/he does not have control of the environment. The

simplest way is to make a tour of the environment giving one an idea of what and how it is. Afterwards, the user becomes more active and interacts with the system. This stage is not necessarily the stage of understanding, but it will be the stage of pre-comprehension, which varies from person to person. Only when there is complete control of the interactivity can the users enter into the environment of the teaching system and become protagonists of their own learning within, of course, the limits of the information content of the system itself.

Since learning strongly depends on personal attitudes, it is difficult to define, a-priori, which is the best approach: deduction or induction, or both. Probably the deductive approach is more natural for software tools, in which rules are defined a-priori.

The aim of this paper is to present a method allowing the development of an educational tool. This method has been implemented in our multimedia laboratory and has been applied to space missions, which is the subject of our research. It may also be used for other subjects. Our approach is based on formal ontological principles, which are summarized in the following sections. Then we present the structure of our system and its semantics and, finally, the applications we have developed.

Ontology & Formal Ontology

Recently, computer science and, in particular, artificial intelligence have developed a new discipline called formal ontology. Formal ontology is an extension of the more traditional knowledge representation field because generic knowledge representation strategies are not sufficient to achieve the goal of engineering a piece of knowledge into a software system. This means that new tools or strategies are needed to find the best structure of knowledge, its disaggregation, relations between its atoms, and the hierarchy of its parts. Since it is currently considered worth thinking of knowledge as an entity by itself, before planning its engineering, it is important to outline which parts of it are devoted to specific domains and which are more general. This is why it is necessary to switch from knowledge considerations to ontological considerations.

Briefly, formal ontology has been defined, (Cocchiarella, 1991), as: "the systematic, formal, axiomatic development of the logic of all forms and modes of being ". Even though the genuine interpretation of the term "formal ontology" is still a matter of debate, it takes into account both the meanings of the adjective "formal": that is synonymous of "rigorous", while ontology means "related to the forms of being". Therefore, "what formal ontology is concerned with is not so much the bare existence of certain individuals, but rather the rigorous description of their forms." (Guarino, 1995).

Formal ontology provides several hints about knowledge treatment in a system but here we are only considering those related to the knowledge structure. Thus, our approach is based on:

- an analysis of what is known about a certain argument,
- its subdivision into different homogeneous pieces, let us say atoms,
- the understanding of what relations there are among them, and
- the selection of some of these parts, that are explicitly devoted to the software system for handling the knowledge and above all for handling its ontological part.

This involves the sharing between the static knowledge engineered into the system (that is the unalterable part, the part constituting the system) from knowledge that is alterable and may be modified by the user (in our approach the teacher). Although we will not go here deeper into the technical aspects concerning formal ontology, we describe how we have used formal ontological principles. From our point of view, an ontologically correct educational software system needs a proper knowledge representation, which will be presented in the next section, in which some of its components are devoted to handling explicitly ontological problems and then to carry out the computation of the knowledge transfer. From several points of view it is crucial to have some components devoted to knowledge processing: first because it increases the transparency of the system, and also because it allows an easy identification of those parts to be considered and modified for knowledge upgrades or changes.

Knowledge representation

As mentioned above, knowledge is represented by its own domain and its transfer educational structure. The former is the object of the system, that is to say the domain of the teacher's knowledge which one handles

explicitly and thus can be changed. The second is the system, i.e. the way by which the knowledge is transferred and engineered into the system and which cannot be altered by the user.

In our approach, the transfer of the domain knowledge is based on the assumption that any teaching process may be represented as though it were a description of a trip. So our system is based on the trip cycle, as shown in figure 1. This approach contains the essence of teaching. In fact the teacher, introducing and explaining an argument, starts from certain assumptions, considered the initial set up, develops them by intermediate states, by concepts, and reaches the final destination which is the conclusion of his reasoning. Since our system is also dedicated to the user, we have also introduced the concept of error, as though it were another path, different from the correct path, which produces a warning and induces another trial.

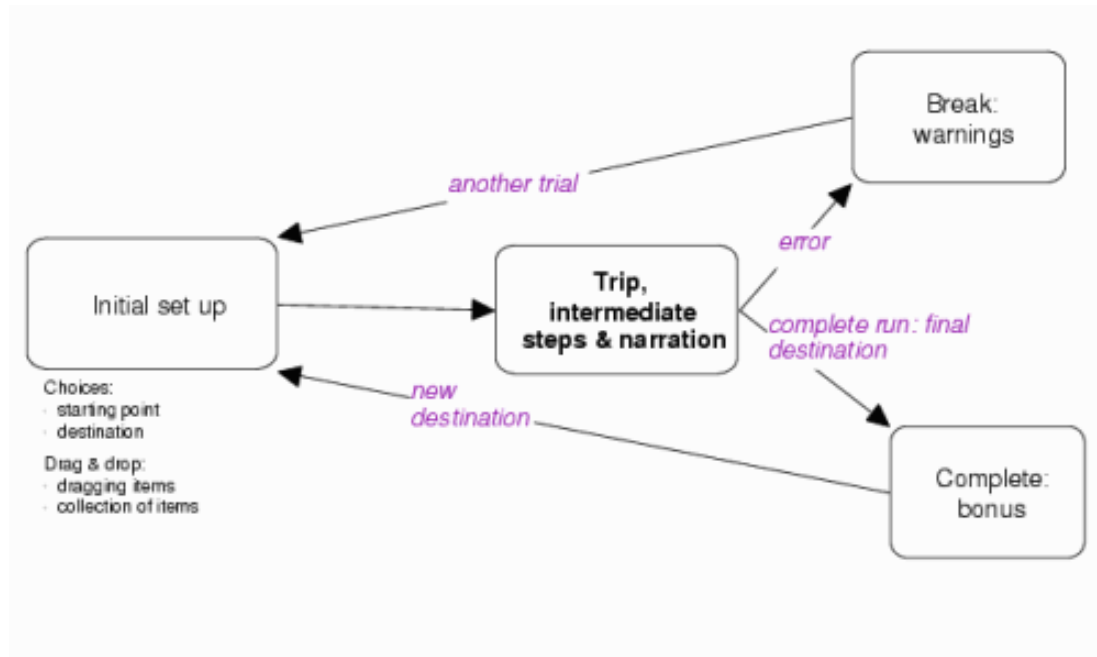


Figure 1. State diagram: the trip cycle with bonus or warnings

This linear, mono-dimensional, structure can be applied to describe the phenomena which are the subject of the knowledge, in a very natural way. So, regardless of the complexity of the phenomena to be treated, we can always describe them by a linear procedure. It is worthwhile pointing out that we are not building a model of something, but only a description of it, so we do not use the approach of many modeling systems, in which there are feedback loops to represent dynamic systems, and the evolution of their quantities. We are simply describing a process choosing a starting point and a final destination, with several intermediate steps, according to the level of detail of the description chosen. Since in the description of phenomena or events we may have many starting points and many destinations, many paths linking them together, and paths passing through inner steps, our system is also able to represent all those figures.

Semantics

Let us now show the semantics we use to describe the process in our domain knowledge. The semantics of our language is defined by categories, classes and descriptors.

Categories are sets of classes representing the same status or event. Categories are only conceptually and do not enter into the software development. They are used to describe the potential links between sets of classes. For these reasons we do not describe their use although they are included in our Planetary Mission project, to render clearer the structure of our semantics.

Classes are the elements in which the narration has been segmented. They represent the minimal self-consistent information in which the subject of knowledge has been divided. They contain a certain number of descriptors and are the main elements of our semantics.

Descriptors are primitive elements which define the subject of the knowledge and allow the input of its relation to classes. Each descriptor is associated with a node, which can be considered the physical element which links one descriptor to another by arrows. The strict relationship between each node and its descriptor and the connection between one descriptor and another defines a graph.

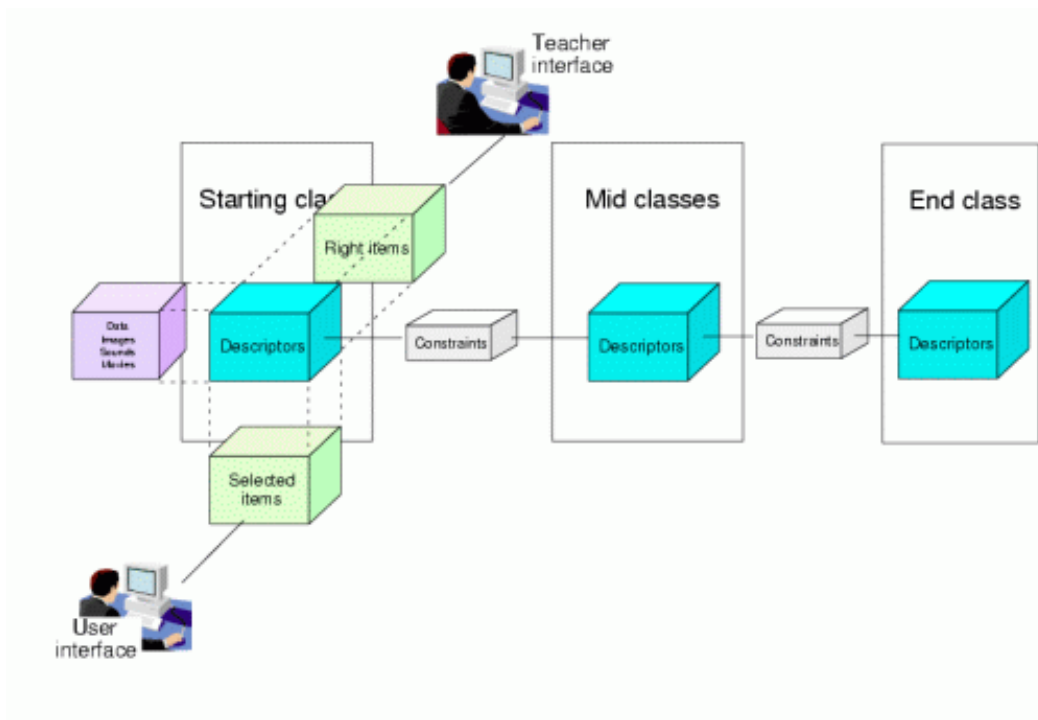


Figure 2. Elements of ontology. Descriptors inside classes shown

The domain knowledge is segmented into self-consistent pieces, so the whole domain knowledge is fragmented but complete. In figure 2 ontology elements and its semantics are shown. Each class may contain several descriptors, since there are different choices or potential paths that can be used during the narration. Each class, being a step of the narration, contains different descriptors to follow different paths of the narration but equivalent from the point of view of semantics. Descriptors and nodes are connected together to form a graph that is the plotted structure. Nodes contain the constraints of the narration. The graph formed by classes and nodes summarizes the plot of the knowledge and represents the logic links. All possible events occurring in the plot are shown by graphs.

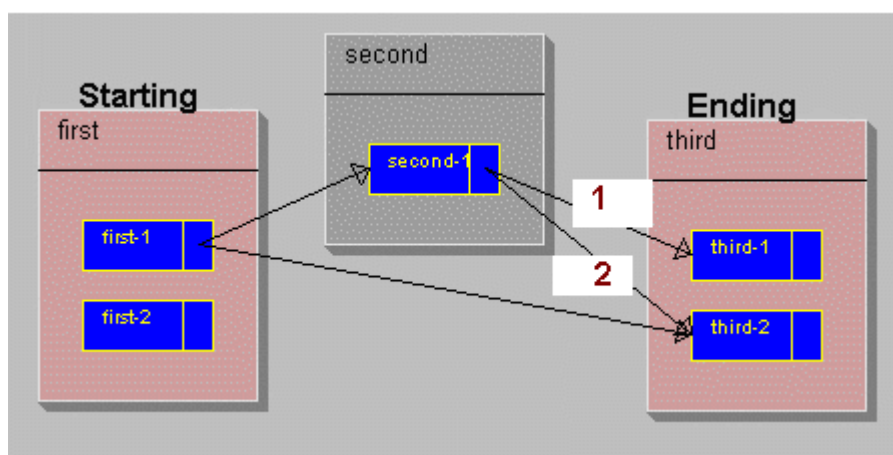


Figure 3. Frame of knowledge segmented into classes

We might consider classes and descriptors as semantic structures, while the nodes, with which they are associated, are their syntactic counterparts (see figure 3). In order to make a distinction between two or more descriptors in the same class, some constraints are to be added to descriptors by the constraint buttons as shown in figure 4. An example of how some constraints, between different classes, are to be inserted is shown in figure 5.

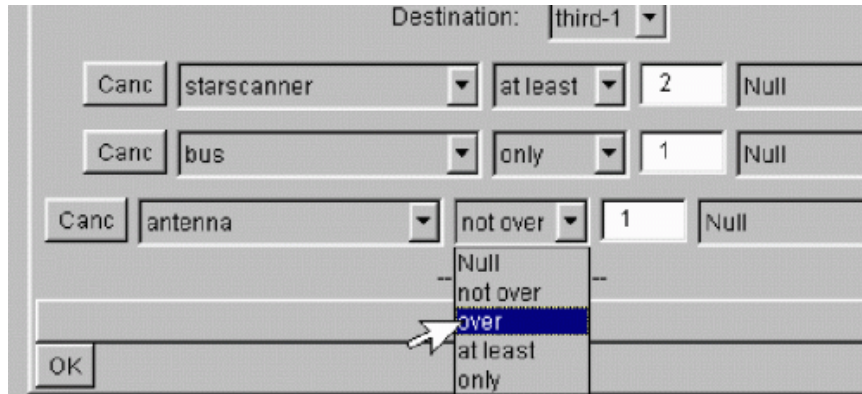


Figure 4. Example of constraint buttons for local paths inside the class

Since, as mentioned above, the system is based on different paths related to the subject of the knowledge domain, during each run of the system only one path will be taken, and many runs will be typically needed by the user to understand the subject being learned. Then the whole domain knowledge will be built as a net, in which each knowledge atom will be linked to others to form the whole net information necessary.

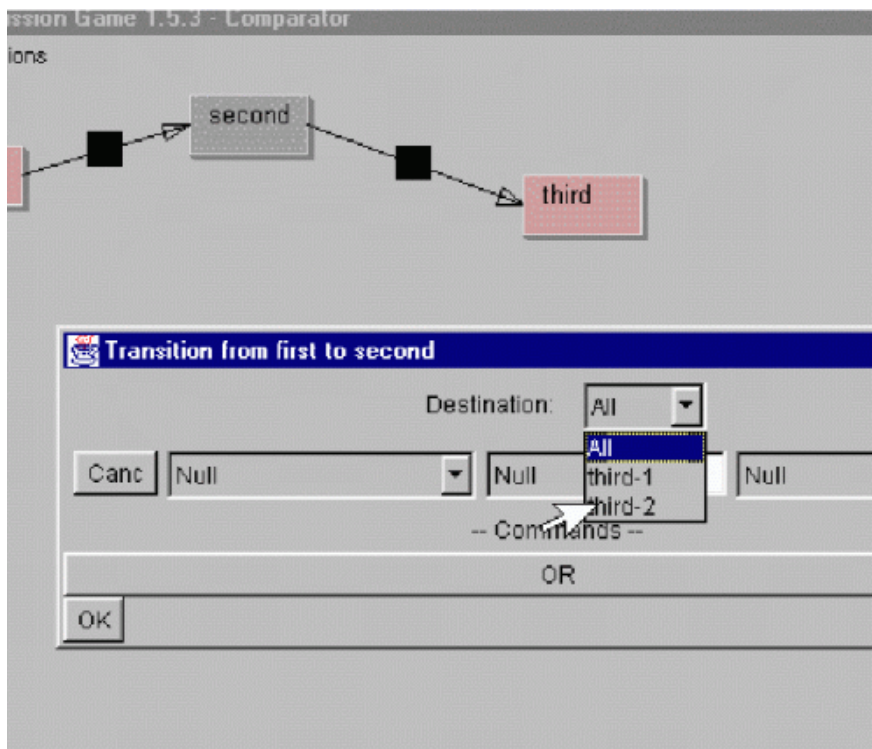


Figure 5. Graph between classes and constraint buttons applied to nodes

Knowledge graph

Since the graph is a very common data structure to represent conceptual knowledge (Luger et Stubblefield, 1998) or semantic networks (Nilsson, 1998), as shown in Figures 5, our graph is the set of nodes and arrows connecting them (Luger et Stubblefield, 1998). Each node has a label to distinguish its descriptors. Drawing arrows between nodes means mapping out possible paths in the narration; we do not deal with graphs in which

there are possible loops, because our semantic structure does not allow such structure in paths. Vice versa our semantics may manage forks from each node because we may deal with graphs and not simply with trees which are simpler graphs. The use of graphs instead of trees gives more flexibility to the plotted description because we may have multiple paths leading to the same result.

Each run of the system takes a path of the narration and this path is a level of knowledge acquired by the learner. The whole domain is only completely described by all paths, so a good learning system implies several trials, that is to say several runs. Then the entire knowledge can be taught as compounded by the descriptors graph (path graph) plus the conditions for proceeding. The minimum necessary conditions for paths among classes are: not cyclic and at least one complete path. Without at least one complete path, the classes graph is nonsense, because one misses some links and has no complete information needed to run the system.

Class graph & conditions

Once we have built up a graph node, we also obtain a class graph. Building relations between nodes also means also building relations between the descriptors contained in each class. The system uses link nodes to infer link classes. Link classes can be labelled, it is possible to put a label, that is to say a condition for proceeding, controlling the direction of the whole narration along that path. These conditions can break the narration if false and trigger a warning to the user. Vice versa, the narration proceeds to the next class. Each class may reach the next only if conditions defined in arrows linking classes are satisfied. Null conditions are considered true.

A typical sequence of classes, linked one by one as in a list, is shown in figure 5. In addition to this simple structure, branches may also be parallel, that is to say one may have forks from some classes. Conditions between classes are based on Boolean logic, with AND and OR operators. Each OR-branch is labeled by a selected destination or by ALL destinations. This allows a partial conditioning, i.e., to condition only a path leading to a specific destination, in order to be able to distinguish some paths from others according to the final destination. This is why we may consider OR-branches as case-branches. AND-branches are nested into OR-branches. The logical sentence we adopt is the disjunctive normal form (DNF as defined in Barwise et Etchemendy, 1993). At the bottom we have conjunctions, AND-relations between atomic conditions, and above disjunctions, OR-relations, which is a standard way to represent every complex logical proposition.

Operation on the system

The operations to be carried out on the system are defined by teacher and user modes.

Teacher mode. In this mode one defines all possible steps and paths and all possible statements (and their features and constraints) to begin and carry out the trip. The teacher mode acts on classes and descriptors of the system producing directories containing images, data, movies and sound files related to the topics of the domain knowledge selected. In teacher mode it is also possible to define the conditions to proceed through the steps.

User mode (or learner mode). In this mode, the user can make choices to define the main parameters of the trip, or to find the atoms of knowledge, which are defined as paths among all those possible. The user collects all the items necessary to define and complete the trip. During this phase, the learner acts as though he were teaching himself, selecting the best items or statements describing the domain knowledge. Every time the step forward is subject to an error, the cognitive feedback system gives a warning to correct the procedure the user has set.

Between the teacher mode and user mode there is a unit, named *Comparator*, in which the domain knowledge is explicitly transferred by means of simple rules defined in teacher mode. This unit maintains and uses the *transfer educational structure* which is statically engineered and which cannot be altered by any users, neither teacher nor learner. The teacher mode acts to modify or implement the knowledge but not the structure of the comparator. It uses the domain knowledge, properly prepared in teacher mode, comparing the choices made by the teacher with those made by the users (learners), carrying out the task of interactively teaching through warnings and suggestions when learners make errors. In figure 6 we can see the two interfaces linked to the system kernel. One is used to set up the knowledge (teacher mode), the other is used to learn (user mode).

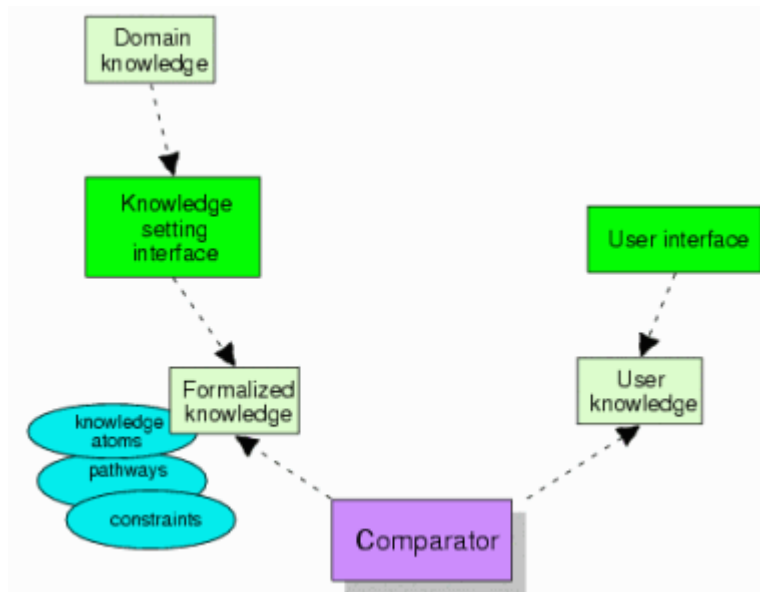


Figure 6. The comparator between the teacher and the learner mode

Graphics interfaces

Let us now show how a graph is built and its graphics interface designed from scratch on the basis of our semantics. We may start by activating the teacher mode. If the computer window is blank, we need to create classes and related descriptors by simply clicking on the classes and embedded descriptors using the relative buttons (see figure 7). They produce the corresponding icons in the computer window and directories on the computer hard-disk.



Figure 7. Main button field

Operations are defined in teacher mode as relationships among all the ontological parts. They are: possible transition between areas, condition of item transferring, variables monitoring the system (linked to items and areas properties — displayed), conditions for proceeding between classes (linked to lists of items in areas) and special conditions (global conditions on system variables). When all the constraints and connections have been defined the system is saved and only the teacher can modify it.

The graphics user interface is, on the other hand, what the user deals with in *user* or *learning mode*. The user will find all the items describing the knowledge domain. The interface contains: storage areas, empty, storage areas with icons, decision selectors to carry out the trial/path in domain knowledge, and some monitors showing system variables. Thus the user, simply, drags and drops inside a proper iconized area descriptors in order to define the logical links between the steps of the narration. Only when the construction is well defined and correct will the system run otherwise it shows a warning with information on possible corrections.

Application: a case study

Space Mission Games

Using Formal Ontology as presented above, we have built up an educational tool about planetary missions. We referred to some detailed scientific documents on space mission design (Wertz et Larson, 1999; Doody, 2001) to identify the main parameters considered as affecting a generic space mission. Several real planetary missions

documented by NASA (JPL-NASA at <http://www.jpl.nasa.gov/>, NASA missions at <http://www.jpl.nasa.gov/missions/>, and NASA database at <http://photojournal.jpl.nasa.gov/>) and ESA (ESA, 2004), have been considered to tune mission parameters and main items with real mission quantities. We have fragmented a hypothetical mission into several pieces, mission classes, and within these we have identified several possible different cases. These cases, descriptors, are the nodes of our graph by which we outline all the different ways to carry out a planetary mission. Each node descriptor, in our case defined by the components of the mission, rocket, probe, target planets and related to the instrumentation allowing the fulfillment of the mission, are linked to movie fragments which are edited in a whole sequence when the mission is set up. A proper interface to QuickTime has been built up for AVI and MPEG movies. The movie shows the mission as it proceeds and its results, following the graph path from the starting node (the launcher) to the final one (planet destination).

Classes & Descriptors

In this Space Mission Game, we have three formal categories: launch, trip, exploration. They may be subdivided into the following classes (plot segments): platform, lift-off, flyby, planet target, arrival, probe, etc. The corresponding descriptors, contained in classes, are: types of launchers (static and/or dynamic state), different flight dynamics, different planets, different landing approaches, different probes, etc.

Connections: Links & Conditions

Let us show, by examples (figure 8) how to set up teacher mode and how to plan to reach Mars. We may select Ariane as launcher (static mode). The first descriptor inside the Platform class is, in fact, Ariane. It is linked, by an arrow, to Ariane in Lift-off class (dynamics mode), and then to Mars, in planet target class and so on. Since the system accepts more links, we may add other pathways, for instance starting with the launcher Saturn or other launchers instead of Ariane, always leading to Mars. The link is simply obtained by dragging the arrow between the descriptors. Constraints are introduced by clicking on the arrows as shown in figures 4 and 5.

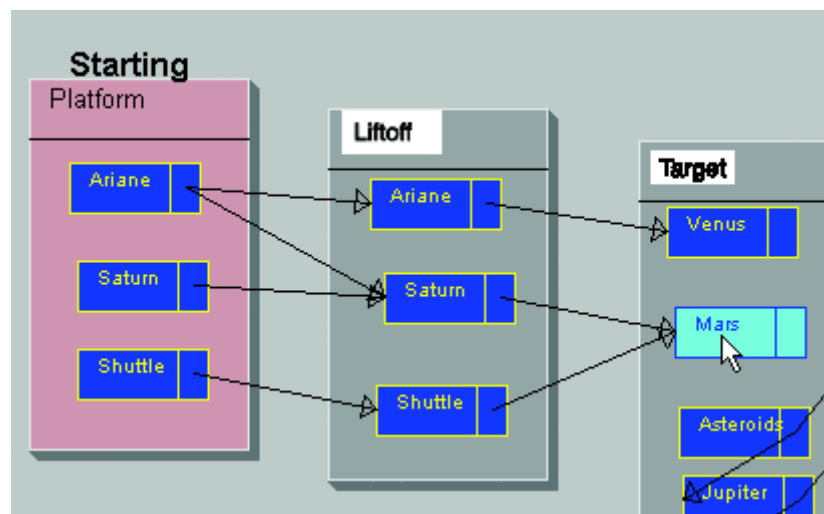


Figure 8. Example of configuring, by multiple arrows, the system in teacher mode

In user mode, see figure 9, users select the scientific mission (for instance to study the magnetic field, atmosphere, etc.) and the target planet (from Earth to Uranus). All the components are introduced inside the rocket silhouette by drag&drop operations, which are in the right part of the picture. In the upper left the chosen instrument for the scientific mission, in the middle the instruments to perform the space mission (from the navigation sensors to power, etc) and in the lower part the suitable amount of fuel.

When the required conditions are defined the space probe is set up for its mission. The Launch Button allows the mission to run and to be visualized. Vice versa when a set up error occurs, a warning related to the mission failure (see figure 10) is shown. The warning also contains a link to the space mission manual where there is information allowing the user to understand his error and correct it.



Figure 9. Graphic user interface of Space Mission



Figure 10. Warning example

Conclusions

In this paper we have presented a method to develop an interactive educational tool able to describe a self-consistent process of knowledge learning based on the description of a defined phenomenon. The structure of our system is based on formal ontological principles for a robust knowledge frame and good awareness of knowledge levels involved during the teaching process. The knowledge frame is expressed by categories (only formal), classes and descriptors. Using such items, semantic rules based on graphs, in conjunction with Boolean logic, have been developed.

The system contains two main units: one used by the teacher (to set up the knowledge) and one used by the learner (to learn by trial and error).

In teacher mode the system is mainly based on two features: paths between classes and related constraints between classes. By these two approaches the learning system is conditioned. In user mode, on running the system, the user knows which events are encountered in each class, and from the conditions, the user knows whether he will be able to proceed or not from one class to another (of course along its path), discovering the information contained in the system.

The user will have to choose or to set up elements and their related properties, which will be evaluated during the system run. In order to make the system attractive, a user friendly interface has also been created.

The advantage of our approach is that it is based on a mono-dimensional linear procedure resembling the form of a narrative. This structure is easily transferable into a software tool which, by using the GUI tools, allows one to use the computer as though it were a piece of paper on which a pencil can draw a knowledge graph.

The first release of an educational tool has been set up and called Space Mission and it is devoted to a planetary mission. It has been presented in several exhibitions to acquire feedback and in order to be closer to user requirements. The open structure of our educational tool allows it to be applied in other contexts. We intend to apply the current structure in other fields of application, including medicine, in order to evaluate its flexibility and its ontological robustness in depth. Space Mission can be downloaded from: <http://yoda.bo.isac.cnr.it/SpaceMission/> It runs on Windows and MacOS X operating systems. A guide is also included.

References

Barwise, J., & Etchemendy, J. (1993). *The Language of First Order Logic* (3rd Ed.), Stanford, CA: CSLI Publication.

Cocchiarella, N. B. (1991). Formal Ontology. In H. Burkhardt & B. Smith (Eds.), *Handbook of Metaphysics and Ontology*, Munich: Philosophia Verlag, 640-647.

Doody, D. (2001). *Basics of Space Flight JPL D-20120*, retrieved 9 December 2004 from <http://www.jpl.nasa.gov/basics/>.

ESA (2004). *European Space Agency official site*, retrieved 9 December 2004 from <http://www.esa.int/>.

Guarino, N. (1995). Formal Ontology, Conceptual Analysis and Knowledge Representation. *International Journal of Human and Computer Studies*, 43 (5/6), 625-640.

Guarino, N. (1998). Formal Ontology and Information Systems. In N. Guarino (Ed.), *Proceedings of the 1st International Conference on Formal Ontologies in Information Systems*, Amsterdam: IOS Press, 3-15.

Luger, G. F., & Stubblefield, W. A. (1998). *Artificial Intelligence, Structures and Strategies for Complex Problem Solving* (3rd Ed.), Italy, Milan: Addison Wesley Longman.

Nilsson, N. J. (1998). *Artificial Intelligence, A New Synthesis*, San Francisco, CA: Morgan Kaufmann.

Wertz, J., & Larson, W. (1999). *Space Mission Analysis and Design, Space Technology Library* (3rd Ed.), Torrance, CA: Larson and Microcosm inc.