Chapter 15
An ontology-driven System for e-learning and knowledge Management

Gilbert Paquette

In Paquette, Gilbert (Ed.). (2010). *Visual knowledge and competency modeling - From informal learning models to semantic Web ontologies*. New York : IGI Global.

Abstract

Between 2003 and 2008, within the LORNET research network (www.lornet.org), our team has been designing and developing TELOS, an innovative operation system for eLearning and knowledge management environments that is driven by a technical ontology. After presenting the underlying principles of this system, we will develop a graphic model of the resulting ontology that captures the conceptual architecture of the system. Next, we will present the main aggregation modeling tool and the way it is related to the TELOS Ontology. Finally, we will illustrate how the ontology is used to drive the system at run-time. The conclusion will discuss the contribution of this research to the field of ontological engineering of software systems.

15.1 PRINCIPLES FOR AN OPERATIONS' SYSTEM

At the turn of year 2000, new concepts had emerge from various fields such as Web-based programmable learning portals, service oriented frameworks, model-driven and ontology-driven architectures, multi-actor scenarios and workflows. These main technological trends have deeply influenced our work to produce more flexible, powerful, yet user-friendly elearning environments.

We aimed to go one level up, enabling the *aggregation of custom-made platforms* or portals in a way similar to desktop integration that has enabled the interoperation of components from different sources. We have designed the Technology Enhanced Learning Operating System (TELOS) on the same interoperability principles. The TELOS architecture aims to extend portal assembly in ways enabling technologists to built their own platforms. These platforms would foster a variety of distributed learning environments or models such as electronic performance support systems (EPSS), communities of practice, formal on-line training and technology-based classroom, and different forms of blended learning or knowledge management environments.

As the project was starting, *Service-oriented frameworks* (Wilson, Blinco and Rehak 2004) such as ELF (2007) or OKI (2007) were proposed to lower the costs of integration, and to encourage more flexibility and simplification of software configurations. Such a framework could also create a broad vocabulary that could be extended to an ontology. The TELOS conceptual framework presented in section 2 would also be designed as a service oriented framework, facilitating the aggregation of services to create custom-made platforms and applications.

This has led us naturally to a *model-driven, ontology-driven architecture* (Kleppe, Warmer and Bast 2003). The main gain of model-driven architectures (MDA) is the generation of the code from the model in successive layers, the model being reusable in other contexts with few adaptations. Ontology-driven architectures (Tetlow et al. 2001; Davies, van Harmelen and Fensel 2002) add to this paradigm an explicit ontology structuring of the objects processed by the system, acting as its executable blueprint. MDA therefore put more emphasis on the platform independent model (PIM), reducing the work on platform specific (PSM) and code models. Ontology-Driven Architectures foster a programming style analogous to the Prolog programming language. Here the declarative part is encoded in the ontology, in our case through OWL-DL statements. The execution part is encoded in queries prepared for an inference engine that processes the queries. The result of a query is to trigger the execution of some of the services.

Another key architectural idea is the concept of *multi-actor learning designs and workflows*, as the main structure of the various environments produced using TELOS. We wanted to avoid some of the weaknesses of our previous virtual campus models and most commercial platforms, where actors only interact within mono-actor environments that do not really take in account collaborative processes. As we have discussed in chapter 8, this question is now solved partly in workflows modeling languages such as BPMN (Correal and Marino, 2007) and in eLearning design specifications like IMS-LD (2003) Multi-actor learning designs and workflows provide a central aggregation mechanism grouping actors, the operation they perform and the resources they use or produce from or for other actors. Based on this work, a multi-actor scenario editor and execution engine was planned as a central piece of TELOS (Paquette and Rosca 2003; Magnan and Paquette 2006)

15.2 BUILDING THE ARCHITECTURE OF *TELOS*

Initially, we have tailored the Rational Unified Process (RUP) to the needs of the project. RUP is an adaptable process framework that describes how to develop software effectively using proven techniques. While the RUP encompasses a large number of different activities, it enables the selection of development processes appropriate to a particular software project.

Development Process

As shown on figure 15.1 our initial use of RUP was first focused on the business modeling and the requirements processes, each with a few cycles including phases of inception, elaboration, construction and transition. This has led to a set of architecture documents, the main one being the Use cases and software requirements documents (UC 1.0). Then the focus has moved to the Analysis and Design process with the construction of the Conceptual Architecture (CA 0.7) and the Conceptual Framework (CF 0.8) documents, which includes the TELOS Conceptual Ontology. The first two years followed the RUP quite closely but with long iteration cycles resulting in a set of architecture documents and throw-away prototypes TELOS-1 and TELOS-2.

In the last three years, the team has reduced the length of the iterations, adopting a process closer to Rapid Prototyping in order to achieve workable prototypes. A number of Software Architecture (SA) documents have been written to support the implementation of the TELOS prototypes. TELOS-3 was the first evolutionary prototype on which we could build the following ones. Each year, a test bed was conducted where users would interact with the available prototype within a carefully planned test process. Prototypes 2, 3 and 4 were demonstrated at the LORNET annual conferences. The last one, TELOS-5,

was demonstrated at the ITS-08 conference. This evolution reflects the fact that TELOS does not follow traditional software development processes, being considered as innovative, risky and ambitious by many persons, in other words, a research project.

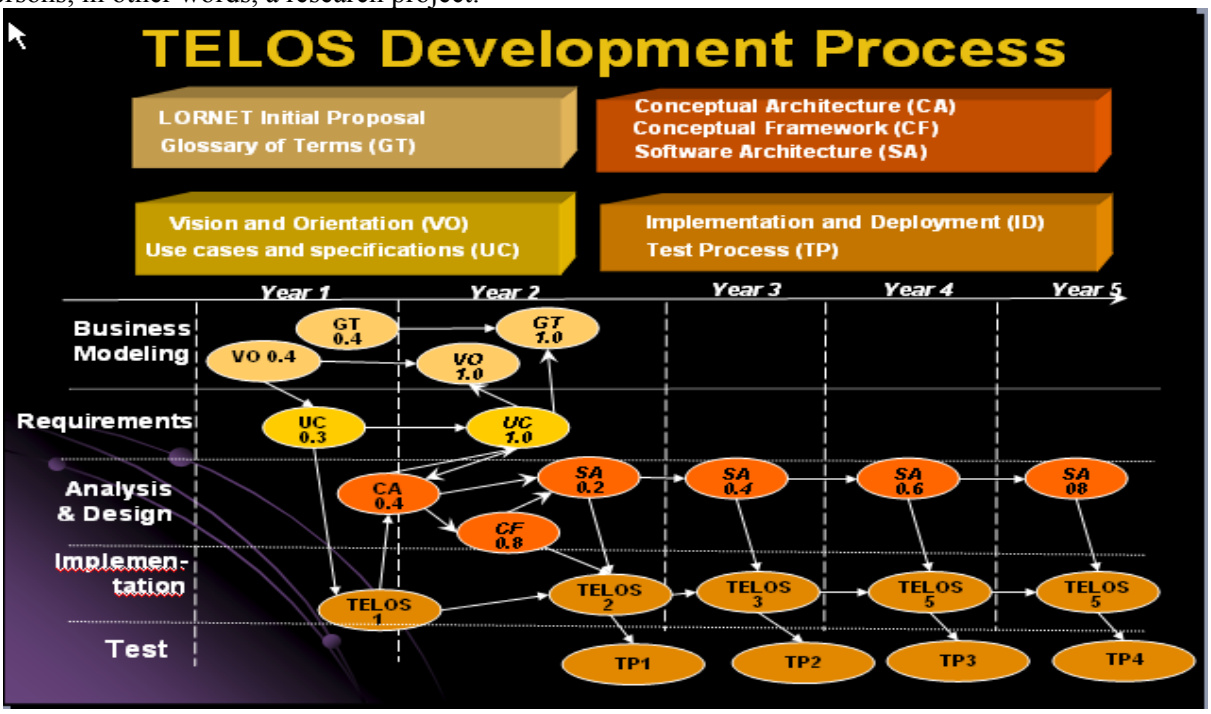

Figure 15.1 – TELOS development process.

Use Cases and Requirements

We will now briefly summarize on Figure 15.2 the Use Cases Specifications and Requirements (UCR 1.0), and the TELOS Conceptual Framework (CF 0.9).

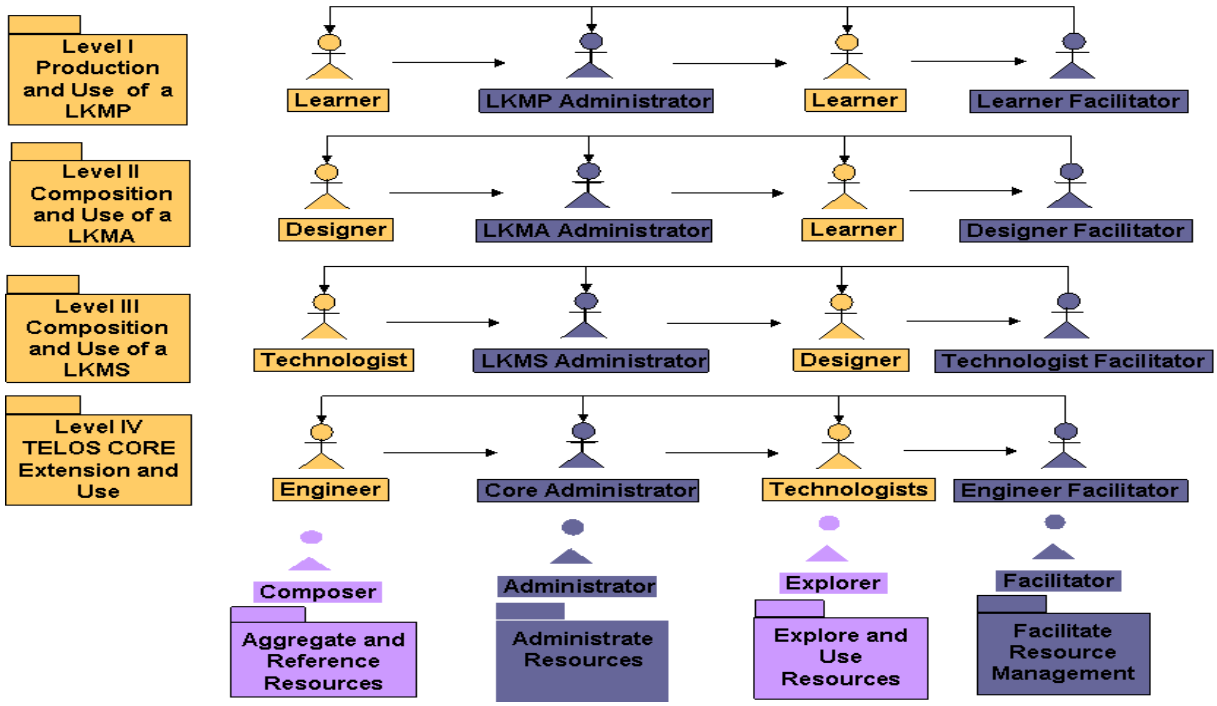

Figure 15.2 - Resource life cycle and system cascade actors

The UCR has undergone 10 iterations, from June 03 to December 04. It groups 30 use case diagrams and descriptions that are packaged as shown on figure 15.2. The use cases at the four levels of the system describe how to build, administrate, use and support a Web-based environment, each being used at each cascade level (rows on figure 15.2). Level IV concerns mainly an engineer extending the TELOS Core that will be used by technologists. At Level III, a technologist uses the TELOS Core to produce a platform, technically called a Learning and Knowledge Management System (LKMS). At level II, a designer uses a platform, to build one or more Learning and Knowledge Management Applications (LKMA), usually called "courses", "learning units", "knowledge management workflows", etc. Finally at level I, using one of these applications, a learner or a knowledge worker will acquire knowledge and produce results (homeworks, documents, performance) that can be grouped in a portfolio or a set of Learning and Knowledge Management Products (LKMP).

Generic resource life cycle use cases (columns on figure 15.2) correspond to four sub-operations (phases) that occur at each of the four cascade levels. In these, a resource is composed, managed (prepared) for use, used by its intended actors, and analyzed to provide assistance. These operations are generally performed in sequence at each of the cascade levels by corresponding actors called respectively composers, administrators, explorers (resource users) and facilitators (acting as analysts to provide assistance and feedback). These operations are generic, being applicable at any cascade level. When they act as composer, learners will have to search for resources in much the same ways as a designer, a technologist or an engineer, even though the content of the resources will differ.

We can use different metaphors to describe these general processes. In a manufacturing metaphor, the resource life cycle corresponds to a process where a product passes through different productions operations. Within the system generation cascade, the TELOS Core is like a factory that produces machine components or complete machines; the products of this first factory are used to build machines that will be used in other factories (LKMSs) to build cars, aircrafts, etc. These transportation machines, will finally be used by their clients to produce some outcome (e.g. to travel).

As a manufacture, the TELOS Core itself starts with a complete set of components to produce LKMS

factories, but it will also be open to improvement, adding new processes and operations, to produce more versatile machines.

Conceptual Architecture

Starting with this elaborated set of use cases, the TELOS conceptual architecture (Rosca 2005) and the TELOS conceptual framework (Paquette, Rosca Masmoudi and Mihaila 2005) were built as a service-oriented framework, bringing it closer to a possible implementation. Figure 15.3 present the main classes of services.

- *Kernel Communication services.* TELOS is built as a distributed architecture on the Internet. To become a node in a Virtual Campus, each user installs a TELOS kernel on his machine that provides basic communication services with other nodes where resources are distributed. These services include for example a service registry, a repository that locates the resources on the nodes of the network, connectors to provide communication with resources built using different technologies, protocol translation and so on.
- *Resource interfacing services.* Basic resources comprise documents in a variety of media formats, tools to process documents, operations that can perform a process automatically and finally persons managing a set of activities on the network. All these resources usually will require to be interfaced in different ways, for example by a communication agent for format translation, through encapsulation for tracing, etc. They will then be stored in a resource repository in order to be reached and to participate in the learning and/or knowledge management processes
- *Resource life cycle services.* These services provide a number of editors for a composer to build, adapt and aggregate resources, thus producing a model of the resource. The provide tools for an administrator to produce instances of the model, as well as interfaces to help users and facilitators interact with an environment instance.
- *Aggregates' management services.* These services provide management functionalities for the main aggregates (or Web portals) used in the Virtual Campus: Core, LKMSs, LKMAs and LKMPs portals. For example, they will help in the storage, modification, display, evolution and maintenance of versions of TELOS Core, the interoperability between platforms (LKMSs), the management of courses (LKMAs) and the LKMPs such as Portfolios.
- *Semantic services.* These services enable users to query or edit semantic resources, for example ontologies or metadata, used to reference resources. Resource publication services enable users to package resources with their semantic references, enabling various kind of resource search, retrieval and display. With these services a user can call upon federated or harvested search operations to display documents, tools, operations (including activities and units-of-learning) related to some domain knowledge and competencies.
- *Common services.* We have grouped in this category all the lower level services that are called by the services in the preceding categories. They correspond to operations that all the actors need to perform while participating in the Virtual Campus.

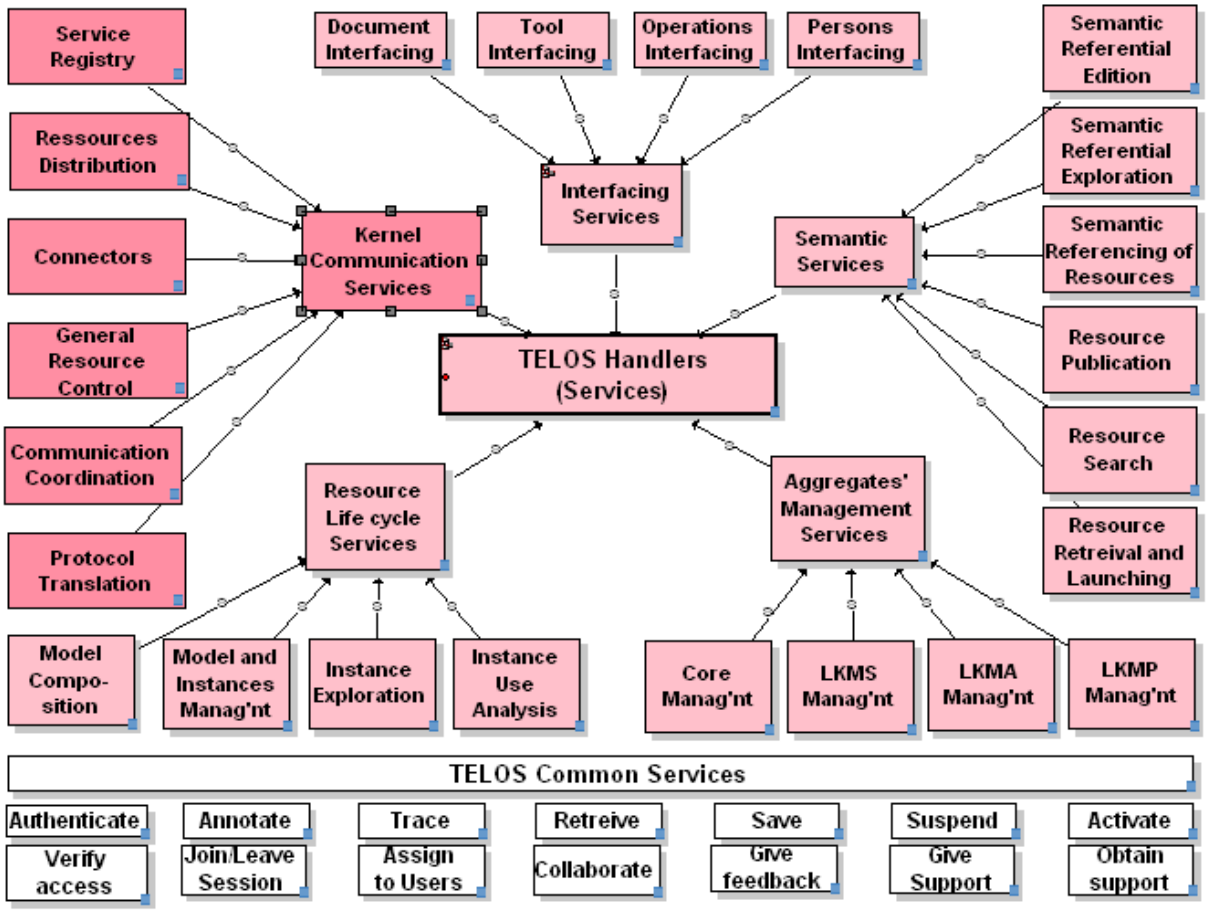

Figure 15.3 -The Virtual Campus Service Oriented Framework

15.3 THE TELOS TECHNICAL ONTOLOGY

In this section we will summarize the general methodology used to develop TELOS. First, we will present the conceptual framework of the TELOS system, and the conceptual ontology derived from it. Next, we will show how this first ontology was adapted to produce the technical ontology that drives the TELOS system.

From Conceptual Framework to Conceptual Ontology

An important goal in the TELOS project was to embed in the system technology-independent models, to help the system survive the rapid pace of technology evolution. Another concern was to favour the reusability of modular components and the flexibility on the system's evolution. For that purpose, the conceptual specifications of TELOS, are not be kept apart from the code of the system as is usually done in software engineering. The TELOS system is able to use ontologies as "conceptual programs". In this vision, the conceptual models are not just prerequisite to the construction of the TELOS system; they are part of the system, as one of its most fundamental layer. These considerations motivated the need for an ontology-driven architecture (ODA).

To achieve that goal, we have translated the use cases and the service-oriented conceptual architecture presented above into an OWL-DL ontology. We have selected to use OWL-DL ontologies (W3C 2004) for a number or reasons. Of the three languages designed by the W3 consortium, OWL-DL has a wide expressivity and its foundation in Description Logic (Baader, Calvanese et al. 2003) guarantees its

computational completeness and decidability. On a more practical side, a growing number of software tools have been designed to process OWL-DL XML files and to put inference engines at work to query ontologies in order to execute the processes in a system.

The graph of figure 15.4 presents the upper level of the TELOS Conceptual ontology that has been constructed using the MOT+OWL editor (Paquette 2008) presented in chapter 10. In TELOS, the actors, the operations they perform and the resources they use or produce are all TELOS resources. This is shown on the graph by using S "is-a-sort-of" links. They are represented as classes linked together with properties such as "perform" and "use or produce".

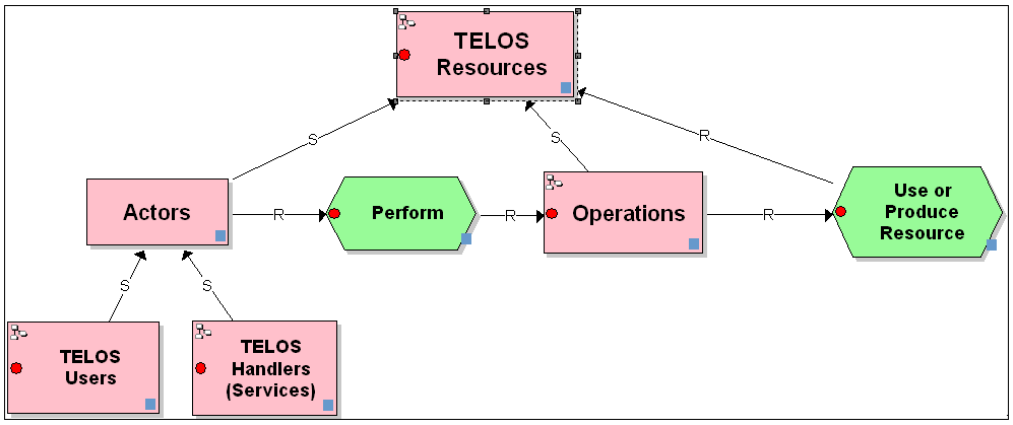

Figure 15.4 - Upper part of the TELOS Conceptual Ontology

Some classes are further defined in sub-models that present sub-taxonomies of classes and their properties. The graph of figure 15.5 shows the taxonomy of TELOS users corresponding to the use cases summarized on Figure 15.2, related by the "canPerform" property to corresponding operations they perform. The taxonomy of operations is further defined in another Operations' sub-model (not shown here) where the operations are linked with the services (handlers) identified previously in the conceptual architecture (Figure 15.3).

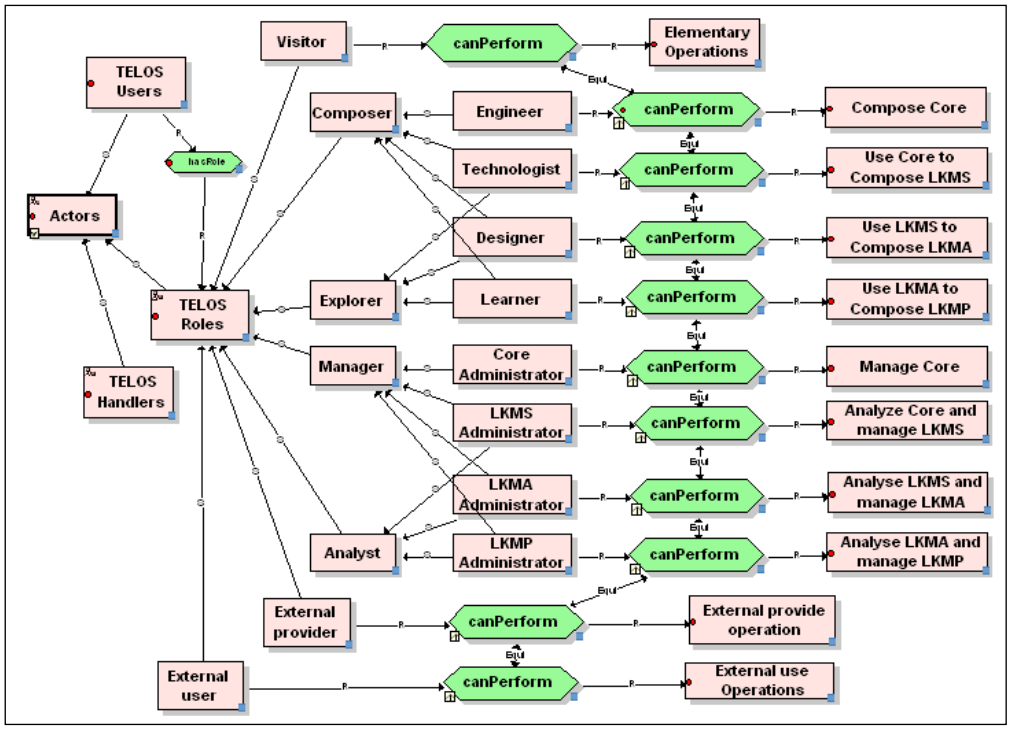

Figure 15.5 - Part of the TELOS Conceptual Ontology focused on Roles and Actors

Another sub-graph describes the taxonomy of resources, including the very important concept of "content" package, which is redefined as resources having semantic descriptions. The upper part of this sub-graph is presented in figure 15.6.

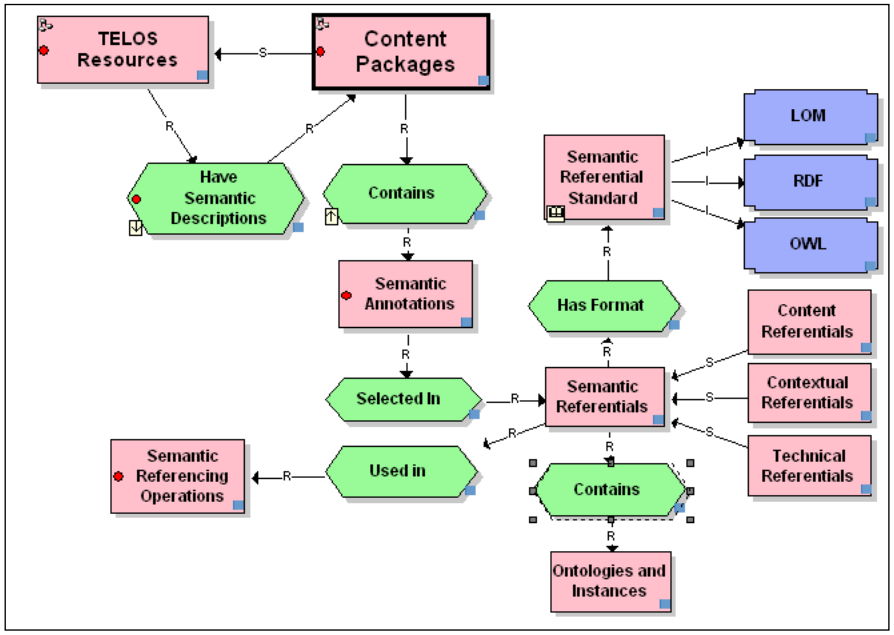

Figure 15.6: Part of the TELOS Conceptual Ontology focused on Resources

From Conceptual Ontogy to Technical Ontology

The conceptual ontology was revised, simplified or expanded to build the TELOS technical ontology shown on figure 15.7, that is integrated as code to drive the operation of the system.

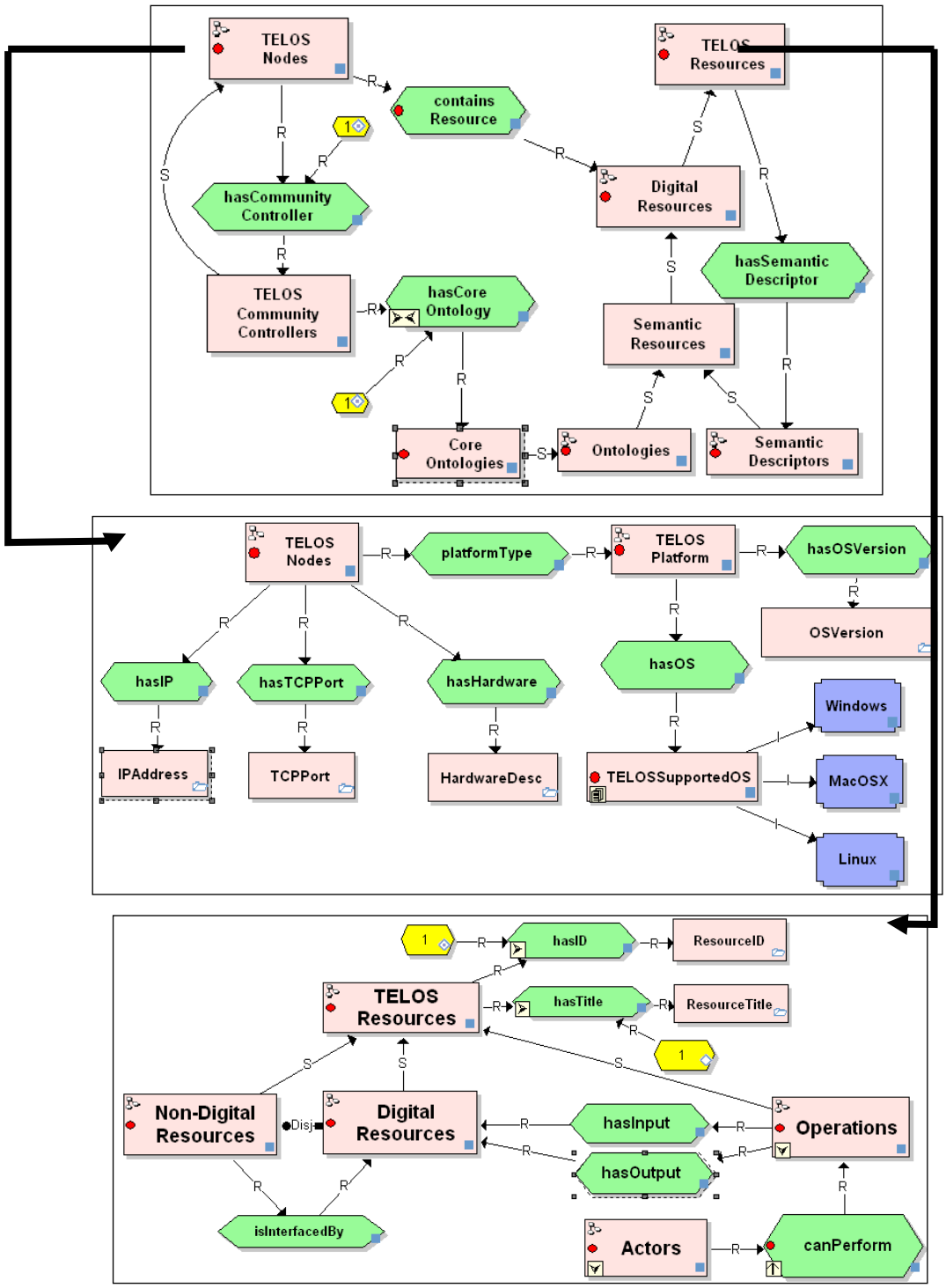

Figure 15.7 - The upper layer of the TELOS technical ontology

First, we had to capture the distributed aspects of TELOS by adding the concept of a "TELOS Node" that was not present in the Conceptual Ontology but defined in the Conceptual architecture. The TELOS Global bus enables the interoperability between different TELOS nodes abstracting their particular physical platform and their network configuration. By connecting TELOS nodes through the Global Bus, we form a dynamic peer-to-peer network. This network may contain special nodes called community controllers, which are basically centralized repositories for resources. The first graph of figure 15.7 is the upper level of the TELOS Technical ontology. It presents the concept of a TELOS node in relation to the concept of a TELOS resource, which was the root of the former conceptual ontology.

The second graph on figure 15.7 presents a sub-model of the previous one defining the concept of a TELOS Node, and the third one presents a more precise, upper level definition of a TELOS Resource that is very similar to the one in the Conceptual ontology. In these graphs, cardinality axioms (hexagons with numbers), disjoint axioms (Disj link) and functional property axioms have been added for more precision. This is essential because this ontology will have to respond to queries using an inference engine that do not tolerate ambiguity.

In these upper layer graphs, we find the interrelations between the main concepts of *Actors*, *Operations* and *Digital Resources*. The Actors' and the Operations' sub-ontologies are directly imported from the Conceptual ontology presented on figure 15.4. The Digital Resources' sub-ontology has undergone more important changes. It is presented on figure 15.8. We see that implementation-focused concepts have been added such as *productID* and *locationURL*. Each of the five main sub-classes: *Documents*, *Atomic Resources*, *Actors*, *Aggregates* and *Digital Operations* are detailed in other OWL-DL models no shown here.

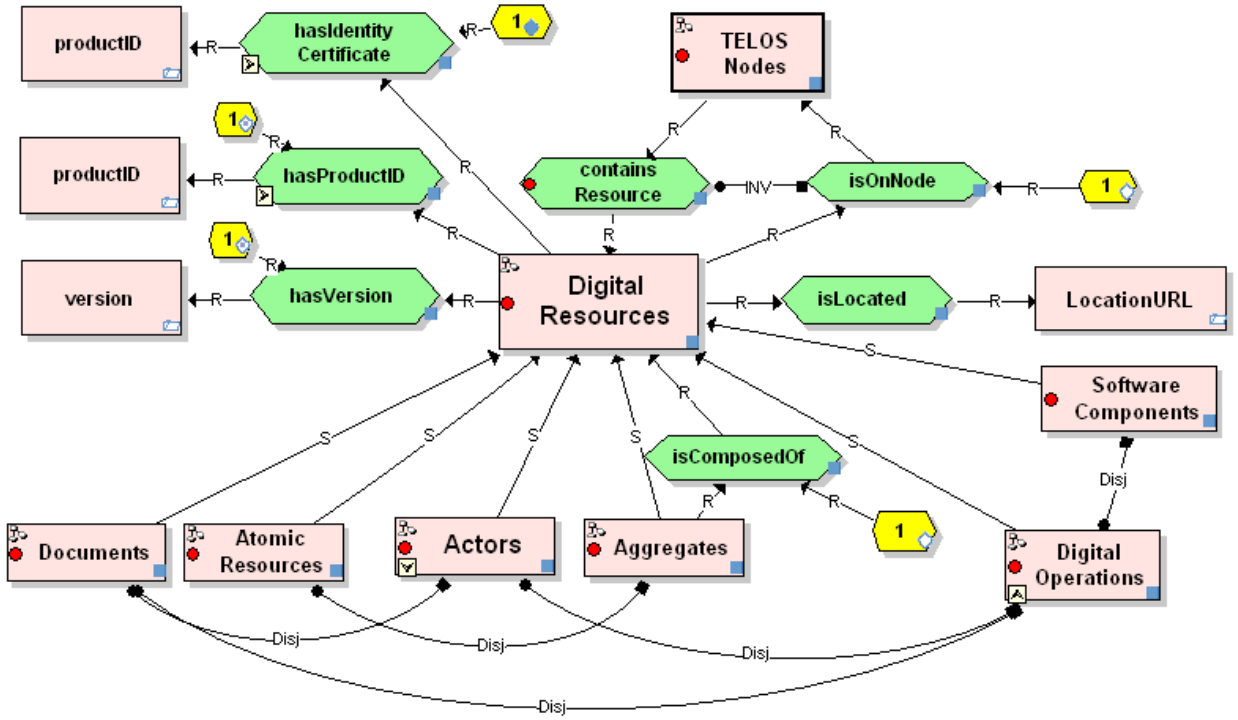

Figure 15.8 -Part of the sub-ontology for Digital Resources

The resources are the persistent data of a TELOS node. The "Aggregates" sub-class is particularly important. This type of resource enables users to create new eLearning tools by gluing existing software components and other resources. It also enables users to model collaborative workflows or scenarios

aggregating actors, the activities they perform and the documents, the software components they use or produce.

The technical ontology we have just presented forms the heart of the semantic layer if TELOS. It is where all TELOS concepts are declared and related together to define the global behavior of TELOS. The semantic layer also contains the domain ontologies created by users that later enables the referencing of the resources with application domain knowledge and competencies.

15.4 TELOS MAIN TOOLS

Before discussing how the TELOS system uses its semantic layer and its Technical ontology for its operations we will first present de main tools in TELOS. Figure 15.9 displays the TELOS desktop main interface in a Web browser, with the three main tools open: the Resource Manager, the Scenario Editor and the Task Manager. The MOWL Ontology Editor presented in chapter 10 is also available from the TELOS desktop.

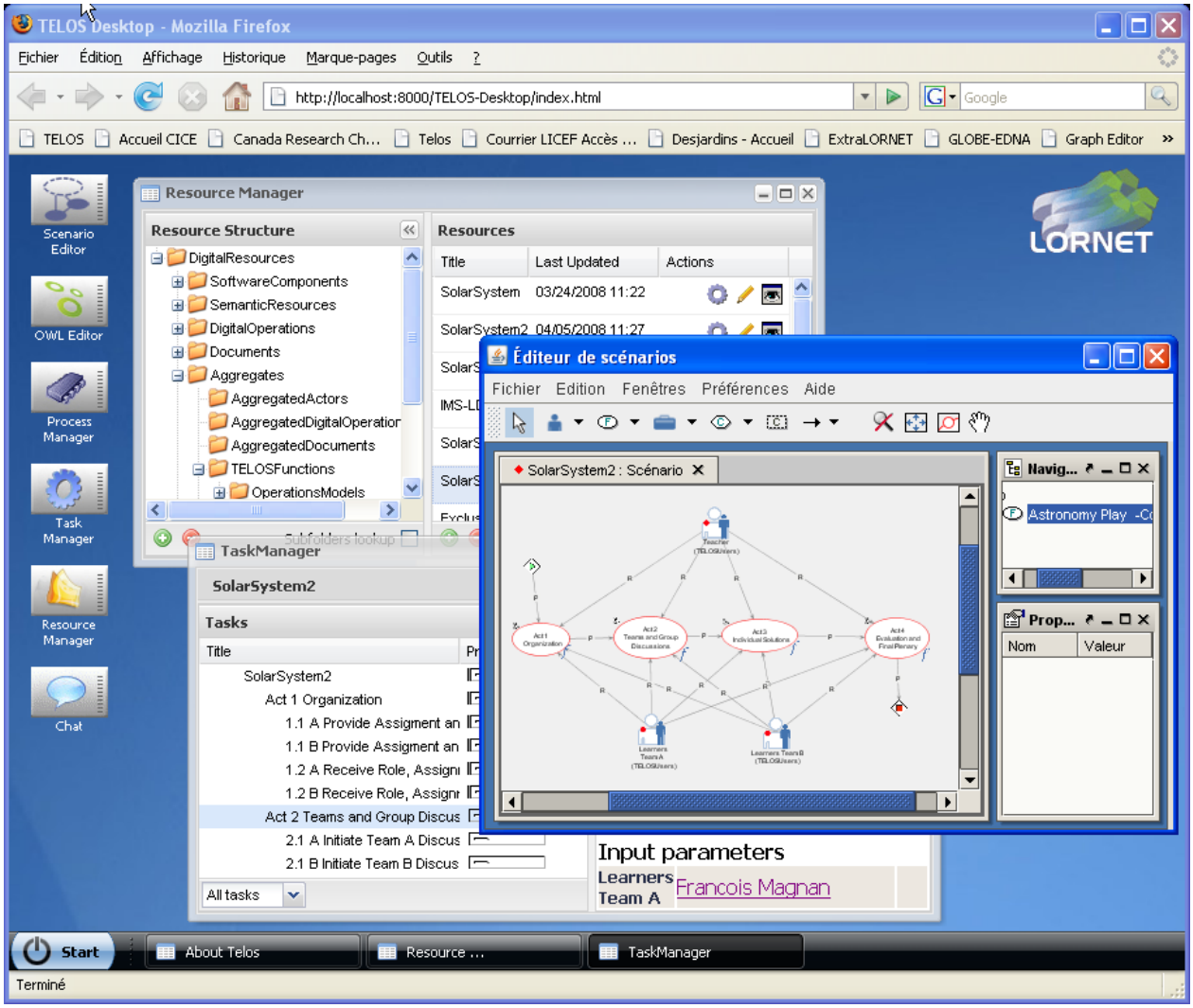

Figure 15.9 - TELOS desktop main interface

The Resource Manager

The TELOS Resource Manager serves to integrate and manage the resources that actors use or produce in TELOS. These resources are classified according to the technical ontology. On the Figure 15.9, the "aggregates" class of the technical ontology is expanded and the "My scenarios" class is selected showing the available instances on its right side. Resources can be added, deleted, moved or duplicated within a class or to other classes.

Once a scenario has been selected, the little icons on the right enable any authorized user to *view*, *modify* or *run* the execution of the resource. These functions differ depending on the type to the resources.

- When a resource is open in modify (write) mode, a key icon (🔑) is added in the action column to prevent two users to edit the same resource at the same time. The resource is transferred on the user's local machine and opened with his associated applications (MS Word for a doc file, PowerPoint for a ppt file, etc...) until it is made available again in the resource manager.
- For a scenario the View and Modify functions open the scenario editor presented in chapter 8, also shown on the second window of figure 15.9; the Run option starts the inference engine that will execute the scenario and present it to its users in the Task Manager.
- For an ontology, the View and Modify functions open the ontology in the Ontology Editor presented in chapter 10.
- Resources of type TELOS Users, viewing or editing opens a User Browser that lets you enter personal information, e-mails, photo, etc....; this information is reachable in scenario execution process, by clickable icons that represent scenario's users. For groups (type Actors/TELOSGroups), a dedicated editor is opened to add or delete individuals from a group.
- Software component can also be integrated into scenarios through operations objects stored in the resource manager. Operations automate some processes during scenario execution. This kind of resource must be a zip archive that contains the binary code of the component plus an XML manifest file that describes its services : name, input parameters, output parameters and arguments order.

The Scenario Editor

The TELOS scenario Editor has been briefly presented in chapter 8. Let us recall that MOT *Concept* symbols serve to represent all kinds of resources: documents, tools, semantic resources, environments, actors (as a resource), activities (as a resource) and data. MOT procedure symbols represent *Functions* that are aggregates of resources (e.g. scenarios) that together achieve a function. Functions can be decomposed into other functions at any depth down to activities enacted by humans, or operations performed automatically by the system. Finally, MOT principles serve to represent actors as well as conditions, depicted by two different sets of symbols. The *Actor* symbols represent users, groups, roles or software agents, all seen as control objects that rule the activities using and producing resources as planned by the scenario model. The *Condition* symbols represent control elements inserted within the basic flow to decide on the following functions, activities or operations to execute.

Now let us look more closely at the operation of the TELOS Scenario Editor. As a user starts elaborating a scenario model, the top level will be a function (generated by default) that represents the whole scenario that is created. This aggregate will be consequently added as a resource into the resource manager. At this top level, it is recommended to add and to link to this function all key actors and resources involved in this scenario. Afterwards, the user will add one or more levels of sub-models, to add more details to the scenario, as shown on figure 15.10. This sub-model shows that the scenario is composed of a sequence of four acts or modules involving three actors and two forum tools. Learning objectives are presented in the first act.

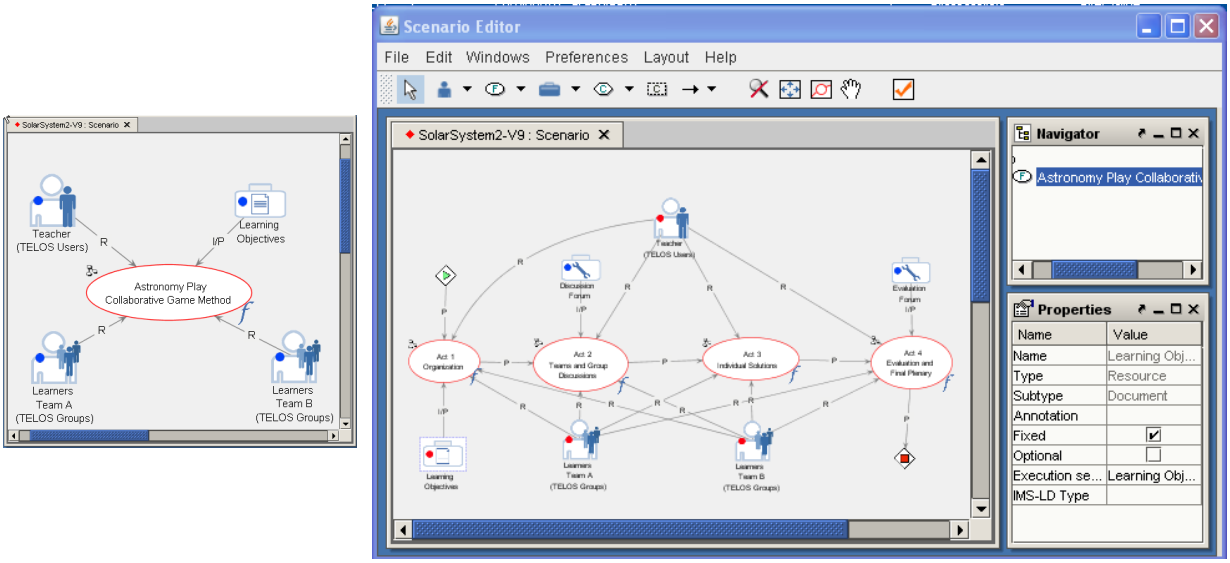

Figure 15.10 - Example of a model and its sub-model

Each graphic object at any level of the scenario can be described by a property sheet grouping properties like its name, type, annotation, etc., the most important one being its execution semantic.

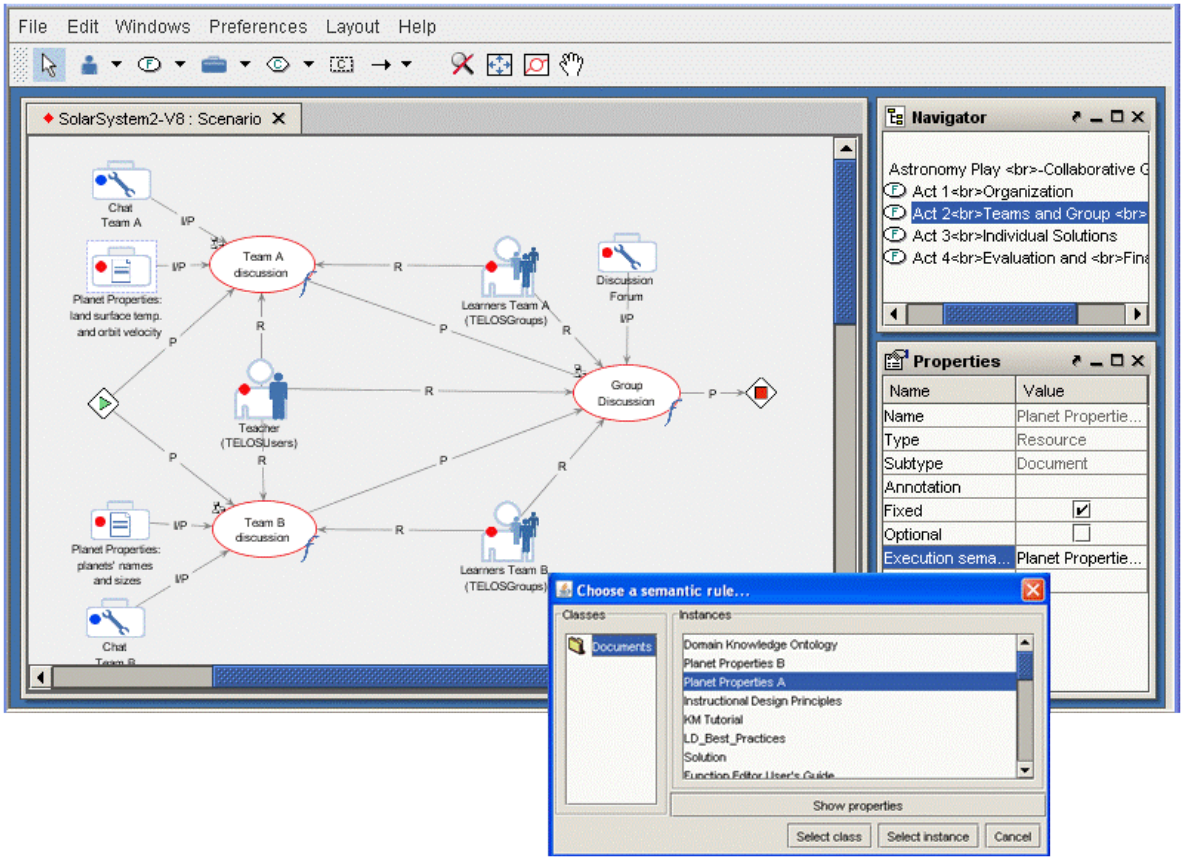

Figure 15.11 - A sub-model of a sub-model

Figure 15.11 presents a sub-model of act 2 that displays a property sheet for the selected object called "Planet Properties land surface temperature and orbital velocity". If we do not tell TELOS what this icon represents in terms of its technical ontology, the system will not be able to process it. This is the role the selected property called "Execution semantic". By clicking on this field, we open the little window of the Ontology chooser that opens a class of resources, here "Documents" and all the available instances that are member of that class. Here, the resource "Planet Properties A", in fact a Powerpoint presentation on the planets, has been selected to be associated to the selected icon. At runtime, this document will be opened from the Resource Manager. In the same way, the teacher, a TELOS user, and the two teams of learners, will be given an execution semantic. The teacher icon will be associated to an individual user acting as the teacher, and the group icons will be associated to a list of precise learners, previously entered in the resource manager. The same can be said for the two chat tools into which each team will interact. Here, the execution semantic is to simply link the icons to URLs that will open in a browser each chat tool at runtime.

The TELOS scenario language provides a high-level programming visual language for TELOS. This generic language is designed for all TELOS users, including students/workers, teachers/designers, technologists and programmers/engineers, when they act, respectively, as composers (see figure 15.2) and use composition handlers (see figure 15.3) at the different levels of the cascade of TELOS aggregates.

In (Paquette and Magnan 2007) we have presented three examples of scenarios. The first case, presented above, is built by a designer that has constructed the course. The second example is less common, showing how a technologist can combine an existing platform with TELOS tools to extend the functionalities of the design environment. The third one has been built by an engineer aggregating four different components built with different technologies, in order to insert automatically the learning objects found by a Google search into a resource manager.

The example on figure 15.12 shows how a technologist can combine an existing platform with TELOS tools to provide a composition environment for a course designer. This design scenario corresponds to the central tasks of the MISA instructional engineering method presented in chapter 8 (Paquette 2001, Paquette 2004). The figure shows part of this scenario that involves using Concept@, Télé-université's actual course design platform, augmented with the TELOS scenario editor and other components. The little window on the right present the tools used by designers in this design scenario.

The design scenario starts with two parallel functions performed by a designer: design of a course backbone using the Concept@ LCMS and the development of a knowledge and competency model for the course using the TELOS ontology editor. Let us note that actually, Concept@ helps produce an activity tree representing the course plan and its subdivision into modules and activities. This is a common situation in most LCMS (Learning Content Management Systems). This tree structure can be exported to a SCORM package. Then we add to the design scenario an operation that automatically transforms a SCORM package to the TELOS scenario format. This last XML file can now be imported in the scenario editor and displayed in the form of a scenario graph where it can be expanded.

Many roles can be defined in Concept@ but this exceeds SCORM's mono-actor capabilities. So information about roles/actors is lost when we open the corresponding graph in the TELOS scenario editor. The next design phase proceeds graphically in the TELOS scenario editor to add manually the actors identified in Concept@.

Within the scenario editor more advanced flow of control can also be added to better personalize learning based on the knowledge and competency model. This is shown by the last three steps of the design scenario on figure 15.12: associate knowledge and competencies to scenarios components, use a resource manager like PALOMA (see chapter 18 and 20) to add complementary resources to the scenario targeting the competencies and, finally parameterize the Competency + software (see chapter 20) for self-diagnosis by the learners of their competencies. We will discuss the question on assistance and personalization in chapter 21.

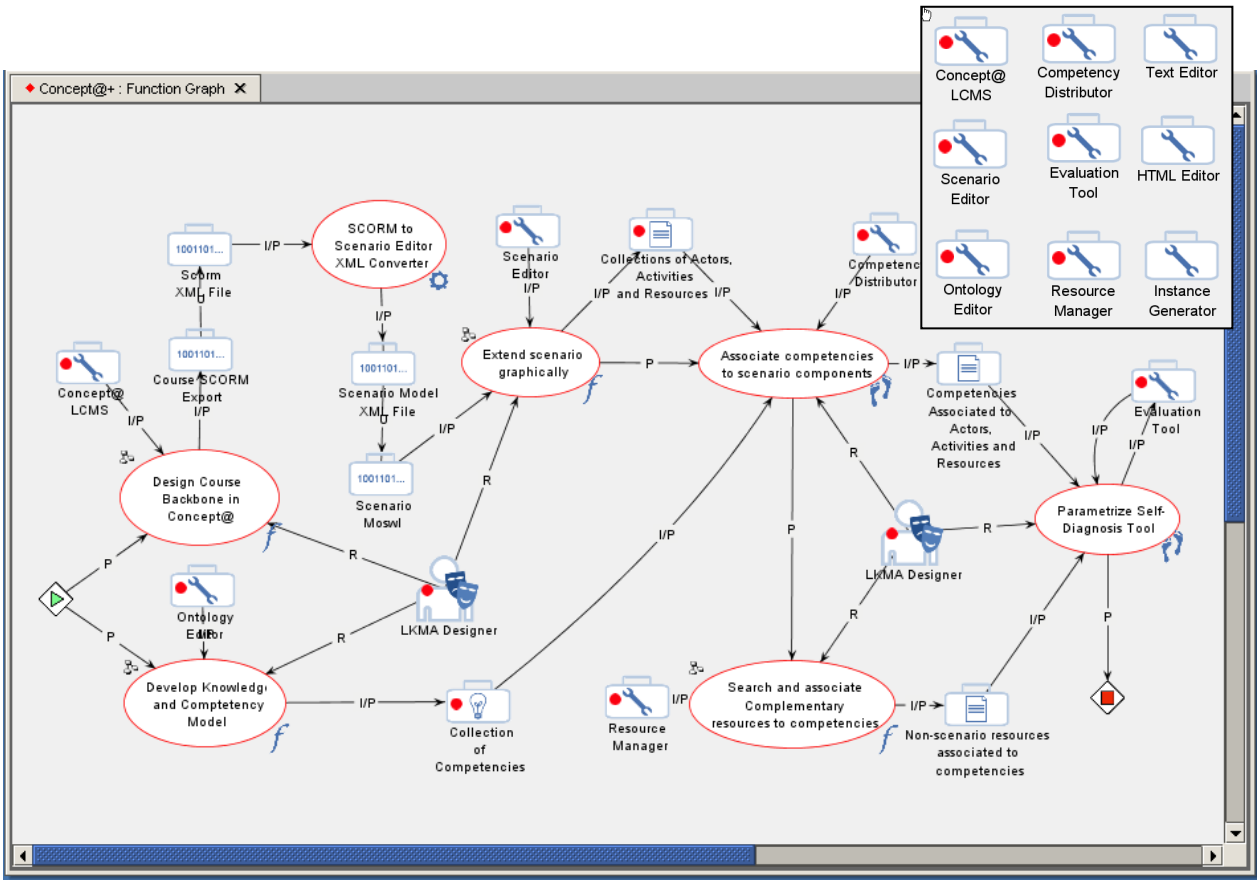

Figure 15.12: Technologist constructing an augmented LKMS platform for designers

A third use of the scenario editor is presented on figure 15.13. It has been built by an engineer aggregating a new service encapsulated in an operation called the "Batch LOM Extractor". This operation takes a set of keywords, a number of LOM records to be found and the name of a destination folder in a repository of learning objects managed by the PALOMA software.

The aim of this aggregated operation is first to make a Google query with the given keywords and collect the specified number of Web sites. Then, the next step will apply a text mining algorithm on each websites to extract automatically part of the metadata according to the LOM standard. Then, each of these metadata records will be inserted into the PALOMA folder identified at the beginning. Finally, this folder will open to show to the user the list of LOMs into the PALOMA software interface.

What we see here is the aggregation of software components built by different groups using different technologies. These software components transfer data from one to another. The Google Search Service is launched using a SOAP Web service connector provided by the TELOS kernel. The Metadata Extractor is a C# component linked to the TELOS kernel by a C# connector; it creates a metadata record in the Dublin Core (DC) format. The DC to LOM conversion is a Scheme program linked to TELOS through a Scheme connector. PALOMA is a Java applet linked to TELOS through a Java connector.

This example illustrates the multi-technology aspect of TELOS and the capability of the scenario editor to aggregate software components as if they were one. The resulting aggregate can now be inserted in any scenario as one of its tool.

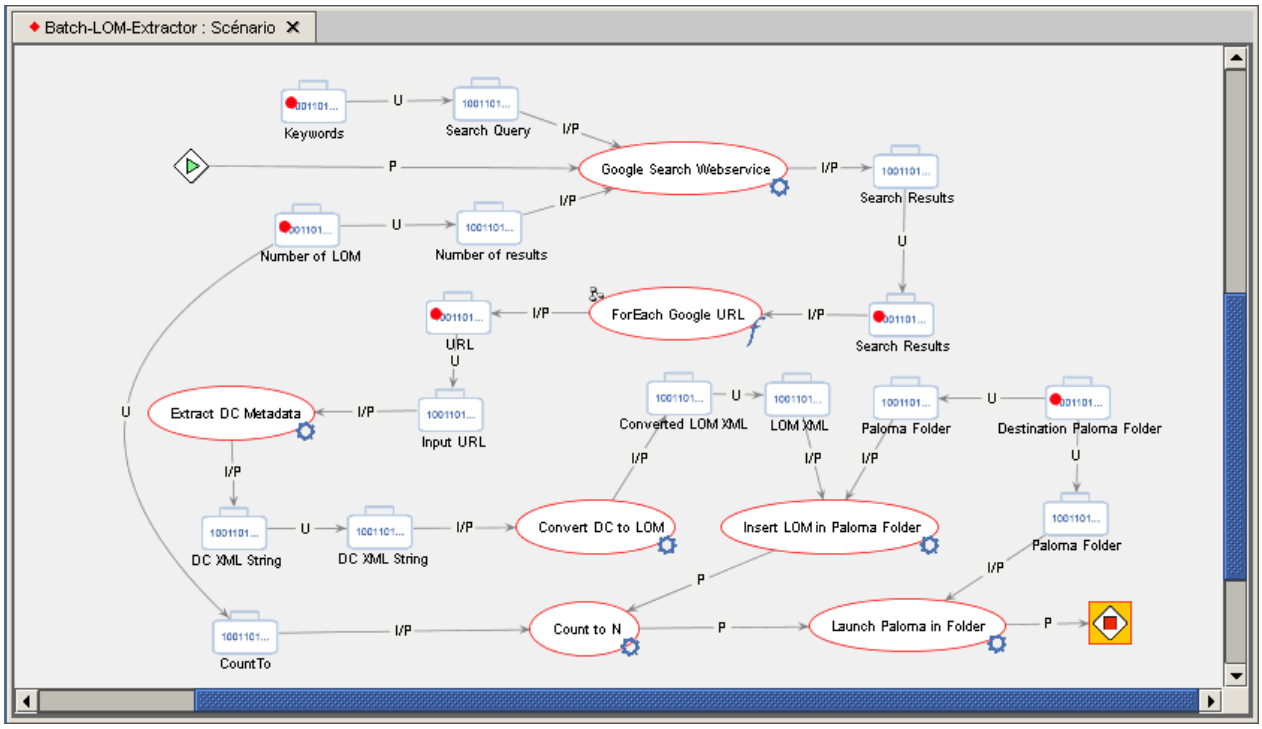

Figure 15.13 - Engineer constructing an operation aggregating services

15.5 ONTOLOGY-DRIVEN SCENARIO EXECUTION

Although these scenarios differ greatly by their goals, their component resources and the level of their actors in the view of TELOS on Figure 15.2, they can all be run by TELOS as long as each resource has an execution semantic based on the TELOS technical ontology. The Scenario Evaluator is the engine that executes a scenario. It coordinates the interactions between actors, activities/operations/tasks and other resources in the scenario at runtime. It looks at each graphic form in the scenario, obtains their location in the TELOS technical ontology and, with this data, it runs the user interface of the task manager accordingly.

The Task Manager

The *TELOS Task Manager* is the tool allowing a user to interact with running scenarios. To invoke the Task Manager, a user clicks on the Task Manager icon on the desktop or select the Task Manager item in the Start menu. The Task Manager will also be invoked automatically when launching a scenario via the Resource Manager.

Figure 15.14 shows the Task Manager for the scenario of figure 15.10 and 15.11 at the very beginning. On the left, the tree view shows the four acts of figure 15.10. Each can be expanded by the + sign, but none is yet active because the execution engine is positioned at the function main level. On the right side of the interface, we see an annotation that has been added to state the general description of the scenario, followed by a list of the input parameters of the function specified on figure 15.10. The Learning Objective is a concrete document specified at design time so it can already be displayed by clicking on its name.

The other three input parameters, two groups of learners and the teacher, have to be instantiated at run time because their execution semantic is a class of the technical ontology. This is what the administrator has to do in order to launch the scenario: look up in the Ontology Chooser the instances of groups and users available in the resource manager (as shown on the little window in the front), select a value for each parameter and click on the *Launch* button.

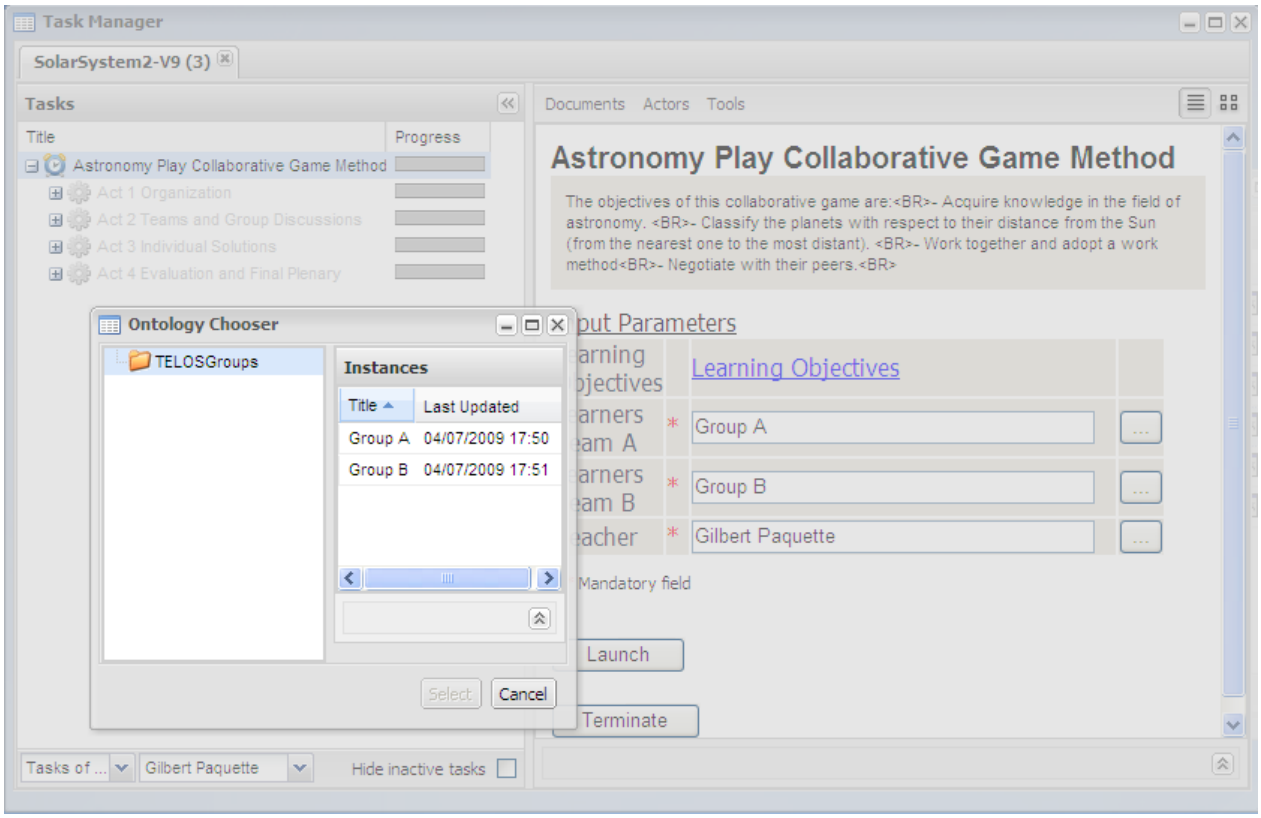

Figure 15.14 - Task Manager Interface

From that moment, the flow of control will move down to the level of the four acts. The first one will be executed, and then the second one and so on, according to the flow of the graph designed on figure 15.9.

Interface for each actor's roles

Figure 15.15 shows the interface in the task manager for team A and team B members at the beginning of Act 2, when they start separate chats to discuss planet properties. It is important for the pedagogy in this scenario that they do not have the same information to analyze. Afterwards, they will share their information in the forum grouping all students and the teacher.

The essential thing here is that the task manager, guided by the technical ontology, presents adapted task manager interfaces to all the participants in the scenario. The teacher sees everything because he is an actor monitoring all the tasks in this scenario. This enables him to see all the documents, actors and task progress for all the learners involved in the scenario.

On the other hand, as shown on figure 15.15, the learners see only the tasks they are involved in and only the document they are suppose to use or produce. For example, learners in team A have terminated activity 1.2A which was their only task in Act 1 and their 2.1A activity is activated. From the Documents

menu or from the Input Parameters section to activity 2.1A, they can launch the document shown on the figure that provides information on the rotational periods, the orbital periods and surface temperature of the planets. In the Actors menu, they see for the moment only the v-cards of their co-learners and of the teacher. And from the tools menu, they have access only to the Science chat A tool.

Meanwhile, learners in team B have access to information on other planet properties, their mass and equatorial radius. They can access their own team mate and the teacher in the Actors' menu and they can open their own chat environment in the tools' menu.

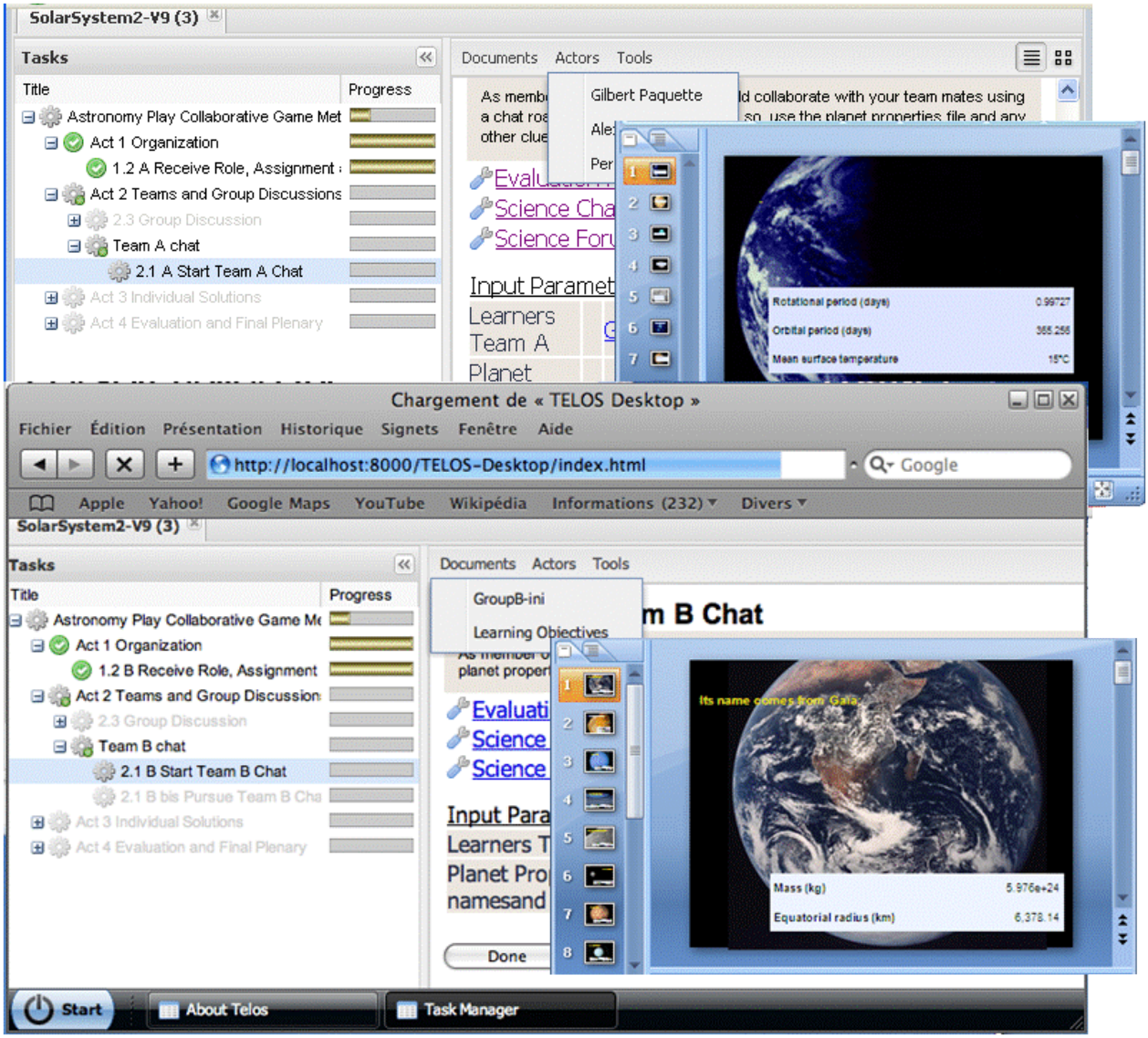

Figure 15.15 - Different views in the Task Manager for team A and team B.

Providing contextual views

Another interesting feature of the task manager is the graphic view presented on figure 15.16 that also uses the structure of the scenario graph and the links between its objects and the TELOS technical ontology. In this mode, the currently selected task is always shown in the center of the graph. Tasks that

immediately precede and succeed the current task are displayed, respectively, on the left and right side of the central node. This way, a user can see the task contextually. To see farther tasks, a user can click on another node and the graphical view will adjust itself. The location bar at the top indicates in which sub-graph the current task belong. A user can use the location bar to navigate to upper sub-graphs.

Here, we see a selected activity where the teacher provides the initial objectives, assignments and information to the learners. We see that this task is followed by two parallel learner tasks 1.2A and 1.2B. Clicking on the persons' icons, shows that 1.1 is performed by the teacher, giving access to its personal information stored in the resource manager in the ontology class "Actor". For task 1.2 A, performed by a group of learners, we see the list of group members stored in the class "Group" of the research manager plus the teacher, here played by the author.

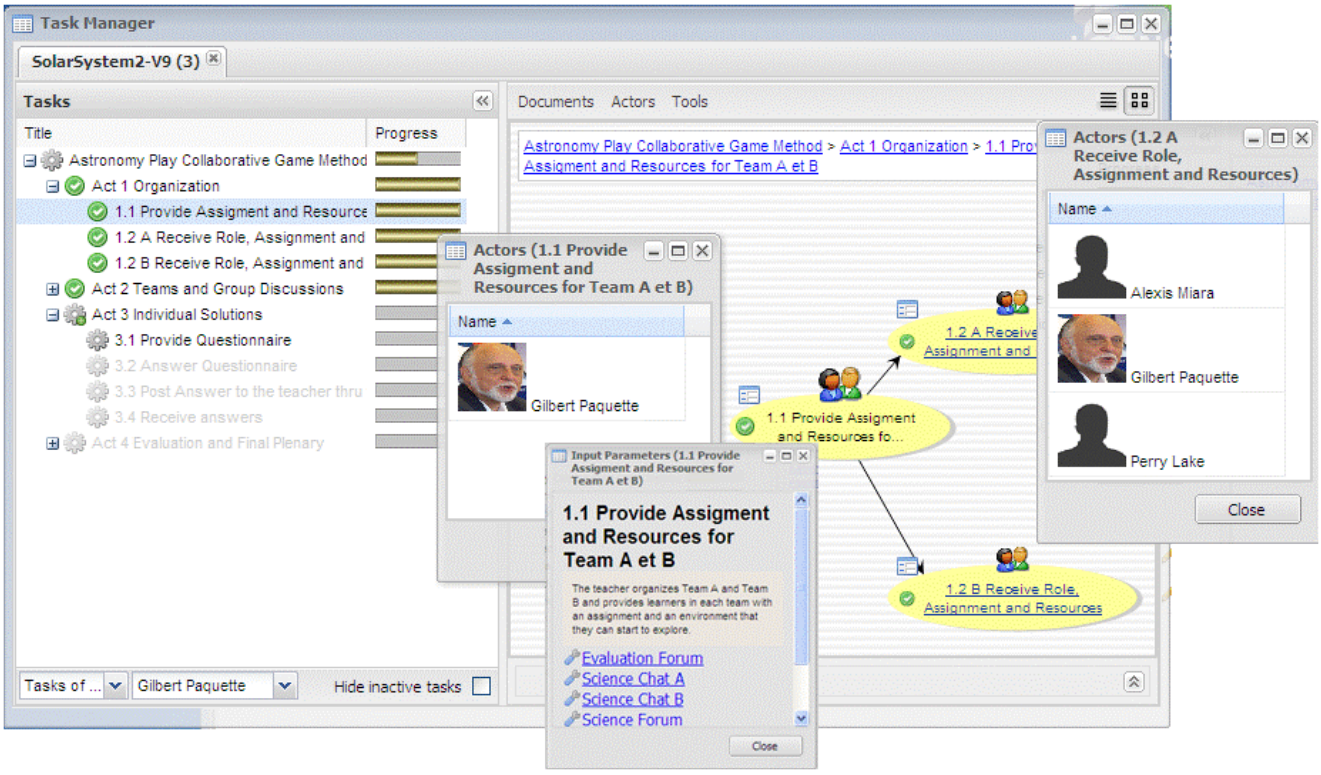

Figure 15.16 - Contextual graphic views in the task manager

Conditions and control at run time

Figure 15.17 shows how a teacher can adapt a scenario at run-time thanks to conditions in the scenario. A sub-model for the team A discussion starts by opening the chat service for team A. Then, the control splits between the learning activity 2.1.A, where team A learners discuss documents on planet properties, and the support activity 2.2.A performed by the teacher where he observes the team A discussion.

The teacher's part is highlighted on the figure. After a certain time activity 2.2.A, observing the chat in team A, the teacher can either stop the discussion or provide additional information (Clue A) to help the learners solve the problem. The learners can also decide to stop, either before or after they have received this additional information. A similar pattern rules the discussion for team B, with the same teacher acting as a facilitator for both teams, each with a different set of planet properties as additional information.

The conditions shown on figure 15.17 are rules expressing the equivalent of IMS-LD level B properties. The decision “Need Clues or Stop Team A?” depends on its input data and the value “true” or “false” that a teacher action will produce in activity 2.2.A. If the value “Stop team A discussion” is true, then the flow of control goes to the end symbol, after which the flow goes up to the main act 2 model. If the value of “Clues A needed” is true, the flow will proceed to the teacher’s activity where he will select a document named “Clue A”. If both input variables are false, the flow will come back to activity 2.2.A. In the task manager, a Web interface, shown on the bottom window, is provided to the Teacher actor to enter a value in the data objects “Stop Team A Discussion” and “Clues A needed”. Depending on these values, the condition “Need Clues or Stop Team A” will be executed to orient the flow of activities. The Condition Edition windows on the Figure is where above rules are edited, here showing the rule “If Clues A needed is true and Stop Team A discussion is false, then go to 2.3A Add clues for team A.

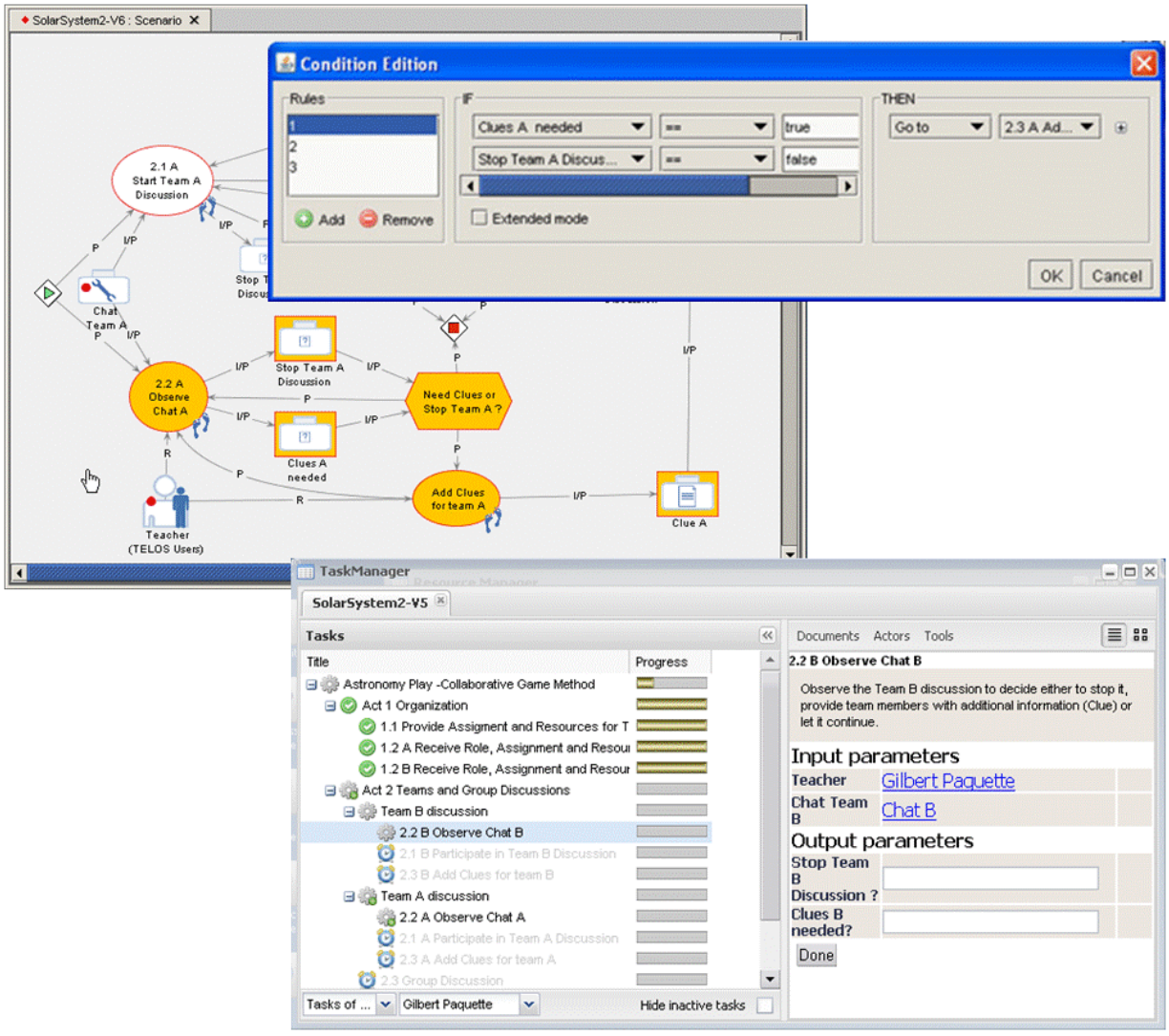

Figure 15.17 - Condition specification and execution at run time

Conclusion to Chapter 15

We now summarize the benefits we expect from this ontology-driven design of TELOS and, more generally, from ontology-driven systems.

1. *Fidelity from Requirements to Code:* Capturing in an ontology the main use cases and conceptual architecture concepts improves the fidelity of the final system with respect to initial requirements. Transforming the Conceptual ontology to a technical ontology embedded in the system ensures that the code will respect the architecture requirements. Also, the ontology-driven aspect of TELOS eases its evolution when new concepts will need to be integrated in the system.
2. *Global Systemic View.* The technical ontology can be seen as a kind Virtual Campus/Enterprise model. It provides a global view to support the cohesion of the activities, from the upper level where an institution can create a global workflow to coordinate its major processes, to the lower levels of a design scenario-based platform and scenario-based applications. An example of this will be given with the GIT project presented in the first section of chapter 19.
3. *Extended set of actors.* Compared to the commercial LCMS in operation this new global approach leads to an unlimited set of actors. At any level any number of actors can be defined and really supported.
4. *Better process coordination.* The fact that the system holds a model of the processes and the support resources leads to better process coordination. Especially in distance universities or distributed organizations, this provides a better assurance that the quality of services will be maintained when the personnel changes or it must provide new products to other actors.
5. *Visible scenarios and workflows*. Learning scenarios or workflows can always be consulted in a Web portal interface such as the Task Manager, links between resources, activities and actors can be seen right away. Each user taking an actor's role can visually see the context of the activities he has to perform, what resources to use or produce and with whom he is to interact with.
6. *Flexible and adaptable environments*. Each environment operates according to a technical ontology which is an integral part of the system. This enables very flexible and adaptable environments. If a new kind of actor, activity or resource needs to be introduced, this is done simply by modifying the instances or classes of the ontology, without changing the main operations of the system.
7. *Resource reusability* is a goal pursued by many advocates of learning object repositories, but it is not that easy to achieve. Using ontologies to annotate each resource within the same framework, and adding connecting operations to take care of possible technology mismatches brings solutions to many reusability problems.
8. *System interoperability*. With TELOS, it is possible to bring different technologies and different platforms to work together. For example, a designer could built a course using a scenario editor in one platform, and transfer it to TELOS to add new functionalities, for example personalized assistance. This process can be designed by defining the aggregation scenario between platforms at the technologist's level.
9. *Modeling for all*. Modeling is not an easy task but it is important enough to make it accessible not only to engineers and technologists, but also to instructional designers, learners and trainers.
10. *Focus on learning and work designs*. Finally, we hope the proposed approach will reduce the technology noise that is often present in eLearning applications when too much time is devoted to solving pure technology problems, instead of focusing on learning problems. We hope the activities will be more focused on pedagogy and on the quality of educational services or knowledge management activities.

These approaches offer new possibilities but also pose additional challenges. The LORNET five year research project having ended, some considerable refinements will happen. We will also need to ensure a

user friendliness of the tools that will be novel to most users. But our hope is that the results achieved here will lead the way to future research and developments and fruitful applications to Web-based learning and knowledge management systems.

References

Baader, F., D. Calvanese, D.McGuinness, D. Nardi and Patel-Schneider, P. (Editors) (2003) *The Description Logic Handbook*. Cambridge University Press.

Correal. D., Marino O. (2007) *Software Requirements Specification Document for General Purpose Function's Editor* (V0.4). LORNET Technical Documents, LICEF research centrer, Télé-université.

Davies, J., van Harmelen, F., Fensel, D. eds. (2002) *Towards the Semantic Web: Ontology-driven Knowledge Management.* John Wiley & Sons, Inc., New York, NY, USA

ELF – eLearning framework (2007) http://www.elframework.org/, last consulted June 14, 2007.

IMS-LD (2003). IMS Learning Design. Information Model, Best Practice and Implementation Guide, Binding document, Schemas.

Kleppe, A. G., Warmer, J. B., & Bast, W. (2003). *MDA explained : the model driven architecture : practice and promise.* Boston: Addison-Wesley.

Magnan, F. and Paquette, G. (2006) TELOS: An ontology driven eLearning OS, SOA/AIS-06 Workshop, Dublin, Ireland, June 2006

OKI – Open Knowledge Initiative (2007) http://www.okiproject.org/, last consulted June 14, 2007

Paquette G. (2008) Graphical Ontology Modeling Language for Learning Environments, *Technology, Instruction., Cognition and Learning* , Vol.5 , p.133-168, Old City Publishing, Inc.

Paquette G and Magnan F. (2008) *From a Conceptual Ontology to the TELOS Operational System*. ITS Workshop on "Ontology-based Learning Resource Repositories, www.lornet.org, last consulted December 2008)

Paquette G. and Magnan F. (2007 *Learning Resource Referencing, Search and Aggregation At the eLearning System Level.* LODE Workshop, ECTEL-07 Conference, Crete, September 18-21, 2007

Paquette G. and Rogozan D. (2006) *Primitives de représentation OWL-DL* - Correspondance avec le langage graphique MOT+OWL et le langage des prédicats du premier ordre. TELOS documentation. LICEF Research Center. Montreal, Québec.

Paquette, G. (2001) Designing virtual learning centers. In H. Adelsberger, B. Collis, J.P.E., ed.: *Handbook on Information Technologies for Education & Training*. International Handbook on Information Systems. Springer-Verlag 249–272

Paquette, G. (2003). *Instructional Engineering for Network-Based Learning.* Pfeiffer/Wiley Publishing Co, 262 pages, décembre

Paquette, G. and Magnan, F. (2008) An Executable Model for Virtual Campus Environments in H.H. Adelsberger, Kinshuk, J.M. Pawlowski and D. Sampson (Eds.), International *Handbook on Information Technologies for Education and Training*, 2nd Edition, Springer, Chapter 19, pp. 365-405, June 2008

Paquette, G., Rosca, I. (2003) Modeling the delivery physiology of distributed learning systems. *Technology, Instruction, cognition and Learning* 1-2 183–209

Paquette, G., Rosca, I., Masmoudi, A., Mihaila, S. (2005) *Telos conceptual framework* v0.8. Lornet technical documentation, Télé-Université

Paquette, G., Rosca, I., Mihaila, S., Masmoudi, A. (2006) Telos, a service-oriented framework to support learning and knowledge management. In Pierre, S., ed.: *E-Learning Networked Environments and Architectures: a Knowledge Processing Perspective.* Springer-Verlag

Rosca, I. (2005) TELOS Conceptual Architecture, version 0.5. LORNET Technical Documents, LICEF research centrer, Télé-université, Montreal.

Tetlow, P., Pan, J., Oberle, D., Wallace, E., Uschold, M., Kendall, E. (2001) *Ontology driven architectures and potential uses of the semantic web in systems and softrware engineering.* http://www.w3.org/2001/sw/BestPractices/SE/ODA/051126/

W3C (2004) *OWL Overview Document* (http://www.w3.org/TR/2004/REC-owl-features-20040210/)

Wilson, S., Blinco, K., Rehak, D. (2004) *Service-oriented frameworks: Modelling the infrastructure for the next generation of e-learning systems.* White Paper presented at the alt-i-lab'04