Expert Systems with Applications 41 (2014) 1609–1621

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An expert system hybrid architecture to support experiment management

Antonino Fiannaca [a,*], Massimo La Rosa [a], Riccardo Rizzo [a], Alfonso Urso [a], Salvatore Gaglio [b]

[a] ICAR-CNR, National Research Council of Italy, Viale delle Scienze, Ed. 11, 90128 Palermo, Italy
[b] DICGIM, Universitá di Palermo, Viale delle Scienze, Ed. 6, 90128 Palermo, Italy

ARTICLE INFO

Keywords:
Expert system
Workflow management system
Hybrid architecture
Ontology

ABSTRACT

Specific expert systems are used for supporting, speeding-up and adding precision to in silico experimentation in many domains. In particular, many experimentalists exhibit a growing interest in workflow management systems for making a pipeline of experiments. Unfortunately, these type of systems does not integrate a systematic approach or a support component for the workflow composition/reuse. For this reason, in this paper we propose a knowledge-based hybrid architecture for designing expert systems that are able to support experiment management. This architecture defines a reference cognitive space and a proper ontology that describe the state of a problem by means of three different perspectives at the same time: procedural, declarative and workflow-oriented. In addition, we introduce an instance of our architecture, in order to demonstrate the features of the proposed work. In particular, we model a bioinformatics case study, according to the proposed hybrid architecture guidelines, in order to explain how to design and integrate required knowledge into an interactive system for composition and running of scientific workflows.

1. Introduction

Expert Systems (ESs) are designed to support users in decision making process. One of the most important features of a ES is the capability to make automated inference and reasoning. In the last years, other type of systems, called Workflow Management Systems (WFMSs) (Hollingsworth, 1995), have been developed in order to support researchers in scientific fields, such as biology or chemistry. In details, a WFMS implements a process management approach (Ko, 2009) that allows user to produce his own workflow in order to solve a given problem.

In this paper, we propose a novel hybrid architecture for expert systems that support experimentalists in process management, by means of an execution environment that give scientists assistance to build a workflow of operations, using elementary components and/or reusing previously provided sub-workflows. In order to accomplish the realization of an intelligent WFMS, this architecture proposes to arrange both the domain knowledge and all the components (data, tools and services) necessary to produce a scientific workflow in a cognitive[1] space: each point in this reference space represents a cognitive status of the system in terms of knowledge representation, reasoning inference and workflow design. By means of this approach, we aim at integrating three different representations of a problem as a coherent design method for expert systems. The integration of these representations is done with respect to two point of views: the planning area and the workflow building one.

The knowledge of the proposed architecture is organized by means of an ontology in a twofold manner: declarative and procedural knowledge are integrated following a biological inspired point of view. Although they usually are considered as two different reasoning approaches, we assume declarative knowledge as integrated on a procedural knowledge, according to the modularity processes in the human brain. In facts, as (Ten Berge & Van-Hezewijk, 1999) stated, the brain has two kinds of memory: the procedural one is responsible for physical activities, just like a technique applied when necessary; whereas the declarative one contains the symbolic knowledge, responsible for storage of facts and events. In other words, we follow the elementary observation that different problems require different approaches and most of them can not be solved only with a sequence of activities or, alternatively, with a set of cognitive skills.

Furthermore, the proposed architecture also aims at defining and executing those activities that are critical for achieving specific objectives and delivering desired outputs. For this reason, we investigated the workflow-oriented approach, following a business process point of view (Thurner, 1998).

* Corresponding author.
E-mail address: fiannaca@pa.icar.cnr.it (A. Fiannaca).

[1] In the sense it is related to artificial intelligence and knowledge representation, without any psychological meaning.

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.08.058

According to this point of view, the hybrid architecture integrates a knowledge management component that contains all the elements necessary to manages both interaction among processes, and all the inputs and outputs that bond these processes together. This way, each process is handled as an integrated set of activities that uses resources to transform inputs into outputs or, in other words, an experiment obtained with the proposed architecture can be considered consistent whenever several processes are interconnected using such input–output relationships. The result of this approach is the composition of a flowchart (sequence of activities) or a workflow of tasks.

As previously stated, the proposed hybrid architecture aims at integrating the three main approaches described above (procedural, declarative and workflow-oriented one). According to the coexistence between two knowledge representation related to the human reasoning, our architecture uses both declarative and procedural approaches at different times, taking advantage of their different features. Moreover our proposed architecture, following the workflow-oriented approach, is able to guide scientists to the composition of a workflow by handling some elementary units that are interrelated via semantic interconnection elements; their aggregation generate sub-processes that allow the modularization of workflow itself. In this contest, the proposed ontology guarantees the coexistence of these three different approaches, in fact it contains some essential entities and relationships allowing reasoning in a twofold manner (declarative and procedural) over an abstract workflow of operations.

The paper is structured as follows. In the next section we briefly discuss some related works in the field of previously cited approaches. In Section 3 and Section 4 we define in details our hybrid architecture as a three–folded cognitive space supported by the ontology that model knowledge and workflow organization. In Section 5, we detail how the proposed hybrid architecture can be implemented in order to produce a consistent workflow of operations that can solve a bioinformatics issue. Finally, in the last section we provide conclusive remarks.

2. Related work

In the last years, a combined declarative-procedural approach was adopted for modelling some complex systems oriented to human–computer interaction. This hybrid approach seems to be useful especially for Decision Support Systems (DSS). In the medical field, (Mulyar, Pesic, Aalst, Peleg, & Carmel, 2008) proposes to add flexibility to classical clinical computer-interpretable guidelines, introducing a procedural component. More in detail, declarative approach lets the user decide how to work depending on the possible scenarios; in turn, it introduces a set of tasks and some dependencies between these tasks. A similar idea was proposed by Smelik, Tutenel, De Kraker, and Bidarra (2011) in the field of computer graphic, where an interactive declarative module was integrated to procedural modelling of virtual worlds. This way, designers are supported on stating what they want to create, reducing the complexity of making virtual worlds by combining semantic-based modelling with manual and procedural approaches. In particular, as it will be illustrated in next sections, our proposed system exploits the same technique adopted by Smelik et al. (2011), regarding the use of a layered data structure.

Unfortunately, the above described architectures are not able to handle systems that also require to produce and execute a pipeline (i.e. a workflow) of customized processing services. As previously stated, these kind of systems, also called WFMSs, implement a process logic inherited by business process management in order to build scientific workflows in many research topics. Scientific workflows differ from traditional business workflows (Ludäscher et al., 2006). The main difference is that business workflows focus above all on control flow, whereas scientific workflows are dataflow-oriented. This difference influences in their execution models and visual formalism. In fact business workflows are usually represented by means of flowcharts or state transition diagrams. Scientific workflows, on the other hand, are shown as dataflow process networks (Lee & Parks, 1995), in which each process is composed of a dataflow actor, representing a single processing step.

Nowadays, two of the most used and famous WFMS for scientific workflows are Taverna (Hull et al., 2006) and Kepler (Altintas et al., 2004). Taverna is a system that integrates services, tools an databases available both locally and on the web, in order to build and run workflows for complex biological and bioinformatics tasks. The system uses a GUI that integrates a workflow designer with drag and drop components. Kepler models a scientific workflows through the composition of processing components, called actor, that interact each other by means of interfaces. A workflow execution can be defined using an object called director that sets up the execution order and the communication details of the actors involved into the workflow. Both Taverna and Kepler are oriented to support the researchers, simplifying the selection and execution of predefined workflows or atomic processing components in order to compose the desired scientific workflow. Nevertheless, a typical user of these systems should have the necessary domain knowledge and skill in order to choose and link together the proper tools to accomplish an experiment; in fact, available WFMSs have not a reasoning component that supports user, by suggesting and assembling the proper tools and services in order to build the desired goal. The need of an intelligent system that can support the automatic composition of services and their smart linking during workflow design represent, in fact, one of the requirements for scientific workflow management system (Ludäscher et al., 2006).

A first attempt to incorporate a declarative approach, into a workflow system, was proposed by Moreno and Kearney (2002). More in detail, authors integrate an artificial intelligence planning phase that generates a sequence of activity linked by dependences; each activity will be translated into a partial job of the workflow, corresponding to the partial plan. This way, they are able to manage alternative control flows and different incomplete portion of plans.

Later, authors in Chung et al. (2003) introduced another project that aim to extend standard workflow systems with dynamically changing processes. In order to obtain an adaptive workflow, authors added a knowledge-based component to WFMS, that enable system to make reasoning about processes within the problem domain. The reasoning contribution is used, according to the declarative approach, for the selection of some plans, that define a set of tasks, together with their ordering constraints. Those tasks can be planned at different hierarchical levels (considering also sub-tasks) and represent the structure of processes, that will be implemented into the workflow.

With regards to the last two works, in order to add decision making features to workflow management, we explicitly consider a procedural component that is combined with a declarative approach. Considering all the previous systems, our architecture uses both a declarative and procedural approach and it integrates them with a process oriented approach in order to provide decision making and workflow management features at the same time. Moreover, we developed an ontology, called Data-Problem–Solution-to-Experiment, in order to organize the knowledge base of the proposed architecture according to the three-folded approach. Ontologies, in fact, are usually adopted in order to organize in a well structured way the knowledge of an expert system (Chandrasekaran, Josephson, & Benjamins, 1999) Expert systems can support different kinds of application domains depending on the content and the organization of its own knowledge-base

(KB). For example, authors in Chen, Huang, Bau, and Chen (2012) used two different ontologies in order to model the knowledge about both drugs and patients. By means of that knowledge structure, they adopted a set of rules and an inference engine in order to analyse the symptoms of diabetes and to suggest treatment. Ontological knowledge-based systems have been adopted in a wide range of application domains, from the management of supply chains (Cheung, Cheung, & Kwok, 2011) and gas and oil facilities (Zarri, 2011), to the decision support for managing emergencies (De Maio, Fenza, Gaeta, Loia, & Orciuoli, 2011). An interesting approach was suggested by Shue, Chen, and Shiue (2009): they developed a knowledge-based expert system to assess financial quality of an enterprise considering a domain knowledge, structured with an ontology, in order to model the accounting categories and their relationships, and an operational knowledge consisting of a set of production rules which use the domain knowledge to automatically process the assessment.

3. Hybrid architecture

As stated in the previous section, the proposed hybrid system collects at the same time three different perspectives: declarative, procedural and workflow-oriented one. The coexistence of these different approaches to the same architecture is guaranteed by assuming a working space that can be visually represented as a three-folded cognitive space, where each component represents a perspective (view) of a proper approach. It is important to notice that this space does not define an euclidean or metric space, but it allows us to underline relationships among the three main concepts at the basis of our hybrid architecture. In other words, when an expert system implementing this architecture is running, a point inside the cognitive space will identify the current state of the system in terms of three well defined views.

Fig. 1 provides a visual representation of the cognitive space implemented in the hybrid architecture; the main characteristics of each component are reported in the following.

Abstraction layer perspective (based on procedural approach):

1. it is responsible for "how to achieve" a specific result for an input problem;
2. it allows the decomposition of complex problems into different abstraction layers;
3. it manages the organization of elements of the different layers.

Decision making level perspective (based on declarative approach):

1. it is responsible about"what to do" given an input problem;
2. it deals with unstructured data;
3. it manages all decision making steps for the problem solving process.

Workflow timeline perspective (based on workflow oriented approach):

1. it is responsible for the generation of a workflow of operations, dealing with the execution of each processing element;
2. it is responsible for the running all the algorithms and services, taking care of the management and organization of issues related to inputs-outputs interface;
3. it allows reconfiguration of each selected tool or service, with back-tracking features;
4. it collects all the intermediate results, saving the process representation of the problem.

In the next subsections, first we will explain how the reasoning activity of the proposed architecture integrates with the above described perspectives. Moreover we will analyse the three perspectives, one by one, explaining what they represent for our hybrid architecture. Afterwards, we will describe these perspectives pairwise, underlining other interesting features of the proposed space.

3.1. Reasoning activity

The reasoning capabilities of the proposed architecture integrate both a direct reasoning according to user request, and the ability to takes into account the available resources and knowledge for decision making.

According to the guideline suggested by Baker et al. (2002), the reasoning activity is composed of the following steps, reported in the Fig. 2.

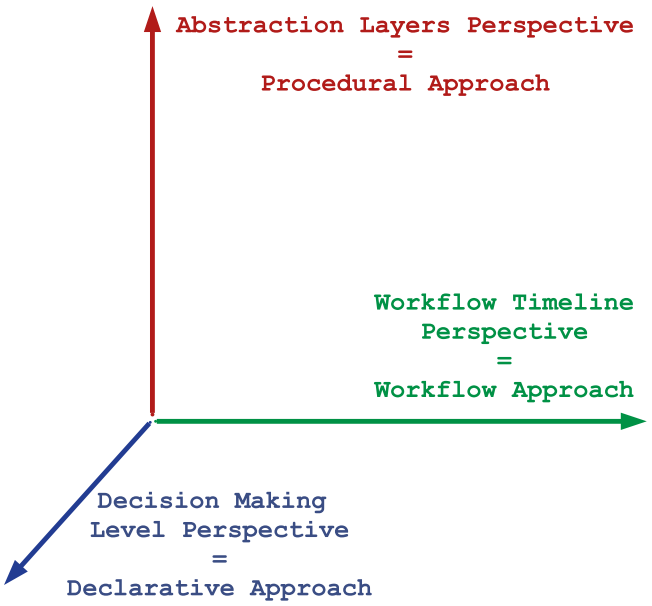

Fig. 1. A visual representation of the cognitive space of the hybrid architecture. It integrates three perspective for the problem, i.e. abstraction layers (based on procedural approach), decision making levels (based on declarative approach) and workflow timeline (based on workflow oriented approach).

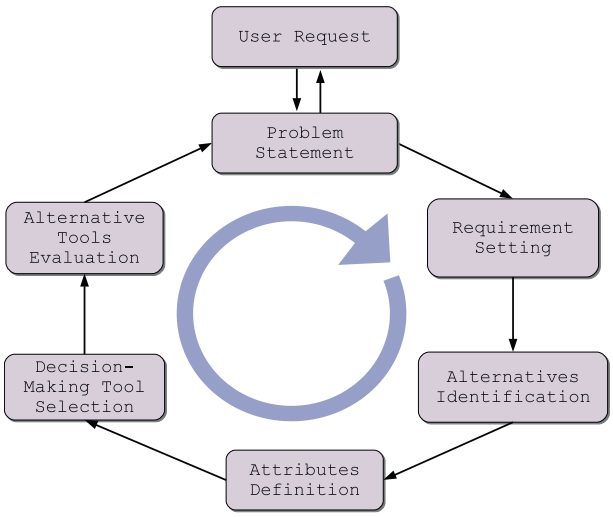

Fig. 2. Reasoning activity guidelines for the proposed architecture.

1. Problem statement: In this phase, the reasoning process identifies the current goal (also in case of complex decision problems) according to the user request and the actual knowledge of the system.
2. Requirement setting: Constraints describing the set of the admissible solutions to the problem detected in step 1 are analysed: i.e. for any possible solution it has decided unambiguously whether a strategy is acceptable or not.
3. Alternatives identification: Alternatives strategies or heuristics offering different approaches for finding a solution have to be evaluated, in order to better match with the user desired goal and the boundary conditions.
4. Attributes definition: In order to support the comparison of different alternatives, it is necessary to define discriminating criteria to measure how effectively well each alternative achieves the goal or almost a sub-goal.
5. Decision-making tool selection: Although it could exist several tools for solving a decision problem, the selection of the appropriate tool depends on the concrete decision problem, as well as some characteristics of a tool (requirement of additional resources, computational complexity) or computing power. The selected tool is proposed to user with a list of pros and cons.
6. Alternative tools evaluation: Since more than a tool can satisfy discriminating criteria, it is shown to the user a set of the most promising alternative tools and/or services, once again with a list of pros and cons for each tool and/or service. In complex problems, the proposed alternatives may also call the attention of the user, that could add further goals or requirements to the decision model.

The above described reasoning phases represent the reasoning guidelines implemented in our architecture. In the next sections it will be explained how these concepts can be matched with the three-folded perspective of our cognitive space. In particular, the most of reasoning activity steps will be recalled in Section 3.5, "Abstraction Layer Vs Decision Making Level".

3.2. Abstraction layer perspective

This perspective deals with an input problem according to its complexity. In facts, for each problem, the proposed architecture can support several views: from the top abstraction layer (i.e. the problem itself) to the bottom abstraction layer (i.e. tool/service instances). At each intermediate abstraction layer, the system architecture shows a different representation of the problem. Therefore, procedural analysis of the problem is done such as a top-down process, starting from its conceptual representation and refining its specification to an executable form in several refinement steps. The number of layers can change according to the implementation of the system architecture, and it is related to the detail level provided by the problem solving strategy. As reported by Smelik et al. (2011), we identify at least three mandatory layers, able to represent a problem with our architecture: the *Problem* (highest abstraction), the *Solution* (medium abstraction) and the *Object* (lowest abstraction) one. Problem Layer contains the problem definition and the input parameters. Here the problem is separated in a set of tasks, arranged according to the hierarchy of problems and sub-problems of minor complexity. Once a specific task is identified, the Solution Layer provides a platform allowing user to select alternative strategies and/or heuristics to solve this task. Finally, at Object Layer, it will be explored all the set of algorithms and/or services able to meet the target provided by the previous strategy.

As defined, each abstraction layer executes some processes where implementation details are hidden from all the processes at other layers, in fact each abstraction layer implements its specific procedural analysis. Since this perspective does not contain any problem solving method, only a cross-view among abstraction layers and decision making levels can provide resolution methods for a given input problem.

Fig. 3 shows the procedural approach for solving a problem at the three main abstraction layers. The choice of task(s), strategy(ies) and tool(s) is done by means of the declarative approach, representing the reasoning component of the proposed architecture. The integration of the declarative and procedural approach is done according to the modularity processes in the human brain as proposed by Ten Berge and VanHezewijk (1999) and previously described in Section 1.

3.3. Decision making level perspective

This perspective reports the main element of the decision making activity of the system. This activity in our architecture is organized in functional modules. Each module has two elements: the knowledge, containing information about a specific problem, and the expertise (skill), being able to making reasoning about its knowledge. For this reason, each module takes care of a specific part of the reasoning process and it is responsible for making decisions about a well defined goal.

Since problems could be very complex, their management could be difficult and, consequently, some large decision modules, i.e. containing a lot of knowledge and expertise, could be needed. For this reason, it is convenient to split problems into sub-problems (as said in the previous section), and, in turn, to build a hierarchy of modules and sub-modules, each of them containing the knowledge and expertise required to model and solve those specific sub-problems. The data structure used to link modules is the multi-tree.

For a specific problem we can represent this perspective by means of a treemap representation (Johnson & Shneiderman, 1991). A treemap is a visual representation for displaying tree-structured data as a set of nested boxes. A treemap allows us to show attributes of each leaf module using size and colour coding, providing an overall view of the entire hierarchy and making the navigation of large hierarchies much easier. An example of this

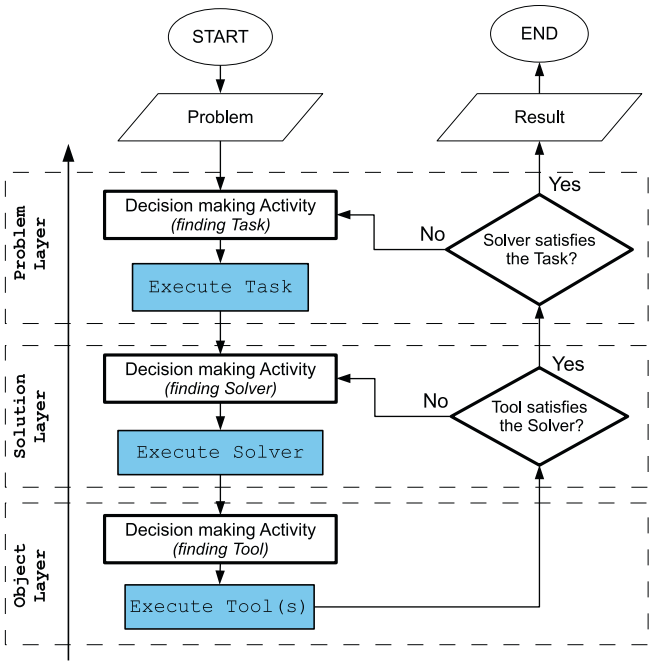

Fig. 3. Abstraction layer perspective. Flow-chart representation of procedural approach template, consisting of three abstraction layers.

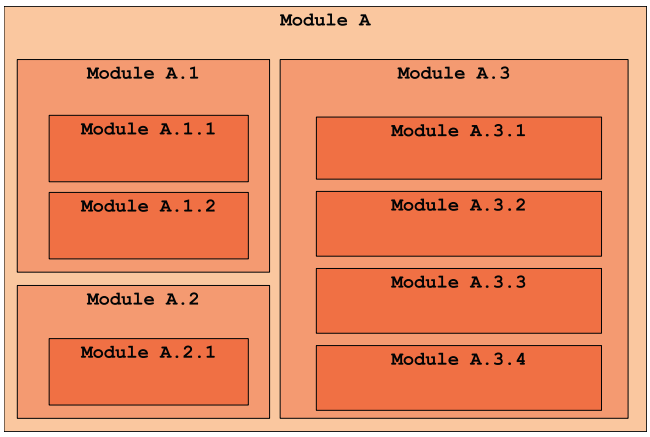

Fig. 4. Decision making level perspective. Hierarchical relationship among eleven decision making modules represented by means of a treemap.

representation is shown in Fig. 4. Here the problem solved by decision making activity of module A, can be broken down into three sub-problem, and so on. For instance, module A.3 contains four different ways to handle with a part of it. All the modules are clearly separated and users can compare modules and sub-modules even at varying depth in the tree, also detecting mutually related properties among modules. As we will show later, this representation is useful to offer a view of the whole space of the hybrid architecture.

3.4. Workflow timeline perspective

This view traces all the phases of the workflow generation. This one appears as the results of a sequence of decision making processes at different abstraction layers, where the main goal, subtasks, business processes and internal/external tools are specified.

In general, the obtained workflow is a collection of tasks organized to accomplish some business process. A task is performed by one or more software (e.g. preprocessing tools), or by means the human interaction (e.g., providing input commands), or a combination of these. In addition, the workflow defines the order of task invocation, task synchronization, and information flow (data-flow).

In Fig. 5a simple workflow of a generic problem is shown.[2] Yellow elements are the used algorithms/services, green files are input/output data and lilac blocks are tool resources. For the sake of simplicity, we can consider this workflow as represented at the object abstraction layer. The proposed hybrid architecture supports the evolution, replacement, and addition of workflow applications, as well as the re-engineering of system components and processes; in facts, users can navigate along this view in order to interact with the sequence of tools, changing algorithms, services and/or parameters. Notice the workflow does not clearly contain any decision element, because, as we will show later, the reasoning activity is accomplished by the decision making level.

In the following we analyse the interaction between each pair of perspectives, in order to discuss the features of these projection planes.

3.5. Abstraction layer Vs Decision making level

This view is the combination of abstraction layer and decision making level perspectives. From the decision making activity point of view, we can define this view as the abstraction-decomposition of the reasoning process. In fact, with respect to Fig. 2, we can identify respectively *Problem Statement* and *Requirement Setting*, as a decision making activity at highest abstraction layer (Task Layer); *Alternative Identification* and *Attributes Definition*, as the activity at medium abstraction layer (Strategy Layer); *Decision Making Tool Selection* and *Alternative Tools Evaluation*, as the activity at lowest abstraction layer (Tool Layer).

A representation of the two combined perspectives is reported in Fig. 6, where decision making modules are arranged into three layers, according to Fig. 3. It is important to notice that the hierarchical organization of the modules along the abstraction layers perspective, is the unpacked view of the treemap representation in Fig. 4. In Fig. 6, each rectangle (each module) makes reasoning with an inference mechanism that can be represented, for instance, with a decision tree (the blue tree-like structure) where each leaf node can give focus to a lower layer decision making module. By means of the combination of this two perspectives, we can navigate through the hierarchy of the entire reasoning tree for exploring sub-modules in different abstraction layers; this way, an user can interact with this view in order to both see in a glass-box the inference mechanism behind the reasoning of the system and learn about each reasoning path, working at task, strategy and tool layers. Communication between decision modules is managed from parent to child: each child is able to solve a specific task that its parent can only propose to solve, without having the knowledge about it. As shown in Fig. 6, the parent module A can give focus to child module A.1 in order to request the solution about a specific strategy and, in turn, the module A.1 can give focus to its child A.1.1 to reasoning about the tool.

3.6. Abstraction layer Vs Workflow timeline

This view allows to see the data pipeline at different abstraction layers, according to the problem complexity. In facts, it shows several views of the workflow: from the top abstraction layer, where the workflow is composed by the sequence of tasks, to the lowest abstraction layer, where the workflow is composed by the sequence of executed tools.

Fig. 7 shows an example of this view, where the problem is trivially solved by one task, called "Main Task". This view is able to represent only dependences among tasks, strategies and tools, without information about decision process on the selection of a specific strategy or tool; for instance, in Fig. 7, the element "Strategy 1" could have a plan for executing "Tool 1" and "Tool 2" in sequence (if they belongs to the same decision making module) or two different plans for respectively "Tool 1" and "Tool 2" (if they belong to two different decision making modules). This last information is given by the decision making level perspective.

3.7. Decision making level Vs Workflow timeline

This view allows to investigates the interaction between decision making level and workflow timeline perspectives. In particular, when an user analyses a part of the whole pipeline, he can capture reasoning behind a single tool, exploring decision making modules responsible for the choice of this tool. This feature allows the user to modify the sequence of tools and to change algorithms and/or parameters; in fact, the proposed hybrid architecture supports the evolution, replacement, and addition of workflow applications, as well as the re-engineering of system components and processes.

Fig. 8 reports six decision making modules and three tools in a view containing both perspectives. In this figure, with respect to Fig. 4, boxes representing modules are arranged in a three-dimensional treemap, where only active modules are reported, i.e. those modules that lead to make an tool. According to the Fig. 5, tools

[2] For interpretation of color in Fig. 5, the reader is referred to the web version of this article.

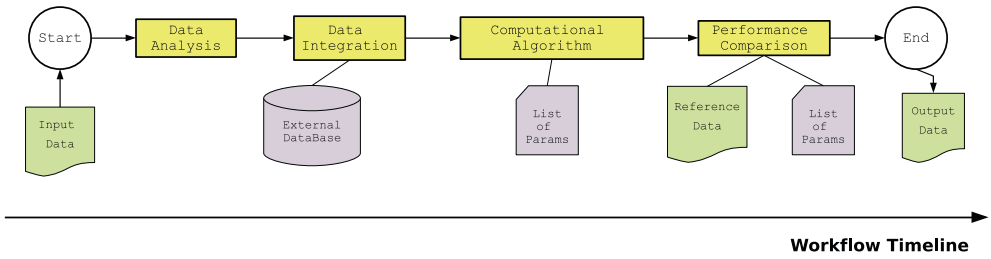

Fig. 5. Workflow timeline perspective. A view of a simple sequence of algorithms. No information about abstraction layers or decision making modules is taken into account.

follow the workflow timeline, that give us the sequence (execution order) of tools into the data pipeline. Fig. 8 also shows that, in order to resolve a required strategy, more than a tool could be managed by the same decision making module. For example, two tools (here called `Tool 1`, `Tool 2`) have been executed under the supervision of the module `A.1` and then it gives the focus back to the parent module `A`.

4. Knowledge organization

In the previous Sections, we described our three-folded architecture for the design of expert systems. In order to fully define and structure the knowledge at the basis of our system, we also developed an ontological paradigm designed to exploit the main features of the hybrid architecture. In the next subsections we explain in details the main features and concepts of our ontology and we explain how this ontology matches with the three perspectives.

4.1. DPS2E ontology

In our proposed architecture, we adopt a general purpose ontology called Data-Problem–Solution-to-Experiment (DPS2E) that matches, as we will explain up next, with the concepts discussed in Section 3. DPS2E represents an extension of our previously published ontological specification for knowledge organization in expert systems design, called Data Problem Solver (DPS) (Fiannaca, La Rosa, Rizzo, Urso, & Gaglio, 2012c; Fiannaca, Gaglio, La Rosa, Rizzo, & Urso, 2012a; Fiannaca, La Rosa, Rizzo, Urso, & Gaglio, 2013). At the basis of our new ontological paradigm, whose main assumptions have been briefly anticipated in Fiannaca, La Rosa, Gaglio, Rizzo, and Urso (2012b), there is the link between declarative knowledge, i.e. the organization of the knowledge, usually provided by an expert of the domain, into a stable structure for problem specification in an application domain; and procedural knowledge, meant as how declarative knowledge can be put together in order to obtain a sequence of processing components (workflow).

In DPS2E, in fact, we clearly define a Knowledge Area (KA), responsible for its declarative assumptions, and an Execution Area (EA), which models the concepts of workflow and experiment. DPS2E ontology, implemented using Protege ontology editor (Noy et al., 2001), is shown, using an UML notation (Kogut et al., 2002), in Fig. 9. From here on, **bold font** will be used for the classes, and `typewriter font` will be used for the relationships among classes. In the upper part of Fig. 9 it is shown the KA, whereas in the lower part there is the EA. DPS2E is defined in terms of four main entities: **Data**, **Problem**, **Solution**, **Experiment**. All of these concepts are modeled by means of metaclasses, i.e. a class whose instances are other classes. In the same way a class defines the features of its objects, a metaclass sets the behavior of other classes and their instances. **Problem**, **Data** and **Solution** concepts belong

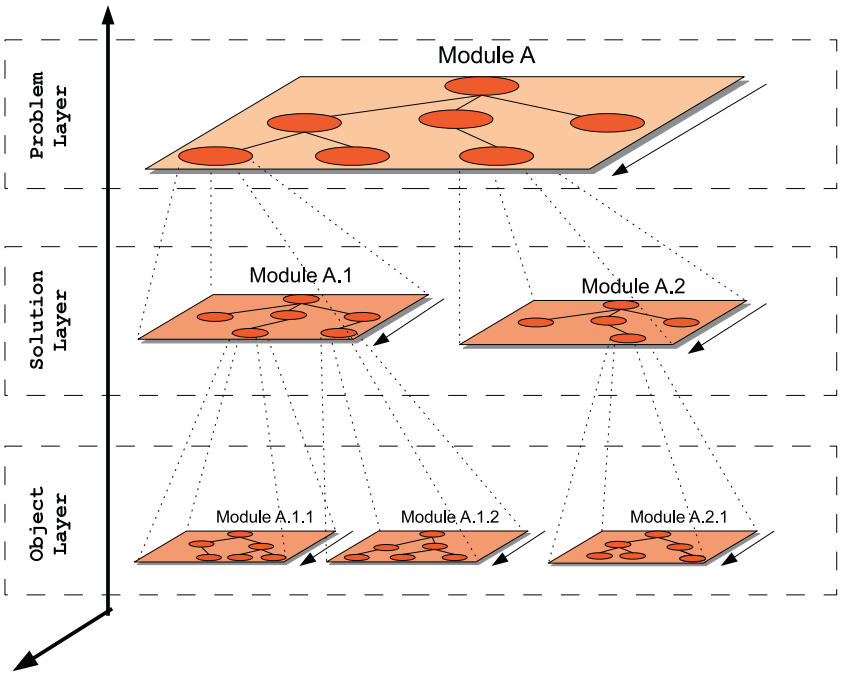

Fig. 6. Abstraction layer Vs. Decision making level perspectives.

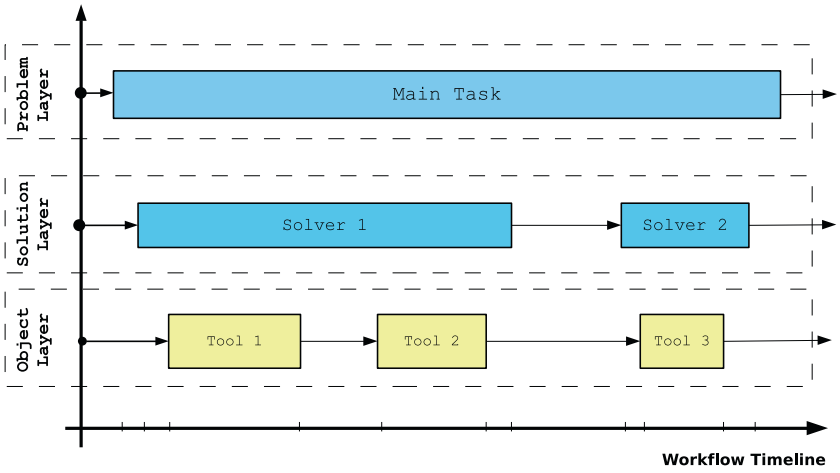

Fig. 7. Abstraction layer Vs. Workflow timeline perspectives.

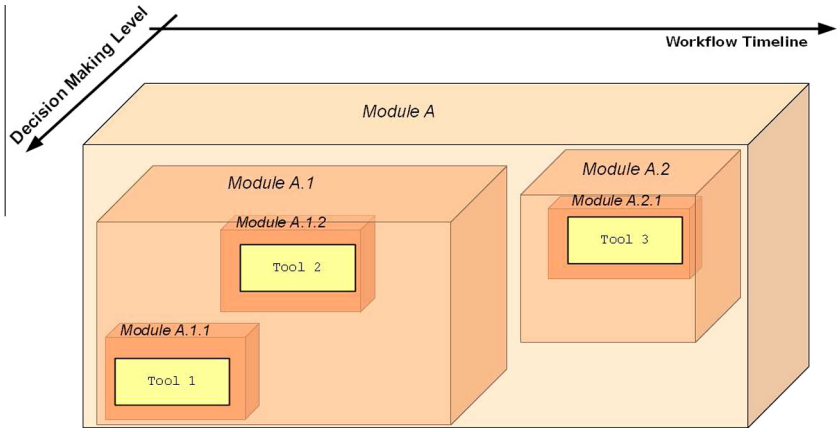

Fig. 8. Decision making Level Vs. Workflow timeline perspectives.

to the KA. **Problem** metaclass represents the actual formulation of issues that can be addressed in a certain application domain. **Data** metaclass encloses all the information, in terms of input and output, needed to deal with a problem. **Solution** metaclass models the necessary skill and expertise in order to solve a problem. **Problem** and **Data** concepts are linked with the mutual relation `uses/isUsedBy`; **Problem** and **Solution** are bound with the mutual relation `resolves/isResolvedBy`. This well separated modeling among **Data**, **Problem** and **Solution** clearly distinguishes among a problem to address from the way to solve it and the information content required to solve it. On the other hand it allows to take advantage of re-usability and modularity properties of ontologies. Each of these three main branches, in fact, can be updated or modified without changing the other ones. For example it is possible to add new **Solution** instances for a given problem without modifying the definition of the problem itself. Unlikely the original DPS paradigm, in DPS2E we introduce the concept of **Experiment**, a metaclass representing the implementation of a **Problem** in terms of its workflow representation and actual running of all its processing components. **Data**, **Problem**, **Solution** and **Experiment** describe very general abstract concepts: their ontological definition will be discussed in the following Sections.

4.1.1. Knowledge Area

The Knowledge Area of DPS2E ontology collects **Data**, **Problem** and **Solution** metaclasses and their ontological models. **Problem** metaclass, meant as "what to do" in an application domain, is simply instantiated as a hierarchy of **Task** metaclasses. This decomposition allows to consider the problems to solve at different abstraction levels, from more complex goals to simpler ones. The slots of **Task** metaclass are reported in Table 1. Each **Task** has a sub-task field that can contain one or more subtasks of decreasing complexity. Precondition slot indicates if a task requires the previous accomplishment of other tasks. Finally each **Task** has a link with an instance of **Data** metaclass, through the `uses` relationship, and with a **Solution** instance, by means of the `isResolvedBy` relation. The remaining slots are described in Table 1.

Data concept aims at modeling the inputs and outputs related to an application domain. It is instantiated through the **Data_type** metaclass, representing an aggregation of domain-dependent information. For example, **Data_type** can model proteomic data, sequence data and so on. Each **Data_type** is characterized by a set of **Attributes**, that are typical properties of each **Data_type** instance. The generic features of each **Data_type** element are summarized in Table 2. A **Data_type** metaclass is coded through one or more instances of **Data_Format** class: for example in the case of genomic sequence data, **Data_Format** can be fasta or genbank format.

Solution part of DPS2E ontology is expanded with the **Solver** metaclass, whose instances describe groups of resolving methodologies and strategies for a given **Task**. In the same way a **Task** can be decomposed into subtasks of lesser complexity, a **Solver** can have other **Solvers** that deal with narrow parts of the original solving strategy. Each **Solver** instance, whose slots are shown in Table 3,

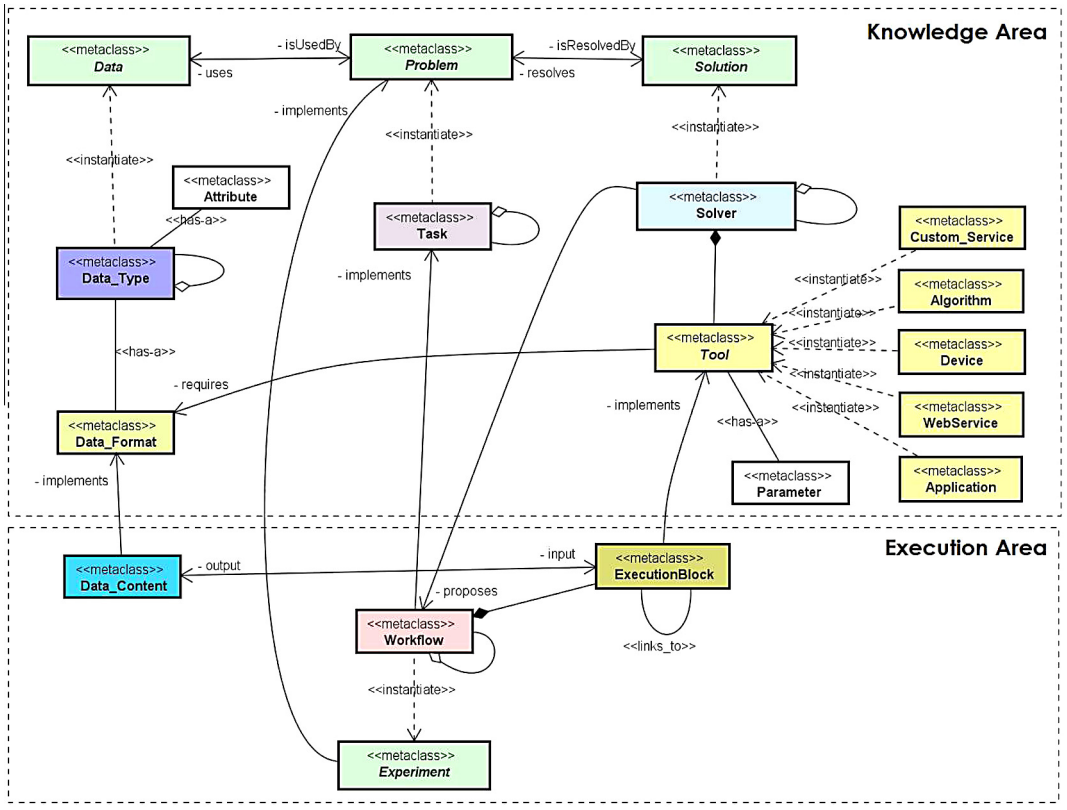

Fig. 9. Data-Problem–Solution-to-Experiment (DPS2E) ontology specification using UML notation.

is defined in terms of its pros, cons, computational approach (e.g. probabilistic or deterministic) and it is linked with the `resolves` relation to an instance of **Task** ontology. This bound determines a map between a task to resolve and its solving components. Finally each **Solver** element, in order to be actually processed, runs an instance of **Tool** metaclass. **Tool** models the processing units that are able to fulfill the solving strategy described by Solver's instances. In our DPS2E paradigm, **Tool** metaclass encloses the concept of external **Device**, **Web Service**, **Application**, **Algorithm** and **Custom Service**, i.e. a programmable component such scripts or simple code snippets. The slots of all **Tool** classes are shown in Table 4. Each **Tool** has a list of pros and cons, a list of inputs and outputs, meant as links to **Data_Format** objects, its computational complexity, bibliographical references, and a set of parameters modeled into a **Parameter** class.

4.1.2. Execution Area

The Execution Area represents the main improvement with respect to the original DPS paradigm. We introduced this new part in our ontology in order to build a bridge between the Knowledge Area and its implementation and visualization through a scientific workflow. KA, in fact, models the knowledge about an application domain only in terms of a set of issues (**Problem**), the inputs/outputs (**Data**), the solving strategies and their corresponding tools (**Solver** and **Tool**, respectively). EA is responsible for the organization of an abstract sequence of tasks, strategies and tools into a concrete workflow of operations, taking care of all the needed connection among workflow's components. The **Experiment** metaclass is the entity that translates into a workflow specification a **Problem** instance found in the KA. **Experiment** concept is instantiated by the **Workflow** metaclass. **Workflow** is defined in a recursive way, i.e. a **Workflow** class is composed of one or more **Workflow** classes through the composition relation. In our ontological paradigm, a **Workflow** is an aggregation of atomic entities called **ExecutionBlock**, that represent a single processing unit in a scientific workflow environment. **Workflow** and **ExecutionBlock** are bound to the concepts of KA in a straight way. **Workflow** instances match with **Task** instances and ExecutionBlock instances match with instances of **Tool** class, so that each task has its own workflow representation seen as a succession of **ExecutionBlocks**. Since, as we described in the previous Section, a **Task** can be decomposed into simpler subtasks, the recursive definition of **Workflow** class allows us to deal with task hierarchy considering a global workflow as composed of nested workflows, each one with

Table 1
The slots for **Task** metaclass in DPS2E ontology.

Slot name	Type	Inverse	Cardinality	Comment
`name`	String		*1*	Problem's name
`description`	String		*1*	A brief explanation
`uses`	Data	`isUsedBy`	*1..n*	The type of data needed
`isResolvedBy`	Solver	`resolves`	*1..n*	The implementation of the task
`sub-task`	Task		*0..n*	Task(s) that can manage a sub-process
`precondition`	Task		*0..n*	Required task(s) to be previously run
`reference`	String		*1..n*	One or more bibliographic references

Table 2
The slots for **Data_Type** metaclass in DPS2E ontology.

Slot name	Type	Inverse	Cardinality	Comment
name	String		*1*	Data's name
description	String		*1*	A brief explanation
isUsedBy	Problem	uses	*1..n*	The problem that uses the data
applicationDomain	String		*1*	Indicate the application domain
attributes	Attribute		*0..n*	A list of data attributes

Table 3
The slots for **Solver** metaclass in DPS2E ontology.

Slot name	Type	Inverse	Cardinality	Comment
name	String		*1*	Solver's name
description	String		*1*	A brief explanation
runs	Tool		*1*	The tool that implements Solver goal
resolves	Problem	isResolvedBy	*1*	The calling task
sub-solver	Solver		*1..n*	A list of component sub-solvers
proposes	Workflow		*1..n*	The proposed Workflow instance implementing the Solver
approach	String		*1*	Give the purpose of the tool
pros	String		*0..n*	A list of strong points for solving a task with a tool
cons	String		*0..n*	A list of weak points for solving a task with a tool
reference	String		*1..n*	Bibliographic reference(s) about the Solver solution

its own set of **ExecutionBlocks**. The slots for **Workflow** and **ExecutionBlock** classes are shown respectively in Tables 5 and 6. Finally in the EA **Data_Format** metaclass is matched to **Data_Content** entity, which is the real information content provided to **ExecutionBlocks** as input and obtained as output after a processing step. The ontological model of the EA can also be used to store pieces of already executed workflows, that can be suggested by Solver's instances by means of the proposes relationship between **Solver** and **Workflow** classes. This way the EA part of DPS2E ontology represents a kind of dynamic memory that is enriched as the scientific workflows are executed, so that it promotes the re-use of workflows.

4.2. DPS2E and hybrid architecture

DPS2E paradigm has been designed in order to support the proposed hybrid architecture and its cognitive space. Looking at Fig. 3, the **Problem** part of the ontology, in fact, can be matched with the highest abstraction layer, providing the knowledge useful to describe the tasks by means of the slot defined into the ontology (e.g. sub-task, isResolvedBy, pros, cons). The **Solution** part of the ontology defines the way to solve a task, and it can be matched with the Solution layer, because it defines the knowledge useful to describe strategies and heuristics in terms of approach, computational paradigm and implementing tools. The **Tool** part of the ontology, then, corresponds to the **Object** layer, with its knowledge related to algorithms, web services, devices and applications that actually run a **Solver** instance (i.e. a strategy). The KA of DPS2E ontology, then, can be populated with facts, i.e ontology instances, representing the information content needed by the decision-making perspective in order to perform its reasoning activity. KB's population is done by a human expert of the domain or extracting information by hand from scientific literature, following the templates defined in the ontology. The decision-making procedure can be carried out through a set of IF–THEN production rules and an inference engine. It is possible to define both general rules, based on the relationships among ontology entities, such as the isResolvedBy relation between **Task** and **Solver**, and more specific rules, that take also into account the slot values of the instances. Moreover, facts and rules belonging to the KA can be seen as high level rules whose purpose is to aid in the assembling of an abstract workflow composed of task, solver and tools instances. On the other hand, facts and rules related to the Execution Area guarantee the consistency of the workflow itself. These low level rules, in fact, deal with the workflow implementation details, like for instance the correctness of inputs and outputs, the need of data conversion or the priority and precedence among the **ExecutionBlocks**. Fact and rules of the EA, therefore, allow to translate an abstract workflow, seen as a sequence of tasks and solvers, into a concrete workflow that can be actually executed (Ogasawara, Paulino, Murta, Werner, & Mattoso, 2009; Ludascher, Altintas, & Gupta, 2003).

5. Case study

The concept of hybrid architecture for expert systems as reported in this work is quite general, in fact in order to be used in

Table 4
The slots for **Tool** metaclass in DPS2E ontology.

Slot name	Type	Inverse	Cardinality	Comment
name	String		*1*	Tool's name
description	String		*1*	A brief explanation
input	Data_Format		*1..n*	Type of input data
output	Data_Format		*1..n*	Type of output data
parameters	Parameter		*0..n*	A list of input parameters
pros	String		*1..n*	A list of tool's strong points
cons	String		*1..n*	A list of tool's weak points
complexity	String		*1*	Computational complexity
reference	String		*1..n*	Bibliographic reference(s) about the tool

Table 5
The slots for **Workflow** metaclass in DPS2E ontology.

Slot name	Type	Inverse	Cardinality	Comment
`name`	String		*1*	Workflow's name
`description`	String		*1*	A brief explanation
`implements`	Task		*1*	The Task implemented by the workflow
`sub-workflow`	Workflow		*1..n*	A list of component nested workflows
`block`	ExecutionBlock		*1..n*	A list of component ExecutionBlocks
`meta-info`	String		*1..n*	A list of extra information (i.e. author, score)
`reference`	String		*1..n*	Bibliographic reference(s) about the workflow

Table 6
The slots for **ExecutionBlock** metaclass in DPS2E ontology.

Slot name	Type	Inverse	Cardinality	Comment
`name`	String		*1*	ExecutionBlock's name
`description`	String		*1*	A brief explanation
`implements`	Tool		*1*	The Tool implemented by the ExecutionBlock
`links_to`	ExecutionBlock		*1..n*	A list of linked ExecutionBlocks
`input`	Data_Content		*1..n*	A list of input Data_Content
`output`	Data_Content		*1..n*	A list of output Data_Content

an application scenario, it must be instanced (or customized) for that specific domain. This architecture, in fact, has been created to support cognitive system designers who want to analyse, configure and exploit together their strategies, tools and/or processes in a semi-automatic system for analytical pipelines. In this Section, we present the main features of a customization of the proposed architecture for a bioinformatics application scenario, i.e. the identification of protein interaction sub-network markers that are correlated with cancer metastasis.

5.1. Bioinformatics scenario

One of the most interesting challenge in cancer research is the identification of protein interaction sub-network markers that are correlated with cancer metastasis. The study of angiogenesis and metastasis development among cancer patients, usually takes into account only clinical and pathological risk factors (i.e. patient age, tumour size and so on). Anyway, it has been demonstrated that these factors can be considered a secondary manifestations rather than primary mechanisms of disease (Wang et al., 2005; van 't Veer et al., 2002). For this reason, a main challenge in bioinformatics is the prediction of the risk of metastasis, by means of the identification of a new prognostic markers that are more directly related to disease (Chuang, Lee, Liu, Lee, & Ideker, 2007).

In the last years, bioinformatics researchers produced several works in this field, proposing different strategies and heuristics that could help the prediction of tumour markers. A few authors, such as (Chuang et al., 2007; Nibbe, Koyutürk, & Chance, 2010; Dao et al., 2011), take into account both protein–protein interaction (PPI) sub-networks (protein complexes) and gene expression profiles that demonstrate a differential expression with respect to carcinogenesis phenotype; in facts, each protein complex is suggestive of a distinct functional pathway, that can provide novel hypotheses in organisms analysis (Sharan, Ulitsky, & Shamir, 2007).

As regarding our methodology, we suppose to collect as many information as possible about the problem of the identification of tumour markers. In addition, we suppose a domain expert, in conjunction with a knowledge engineer, is able to identify some key assumptions that can be used to define entities and their relationships, as described into the DPS2E ontology. This process can be done by means of some specific "extraction tables", such as that showed in Table 7.

Once all the available knowledge and the potentially useful resources (i.e. data frame, tools, web services and so on) of the problem are arranged or linked into the DPS2E ontology schema, it is possible to complete the implementation of the hybrid architecture, by adding a reasoning engine and producing suitable rules as discussed in Section 4.2.

In particular, for the proposed scenario, we build a small, but for our demonstration exhaustive, ontology considering state-of-the-art literature about this topic. In the following Section is reported the DPS2E ontology instance.

5.2. Ontology instance

In this scenario, as a start point we consider the "identification of protein networks for disease classification", that, according to DPS2E ontology, represents the **Problem** concept. As regards the data input we consider a list of protein–protein interactions and as data output a list of marked protein network, that could be responsible for some specific diseases. According to the related

Table 7
Example of extraction table. These tables aims at supporting domain expert and/or knowledge engineer in designing structured knowledge. In facts, finding the assumptions that resolves these queries (e.g.. "In order to resolve a goal of interest that operate on a specific data, which method(s), implementing a proper strategy(ies), could be used?"), it is possible to create ontology entities and relationships.

Goal of interest (*Task*)	**Operate on** (*Data_Type*)	**Strategy** (*Solver*)	**Method** (*Tool*)
Complex Clustering	PPIN	Local Search	MCODE
Complex Clustering	PPIN	Flow Simulation	MCL
PPIN Filtering	PPIN/PPI	Remove FP	Betweenness Centrality
PPIN Filtering	PPIN/PPI	Add FN	Detect Defective Cliques
Marker Identification	PPIN	Integrative-omics	Overlaying + Significance Eval.

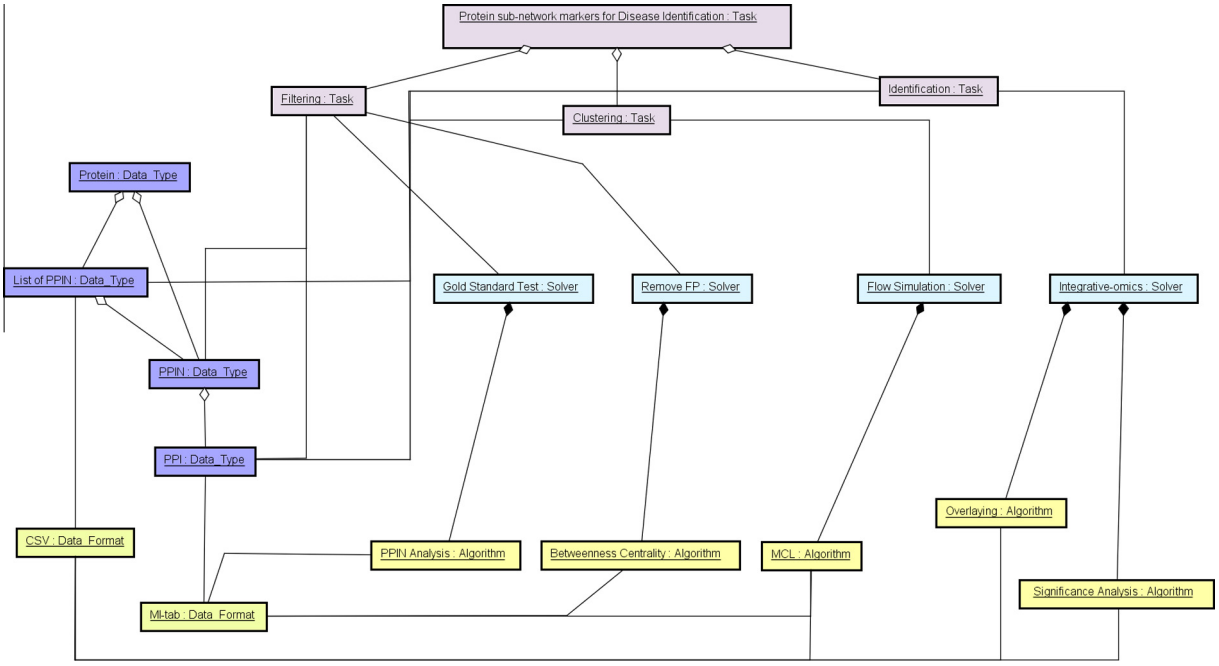

Fig. 10. A part of the Knowledge Area of the DPS2E ontology instance for the problem of the "identification of protein networks for disease classification".

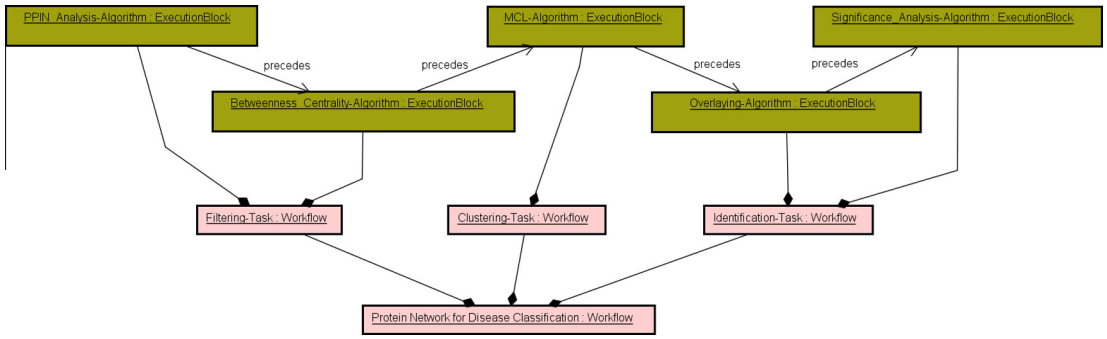

Fig. 11. A part of the Execution Area of the DPS2E ontology instance for the problem of the "identification of protein networks for disease classification". The "precedes" relationship gives information about the execution timeline.

literature, this problem could be arranged in three main tasks: filtering, clustering and identification. For instance, the first task has been handled by some authors (Ucar, Parthasarathy, Asur, & Wang, 2005) with a topological approach; in facts, they developed some graph-based algorithms in order to eliminate redundant false positive interactions from the original PPI dataset. This preprocessing strategy points to increase the reliability of PPI-Network. As regarding the second task, i.e. finding meaningful groups of biological units, a number of approaches have been proposed and a lot of them are based on clustering (Chua, Ning, Sung, Leong, & Wong, 2008; Gao & Sun, 2009). A well-know algorithm is Markov Clustering Algorithm (MCL) (Enright, Van Dongen, & Ouzounis, 2002), that divides the graph by means of "flow simulation paradigm". In facts, it separates the graph into different segments, with an iteration of simulated random walks within a graph. Once sub-networks are obtained, it is possible to identify those complexes that demonstrate a differential expression with respect to carcinogenesis phenotype, by means of an integrative-omics approach proposed in Nibbe et al. (2010).

Using these elements, we could obtain some putative disease protein sub-networks. Ultimately, in order to face with this case of study, we populated the KB with the knowledge about three main tasks (filtering, clustering, identification), three different solver approaches (statistical, topological, integrative-omics) and five tools (both algorithms and applications).

For clarity reason, the problem ontology has been decomposed in two figures: Fig. 10 shows a part of the Knowledge Area, Fig. 11 contains a part of the Execution Area.

For lack of space, Fig. 10 contains only a small part of the structured knowledge related to the current scenario, showing only a sub-set of elements, that represent a possible solution for the problem. Obviously, starting from this knowledge organization, it is possible to increase the knowledge base by adding other tools implementing already identified solvers and/or new solvers coding different strategies/approaches that solve the three main tasks.

The knowledge related to the **Task**, **Solver** and **Tool** is used in order to perform the decision making activity, respectively at the three abstraction layers (problem, solution, object) as shown in Fig. 6. This knowledge will be organized in modules, as explained in Section 3.3. For example, in the proposed scenario it is possible to define three modules, namely *Filtering*, *Clustering* and *Identification*, that contain the expertise needed to perform the decision making activity for the choice of the proper solving strategy.

As regards the DPS2E Execution Area for the current scenario, the Fig. 11 reports a set of execution blocks used for obtaining a concrete workflow that implements the case study. This concrete

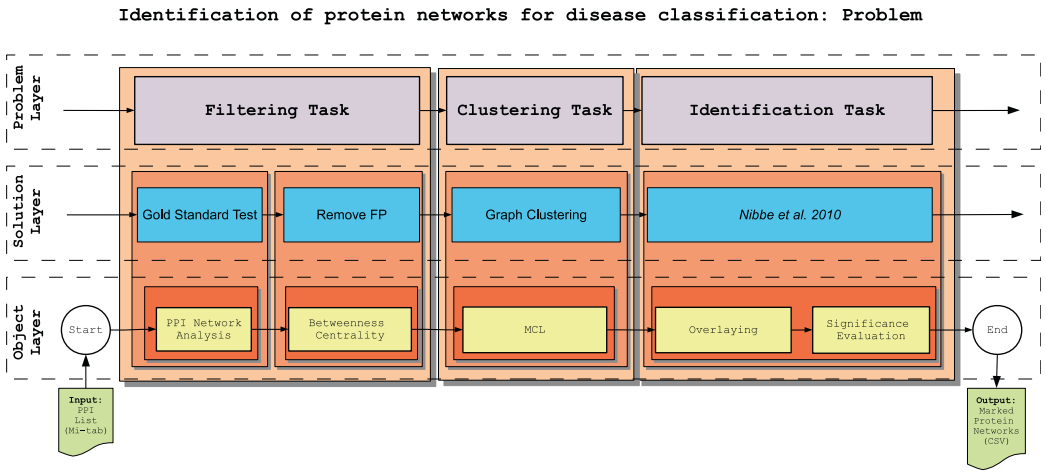

Fig. 12. A possible workflow for the current scenario. The layout shows the workflow in terms of Tasks (purple boxes), Solvers (blue boxes) and tools (yellow boxes). Each component of this workflow is the result of the decision-making activity, at different abstraction layers, of the proposed hybrid architecture for expert systems. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

workflow represents a possible solution for the case study, just using the knowledge available into the showed DPS2E Knowledge Area. In Fig. 11, it is important to notice the execution blocks layout: in facts it has been arranged according to the "precedes" relationship, in order to show the execution timeline. Obviously, each execution block contain information about (eventually formatted) input file(s), executed tool(s), parameter(s) and output file(s).

Following the three different approaches (procedural, declarative and workflow-oriented one) integrated into the described cognitive space (see Section 3) and the reasoning mechanism introduced in Section 3.1, it is possible to support an experimentalist in his own activity, from the design of an abstract workflow to the implementation of a concrete workflow. The resulting workflow can, eventually, be run on the most recent WFMS, like Taverna, whenever the adopted tools are available.

Another point of view of the final workflow for the "identification of protein networks for disease classification" problem is shown in Fig. 12. The workflow layout respects the color pattern of DPS2E ontology: tasks are represented with orange rectangles, solvers with blue rectangles and tools with yellow rectangles and each component of this workflow is the result of the decision-making activity, at different abstraction layers, of the proposed hybrid architecture. In particular, the Filtering task consists of the concatenation of two different strategies, implemented by two different algorithms; whereas the last task, i.e. Identification, is composed of one strategy consisting of the sequential execution of two algorithms. Of course this is not the only workflow resolving the problem: the possible alternative tools and strategies (solvers) depend on the richness of the KB, organized according to DPS2E ontology.

6. Conclusion

Nowadays, many experimentalists require intelligent systems that can support them for laboratory experiments. In this work, we proposed a novel hybrid architecture for designing expert systems able to support experiment management. The proposed architecture is defined in a cognitive space composed by three different approaches: procedural, declarative and workflow-oriented. Following the introduced reasoning mechanism over them, it is possible to support an experimentalist in his own activity, from the composition of an abstract workflow to the implementation of a concrete workflow.

The proposed hybrid architecture is supported by an ontological paradigm, called Data-Problem–Solution-to-Experiment (DPS2E), that allows to organize the knowledge of an expert system following the guidelines summarized by the architecture.

The proposed architecture have been used for design a prototype system in bioinformatics field. Modelling the knowledge related to that scenario according to the DPS2E ontology, the decision-making component of the architecture, implemented by means of a set of production rules and an inference engine, is able to produce a scientific workflow, as a combination of the instances (facts) contained into the knowledge base, or reusing some pieces of sub-workflows previously developed. The resulting workflow can, eventually, be run on the most recent WFMS, like Taverna, whenever the adopted tools are available.

References

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., & Mock, S. (2004). Kepler: An extensible system for design and execution of scientific workflows. *2004 Proceedings 16th International Conference on Scientific and Statistical Database Management* (pp. 423–424). IEEE. http://dx.doi.org/10.1109/SSDM.2004.1311241.

Baker, D., Bridges, D., Hunter, R., Johnson, G., Krupa, J., Murphy, J., & Sorenson, K. (2002). Guidebook to decision making methods. Technical Report. Department of Energy, USA.

Chandrasekaran, B., Josephson, J., & Benjamins, V. (1999). What are ontologies, and why do we need them? *Intelligent Systems and Their Applications. IEEE.* .

Chen, R. C., Huang, Y. H., Bau, C. T., & Chen, S. M. (2012). A recommendation system based on domain ontology and SWRL for anti-diabetic drugs selection. *Expert Systems with Applications, 39*, 3995–4006. http://dx.doi.org/10.1016/j.eswa.2011.09.061.

Cheung, C., Cheung, C., & Kwok, S. (2011). A knowledge-based customization system for supply chain integration. *Expert Systems with Applications, 39*, 3906–3924. http://dx.doi.org/10.1016/j.eswa.2011.08.096.

Chua, H. N., Ning, K., Sung, W. K., Leong, H. W., & Wong, L. (2008). Using indirect protein-protein interactions for protein complex prediction. *Journal of Bioinformatics and Computational Biology, 6*, 435–466.

Chuang, H. Y., Lee, E., Liu, Y. T., Lee, D., & Ideker, T. (2007). Network-based classification of breast cancer metastasis. *Molecular Systems Biology, 3*, 140. http://dx.doi.org/10.1038/msb4100180.

Chung, P., Cheung, L., Stader, J., Jarvis, P., Moore, J., & Macintosh, A. (2003). Knowledge-based process management – An approach to handling adaptive workflow. *Knowledge-Based Systems, 16*, 149–160. http://dx.doi.org/10.1016/S0950-7051(02)00080-1.

Dao, P., Wang, K., Collins, C., Ester, M., Lapuk, A., & Sahinalp, S. C. (2011). Optimally discriminative subnetwork markers predict response to chemotherapy. *Bioinformatics, 27*, 205–213. http://dx.doi.org/10.1093/bioinformatics/btr245.

De Maio, C., Fenza, G., Gaeta, M., Loia, V., & Orciuoli, F. (2011). A knowledge-based framework for emergency DSS. *Knowledge-Based Systems, 24*, 1372–1379. http://dx.doi.org/10.1016/j.knosys.2011.06.011.

Enright, A. J., Van Dongen, S., & Ouzounis, C. A. (2002). An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research, 30*, 1575–1584.

Fiannaca, A., Gaglio, S., La Rosa, M., Rizzo, R., & Urso, A. (2012a). An intelligent system for building bioinformatics workflows. In *6th International Conference on*

Complex, Intelligent, and Software Intensive Systems (CISIS) (pp. 212–218). IEEE. http://dx.doi.org/10.1109/CISIS.2012.141.

Fiannaca, A., La Rosa, M., Gaglio, S., Rizzo, R., & Urso, A. (2012b). An ontological-based knowledge organization for bioinformatics workflow management system. *EMBnet Journal, 18*, 110–112.

Fiannaca, A., La Rosa, M., Rizzo, R., Urso, A., & Gaglio, S. (2012c). An ontology design methodology for knowledge-based systems with application to bioinformatics. In *2012 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)* (pp. 85–91). IEEE. http://dx.doi.org/10.1109/CIBCB.2012.6217215.

Fiannaca, A., La Rosa, M., Rizzo, R., Urso, A., & Gaglio, S. (2013). A knowledge-based decision support system in bioinformatics: An application to protein complex extraction. *BMC Bioinformatics, 14*, S5.

Gao, L., & Sun, P. (2009). Clustering algorithms for detecting functional modules in protein interaction networks. *Journal of Bioinformatics and Computational Biology, 7*, 217–242.

Hollingsworth, D. (1995). The workflow reference model. Technical Report 1. Workflow Management Coalition.

Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M. R., Li, P., et al. (2006). Taverna: A tool for building and running workflows of services. *Nucleic Acids Research, 34*, W729–W732. http://dx.doi.org/10.1093/nar/gkl320.

Johnson, B. & Shneiderman, B. (1991). Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proceedings of IEEE conference on visualization* (pp. 284–291).

Ko, R. K. L. (2009). A computer scientist's introductory guide to business process management (BPM). *Crossroad, 15*, 11–18. http://dx.doi.org/10.1145/1558897.1558901.

Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., et al. (2002). UML for ontology development. *The Knowledge Engineering Review*, 17. http://dx.doi.org/10.1017/S0269888902000358.

Lee, E. A. & Parks, T. M., (1995). Dataflow process networks. http://dx.doi.org/10.1109/5.381846.

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., et al. (2006). Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience, 18*, 1039–1065. http://dx.doi.org/10.1002/cpe.994.

Ludascher, B., Altintas, I., & Gupta, A. (2003). Compiling abstract scientific workflows into Web service workflows. *15th International Conference on Scientific and Statistical Database Management* (pp. 251–254). IEEE Comput. Soc.. http://dx.doi.org/10.1109/SSDM.2003.1214990.

Moreno, M., & Kearney, P. (2002). Integrating AI planning techniques with workflow management system. *Knowledge-Based Systems, 15*, 285–291. http://dx.doi.org/10.1016/S0950-7051(01)00167-8.

Mulyar, N., Pesic, M., Aalst, W. M. P. V. D., Peleg, M., & Carmel, M. (2008). Declarative and procedural approaches for modelling clinical guidelines: Addressing flexibility issues. *Knowledge Creation Diffusion Utilization, 4928*, 335–346.

Nibbe, R. K., Koyutürk, M., & Chance, M. R. (2010). An integrative-omics approach to identify functional sub-networks in human colorectal cancer. *PLoS Computational Biology, 6*, 15.

Noy, N., Sintek, M., Decker, S., Crubezy, M., Fergerson, R., & Musen, M. (2001). Creating semantic web contents with Protege-2000. *IEEE Intelligent Systems, 16*, 60–71. http://dx.doi.org/10.1109/5254.920601.

Ogasawara, E., Paulino, C., Murta, L., Werner, C., & Mattoso, M. (2009). Experiment line: Software reuse in scientific workflows. In M. Winslett (Ed.), *Scientific and Statistical Database Management* (pp. 264–272). Berlin, Heidelberg: Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-02279-1.

Sharan, R., Ulitsky, I., & Shamir, R. (2007). Network-based prediction of protein function. *Molecular Systems Biology, 3*, 88.

Shue, L., Chen, C., & Shiue, W. (2009). The development of an ontology-based expert system for corporate financial rating. *Expert Systems with Applications, 36*, 2130–2142. http://dx.doi.org/10.1016/j.eswa.2007.12.044.

Smelik, R. M., Tutenel, T., De Kraker, K. J., & Bidarra, R. (2011). A declarative approach to procedural modeling of virtual worlds. *Computers & Graphics, 35*, 352–363. http://dx.doi.org/10.1016/j.cag.2010.11.011.

Ten Berge, T., & VanHezewijk, R. (1999). Procedural and declarative knowledge: An evolutionary perspective. *Theory Psychology, 9*, 605–624. http://dx.doi.org/10.1177/0959354399095002.

Thurner, V. (1998). A formally founded description technique for business processes. *Proceedings International Symposium on Software Engineering for Parallel and Distributed Systems* (pp. 254–261). IEEE Comput. Soc.. http://dx.doi.org/10.1109/PDSE.1998.668193.

Ucar, D., Parthasarathy, S., Asur, S., & Wang, C. E. C. (2005). Effective pre-processing strategies for functional clustering of a protein-protein interactions network. *5th IEEE Symposium on Bioinformatics and Bioengineering BIBE05* (pp. 129–136). IEEE. http://dx.doi.org/10.1109/BIBE.2005.25.

van 't Veer, L. J., Dai, H., van de Vijver, M. J., He, Y. D., Hart, A. A. M., Mao, M., et al. (2002). Gene expression profiling predicts clinical outcome of breast cancer. *Nature, 415*, 530–536. http://dx.doi.org/10.1038/415530a.

Wang, Y., Klijn, J. G. M., Zhang, Y., Sieuwerts, A. M., Look, M. P., Yang, F., et al. (2005). Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *Lancet, 365*, 671–679. http://dx.doi.org/10.1016/S0140-6736(05)17947-1.

Zarri, G. P. (2011). Knowledge representation and inference techniques to improve the management of gas and oil facilities. *Knowledge-Based Systems, 24*, 989–1003. http://dx.doi.org/10.1016/j.knosys.2011.04.010.