Scalable Computing: Practice and Experience
Volume 13, Number 1, pp. 73–83. http://www.scpe.org
ISSN 1895-1767
© 2012 SCPE

MULTI-AGENT ARCHITECTURE IN SEMANTIC SERVICES ENVIRONMENT

CRISTINA MINDRUTA[*], VICTOR ION MUNTEANU[†], VIOREL NEGRU[‡] AND CALIN SANDRU[§]

Abstract. A semantic enabled multi-agent architecture for solving non-linear equations systems by using a service oriented approach is proposed. The service oriented approach allows us to access already implemented methods for solving complex mathematical problems. The semantic descriptions of these services provide support for intelligent agents. The proposed architecture is a framework with two extension areas, agent society and domain ontology, with possible application in other domains too.

Key words: multi-agent system, non-linear equations systems, semantic services

1. Introduction. The main goal of this paper is to propose a multi-agent architecture for solving non-linear equations systems. This architecture is designed around a semantic-based solving paradigm supported by an ontology of service descriptions. That allows us to define a multi-agent expert system in a semantic services environment.

Similar work has been done in the MONET project [3] which had the aim to provide a set of web services together with a brokering platform in order to facilitate means of solving a particular mathematical problem. The semantic representation for the mathematical objects was done using OpenMath [20] (MathML [22] was cited also).

GENSS (Grid-Enabled Numerical and Symbolic Services) project [1], like MONET, tries to combine grid computing and mathematical web services using a common open agent-based framework.

In [12] is discussed the matchmaking of semantic mathematical services described using OpenMath.

The architecture we propose is being built based on past experience in designing NESS, a non-linear equations systems solver, and EpODE, an expert system dedicated to ordinary differential equations. NESS [15] is an intelligent front-end for solving non-linear equation systems, developed in CLIPS. Starting from the features of the system to be solved and of the numerical methods, human expert uses domain knowledge (numerical analysis) and heuristics to choose the most suitable method, to interpret the results (intermediary and final), and to restart the solving process in the case of failure. NESS uses task oriented reasoning. A MAS architecture based on UPML has been proposed and instantiated for NESS [18]. EpODE was initially realized as a monolithic expert system [17] and has been re-engineered as a semantic services oriented framework [14]; the solving methodology is workflow-oriented, being realized by integrating semantic services with process modeling.

While having similar main functional objective with NESS, the architecture proposed in this paper is more flexible due to the semantic services component and to the new society of agents designed accordingly.

We have designed a multi-agent architecture that will implement a task-oriented solving model, with a core semantic-based solving paradigm. Typically, multi-agent architecture offers flexibility, scalability and mobility, important quality attributes when dealing with a large number of software services. The agents in the architecture have capabilities ranging from semantically searching for services to providing an execution plan for the given problem. The problem that is to be solved is passed to the multi-agent system as input data. The expert agents in the system analyse the problem and propose an execution plan in order to find the solution. The execution plan is ran and constantly monitored (adjustments are made if needed), and the result of the execution is returned to the user.

The execution plan contains numerical methods which are offered by software services.

We have also designed a semantic services ontology in order to support the semantic-based solving paradigm. For semantic descriptions we have decided to use WSMO (Web Services Modelling Ontology) [2]. In their work, Sorathia et al. [19] analyse several ontologies which were used as an approach to service annotation for discovery, selection, composition, execution and monitoring. Although addressing mainly the challenge of

[*]Department of Computer Science, Faculty of Mathematics and Informatics, West University of Timişoara, Timişoara, Romania (cmindruta@info.uvt.ro).
[†]Department of Computer Science, Faculty of Mathematics and Informatics, West University of Timişoara, Timişoara, Romania (vmunteanu@info.uvt.ro).
[‡]Department of Computer Science, Faculty of Mathematics and Informatics, West University of Timişoara, Timişoara, Romania (vnegru@info.uvt.ro).
[§]Department of Computer Science, Faculty of Mathematics and Informatics, West University of Timişoara, Timişoara, Romania (csandru@info.uvt.ro).

semantic interoperability between relevant service ontologies, we may identify, in this comprehensive literature review providing insight in the state-of-the-art in services ontologies, that WSMO and WSAF [13] are relevant computational ontologies. WSAF is focused on agent mediation and supports dynamic service selection based on QoS (non-functional properties). On the other hand, WSMO is focused on service mediation based both on functional and non-functional properties, allowing more flexibility.

Our approach considers a semantic services context, which is able to offer semantic information useful to the system of agents. In this context, the proposed multi-agent system uses a specific ontology containing concepts, relations and axioms defined for the non-linear equations systems domain, and has an extensible database of semantic descriptions for services implementing numerical methods.

The system implements a core paradigm for solving problems, based on semantic matching between problem properties and numerical method capabilities. Numerical methods are identified based on their semantic descriptions that reflect the properties of the problem for which the method is appropriate. The method selection can be realized by the user based on his own expertise, by the user based on system recommendation and estimations, or automatically by the multi-agent system.

This core paradigm is included in a more flexible approach to solve the problems, that implies coordinated activity in the society of agents and with the user. This can result for example in starting to solve a problem with a numerical method and, from a given step, to continue with another numerical method, based on intermediary results and performance of the system.

Such a flexibility is provided by the proposed multi-agent architecture and covers a large area of user skills, from users with simple mathematical skills, for which the system could provide a solution based on its own expertise, to users with very high mathematical skills, which want to experiment solving new types of problems and using new numerical methods. The experience gained by the later category of users is also captured by the multi-agent system in new methods and new characteristics of the existing ones, thus improving its capabilities.

The paper is structured as follows. Motivation and example use cases are covered in section 2. The conceptual model is covered in section 3. Section 4 is dedicated to the semantic descriptions of the ontology, services and goals used to support the core solving paradigm. Section 5 presents the proposed multi-agent architecture, based on the task-oriented model and integrated with the semantic infrastructure. Conclusions and future work are discussed in section 6.

2. Motivation and use cases. Service Oriented Computing is an emerging computing paradigm in the context of distributed computing. It is already largely accepted and implemented. For example, Cloud Computing is one of the most relevant chapter in service oriented computing. In this context a lot of services in different application domains are available but they are still under-exploited. One of the important challenges is the automation of discovery and composition of services, supported by semantic technologies. The work presented here focuses on the modelling and exploiting the domain of non-linear equation systems, but the presented architecture has flexibility points allowing for its extension to other domains.

We outline here a multi-agent architecture in the context of semantic services intended to provide an expert system to mathematicians. Assuming that different methods for solving non-linear equation systems are implemented in services, the proposed system assists the user in using them and exploring new solving methods.

The following examples of possible use cases for non-linear equations systems solver are gradually more comprehensive.

2.1. Use Case 1 - Solve automatically. The user will provide the non-linear system problem and the system will present the solution. The user can also provide a time limit for finding the solution or the system will use a default value. If the system is not able to find a solution in the imposed amount of time, the user will be informed. The user can establish a new time limit.

2.2. Use Case 2 - Solve under user control. The user will provide the non-linear system problem and the set of session restrictions (time limit, error level, a.s.o.). He can select a numerical method for solving the equations system or can let our system to select one based on its expertise. The user can control the intermediary results and may interrupt the execution to select another numerical method to continue the solving process and/or to modify session restrictions.

2.3. Use Case 3 - Support research. The researcher can add new numerical methods for solving non-linear equations systems. These numerical methods must have been implemented as software services and the researcher adds to the system their semantic descriptions. The new added numerical methods can be then explored inside the previous use cases.

3. Conceptual model.

3.1. Core solving paradigm. The system implements a core paradigm for solving problems, based on semantic matching between problem properties and numerical method capabilities.

The *domain ontology* describes the following domain:

- *Problem*: A problem is defined by its input data and may get an unique identifier for future recognition in the system. A problem is characterized by a set of properties derived from its input data.
- *Method*: Problems can be solved with methods (numerical methods). A problem can be solved with a numerical method or with a composition of numerical methods. The composition of numerical methods can be predefined or can be created in collaboration with the user. A method is uniquely identified. Besides the problem input data, a method has a specific set of input data. A method is characterized by a set of properties.
- *Session*: A problem is solved under a set of restrictions that define the solving session.
- *Solution*: A solution to a problem is obtained invoking software services.
- *Matching rules*: The expert system recommends numerical methods based on a set of matching rules. The matching rules take into account problem properties and method capabilities.

The *solving paradigm* is focused on matching semantic descriptions, and has the following phases:

- Compute problem properties.
- Identify numerical methods by matching the problem properties with methods capabilities.
- Select numerical method.
- Apply method.

3.2. Task oriented reasoning. Although sometimes associated with an activity to execute, the concept of task is mostly intended to abstract a specific goal to be achieved [8, 2, 5]. In this regard, tasks definitions do not explicit the particular method to use in order to achieve the goal, but rather give a description of the state of the world to be achieved.

The operational aspect can be abstracted in the concept of a problem solving method (PSM). In their analysis on research on using PSMs in developing Semantic Web applications and based on their practical experience with developing a software using PSMs as conceptual building blocks, the authors in [16] conclude that the fundamental PSM construction techniques used when developing new applications are sufficiently powerful and flexible to build very complex systems.

A PSM describes how to achieve a result based on a set of input data. The goal of a PSM can be regarded as a procedural one in order to obtain a result according to the method specification. The meta-properties of the methods to be mentioned in this context include input, output, precondition, postcondition, sub-task.

The task oriented model is the abstract methodological basis for the proposed non-linear equations system solver, task oriented reasoning being a natural approach for our multi-agent system. Starting from a particular task, one can build a hierarchy of tasks and PSMs that can be considered as the plan for solving the root task. Figure 3.1 represents how may the task-oriented model be applied in the non-linear equations systems solver.

4. Semantic model.

For semantic representations we have adopted Web Services Modelling Ontology. It is a natural approach taking into consideration the fact that, according to [6], WSMO provides a formal ontology for describing Web services based upon WSMF and, in addition, embodies a number of principles that are derived either from the UPML framework, and therefore PSMs research in general, or from the principles underlying the Web and service-oriented computing.

WSMO is a conceptual model that provides ontological specifications for the core elements of semantic Web services. WSMO defines four basic types of elements: ontologies, goals, services and mediators. Ontologies are used to describe application domain ontologies, i.e. entities, relations and constraints in the problem domain. Services are described by non-functional properties, by functionality defined with capabilities, and by behavior defined with interfaces that describe two perspectives: communication and collaboration. Goals have descriptions similar to services, but describe the service requester point of view. Mediators describe mediation entities used to overcome structural, semantic, conceptual disparities.

WSMO is appropriate for modelling the proposed solving paradigm, because it offers a clear separation between goals and services. Services offer numerical methods or compositions of methods from our paradigm,

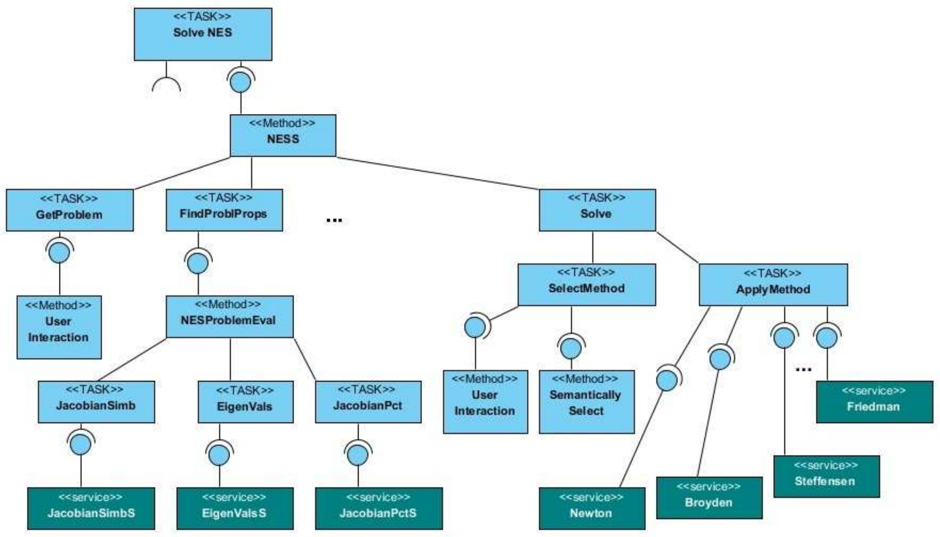


Fig. 3.1: Task-oriented model applied to NESS

and goals are dynamically built for each problem to be solved.

We have used Web Service Modeling Toolkit [9] as support for the modelling activities.

4.1. Knowledge representation. Ontologies. In [14], starting from an existing expert application for solving ordinary differential equation systems, an architecture of services and workflows in a semantic environment have been designed. This was supported by a base ontology for non-linear problems `NonLPOnto`, and a specialization for ODE (ordinary differential equations) systems.

We benefit now from the extensibility of this base ontology to model the non-linear equations systems expert semantics as another extension of this ontology.

`NonLPOnto` represents a pattern for ontologies specific to different non-linear problem categories. One of these categories is represented by the non-linear equations systems (NES). As with ODE systems, the main concepts defined in `NESOnto` are specializations of concepts defined in `NonLPOnto`(figure 4.1).

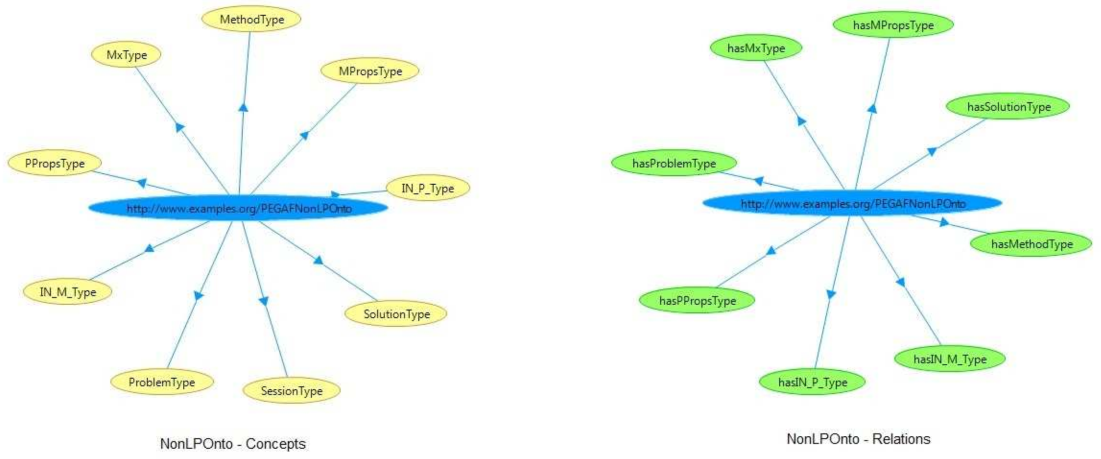


Fig. 4.1: `NonLPOnto` represented with WSMT

`NonLPOnto` is designed as a high level WSMO ontology for modeling mathematical services for solving non-linear problems. According to this model a problem has an input data set and a set of properties. An associated matrix is build based on the input data set, and the characteristics of this matrix are used to specify properties of the problem. A problem is solved in the context of a session that defines specific requirements for computation and result. Solving a problem implies applying a numerical method. A numerical method has also a specific input set and is characterized by a set of properties.

Refining things, `NESOnto` (figure 4.2) contains concepts, relations and axioms that define problem and solution spaces of the non-linear equations systems.

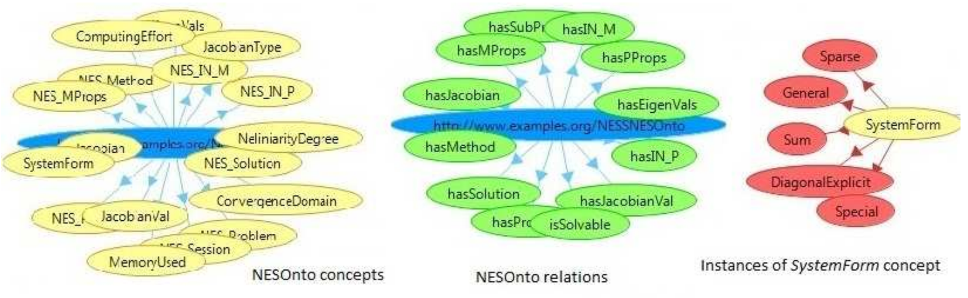


Fig. 4.2: Elements of `NESOnto` represented with WSMT

`NESOnto` defines the concept of problem (`NES_Problem`) in relation to the concepts representing the input data of the problem (`NES_IN_P`) and the properties of the corresponding non-linear equations system (`NES_PProps`).

It also defines the concept of numerical method (`NES_Method`) in relation to the concepts representing the input data of the numerical method (`NES_IN_M`) and the properties of the numerical method (`NES_MProps`). In figure 4.3, the concept `NES_MProps` is represented with its attributes and the relation `hasPProps` between the `NES_Method` and its properties.

The restrictions imposed on the solving session are modelled with the concept `NES_Session` and the solution of the non-linear equations system is modelled with the concept `NES_Solution`.

The associated matrix is modelled with two concepts: `Jacobian` represents the symbolic Jacobian of the system, and `JacobianVal` represents the Jacobian matrix of the system computed in a given point.

The properties of the non-linear equations system are of types defined in specific concepts (ex. `SystemForm`), and for each of these concepts the particular instances (ex. `General`, `Sparse`, `DiagonalExplicit`) have been defined.

The properties of the numerical method have been defined in the same manner. They will be used by the expert agent, that will refine the service selection and composition matching them to the execution constraints represented as an instance of the `NES_Session` concept.

One key element of `NESOnto` is the relation *isSolvable* applied to the problem properties. The relation is used in the matching process between the problem characterized by its set of properties and a numerical method which could solve the specific problem. An instance of this relation is created when a semantically described service can solve the problem.

4.2. Services implementing numerical methods. The capabilities of each service are described using `NESOnto` and an ontology specific to the service that contains one or more axioms. One of the axioms in this specific ontology defines the relation *isSolvable* by expressing the properties of the non-linear equations systems for which the numerical method is appropriate.

In figure 4.4(a) is represented the semantic description of `Broyden_NES` service that implements the numerical method Broyden for solving non-linear equations systems. The precondition in the semantic description states that the method can be used if the relation *isSolvable* exists for the properties of the problem to be solved. The semantics of this relation for the service `Broyden_NES` are defined in the axiom *isSolvableDef* of the ontology particular to the capabilities of this service, and expresses the fact that Broyden method is recommended for non-linear equations systems of general form, with non-singular Jacobian, and of medium (between 10 and 50

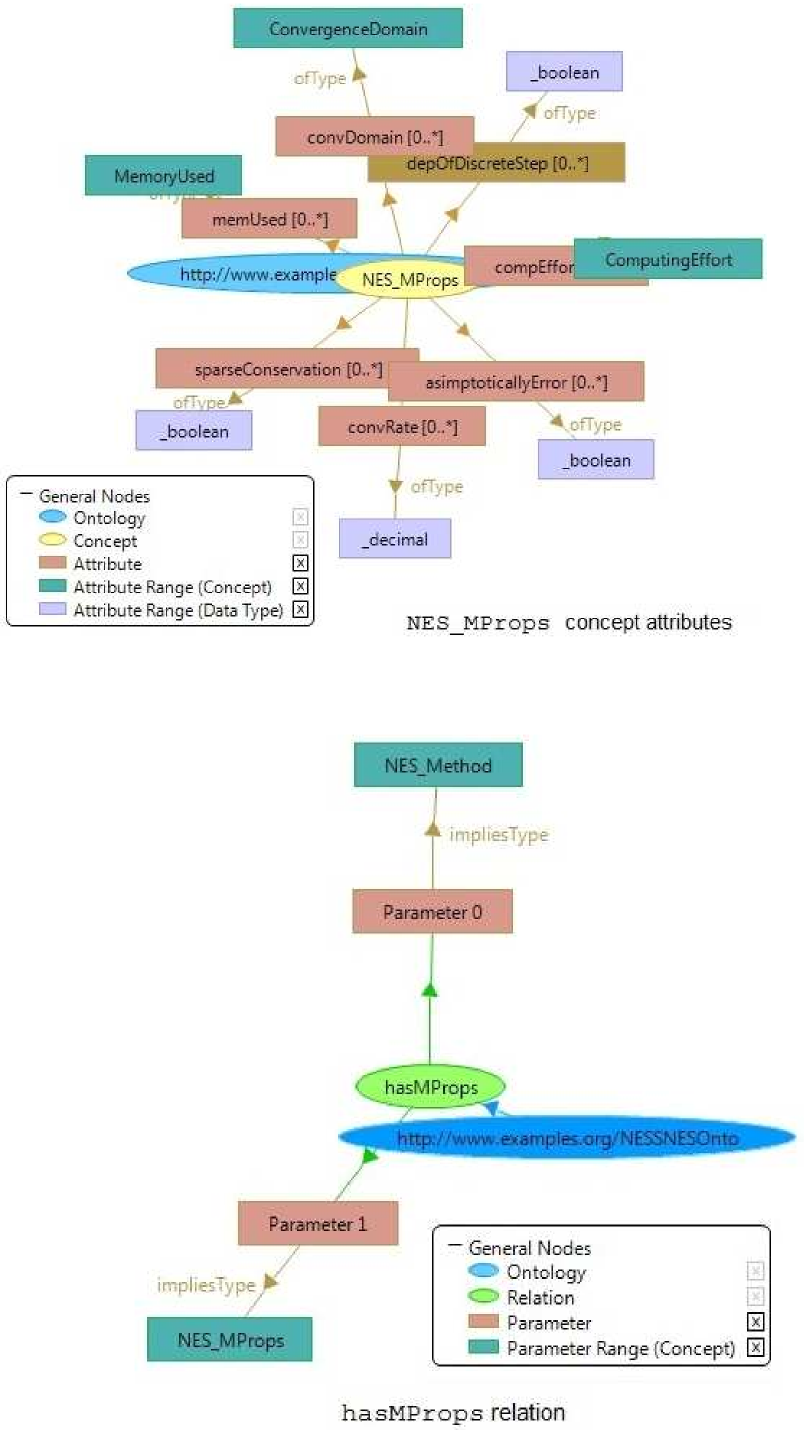
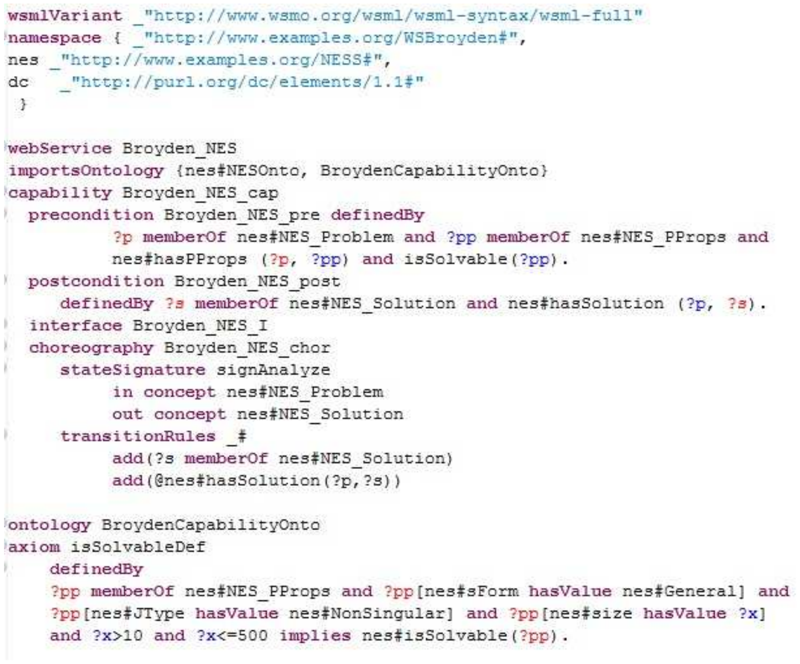


Fig. 4.3: Method and its properties represented with WSMT

equations) or big size (between 50 and 500 equations). This represents a part of the expert knowledge and is implemented in the semantic description of the service.

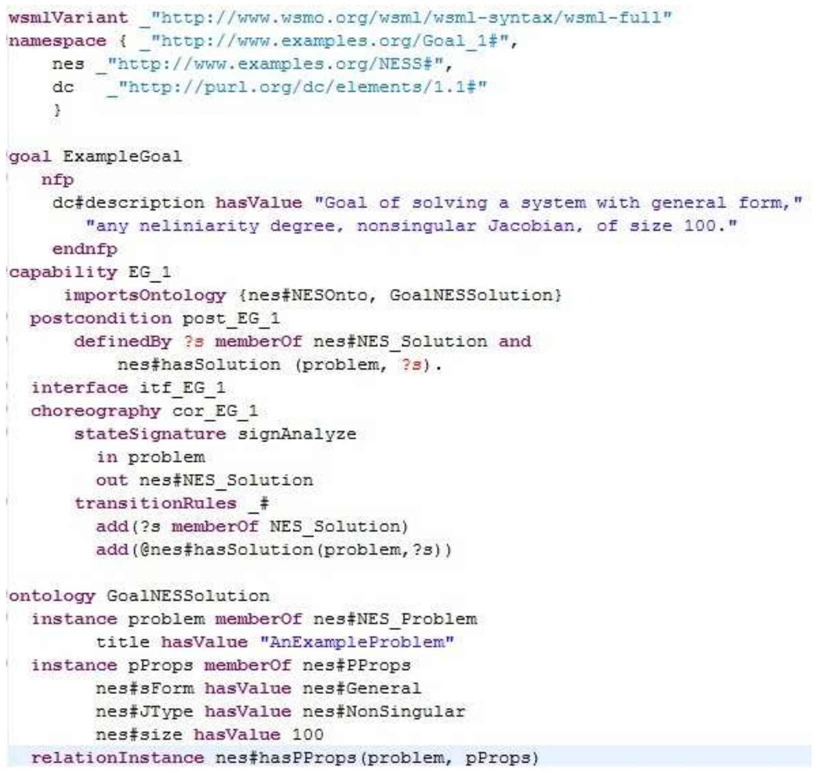


```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-full"
namespace { _"http://www.examples.org/WSBroyden#",
nes _"http://www.examples.org/NESS#",
dc   _"http://purl.org/dc/elements/1.1#"
 }

webService Broyden_NES
importsOntology {nes#NESOnto, BroydenCapabilityOnto}
capability Broyden_NES_cap
  precondition Broyden_NES_pre definedBy
          ?p memberOf nes#NES_Problem and ?pp memberOf nes#NES_PProps and
          nes#hasPProps (?p, ?pp) and isSolvable(?pp).
  postcondition Broyden_NES_post
     definedBy ?s memberOf nes#NES_Solution and nes#hasSolution (?p, ?s).
  interface Broyden_NES_I
  choreography Broyden_NES_chor
     stateSignature signAnalyze
          in concept nes#NES_Problem
          out concept nes#NES_Solution
     transitionRules _#
          add(?s memberOf nes#NES_Solution)
          add(@nes#hasSolution(?p,?s))

ontology BroydenCapabilityOnto
axiom isSolvableDef
    definedBy
    ?pp memberOf nes#NES_PProps and ?pp[nes#sForm hasValue nes#General] and
    ?pp[nes#JType hasValue nes#NonSingular] and ?pp[nes#size hasValue ?x]
    and ?x>10 and ?x<=500 implies nes#isSolvable(?pp).
```

(a)



```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-full"
namespace { _"http://www.examples.org/Goal_1#",
    nes _"http://www.examples.org/NESS#",
    dc   _"http://purl.org/dc/elements/1.1#"
    }

goal ExampleGoal
   nfp
    dc#description hasValue "Goal of solving a system with general form,"
       "any neliniarity degree, nonsingular Jacobian, of size 100."
    endnfp
capability EG_1
     importsOntology {nes#NESOnto, GoalNESSolution}
  postcondition post_EG_1
      definedBy ?s memberOf nes#NES_Solution and
          nes#hasSolution (problem, ?s).
  interface itf_EG_1
  choreography cor_EG_1
      stateSignature signAnalyze
        in problem
        out nes#NES_Solution
      transitionRules _#
        add(?s memberOf NES_Solution)
        add(@nes#hasSolution(problem,?s))

ontology GoalNESSolution
  instance problem memberOf nes#NES_Problem
        title hasValue "AnExampleProblem"
  instance pProps memberOf nes#PProps
        nes#sForm hasValue nes#General
        nes#JType hasValue nes#NonSingular
        nes#size hasValue 100
  relationInstance nes#hasPProps(problem, pProps)
```

(b)

Fig. 4.4: (a) WSML descriptions for BroydenNES service and (b) ExampleGoal goal

4.3. Goals. Goals are dynamically built for each problem. Each goal has its a particular ontology that contains instances of concepts from `NESOnto`. In order to identify the services which are able to solve the corresponding problem, the particular ontology (`GoalNESSolution`) contains an instance of the `NES_PProps` concept which holds the concrete properties of the given problem.

In figure 4.4(b), a goal of solving a non-linear equations systems is described in WSML.

5. Multi-agent architecture. Using a multi-agent system for service discovery and composition enables a decentralized approach to solving non-linear equation systems. This approach is further enhanced by an event driven model and concurrency offered by the agents.

When developing the multi-agent architecture, we had several architectural concerns in mind. The architecture must:

- Allow service operations: publishing, semantic facilitation, invoking (running) etc.
- Provide automatic semantic service selection and composition.
- Detect and recover from a failing service.
- Monitor services.
- Create solving scenarios based on previous solving experience.

In our architecture, the PSMs can be implemented as semantic services or as agents (figure 5.1).

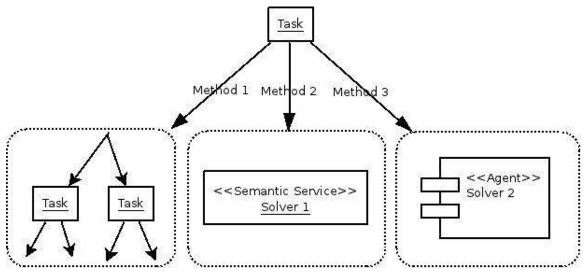


Fig. 5.1: Task structure

5.1. Agents. The multi-agent system is composed of the following agents: service, monitoring, execution, user interface, matchmaking, reasoning and historian. The proposed architecture is depicted in figure 5.2.

The user interface agent handles all communication with the users and provides users with proxy functionality for interacting with the system. User interface agent exposes a REST interface to which users can connect. It communicates with the reasoner, executioner and monitoring agents in order to manage the planning and execution.

The reasoner agent is in charge of creating task-oriented plans to solve the problems it receives from the human interface agent. It uses the domain ontology to create the semantic definitions of the tasks in terms of WSMO goals by processing the input data in order to detect problem properties. Furthermore, the historian agent is used to identify plans that where applied in similar problems. After the plan has been made, it is dispatched to one of the execution agents for processing. Similarly to the WSMX platform, the reasoner agent will use one of the following reasoners: IRIS[1], KAON2[2], PELLET[3] or MINS[4].

The execution agent handles the execution of the work plan. It will be in charge of service invocation and workflow management. The execution agent will query the matchmaking agent to retrieve information like endpoints for semantically compatible services for the current work plan. Service invocation is done through service agents. Also, synchronization with monitoring agents will be done in order to monitor current work plan progress. In case of execution problems, it will ask a reasoner agent to find alternatives/suggest a solution for it.

The monitoring agent has the role of monitoring the current task execution. It will synchronize with execution agents for work plan information. It will also send periodic updates to the user interface agents

[1]http://www.iris-reasoner.org/
[2]http://kaon2.semanticweb.org/
[3]http://clarkparsia.com/pellet
[4]http://tools.sti-innsbruck.at/mins/

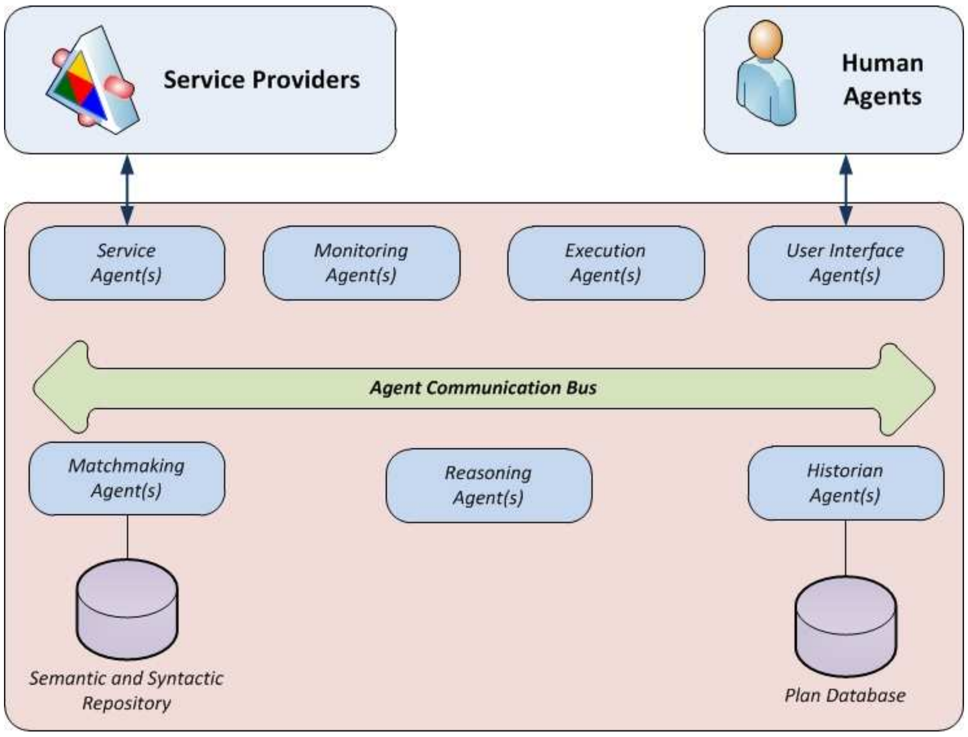


Fig. 5.2: Multi-agent system architecture

so they can relay task progress to the user. Monitoring agents also have the capability of creating execution profiles that they will end to historian agents. These profiles cover the running work plan along with the service decisions made and any errors that may have appeared.

The historian agent is in charge of archiving data and making it available for reasoner agents. It receives problem profiles from monitoring agents and stores them in a database. Storing these profiles is important because they contain information about previous task executions, information that is important for reasoner agents to make decisions.

Service agents have the role of communication with service providers. They can invoke services and also play a proactive role in the system by discovering services that can be integrated within the system. They will look in WSIL[5]/UDDI[6] service repositories in order to identify compatible services based on the ontology.

Last but not least, the matchmaking agents have two responsibilities. One is to handle storing semantic descriptions of domain ontology and of services, and WSDL descriptions of services. This will make the service database available for the system while alleviating the burden of dealing with storage mechanisms. The other responsibility is to offer goal-service matching functionality. Mainly, the execution agent can directly ask for a specific service or can ask for a proper service to be found by performing semantic matching between the service and the goal descriptions. In the proposed architecture we have focused on the relevant components for proving the concepts involved in a dynamic support for solving non-linear equations systems in a semantic services environment. We assumed the existence of the needed content in the Semantic and Syntactic Repository describing web services. Creation and maintenance of this repository is a complex task that will be considered in a future work.

One important component of this multi-agent architecture is the agent communication bus. This bus enables synchronous and asynchronous calls between the agents and is available in a distributed setting.

Figure 5.3 depicts the interaction between agents, interaction that leads to finding a solution for a simple problem.

[5]http://www.ibm.com/developerworks/library/specification/ws-wsilspec/
[6]http://uddi.xml.org/

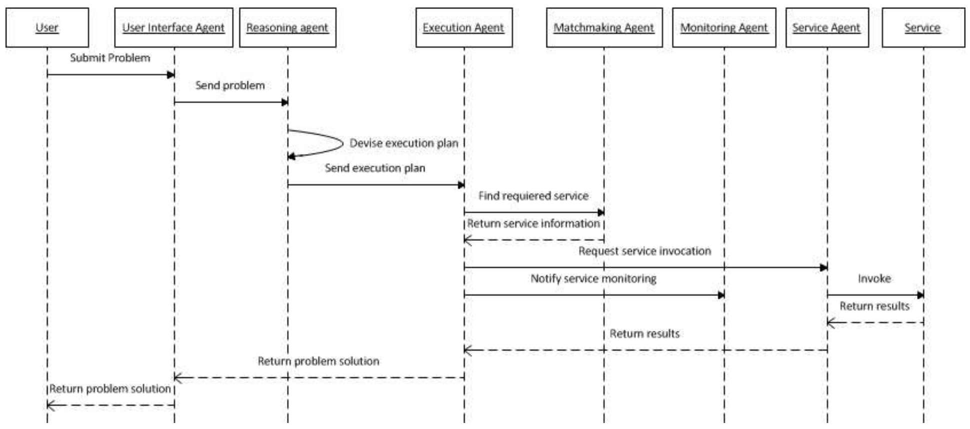


Fig. 5.3: An example of a simple problem solving sequence

5.2. Agent platform. Our approach in implementing the multi-agent system is based on the Akka platform[7]. Akka, while not being a traditional, FIPA compliant [4], multi-agent platform like Jade[8], offers tools to build highly concurrent, distributed, and fault tolerant event-driven applications [21]. Akka allows applications to be written both in Java[9] and in Scala[10] while implementing the actor model. The actor model enables our agents to have reactive behavior in solving the tasks.

Akka's integration with Apache Camel[11] allows agents to send and receive messages using a large number of protocols and APIs. Also, support for the Spring Framework[12] enables faster development.

Matchmaking is an important functionality of such a system. Different approaches are offered by WSMX [7, 11], WSMO-MX [10] or are domain specific and user oriented [23]. We have adopted a simplified version which makes use of a particular pattern in semantic descriptions of the services.

5.3. Framework and extensibility. The proposed architecture has a great flexibility due to the event-driven architectural style of the infrastructure and to the inherent flexibility of multi-agent systems. It can be seen as a framework with two main extensibility areas. The first extensibility area is in the society of agents which can be extended with new types of agents that augment and refine the solving paradigm. The second extensibility area stands in the semantic definitions. Extending the ontology allows to cover more mathematical chapters. Changing the ontology results in adapting to different domains where problems can be solved based on a similar core solving paradigm. Moreover, the two types of extensions can be combined.

6. Conclusions and future work. The task-oriented model makes a clear separation between tasks to be accomplished and methods for solving them. This model is implemented using a multi-agent architecture and is applied to design a solver for non-linear equations systems.

The multi-agent system is integrated with services implementing numerical methods. The services are semantically described in terms of a domain ontology we propose for non-linear equations systems.

Semantic descriptions are realized in WSML, allowing us to benefit from the clear separation between goals and services. This maps over the task-oriented model, specifically over tasks and solving methods respectively.

Specialized agents dynamically build semantic descriptions of the goals based on the properties of problems to be solved, and appropriate services are discovered by semantic matching.

Other agents are implied in user interaction, in building work plans, in managing the system, aiming to offer solutions to different problems or support for the human expert to experiment numerical methods in solving non-linear equations systems.

The multi-agent system is build over Akka platform. The system will be validated in the context of NESS.

[7]http://akka.io/
[8]http://jade.tilab.com/
[9]http://www.oracle.com/us/technologies/java/overview/index.html
[10]http://www.scala-lang.org/
[11]http://camel.apache.org/
[12]http://www.springsource.org/

Our future work has more directions: to extend the definitions of the knowledge in the domain of non-linear equations systems and to combine WSML with OpenMath and MathML; to include the chapter of ordinary differential equations systems and possible other mathematical chapters; to open the framework towards other domains.

Acknowledgments. This work was partially supported by the Romanian Government PNII grant nr. 12118/2008 (SCIPA), by POSDRU/88/1.5/S/49516 structural funds grant, ID 49516 (2009), and by the grant POSDRU 21/1.5/G/13798.

REFERENCES

[1] *Grid-enabled numerical and symbolic services. [Online] Available: http://genss.cs.bath.ac.uk/.*

[2] *Web service modeling ontology. [Online] Available:* `http://www.wsmo.org/`.

[3] M. Aird, W. Medina, and J. Padget, *Monet: service discovery and composition for mathematical problems*, in Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on, may 2003, pp. 678 – 685.

[4] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*, John Wiley & Sons, 2007.

[5] B. Chandrasekaran, *Design problem solving: a task analysis*, AI Mag., 11 (1990), pp. 59–71.

[6] J. Domingue and D. Fensel, *Problem solving methods in a global networked age*, Artif. Intell. Eng. Des. Anal. Manuf., 23 (2009), pp. 373–390.

[7] D. Fensel, F. M. Facca, E. Simperl, I. Toma, D. Fensel, F. M. Facca, E. Simperl, and I. Toma, *The web service execution environment*, in Semantic Web Services, Springer Berlin Heidelberg, 2011, pp. 163–216.

[8] D. Fensel, E. Motta, F. V. Harmelen, V. R. Benjamins, M. Crubezy, S. Decker, M. Gaspari, R. Groenboom, W. Grosso, M. Musen, Enric, E. Plaza, G. Schreiber, R. Studer, and B. Wielinga, *The unified problem-solving method development language UPML*, Knowledge and Information Systems, 5 (1999), p. 2003.

[9] M. Kerrigan, A. Mocan, E. Simperl, and D. Fensel, *Modeling semantic web services with the web service modeling toolkit*, Journal of Network and Systems Management, 17 (2009), pp. 326–342. 10.1007/s10922-009-9130-8.

[10] M. Klusch and F. Kaufer, *Wsmo-mx: A hybrid semantic web service matchmaker*, Web Intelli. and Agent Sys., 7 (2009), pp. 23–42.

[11] S. Komazec and F. Facca, *Whats new in wsmx?*, in The Semantic Web: Research and Applications, L. Aroyo, G. Antoniou, E. Hyvnen, A. ten Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, eds., vol. 6089 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2010, pp. 396–400.

[12] S. Ludwig, O. Rana, J. Padget, and W. Naylor, *Matchmaking framework for mathematical web services*, Journal of Grid Computing, 4 (2006), pp. 33–48. 10.1007/s10723-005-9019-z.

[13] E. Maximilien and M. Singh, *A framework and ontology for dynamic web services selection*, Internet Computing, IEEE, 8 (2004), pp. 84 – 93.

[14] C. Mindruta and D. Petcu, *A semantic services architecture for solving ODE systems*, in Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2010 12th International Symposium on, sept. 2010, pp. 301 –307.

[15] V. Negru, S. Maruster, and C. Sandru, *Intelligent system for non-linear simultaneous equation solving*, in Technical Report Report Series. No, 98-19. RISC-Linz, december 2003.

[16] M. j. O'connor, C. Nyulas, S. Tu, D. l. Buckeridge, A. Okhmatovskaia, and M. a. Musen, *Software-engineering challenges of building and deploying reusable problem solvers*, Artif. Intell. Eng. Des. Anal. Manuf., 23 (2009), pp. 339–356.

[17] D. Petcu, *Expert system for ordinary differential equations. [Online] Available: http://www.info.uvt.ro/ petcu/epode/-main.htm.*

[18] C. Sandru and V. Negru, *Validating UPML concepts in a multi-agent architecture*, in Schedae Informaticae, vol. 15, 2006, pp. 109–126.

[19] V. Sorathia, L. Ferreira Pires, and M. S. van, *An analysis of service ontologies*, Pacific Asia Journal of the Association for Information Systems, 2 (2010), pp. 17–46.

[20] The OpenMath Society, *Openmath. [online] available: http://www.openmath.org/.*

[21] Typesafe Inc, *Akka Documentation Release 2.0. [Online] Available:* `http://doc.akka.io/docs/akka/2.0/Akka.pdf`, March 2012.

[22] W3C, *Mathml. [online] available: http://www.w3.org/math/.*

[23] M. Wilkinson, B. Vandervalk, and L. McCarthy, *The semantic automated discovery and integration (sadi) web service design-pattern, api and reference implementation*, Journal of Biomedical Semantics, 2 (2011), p. 8.

Edited by: Dana Petcu and Daniela Zaharie
Received: March 1, 2012
Accepted: April 15, 2012