Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 981–984

ISBN 978-83-60810-22-4

Multi-Agent Architecture for Solving Nonlinear Equations Systems in Semantic Services Environment

Victor Ion Munteanu, Cristina Mindruta, Viorel Negru, Calin Sandru
Computer Science Department
West University of Timisoara
{vmunteanu, cmindruta, vnegru, csandru}@info.uvt.ro

***Abstract*—A semantic enabled multi-agent architecture for solving nonlinear 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.**

I. Introduction

THE MAIN goal of this paper is to propose a multi-agent architecture for solving nonlinear equations systems. This architecture is designed around a semantic-based solving paradigm supported by a service oriented ontology. That allows us to define expert agents 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 (MathML was cited also).

GENSS (Grid-Enabled Numerical and Symbolic Services) project [2] , like MONET, tries to combine grid computing and mathematical web services using a common open agent-based framework.

In [8] 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 [10] 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 numeric methods, human expert uses domain knowledge (numeric 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 [12]. EpODE was initially realized as a monolitic expert system [11] and has been re-engineered as a semantic services oriented framework [9]; the solving methodology is workflow-oriented, being realised by integrating semantic services with process modelling.

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 been 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 analyze the problem and propose an execution plan in order to find the solution. The execution plan is ran and adjusted if need be, and the result of the execution is returned to the user.

The execution plan contains numeric methods which can be 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)[1]. 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 nonlinear equations systems domain, and has an extensible database of semantic descriptions for services implementing numeric methods.

The system implements a core paradigm for solving problems, based on semantic matching between problem properties and numeric method capabilities. Numeric 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 numeric method and, form

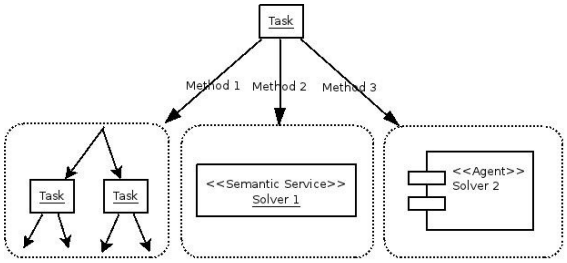

Fig. 1. Task structure

a given step, to continue with another numeric 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 numeric 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. Section II presents the proposed multi-agent architecture, based on the task-oriented model and integrated with the semantic infrastructure. Section III is dedicated to the semantic descriptions of the ontology, services and goals used to support the core solving paradigm. Section IV presents conclusions and future work.

II. Multi-agent architecture

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.

A. 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 [6], [1], [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). 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 nonlinear 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. In our architecture, the PSMs can be implemented as semantic services or as agents (fig.1).

B. Agents

The multi-agent system is composed of the following agents: client, reasoner, executor, monitoring, archiver, historian, service discoverer, service wrapper, and solver. These agents can be seen in fig.2.

The client agent handles the communication with the user. It exposes a graphical user interface which allows the user to interact with the system. The client agent communicates with the reasoner, executor and monitoring agents in order to manage the planning and execution.

The reasoner agent receives the problem from the client agent and creates the work plan to solve the current problem. It uses the domain ontology to create the semantic definitions of the tasks in terms of WSMO goals and will interact with the WSMX platform in order to find solving methods for a specific task. The historian agent provides the reasoner with plans that were applied in similar problems. The reasoner agent interacts with the executor agent when it needs to update the plan or when an alternative path is needed.

The executor agent handles the execution of the work plan. It communicates with the reasoner agent in order to receive and detail the work plan. The executor accesses the WSMX platform endpoint and queries it in order to find semantically compatible services for the current step in the work plan. It will communicate with the monitoring agent in order to report task progress.

The monitoring agent has the role to monitor the current task execution. It will send task information to the client agent, which in turn will notify the user about the current state of the execution. The monitoring agent will also create an execution profile and will send it to the archiver for storage.

The historian agent receives the current problem profile from the reasoner agent and it will look in the archive for similar profiles. These profiles are sent back to the reasoner so that it can make a decision based on past executions.

The service discoverer agent looks for new services that can be integrated in the system. It will look in WSIL/UDDI service directories in order to identify compatible services based on the ontology, and will search the JADE's directory facilitator for compatible solver agents.

The service wrapper handles services with no semantic description. It generates it's own WSML file for the service it is wrapping around and publishes it in the WSMX platform.

C. WSMX Platform

The web services that the system uses expose their capabilities, ways to interact with them, and ontologies through WSMO-compliant (WSML) descriptions. These web services

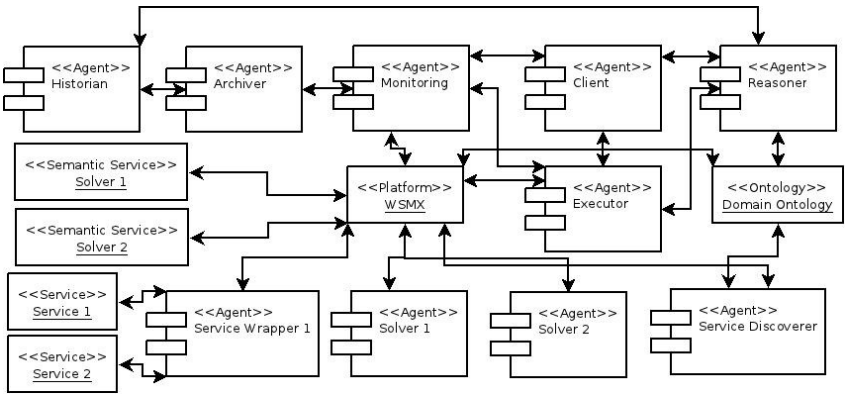

Fig. 2. Multi-agent system architecture

will be deployed on the WSMX platform. The WSMX platform can match the semantic descriptions of web services with semantic descriptions of goals provided by the agents, and can invoke the execution of those services. On the WSMX platform will be deployed semantic services, semantic agents that can wrap around non semantic services, and semantic agents that act alone. These services will execute the tasks given by the executor agent.

III. Semantic model

WSMO is appropriate for modelling the proposed semantic-core solving paradigm, because it offers a clear separation between goals and services. Services will offer numeric methods or compositions of methods from our paradigm, and goals are dynamically built for each problem to be solved.

1) Ontologies: `NESOnto` (fig.3) contains concepts, relations and axioms that define problem and solution spaces of the nonlinear equations systems.

`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 nonlinear equations system (`NES_PProps`). It also defines the concept of numeric method (`NES_Method`) in relation to the concepts representing the input data of the method (`NES_IN_M`) and the properties of the method (`NES_MProps`). The restrictions imposed on the solving session are modelled with the concept `NES_Session`, and the solution of the nonlinear 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 problem are of types defined in specific concepts (e.g. `SystemForm`), and for each of these concepts the particular instances (e.g. `General`, `Sparse`, `DiagonalExplicit`) have been defined.

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

2) Services implementing numeric methods: The capabilities of each service are described using `NESOnto` and an ontology specific to the service. One of the axioms in this specific ontology defines the relation *isSolvable* expressing the properties of the nonlinear equations systems for which the method is appropriate.

In fig.4 (a) is represented the semantic description of `Broyden_NES` service that implements the numeric method Broyden for solving nonlinear 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 nonlinear equations systems of general form, with nonsingular Jacobian, and of medium or big size. This represents a part of the expert knowledge and is implemented in the semantic description of the service.

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 fig.4 (b), an example goal of solving a nonlinear equations system is described in WSML.

IV. 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 nonlinear equations systems.

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

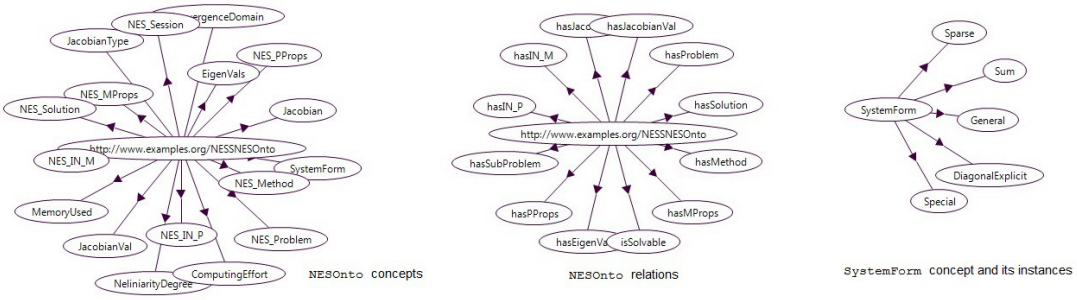

Fig. 3. `Elements of NESOnto` represented with WSMT

```
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 implies nes#isSolvable(?pp).
```

(a)

```
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. (a) WSML descriptions for `BroydenNES` service and (b) `ExampleGoal` goal

Semantic descriptions are realized in WSML, allowing us to benefit from the clear separation between goals and services. This maps over our 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 in order to offer solutions to different problems or support for the human expert to experiment methods to solve nonlinear equations systems.

The multi-agent system is implemented in JADE [4]. The reasoner agent uses JESS [7] as a rule engine together with the domain ontology in order to devise the execution plan. The system will be validated in the context of NESS.

In our future work we will extend the definitions of the knowledge in the domain of nonlinear equation systems and will try to combine WSML with OpenMath and MathML in expressing them.

V. Acknowledgment

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] WSMO. http://www.wsmo.org.
[2] GENSS project. http://genss.cs.bath.ac.uk, 2004.
[3] M.L. Aird, W.B. 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*, pages 678 – 685, may 2003.
[4] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic 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.*, 11:59–71, October 1990.
[6] Dieter Fensel, Enrico Motta, Frank van Harmelen, V. Richard Benjamins, Monica Crubezy, Stefan Decker, Mauro Gaspari, Rix Groenboom, William Grosso, Mark Musen, Enric Plaza, Guus Schreiber, Rudi Studer, and Bob Wielinga. The unified problem-solving method development language UPML. *Knowl. Inf. Syst.*, 5:83–131, March 2003.
[7] Ernest Friedman Hill. *Jess in Action: Java Rule-Based Systems*. Manning Publications Co., Greenwich, CT, USA, 2003.
[8] Simone Ludwig, Omer Rana, Julian Padget, and William Naylor. Matchmaking framework for mathematical web services. *Journal of Grid Computing*, 4:33–48, 2006. 10.1007/s10723-005-9019-z.
[9] Cristina Mindruta and Dana Petcu. A semantic services architecture for solving ODE systems. *Symbolic and Numeric Algorithms for Scientific Computing, International Symposium on*, 0:301–307, 2010.
[10] Viorel Negru, Stefan Maruster, and Calin Sandru. Intelligent system for non-linear simultaneous equation solving. In *Technical Report Report Series. No, 98-19. RISC-Linz*, december 2003.
[11] Dana Petcu. EpODE. http://www.info.uvt.ro/~petcu/epode/main.htm.
[12] Calin Sandru and Viorel Negru. Validating UPML concepts in a multi-agent architecture. In *Schedae Informaticae*, volume 15, pages 109–126, 2006.