Support for Goal Oriented Requirements Engineering in Elastic Cloud Applications

Zill-e-Subhan[1], **Robail Yasrab**[2], **Muhammad Farhan**[3], **Abdul Qahhar Mohsin**[4], **Muhammad Munwar Iqbal**[5]

[1] National University of Computer and Emerging Sciences (FAST) Lahore,
[2,4] e-Lecturer Virtual University of Pakistan,
[3,5] Department of Computer Science & Engineering, UET Lahore

Abstract
Businesses have already started to exploit potential uses of cloud computing as a new paradigm for promoting their services. Although the general concepts they practically focus on are: viability, survivability, adaptability, etc., however, on the ground, there is still a lack for forming mechanisms to sustain viability with adaptation of new requirements in cloud-based applications. This has inspired a pressing need to adopt new methodologies and abstract models which support system acquisition for self-adaptation, thus guaranteeing autonomic cloud application behavior. This paper relies over state-of-the-art Neptune framework as runtime adaptive software development environment supported with intention-oriented modeling language in the representation and adaptation of goal based model artifacts and their intrinsic properties requirements. Such an approach will in turn support distributed service based applications virtually over the cloud to sustain a self-adaptive behavior with respect to its functional and non-functional characteristics.

Keywords
Cloud Applications, Neptune, Intention, Goal Oriented Requirements Engineering (GORE)

1. INTRODUCTION

With the ever increasing need to utilize computing resources as non-scale bounded computing infrastructure, which cost a fortune, the shift towards the cloud computing as a new pay-as-you-need computing paradigm has been announced to help in reducing the cost and maintaining the system locality without necessarily accepting risks implicated by infrastructure fragility [10]. During early stages of the cloud, IBM had provided a definition to cloud as:

"An all-inclusive solution in that the entire computing resources (software, hardware, storage, networking, and so on) are offered quickly to users as demand dictates" [1]

In other words, the convention naming of X-as-a-Service (XaaS) or Software-as-a-Service (SaaS) now is increasingly used as a generalized aspectual adoption of services in the cloud. A influential fundamental idea is computing in the course of service-oriented architectures (or simply SOA) [2]: "delivery of an integrated as well as orchestrated suite of processes to an end-user in the course of composition of together insecurely plus strongly coupled processes, plus services" frequently network based. Connected ideas are component-based system engineering [3]: orchestration of dissimilar components in the course of workflows, as well as virtualization. In an service-oriented architecture environment, end-users demand an IT service (or an incorporated set of such services) at the preferred quality, functional as well as capacity level, plus obtain it either at the time demanded or at a particular later time. Service detection, brokering, plus reliability are significant, as well as services are typically intended to interoperate, as are the complex made of these services. The solution to a SOA structure that facilitates workflows is componentization of its services-based delivery. An incorporated vision of service-based actions is offered through the idea of a workflow. IT supported workflow signifies a series of structured actions that happen in information assisted problem reservation. In the situation of cloud computing, the main questions should be whether the fundamental arrangement is accommodating to the workflow-oriented view of the world. This comprises on-demand access to individual as well as combined computational plus other autonomics, resources, capability to group resources as of potentially diverse "clouds" to deliver workflow outcome, suitable level of security as well as privacy, etc. [4]. In essence the pressing need to proximate challenging interaction between system components to serve accomplishment of strategically set goals led to the forming of new Intention-Cloud-Based-Model [5], in which functional and non-functional system characteristics are reified by actors based distributed intentionalities. This inception had led to a new way of developing applications in the cloud, and thus cooperatively rendered for more adoption of elastic and dynamic computational resources.

In this paper we use Neptune1 as an adaptive cloud application development framework, which supports adaptation of runtime requirements through self-management system and an intention-based modeling

Manuscript received April 5, 2013
Manuscript revised April 20, 2013

language which stimulate an abstraction of Goal Oriented Requirement model. We use intentions as to describe the instantaneous behavior in system components to operationalize goals into functional and non-functional. Neptune uses the semantic description underlined by goals in order to find necessary requirements needed to fulfill these goals, thus by match and build necessary links between multi-stage levels of goals.

2. PROBLEM STATEMENT

Researchers continually seek extra well-organized as well as agile software engineering methods, particularly as solutions intended for repeatedly demanding applications are necessary. Against this environment, wide variety of software engineering paradigms has been planed (for example structured programming, procedural programming, object-oriented programming, declarative programming, application frameworks, design patterns as well as component-ware). An Intra-

Agent view perspective of software engineering has the following components [7]:

- **Social** -- who are the related actors, what do they want? What are their duties, what are their potentials?
- **Intentional** -- what are the applicable objectives as well as how do they interrelate? How are they state met, as well as through whom?
- **Process-oriented** -- what are the applicable business/computer procedures? Who is accountable for what?
- **Object-oriented** – what are the applicable matter as well as classes, along by their inter-relationships?

The require for transform can happen because to dynamic transforms in the users' requirements as well as the environment in that the system operates. Thus by expanding users instant needs and sometimes to fulfill infrastructure demands also, this will affect how application will function in order to successfully adapt to these new changes. Dynamic web-service composition permits an application to switch one web based service intended for another, at runtime.[9] However, finding and linking new services in a way to adapt new changes in application functionality or yet ones that are functionally the similar however demonstrate diverse QA levels, is a difficult process. An increased challenge took place when applications became distributed over the cloud and has a plenty of web based services to select from. The majority of the present research on dynamic web service composition spotlights on determining the most excellent match among parameters demanded through the application as well as the ones advertised through the services. The most excellent match, though, is hard if not possible to find in a reasonable time, known the dynamic nature as well as scalability of the cloud. Bounded through these restrictions, our approach presents to harness globally initiated changes in requirements of applications sitting in a cloud environment, through formal semantic description of these requirements in order to fulfill desired system goals.

3. PROPOSED SOLUTION

It outlines clear (and desirable) that applications should adjust themselves to altering requirements, particularly all through procedures time. Though, in an random case, formulating an implementation adaptable is a hard procedure. It has to be re-architected in similar a way that is able to rationalize its working all through requirements transformation. This means that it should be able to:

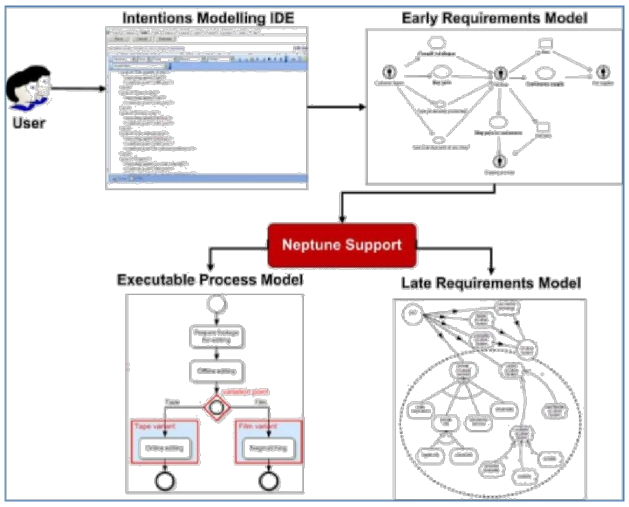
1. Dynamically decompose its part artefacts driven through their socio-intentional potentials.
2. Dynamically recognize objective, requirements, that will require runtime adaptation;
3. Start the search intended for novel services, components that enhanced address adapted objectives;

In this paper, we look at these steps, thus to capture runtime requirements needed to search and find atomic system components best fit to fulfill system goals and objectives, while we keep to a later stage verification and assurance rendered from value proposition to acting stakeholders also attracting alternative decisions to trade between actors goals and services seeking better alternative scenario based actions needed to be taken to satisfy these goals.

1. MODELLING GOALS OVER THE CLOUD

Since applications over the cloud can cover broaden range of heterogeneous platforms, the most appropriate way to model goals over the cloud is using GORE abstract models, to this end actors in goal based models are represented as stakeholders as well as their objectives have been recognized, a strategic foundation model decides in the course of a means-ends examination how these objectives (as well as softgoals) are able to really be satisfied in the course of the assistance of other actors. A planned rationale model is a graph by means of four kinds of nodes -- objective, job, resource, as well as softgoal -- plus two kinds of links -- outlined-ends links and procedure decomposition links. A strategic rationale graph detains the association among the objectives of each actor as well as the dependences in the course of which the actor expects these reliance to be fulfilled. Cloud service provider in an e-commerce economic model is represented by shop front owner, shipping provider, payment facilitator. These actors have dependency relationships between each other presented as depender and dependee so for a strategic relationship between a shop-front owner and a buyer for ex. in order to achieve a goal

“provide best value over price”, the depicting process model to accomplish such goal when a buyer submits his orders over shop-front coactively by sharing this goal with different other cloud service providers like for ex. payment facilitator, packaging, delivery options provided by shipper, these will represent a dependee links as opposed to shop-front owner which in turn a acts as depender. Actors in the cloud are initiators of goals and sub-goals, when goals are objectives the system and/or its subsidiary components have to achieve under certain conditions or constraints. Goals can be of two types according to their commitment to achieve, soft-goals which cannot be foreseen as in state of achieved according to their non-deterministic nature, thus can instead infer either positively or negatively the achievement of hard-goals which can be accomplished when transitioned to a specific state. While soft-goals can be “metricised” through tasks and functions, hard-goals can be achieved by soft-goals and tasks. During the specification of goals from their semantics, one goal statement could be complex like “Achieve Better Customer Satisfaction”, such type of goal is usually called Strategic Goals, which can be proposed on an organizational level, sometimes extracted from organization vision. Two other levels of goal decomposition are High Level/Core objectives as well as Lower-level/Operational objectives. High level aims are nothing however core objectives described all through



requirements elaboration in the cloud, so in that case these goals are identified only during elaboration of requirements. A new introduction of third level which is represented as macro staged level called NBLO (Neptune Base Language Object) level. At this level Neptune semantically searches to find link through service agreements with high system model abstracts to other cloud service providers using meta object abstractions. That link provides required resources to be utilized as services for the system to perform functionally (Figure 2).

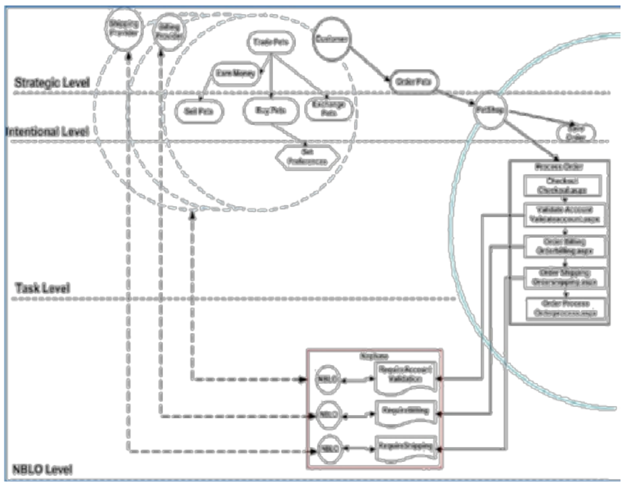


Figure 2: Goal abstraction levels in the cloud

Explaining objectives AND/OR structure through describing objectives as well as their refinement/conflict links in anticipation of transferable fundamentals are attained. The procedure of recognizing objectives, describing them exactly, as well as relating them in the course of positive/negative involvement links is generally a blend of top-down as well as bottom-up sub-processes; offspring objectives are recognized through asking HOW questions regarding objectives previously acknowledged whereas parent objectives are recognized through asking WHY questions regarding goals as well as operational necessities previously recognized (Figure 3). The elaboration process of goals into goals and sub-goals can be anticipated through our cloud intention modeling methodology, in which writing intentions that drive process and sub-process models, the process model is annotated with goals and sub-goals, where the lowest level in goal tree are the Operational Goals, which will be supported by Neptune Functions as tasks at the operational level. Objectives have to clearly be specified accurately to facilitate verification/validation, requirements elaboration, negotiation, conflict management, replacement, explanation as well as evolution.

Figure 3: Capturing Early and Late Requirements with Neptune

2. INTENTION-CLOUD-BASED MODEL

The intention model is composed of flow model and logic model. Whereby, I representing the intention model can be defined as:

I = < T, cp >

Where T represents a set of tasks to be executed to achieve the goals, which define the Intention model and cp, represents a set of conditions which define and/or control the associated task set. In other words, T represents the flow model, and cp represents the control flow model of a given Intention model (Figure 4). [10]

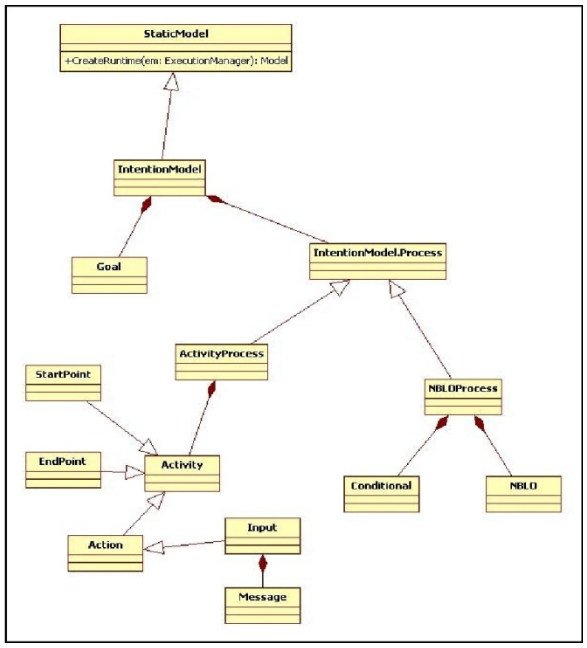


Figure 4: ICBM Meta Model

IV. CASE STUDY: DESIGN PETSHOP SCENARIO VIA GOAL MODELLING

Using Microsoft PetShop blueprint as an online e-commerce store-front, we re-engineer PetShop architecture to be a cloud based application driven by GORE methodologies. Based on is goal modeling methodology, two models are available Strategic Dependency and Strategic Rationale. We first need to specify who are the main actors and their goals (Listing 1).

```
<PetShopInte
    ntion>
    <goals>
        <actor  id="PetShop"/>
        <actor id="Product
        manager"/> <actor
        id="Cart"/>
        <actor id="Account
        controller"/> <actor
        id="Billing provider"/>
        <actor id="Shipment
        provider"/> <actor
        id="Order manager"/>
        <actor id="Cart
        controller"/>
```

Listing 1: representation of actors using intention modeling

Goals and soft-goals are of multi abstraction levels starting from strategic, core and operational, strategic goals are represented in two models, strategic dependency and strategic rationale. In our *PetShop* scenario, “Consult Catalogue” is a strategic goal for actor “Customer”. Since intentions identify locality attributions of actors for the specification of their instant behavioral needs, in cloud environment the effort behind using intention modeling is to identify actor’s needs. Hence in order to share the semantic description of actor requirements, an exposition of these requirements to Neptune would harness more cloud service providers to be linked through interfaces through dependencies as actors in strategic dependency model. As seen in (Listing 2) alternative optional intentions can be presented by an actor to support progression of a strategic goal i.e. “Consult Catalogue”; a formal way of presenting goal decomposition into goals and sub-goals.

```
<goal id="See pet details" mode=”achieve”>
            <supporting actor="PetShop"/>
            <contribute goal="Consult
            catalogue"/> <contribute
            goal="Order pets"/>
        </goal>
        <goal id="See pet by search"
            mode=”achieve”> <supporting
            actor="Product manager"/>
            <contribute goal="Consult
            catalogue"/> <contribute goal="Order
            pets"/>
        </goal>
        <goal id="See pet by category"
            mode=”achieve”> <supporting
            actor="Product manager"/> <contribute
            goal="Consult catalogue"/> <contribute
            goal="Order pets"/>
        </goal>
```

Listing 2: Goal decomposition using intention modelling

The process model is annotated with intentional goals and their attributes; this differentiates between actor intentional goals and strategic ones (Listing 3). Actions which entail tasks also have contribution relationship through annotations as in (Listing 2), so each action in the process flow is linked accordingly to a core goal, this relationship is been presented as semi-formal specification often include keyword, like Achieve, Maintain and Avoid verbs in KAOS [09].

```
<process id="1" text="Consult catalogue"
        type="activity"> <strategic goal="Consult
        catalogue"/> <startpoint id="Start point">
            <moveto>Choose way of
        consulting</moveto> </startpoint>
        <action id="Choose way of
            consulting"> <contribute
            goal="See pet details"/>
            <input type="text">
                <message><![CDATA[index.aspx]]></mess
            age> </input>
            <moveto result="search">See pet by
                    search</moveto> <moveto
                    result="category">See pet by
         category</m
        oveto>
        </action>
        <action id ="See pet by search">
            <contribute goal="See pet by
            search"/> <input type="text">
                <message><![CDATA[0]]></me
            ssage> </input>
          <moveto>Search</moveto>
          </action>
```

Listing 3: Goal annotation over process model

Tasks will be called through actions in the flow model using ask construct, here a specification of the required operation to be taken and Neptune function which support it. Support for task decomposition will be specified at Neptune function. The process model contains flow of actions that should be executed respectively and controlled with control model, the same representation can be accomplished through goal decomposition model as strategic goals are decomposed into goals and sub-goals, these in turn represent intentions in the intention model while operationalization of goals occurs at task level, the task decomposition into sub-tasks with "AND" relationship can represent actions in process flow model, when the final state of goal can be set to achieved only when all tasks and sub-tasks are set to completed. The "AND" and "OR" relationships can be represented through control model in the process flow.

```
         <ask                        id="Search"
            NeptuneFunction="search">
            <contribute    goal="See   pet   by
            search"/> <input type="text">
                <message><![CDATA[0]]></me
            ssage> </input>
            <moveto>Show search
        results</moveto> </ask>
```

Listing 4: Support for Neptune Function

Intention model provides a dynamic interface to write functional support for tasks that Neptune should executed at runtime (Listing 4), through this support we may call Neptune function at any point during the flow model execution, as there are number of tasks should be executed within each intention. By accomplishment of these tasks, the state of intention is said to achieved, once intention model is interpreted to generate valid abstract process and control model, these will be used by Semantic Linker to generate BPEL like orchestration model. During this stage Neptune will use task function to discover software services – including services, and software components – ontological and WSDL definition that will associate each task with required services (endpoints) that can be invoked to perform that task. Composer then composes those services and components together to produce a new functionality. The produced composition will be sent respectively to the Execution Engine (e.g. BPEL engine) (Figure 5).

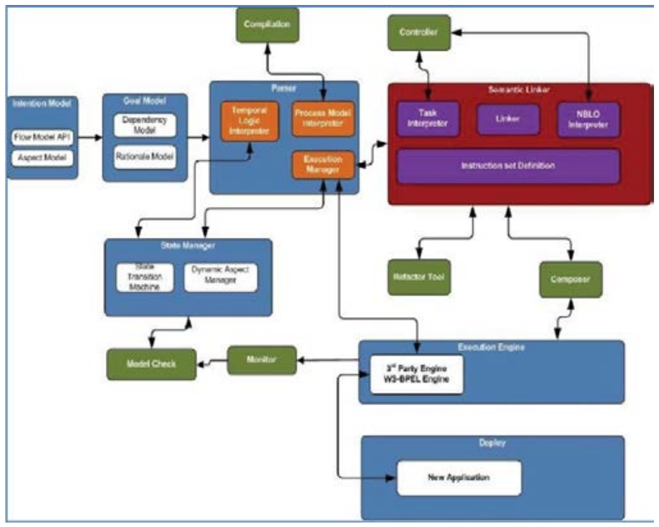


Figure 5: Goal modeling support for Neptune

As the major design objective of Neptune is its capability to generate runtime flexibility. As such, determined whether an adaptation established at runtime or at design time is intrinsically simple; if no re-deployment and re-compilation of the application happened in a customary design sense, as a result the edition are able to be recognized to be made at runtime. The procedure of performing an search inside the PetShop is split among the logic layer as well as the presentation layer. Data is recovered to notify the procedure by means of discrete web pages, like that SeePetsDetails.aspx holds the trigger code that moves the procedure towards one more task. User has two subordinate intentional goals when trying to achieve See pet detail goal, either he performs a task which supports quick access to specific product details or to choose to browse products by category. By decomposing these intentions into process of activities a system modeled by interpreting intention model needs to identify relevant tasks needed to satisfy these intentions, thus by writing required support for Neptune function is expected to trigger compilation of Neptune Base

Language Objects (NBLO's), which will be used by Semantic Linker to link required components to the system.

1. WRITING NEPTUNE SUPPORT

Neptune provides the system-to-be as a new actor to dependency model, at this stage Neptune starts to model late requirements by building same dependency and rationale models for system layers and components, here in our case presentation layer as a depender delegates accomplishment of actor task of viewing Pet details to logic layer, then logic layer provides reasoning to save and pull resource data from data layer. The running of the procedure passes to the logic layer to understand the data, as well as to store it in database. Once stored, the logic moves the running to create another page intended for the user. Similarly, the classes inside the logic layer are able to be said to modify the state of the procedure through demonstrating that a task is complete, as well as that data is incorporate to the data layer structures, as these are the disadvantages of its actuation. Likewise pages inside the presentation layer are able to be said to initiate data to the state of procedure, plus indicate that the task of generating data is complete. Through these means, characterized intentional elements of goals which gain achieve modality exhibit new state for goal fulfillment.

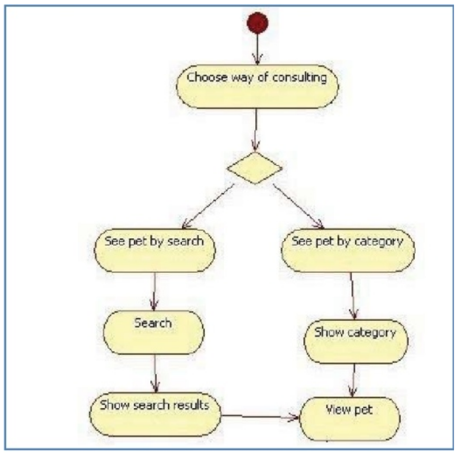


Figure 6: Display Pet Details Process

(Listing 5) demonstrates the explanation of search components inside the logic layer as well as the presentation layer. In this means the nbloSearchPets describes the method getSearchResults within the presentation layer as adding the features PetName, PetDesc and PetLocation to the sessionID element within the state of the process. Similarly, the nbloDisplaySearchResults adds the feature DisplaySearchResults to an constituent Database to explain the reality that operationally. Values are placed intended for the characteristics through way of the return value of the fundamental parts. By itself, PetLocation will take the value of the return the characteristic PetLocation of the object returned through the technique getSearchResults. Other values are able to be specified through the as operator in Neptune, like that the values of the return type are able to be reflected.

```
define nbloSearchPets with
NString sessionID
 {purpose
         {
         feature PetName to
         sessionID ; feature
         PetDesc to sessionID;
         feature PetLocation
         to sessionID;
 }
 actuation
 {
// call the presentation
layer call
         BaseLanguage.Csharp(”searchPL.dll”, ”get
         SearchResults”, sessionID,sync);
 }}
//  logic  layer  component
         define nbloSearchPetByKeyword with
         NString sessionID {purpose
 {
         feature  sessionID.Searching  to  Database  ;
 }
         actuation
 {
 //  call  the  logic  layer
         call
         BaseLanguage.Csharp(”search
         LL.dll”, ”processSearchPets
         ”,sessionID,sync);
 }}
```

Listing 5: NBLO definition of logic and presentation components.

The procedure itself is able to be declared as act from discrete jobs that each necessitates particular information to be accessible inside the state of the procedure. The development model employed inside the PetShop to execute a “See Pet Details” is presented in figure 6. As every stage in the procedure has a associated class, it is able to be said, for instance, that previous to entering the job designated for taking pet details from the products, data have to first be accessible. In this means, axiomatic semantics intended for the tasks are able to be placed foundational on information necessary through the task. Likewise, the job itself necessitates information to be incorporated to the state after its completion, like for the class SearchPets the data relating to show particulars of the pets are the job performed through that part of the procedure.

The subsequent (Listing 6) demonstrates the procedure “See Pet Details”, that necessitates that search data should be accessible in the state element sessionID, as well as that this should be put aside to the database, or rather Database.Searching should be accessible on known searched keyword.

```
task  SearchPets   with  NString  sessionID
// Presenting system as an actor for late
requirements {actor PetShop}{mode achieve}
//GORE
requirements
{requirements
 {
         //resource requirements
         needPetName:require
         sessionID.PetName;
         needPetDesc:require
         sessionID.PetDesc;
         needPetLocation:require
         sessionID.PetLocation; //task
         requirements
queried:require  Database.Searching(sessionID)  ;
 }
 }
```

Listing 6: Task Definition for Pets Searching.

The semantic linking module, when outlines by means of task description, are able to then position the NBLO that are able to offer the data necessary to be completed through the job (Listing 7), offering a level of autonomy among processes description as well as the logic plus presentation layers of the systems. As both processes are synchronous, the running of the actions will establish in sequence, permitting the data to be located, as well as incorporated to the database.

```
task  SearchPets  with  NString  sessionID
// Presenting system as an actor for late
requirements {actor PetShop}{mode achieve}
//GORE
requirements
{requirements
         {
         //need resources for Means-Ends
         needPetName:require
         sessionID.PetName;
         needPetDesc:require
         sessionID.PetDesc;
         needPetLocation:require
         sessionID.PetLocation; //need
         task
queried:require  Database.Searching(sessionID)  ;
}
//as processed by the semantic
 linker actions
 {
         needPetName,needPetDesc,needPetLocation via
         nbloSearchPets; queried via
         nbloSearchPetByKeyword;
}
}
```

Listing 7: Task Definition for Shipping

5. CONCLUSIONS

Intention modelling and Neptune opened new visions over how to expose language semantics over the cloud, this in turn showed extraordinary potential for adding goal modeling and goal requirement engineering through composition of multi actor agency partnering in a socio economical model. It is however still essential to add support for assurance and verification of transitional change of actors goals at runtime and how to operate in competitive market, this will surely reflect more complexity while keeping best value proposition during goal refinery process. Future work will cover assisting cloud features for multi-tenancy, quality assurance through runtime verification of non-functional system requirements linked to softgoals, thus how to maintain system steady-state over large scaled user demand-ship. It is foreseeable for example, to dynamically build on-demand venturing partnership as dependency models through call of agency NBLO's at actor level rather than task level. As is obvious in this paper though, intention modeling has demonstrated promise in realistic application intended for offering autonomous behavior bounded through behavioral as well as architectural issues of how to build dynamic composite service based cloud application.

REFERENCES

[1]. Dustin Amrhein, Scott Quint (2009), Cloud computing for the enterprise: Part 1: Capturing the cloud, Understanding cloud computing and related technologies, Retrieved March 28, 2012, from http://www.ibm.com/developerworks/websphere/techjournal/0904_amrhein/0904_amrhein.html

[2]. Priyanka R. Nayak, "A Seminar Report on Cloud computing", p5, (2009-2010)

[3]. Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, Anand Ghalsasi (2011), "Cloud computing — The business perspective", Decision Support Systems, Volume 51, Issue 1, April 2011, Pages 176-189, http://www.sciencedirect.com/science/article/pii/S0167923610002393

[4]. T. Baker, A. Taleb-Bendiab, M. Randles, Y. Karam (2010), "Support for Adaptive Cloud-Based Applications via Intention modelling", School of Computing and Mathematical Sciences, WSS '10: Proceedings to 3rd Intl, Symposium on Web Services at Zayed University, pp 1-6

[5]. Philip, M., Talebbendiab A., "CA-SPA: Balancing the Crosscutting Concerns of Governance and Autonomy in Trusted Software", Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 2 (AINA'06) (Washington, USA), IEEE Computer Society, pp. 471–475, (2006).

[6]. Winikoff, M, Padgham, L, Harland, J and Thangarajah, J 2002, 'Declarative and procedural goals in intelligent agent systems', in Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning, Toulouse, France, 22-25 April 2005.

[7]. Jordi Cabot and Eric Yu (2008), "Improving Requirements Specifications in Model-Driven Development Processes", First International Workshopon Challenges in Model Driven Software Engineering, http://ssel.vub.ac.be/ChaMDE08

[8]. Sayed G. Tabatabaei, Wan M. Kadir, Suhaimi I. (2008), "Web Service Composition Approaches to Support Dynamic E-Business Systems", V 2, no 16, Communications of the IBIMA, 2008, pp 115-121

[9]. Axel van Lamsweerde (2001), “Goal-Oriented Requirements Engineering: A Guided Tour”, Proceedings of the 5th IEEE International Symposium on Requirements Engineering, p.249, August 27-31, 2001

[10]. Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. (2010), “A view of cloud computing”, Commun. ACM 53, 4 (April 2010), 50-58, http://doi.acm.org/10.1145/1721654.1721672