2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

Agent Enabled Adaptive Management of Cloud Service Provisioning

Mohan Baruwal Chhetri*, Quoc Bao Vo*, and Ryszard Kowalczyk*†
*Faculty of Science, Engineering and Technology
Swinburne University of Technology, Melbourne, Australia
Email: {mchhetri, bvo, rkowalczyk}@swin.edu.au
†Systems Research Institute
Polish Academy of Sciences, Warsaw, Poland

***Abstract*—Infrastructure as a Service (IaaS) providers tend to use heterogeneous hardware environments and virtualization techniques to provision a wide variety of virtual machines that have varying specifications, exhibit varying performance levels, and are offered under different pricing models. Consumers also have very diverse and dynamic infrastructure requirements depending upon the workloads that they intend to deploy on the cloud. Given the diverse and dynamic nature of demand-side requirements and supply-side capabilities, market participants can benefit from decision-support and decision-making tools that automate, augment and coordinate some or all of their decision processes leading to the service exchange. This can be realized through autonomous software agents that can offer the necessary capabilities for adaptive management of the different tasks associated with service provisioning. We use the practical example of on-demand resource provisioning on Amazon EC2 to motivate, illustrate and validate our proposed solution.**

I. Introduction

Everything as a Service (XaaS) [1] is a cloud-based service delivery model in which organizations offer their core capabilities at any layer of the IT stack as cloud services using new and innovative pricing and provisioning models. Similarly, *Service Aggregation* or *Composite Service Provisioning* [2][3] is a service-centric business model that allows organizations to loosely combine component XaaS services in innovative ways to create value-added composite services that span organizations and computing platforms. *Dynamic Business Networks* (DBN) [15] offer service consumers, providers and composite service providers a highly diverse, dynamic and distributed environment, in which they can collaborate to establish on-demand business partnerships based on their requirements and capabilities.

Infrastructure-as-a-service (IaaS) is a popular XaaS business model that typically involves the on-demand, over-the-internet provisioning of virtualized computing resources such as memory, storage, processing and network using a pay-as-you-go model. IaaS providers tend to use heterogeneous hardware environments and virtualization techniques to provision a wide variety of virtual machines that have varying specifications, exhibit varying performance levels, and are offered under different pricing models. IaaS consumers also have very diverse and dynamic infrastructure requirements depending upon the workloads that they intend to deploy on the cloud. For example, some cloud consumers want to run time-flexible, interruption-tolerant tasks under different deadlines and budgets. The *Amazon EC2 Spot Market* offers a cost-effective purchasing option for such customers. Other customers might have tasks that require uninterrupted access to the computing resources for longer durations of sustained use, in which case the *sustained use discount* pricing model offered by Google Compute Engine (GCE) might be an attractive option.

Given the diverse and dynamic[1] nature of consumer requirements and provider capabilities, participants on both sides of the IaaS market, can benefit from decision-support and decision-making tools that automate, augment and coordinate their decision processes and consequently improve the efficiency and effectiveness of the service provision process. Markets that facilitate such intelligent decision-making through the use of Artificial Intelligence are referred to as Smart Exchanges [4][5]. There are two broad categories of intelligence that can be provided to facilitate decision-making in smart exchanges. *Individual intelligence* is primarily used by individual entities in individual markets. *Collective intelligence* refers to market level intelligence where several atomic entities within and across markets interact with each other and there is a co-dependency and co-evolution of markets e.g. end-to-end supply chains.

In this paper, we propose an novel agent-based framework for the adaptive management of dynamic service provisioning. The adaptive management capabilities are realized using *policy-aware software agents* i.e. software agents whose reasoning behaviours are guided by policies. We use the practical example of on-demand resource provisioning on Amazon EC2 to demonstrate how our proposed framework can ensure quality-aware and quality-assured service provision and procurement.

The rest of the paper is organised as follows. We present our motivating scenario to illustrate the need for adaptive management of service provisioning in Section II. In Section III we present our proposed approach for agent-enabled adaptive management. In Section IV, we present an overview of our policy-aware agent framework including agent policies, policy processing model, agent architecture and proof-of-concept prototype implementation. We briefly look at related work in Section V and conclude the paper in Section VI.

[1]Diverse – showing a great deal of variety, dynamic – characterized by constant change, which can be transient (lasting only for a short time) and volatile (liable to change rapidly and unpredictably).

978-1-4673-9618-9/15 $31.00 © 2015 IEEE
DOI 10.1109/WI-IAT.2015.39

II. Motivating Scenario

As a motivating example, we consider the scenario of on-demand resource provisioning on the cloud. We assume that a service entity that we call *Smart CloudPurchaser (SCP)* offers a service whereby it dynamically procures computing resources for its customers to process complex jobs that have varying levels of interruption-tolerance, time-flexibility and budget limits. In addition to having choice of multiple instance types, operating systems, software packages and geographical locations, customers also have choice of purchasing model. Therefore, each time SCP receives a request to procure computing resources, it has to decide which instance type to rent, how many instances to rent and whether to purchase an on-demand instance, bid for a spot-instance or look for resources in the reserved instance marketplace (see Figure 1). If purchasing spot instances, it has to determine the best bid value to use. The choice of purchasing model and bidding strategy can vary depending upon the interaction context (see Figures 2 and 3).

A. Multiple Purchasing Options

SCP can procure the cloud servers from Amazon EC2 using four different purchasing models.

- *On-demand Instances* - EC2 consumers can request on-demand instances at any time and at fixed hourly rates with no long-term commitments or upfront costs. While Amazon tries to maintain sufficient on-demand capacity to meet consumer needs, there is a possibility that during periods of very high demand, an on-demand request may not be fulfilled immediately. However, once an on-demand instance has been launched, the consumer has *uninterrupted access* to it.
- *Reserved Instances* - EC2 consumers can pay a small one-time, upfront reservation fee (for one year or three years) for different types of instances, and then have *guaranteed immediate and uninterrupted access* to them at any-time within the reservation period at lower fixed hourly rates.
- *Spot Instances* - EC2 consumers can bid for unused resources on the Spot Market. They can specify the maximum hourly price they are willing to pay for a particular instance type and the deadline by which they need access to the instance. Customers get access to the requested resource if their bid price is above the spot price and can use it as long as it remains above the spot price. However, if the spot price goes above the bid price, Amazon shuts down the instance with a two-minute notification.
- *Reserved Instances from Marketplace* - EC2 customers can also procure instances on the Reserved Instance Marketplace. EC2 customers with unused reserved instances can re-sell them in the Reserved Instance Marketplace. They can determine the selling price, the quantity and the duration for which they want to sell the reserved instances. Potential customers can search the marketplace for suitable offers and purchase directly from the sellers.

B. Preference based Service Selection

The first step in the procurement of computing resources from Amazon EC2 is the selection of the appropriate cloud server instance type from the list of pre-configured instance types that EC2 publicly advertises. The choice of cloud server is determined from the preferences specified in the request. The preferences may be specified as absolute minimum and maximum values for individual attributes - for e.g. the job completion time, the total budget, or the minimum memory required, minimum storage space and the geographic location. Conditional preferences may also be specified in terms of the infrastructure requirements under different job completion deadlines. For example, if the deadline is too short and the job is high-memory intensive, then it is better to go with a more powerful instance. On the other hand, if the deadline is long, then the objective may be to minimize the cost and go for a cheaper machine.

Fig. 1: Amazon EC2 - Multiple purchasing options

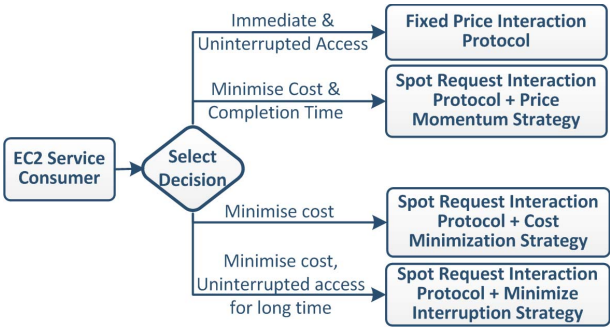


Fig. 2: Context-driven selection of purchasing model

C. Context driven Purchasing Model Selection

Once the instance type to be procured has been selected, SCP has to decide which purchasing model to use in order to procure it. Depending upon the current context, SCP can either try the on-demand purchasing model, the spot-instance purchasing model or the reserved instance marketplace model to rent the resources and fulfil the incoming request. If using the spot-instance purchasing model, it can choose from a number of different bidding strategies (see Figure 2).

D. Concurrent Interactions

At any given time, SCP can procure the same instance type from EC2 using different purchasing models and end up paying different prices for the instance. For example, if SCP receives a job that needs to be completed immediately, it might choose to procure the instances using the on-demand purchasing model. It might also submit additional spot bids in

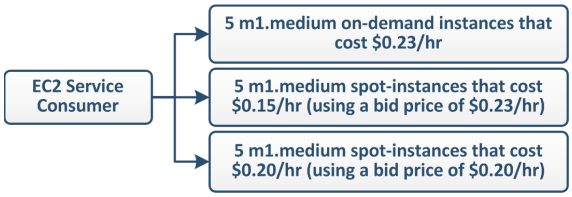


Fig. 3: Concurrent procurement using different purchasing models

different spot markets and at different prices. Figure 3 shows a scenario where SCP is participating in multiple concurrent interactions using different purchasing models.

E. Adaptive Management

Context-driven selection of purchasing model does not always guarantee the successful procurement of cloud resources. For instance, if SCP chooses the spot instance purchasing model, there is no guarantee that the requested cloud server will be procured within the stipulated deadline. The spot request might be unsuccessful for a number of different reasons such as bid *price-too-low*, *capacity-not-available* or *capacity-oversubscribed*[2]. SCP can take a number of different adaptive actions in such a situation.

- *Change bidding strategy.* If the bid status code is *price-too-low*, SCP can either choose to keep waiting for the status to change, or decide to adapt its strategy. If the deadline is large enough, SCP can cancel the current spot request and submit a new one with an updated bid price. The adaptive action is to adjust the bidding strategy but use the same purchasing model.
- *Change purchasing model.* If the bid status code is *price-too-low* and the task completion deadline is small, SCP can decide to cancel the current spot request and submit a new on-demand request. The adaptive action is to use a new purchasing model.
- *Change Server Specification.* If the bid status code is *capacity-not-available* or *capacity-oversubscribed*, SCP can decide to either purchase an on-demand instance or find another server specification which is similar to the currently requested server and resubmit a new spot request. The second adaptive action is to change the server specification.

These simple examples show how adaptive management of the service procurement process can improve the chances of successfully procuring and consuming the requested service under different interaction contexts.

III. Adaptive Service Provisioning

In this Section, we present a conceptual model and framework for the quality-aware and quality-assured provision and procurement of cloud services. We extend the traditional Consumer Buying Behaviour (CBB) model augmented with concepts from intelligent agent technology [6][7], and discuss the main steps involved in the quality-aware fulfilment of a dynamic service request as shown in Figure 4. We also present and discuss our model for the quality-assured procurement of services as shown in Figure 5.

A. Requirements Elicitation

The first phase in the dynamic service procurement process is the requirements elicitation phase (not shown in Figures 4 and 5) In this phase, the service consumer submits its functional and non-functional (QoS) requirements including constraints and preferences. A customizable service is normally characterised by multiple customizable attributes, which can range from low-level technical attributes to high-level business attributes. Expressing preferences and constraints over them is a necessary first condition for quality-aware service provisioning. In order to maximize the chances of reaching an agreement, service consumers and providers have to sufficiently specify their preferences over the service usage terms and conditions. For this, they require preference models which are highly expressive and flexible, easy to use, and support varying levels of complexity. Preference models which satisfy these requirements can be used with a wide range of interaction protocols and decision-making strategies. Service providers can benefit from such models because it allows them maximum flexibility in the offered service quality levels. Similarly, consumers can benefit because they have a better chance of finding service offerings that match their requirements.

If we consider our motivating scenario, every time the service consumer has a job that needs to be executed on the cloud, it submits a request to SCP. The request can specify preferences over the server configuration, the geographic location, the operating system, and the cloud provider. It can also specify the time-sensitivity of the job i.e. how quickly the task needs to be completed, and the budget sensitivity of the job i.e. the budget available for executing the job.

B. QoS-Aware Service Procurement

Intelligent software agents can handle diverse and dynamic consumer requests and select the best decision in each phase of the service procurement process including *Service Selection*, *Provider Selection*, *SLA Negotiation & Contracting*, and *Service Consumption & Monitoring*. We refer to this as *QoS-aware provisioning of services*, because, in each phase, the decision actions are influenced by the preferences specified in the user request and are taken in the best interests of the service consumer.

- *Service Selection.* There can be several functionally equivalent services that can fulfil the user requirements. In this stage, candidate services that can best satisfy the incoming request are identified by evaluating the alternatives based on non-functional criteria. The result of this stage is the *consideration set of services*.

  In our scenario, SCP can refer to a server catalog that lists all details about the offered instance types including price, specification, geographic location and performance data, and short-list the candidate servers that meet the configuration constraints and the performance constraints. The candidate servers can be ranked

[2]Please refer to http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-bid-status.html for a complete list of Spot Bid status codes

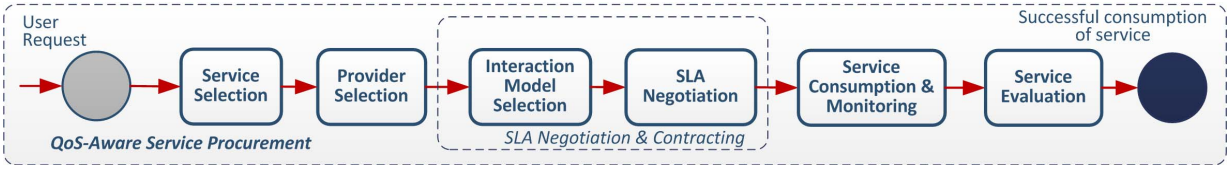


Fig. 4: Agent behaviour for QoS-aware service procurement

using different ranking criteria such as cost, geographic location, specification and performance.

- *Provider Selection.* In this stage, candidate providers that can offer the services identified in the previous stage are short-listed by evaluating service provider alternatives based on different selection criteria including trust, reputation and past performance.

  In our scenario, SCP selects Amazon EC2 as the service provider since it is the only IaaS provider that offers a spot market.
- *SLA Negotiation & Contracting.* In this stage, negotiations are carried out with the short-listed providers from the previous stage to try and find a service offering with the most acceptable service usage terms and conditions. The consumer can negotiate with a single provider or with multiple providers concurrently, using different interaction protocols and decision-making strategies.

  In our scenario, SCP has to determine how to procure the required cloud servers from the short-listed providers. As an illustrative example, it can submit multiple spot requests at different bids for the same server type but in different geographic locations. Once a mutually acceptable agreement is reached with a particular provider, the remaining negotiations can be terminated and the cloud server can be provisioned by the contracted provider and consumed by SCP.
- *Service Consumption & Monitoring.* Once a SLA has been established with a provider, the consumer can start consuming the service for the duration agreed upon. Monitoring of the provisioned service in the consumption phase ensures SLA compliance by all involved parties.

  In our scenario, assuming that SCP is successful in procuring computing resources, Amazon EC2, as contracted service provider, spins up the requested virtual servers with the necessary software required to execute the job of the end-consumer. SCP can execute the task and once it has been completed, decommission the servers and pay for the usage duration.
- *Service Evaluation.* This post-purchase stage involves the evaluation of the overall satisfaction derived from procuring and consuming the service.

  In our scenario, depending upon whether the negotiation stage was successful, and the service was successfully procured and consumed, SCP rates Amazon EC2, the server procured eg. m1.medium, and the bidding strategy used e.g. price momentum strategy. This feedback can help improve the quality of future decision-making.

C. QoS-Assured Service Provisioning

Given the dynamic nature of the service environment, adaptive management becomes necessary to handle any deviations from expected outcomes or behaviour in any phase of the service provisioning process. For example, the user requirements may change either in the *Service Selection* phase or in the *Service Consumption & Monitoring* phase. Similarly, the service provider might use a more conservative strategy in the *SLA Negotiation & Contracting* phase making it difficult to reach an agreement, or terminate the service or violate the SLA terms and conditions in the *Service Consumption & Monitoring* phase. Thus, deviations can be triggered by participants on either side of the exchange and an adaptive management system (see Figure 5) has to be able to handle them seamlessly in order to ensure QoS-aware and QoS-assured service provisioning[3].

1) Adapting to Provider Initiated Changes: The service provider can deviate from the expected behaviour in different phases of the service provisioning process. We consider the possible deviations in each phase of the interaction and discuss how the service consumer can adapt to these deviations.

- *Service Consumption & Monitoring.* A deviation from expected behaviour can occur in the *Service Consumption & Monitoring* phase either due to a SLA violation or service failure. If there is a SLA violation, the service consumer can try and re-negotiate a new SLA with the current contracted provider (as shown by the arrow from *Service Consumption & Monitoring* to *Strategy Reselection* and then to *Interaction Model Reselection* on the top half). If this fails, then the service consumer can try *Provider Reselection* and *Service Reselection* respectively.
- *SLA Negotiation & Contracting.* If the consumer realizes during the course of negotiation with the service provider that the negotiation is going to be unsuccessful, it can either adapt/change its decision strategy (*Strategy Reselection*) or the interaction model itself (*Interaction Model Reselection*).
- *Provider Selection.* If the service consumer selects a particular provider as the candidate for *SLA Negotiation & Contracting* but the selected provider refuses to negotiate with the service consumer, the service consumer can try and find an alternative service provider to negotiate with (*Provider Reselection*).
- *Service Selection.* If the service consumer selects a particular service in the Service Selection phase but is unable to find a provider who is willing to offer that service, it has to then find an alternative service which is functionally equivalent to the initial selection (*Service Reselection*).

[3] While we focus on adaptive management of service provisioning from the consumers' perspective in this paper as shown in Figure 5, it should be noted that the same principles apply to provider side management as well.

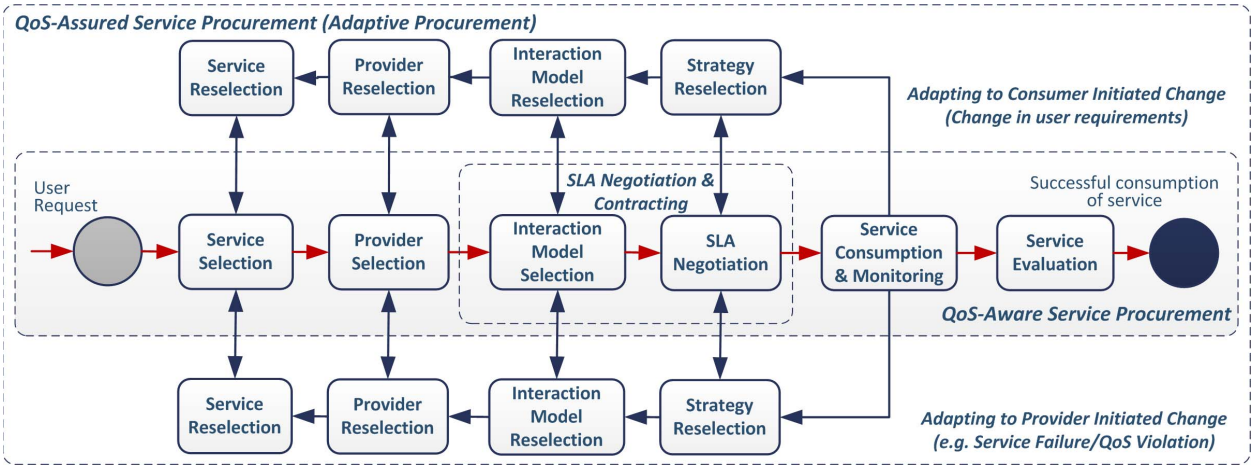


Fig. 5: Adaptive agent behaviour for QoS-Assured service procurement

As an illustrative example, let us consider the scenario where SCP is using a persistent spot request[4] *to process a task. During the task execution, the spot instance can get terminated by Amazon EC2 in the Service Consumption & Monitoring phase due to an increase in the spot price. In such an event, SCP can take a number of different adaptive actions. By default, the spot request is automatically resubmitted, provided the deadline has not expired. Alternatively, if the remaining task completion deadline is not very large, SCP can cancel the persistent request and submit a new spot request with a higher bid price to maximize the chances of it being successful. If the deadline is small, SCP can cancel the persistent request and submit a new on-demand request. As a second illustrative example, let us consider the scenario where SCP submits a spot request to Amazon EC2 but receives a price-too-low status update from Amazon in the Negotiation & Contracting phase. In response, SCP can either submit a new spot request with a higher bid price or submit a new on-demand request.*

2) Adapting to Consumer Initiated Changes: Similar to the service provider, the consumer can also change its requirements and preferences during any stage of service provisioning. We discuss here how the service consumer can adapt to these changes.

- *Service Consumption & Monitoring.* If the consumer changes its QoS requirements during the service consumption phase, the first adaptive action that can be taken is to try and renegotiate the terms and conditions with the currently contracted provider. This can be done either by using the previously used interaction protocol with the same strategy or a different strategy (as shown by the arrow from *Service Consumption & Monitoring* to *Strategy Reselection* on the top half), or with a different interaction protocol (*Interaction Model Reselection*). If that fails, it can try to negotiate with a different provider (*Provider Reselection*) or find a replacement service which can satisfy the updated consumer requirements (*Service Reselection*).
- *SLA Negotiation & Contracting.* If the QoS requirements change during the *SLA Negotiation & Contracting* phase, the consumer can try to accommodate the changes using the current strategy. If that fails, it can try to change the strategy (*Strategy Reselection*) and interaction models (*Interaction Model Reselection*). If that fails as well, it can try to negotiate with a new provider (*Provider Reselection*), or find a replacement service (*Service Reselection*) which can satisfy the updated consumer requirements.
- *Provider Selection.* If the consumer changes its preferences about the provider, then it may be necessary to redetermine the consideration set of providers based on the updated constraints and preferences (*Provider Reselection*).
- *Service Selection.* If the consumer changes its preferences over the requested services, then the consideration set of services which can fulfil the updated consumer requirements may have to be redetermined (*Service Reselection*).

As an illustrative example, let us consider the scenario where SCP receives a request with a long deadline. Because the deadline is long, SCP submits a persistent spot request with a relatively low bid price and executes the task as a long-running interruptible process on the spot instances procured from Amazon EC2. While SCP is processing the task, the consumer decides to change the deadline and make it smaller leaving lesser time to complete the task. Based on the updated deadline, SCP can either continue to use spot instances or cancel the current spot request and submit an on-demand request to ensure that the task is completed within the new deadline. The adaptive mediation applied in this example is to cancel the persistent spot request and submit an on-demand request to ensure quality-assured service provision even under changed user requirements.

IV. Policy-aware Agent Framework

The adaptive management capabilities discussed in Section III can be realized by using software agents. In order to enable adaptive management, software agents need to possess the *ability to externally communicate* with other participants, the *ability to internally reason* about the best course of

[4] A persistent spot request remains active until it expires or the user explicitly cancels it. So if a spot instance gets terminated due to an increase in spot price and the spot request is still active, it automatically gets resubmitted to Amazon.

action to take, and the *ability to coordinate* these two sets of behaviours [9]. These capabilities are realized through two sets of behaviours and one coordination mechanism. The *communicative behaviour*[5] determines how the management agent interacts with other agents or entities, and the *strategic behaviour* defines the internal reasoning capabilities of the management agent. In our framework, we use state-based formalism to model the communicative behaviour of the agents, and policy-based formalism to model the strategic behaviour. Combining these two formalisms enables the loose coupling that is necessary for adaptive management.

In this Section, we first present the different types of policies that are used by our management agent to realize its internal decision-making. We then present a reference architecture for our adaptive management agent and a proof-of-concept prototype implementation.

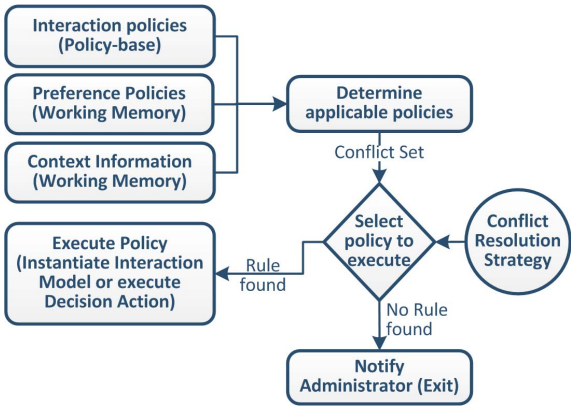
A. Agent Policies

The decision-making behaviours of our policy-aware agents are modelled using the following two types of policies:

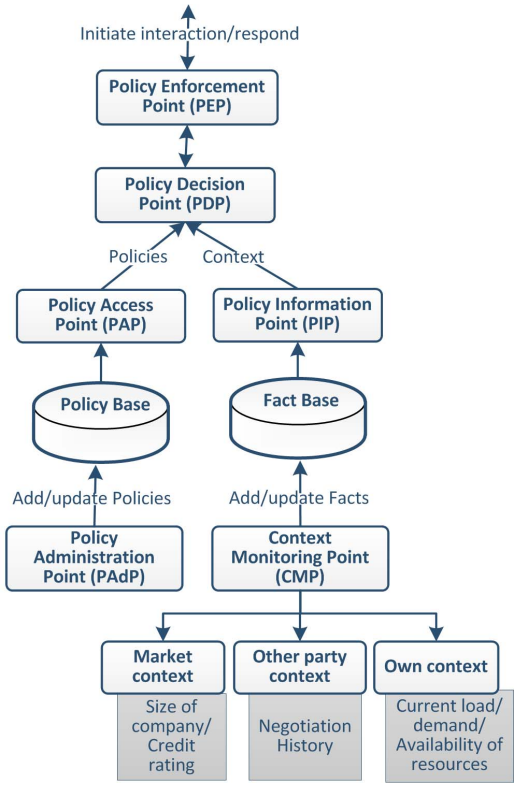
- *Preference policies.* These are essentially directives from human administrators about **what** they want. Policy authors can use them to specify their non-functional requirements and capabilities and preferences and constraints over them. An intelligent agent can reason about these preference statements and use them to make decisions in the *Service Selection*, *Provider Selection*, *Negotiation & Contracting* and *Consumption & Monitoring* phases. The preference policies allow policy authors to define different types of assertions over the QoS requirements and capabilities. More details about the formalism of preference policies are available in [8].
- *Interaction policies.* These are essentially directives about **how** the agents should go about trying to achieve the **what** defined using preference policies. Rather than defining a *single best strategy*, these policies capture a number of different interaction protocols, strategies and tactics that the agent can use to achieve its objectives. Interaction policies can be defined at two levels. At the *macro level*, they provide a high level guidance on which interaction protocols to use with which decision strategy under different context conditions. At the *micro level*, they capture the specific tactics to be used in the different phases of SLA negotiation when using specific interaction protocols under different context conditions. Our model makes use of *condition-action policies* where the condition part captures the interaction context and the action part specifies the interaction protocol, strategy or tactic to be used. More details about the formalism used are available in [8].

B. Policy Processing Model

The policy-aware agent needs the capabilities to reason about the directives it receives from its human administrator in the form of policies. In our framework, the agent uses an embedded reasoning engine that is based on the policy-processing model from XACML[6] [11]. We extend it to enable two main functionalities – coordination and orchestration of the different phases of service provisioning, and atomic management of each individual phase. Figure 6a shows how the policy engine can select the best decision action in each phase of service provisioning. We have presented our policy-



(a) Context-driven selection of decision action



(b) Reference Architecture of Reasoning Component

Fig. 6: Policy Processing Model

[5]The communicative behaviour of agents has to conform to the public interaction protocol that they use to interact with other participants. For e.g. in order to purchase spot instances from Amazon EC2, SCP has to follow the spot request interaction protocol

[6]The original XACML framework is an authorization and access control framework mainly targeted towards web-based systems that need to control access to service resources from incoming HTTP requests.

based orchestration model and mechanism for automated SLA establishment in [9] and [10].

At a high-level, the reasoning engine comprises of three main components:

- *Policy base* - The policy base captures domain knowledge specified by human administrators in the form of preference and interaction policies.
- *Fact base* - The fact base is also referred to as the working memory. It contains relevant context information that can influence the reasoning process.
- *Inference Engine* - The inference engine applies the policies in the policy base to the facts in the fact base and infers the appropriate actions to be taken. Figure 6a shows how it does this in two-phases.

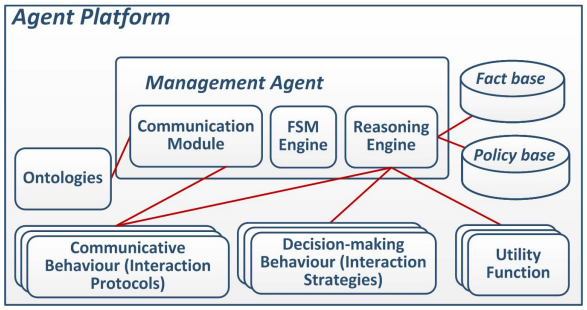
At a more granular level (see Figure 6b), the policy engine consists of the following components:

- *Policy Enforcement Point (PEP)* - This is the main decision enforcing component since it executes the decision actions selected by the Policy Decision Point (PDP).
- *Policy Decision Point* – PDP is the core decision-making component of the policy processing model. It evaluates the current context against applicable policies to select appropriate actions in each phase of the service provisioning process.
- *Policy Access Point (PAP)* - PAP makes available to the PDP all the interaction policies that are in the policy database.
- *Policy Information Point (PIP)* - PIP is the component responsible for retrieving all the relevant contextual information that can influence the decision-making. The context information can include information about the service entity, the other entity involved in the interaction, the market conditions etc.
- *Policy Administration Point (PAdP)* - PAdP, as the name suggests, is the component that is used to manage the policies that are used and evaluated by the PDP. In general, it enables the authoring, deployment and change management of the policy base.
- *Context Monitoring Point (CMP)* – CMP is the component which is responsible for updating the working memory with up to date context information to enable more relevant decision-making.

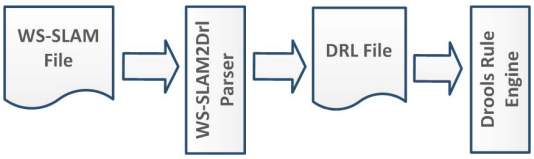
C. Agent Architecture

The reference architecture for the management agent (Figure 7a) has the following core components:

- *Communication Module* – The agent may have to communicate with other agents or with non-agent services e.g. Amazon EC2. If communicating with service providers like Amazon EC2, it uses the API provided by EC2 to communicate with it. If communicating with other agents, it uses appropriate communication ontologies.
  - *FSM Engine* – The communicative behaviour of the management agent is modelled as a Finite State Machine (FSM) and realized using an embedded FSM engine.
- *Reasoning Module* – The reasoning module of the agent comprises of the following:



(a) Reference Agent Architecture



(b) Parsing WS-SLAM rules to Drools

Fig. 7: Smart Cloud Purchaser Agent

  - *Policies* – The management of service provisioning involves two aspects – *global coordination* of the entire process from service selection through to service evaluation, and the *atomic management* of each individual phase of the process, both of which are achieved using policies. The management agent has a set of policies that guide its decision-making. They define what actions the agent needs to take in the different states of the process under different conditions. These policies can refer to externally defined modules including utility functions, interaction strategies and tactics.
  - *Fact base* – The management agent has a working memory where it saves all the context information that it gathers during the service provisioning process. This context information is used for decision making.
  - *Reasoning Engine* – The agent has an embedded reasoning engine which enables it to reason about the interaction context and determine appropriate actions to be taken in the different phases of the service provisioning process.

D. Prototype Implementation

In order to validate our proposed framework for adaptive management of the service provisioning process, we have implemented a proof-of-concept prototype called Smart Cloud Purchaser (SCP) for procuring computing resources from Amazon EC2. We have implemented SCP using the Java Agent DEvelopment (JADE) Framework[7] which is a FIPA-ACL compliant JAVA based framework for building autonomous software agents. The overall service provisioning process is modelled as a finite state machine (FSM). Similarly, the interaction protocols used to interact with Amazon EC2 in the *SLA Negotiation & Contracting* phase i.e. the on-demand interaction protocol (ODIP) and the spot request interaction

[7]http://jade.tilab.com

protocol (SRIP)[8] are also modelled and implemented as FSMs. Each state in the interaction protocol corresponds to a phase that the SLA goes through e.g. *Pending Submission*, *Pending Evaluation*, *Pending Fulfilment* and *Fulfilled*. The transitions between these states are triggered by message exchanges between SCP and Amazon EC2.

JADE uses the concept of behaviours to represent tasks that an agent can accomplish. It supports both simple and complex behaviours and the complex scheduling of behaviours. The `FSMBehaviour` is a composite behaviour which executes its children behaviours according to a pre-defined FSM and is consistent with our model for SLA choreography and orchestration. We use a custom JADE-FSM-Engine [14] to configure the different child behaviours within the composite `FSMBehaviour`. Similarly, we reuse the JADE `DataStore` to store context information that is shared between the different behaviours (corresponding to states in the interaction protocol FSM).

We use the Drools rules engine[9] to implement the reasoning module of the SCP agent. It evaluates the interaction policies against the context information to select appropriate actions in each phase of the service procurement process. We use our custom policy specification language, Web Service - Service Level Agreement Management (WS-SLAM) to write the preference and interaction policies. These policies are parsed into Drools rules using a WS-SLAM2Drools parser (Figure 7b).

V. Related Work

The work presented in this paper draws its motivation from research work presented in [12], which to the best of our knowledge is the first proposal on adaptive management of composite service provisioning process. The main difference between [12] and this work is that in this paper we look at adaptive management of single service provisioning and realize the adaptive management capabilities by integrating agent-based computing with policy-based computing. The second difference between [12] and this work is that we motivate, illustrate and validate our proposed framework using the practical scenario of on-demand resource provisioning on the cloud. While this paper has focussed on the adaptive management of the service provisioning process and discussed how it can be achieved using agent technology and policy-based computing, the policy-based framework and the corresponding models, mechanisms and tools used by the SCP agent have been published in [8][9] and [10]. Similarly, the coordination model and mechanism has been taken from [13], where the authors present an agent-based solution for the coordinated negotiation of SLAs for composite services. We apply the same coordination model to the scenario of on-demand resource provisioning on the cloud since it is ideally suited for managing complex multi-phase interactions.

VI. Conclusion

In this paper we have proposed a novel agent-based framework for the adaptive management of service procurement. We have illustrated through the practical scenario of dynamic resource provisioning on Amazon EC2, how our framework can ensure the QoS-aware and QoS assured provisioning of services. The novelty of our proposal is the use of policy-based computing to realize the internal reasoning & decision making behaviour of the management agent. The use of a policy-based approach for modelling decision-making enables flexible and adaptive decision-making which is necessary for service procurement in diverse, dynamic, volatile and transient environments such as the cloud. While we have focussed on adaptive management of service provisioning from the consumer side and in the context of single-service provisioning, our framework can easily be extended to provider side management of service provisioning and can also be used for adaptive management of composite service provisioning.

[8]Details of the spot request lifecycle are available at http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-bid-status.html

[9]http://www.jboss.org/drools

VII. Acknowledgement

This work is partially supported by the Australian Research Council (ARC) Discovery Project DP110103671 grant.

References

[1] H. E. Schaffer, *X as a service, cloud computing, and the need for good judgment*. In IT professional, 11(5), (pp. 4-5) (2009)

[2] M. P. Papazoglou, P. Traverso, S. Dustdar and F. Leymann, *Service-oriented computing: a research roadmap*. In International Journal of Cooperative Information Systems, 17(02), 223-255 (2008)

[3] C. Weinhardt et. al, *Cloud computinga classification, business models, and research directions*. In Business & Information Systems Engineering, 1(5), (pp. 391-399) (2009)

[4] S. Chichin, M. Baruwal Chhetri, Q. B. Vo, R. Kowalczyk and M. Stepniak, *Smart Cloud Marketplace-Agent-Based Platform for Trading Cloud Services*. In Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on (Vol. 3, pp. 388-395). IEEE (2014)

[5] M. Bichler, A. Gupta and W. Ketter. *Research commentary-designing smart markets*. In Information Systems Research, 21(4), 688-699 (2010)

[6] R. H. Guttman, A. G. Moukas and P. Maes, *Agent-mediated electronic commerce: A Survey*, The Knowledge Engineering Review, Vol. 13, No. 02, 147-149 (1998)

[7] M. He, N. R. Jennings and H. F. Leung, *On agent-mediated electronic commerce*, In Knowledge and Data Engineering, IEEE Transactions on, 15(4), 985-100(2003)

[8] M. Baruwal Chhetri, Q. B. Vo and R. Kowalczyk, *AutoSLAM - A policy–based framework for automated SLA establishment in cloud environments*, In Concurrency and Computation: Practice and Experience, 2013.

[9] M. Baruwal Chhetri, Q. B. Vo and R. Kowalczyk, *Adaptive AutoSLAM-Policy-Based Orchestration of SLA Establishment*. In International Conference on Services Computing (SCC), IEEE, 2014, pp. 472-479.

[10] M. Baruwal Chhetri, Q. B. Vo and R. Kowalczyk, *Supporting Temporal Aspects of SLA Establishment in Auto SLAM Framework*. In IEEE International Conference on Services Computing (SCC), IEEE, pp. 234-241, 2015.

[11] T. Moses et al., *Extensible access control markup language (xacml) version 2.0,* Oasis Standard, vol. 200502, 2005.

[12] R. Kowalczyk and M. Baruwal Chhetri, *Agent Enabled Adaptive Management of QoS Assured Provision of Composite Services*, In Cybernetics and Systems: An International Journal, 40(2), (2009), pp. 68-84.

[13] M. Baruwal Chhetri et. al., *A coordinated architecture for the agent-based service level agreement negotiation of web service composition*. In Australian Software Engineering Conference, IEEE, 2006, pp. 10-18

[14] S. K. Goh, M. Baruwal Chhetri, and R. Kowalczyk, *JADE-FSM-engine: A deployment tool for flexible agent behaviours in JADE*, In Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IEEE Computer Society, 2007, pp. 524-527.

[15] B. Iyer, J. Freedman, M. Gaynor, and G. Wyner, *Web services: enabling dynamic business networks*, The Communications of the Association for Information Systems, vol. 11, no. 1, p. 38, 2003.