Architecting End-to-End Convergence of Cloud Services: An Agent-Based Approach

Djamel Benmerzoug
Department of Software Technologies and Information Systems
Faculty of New Technologies of Information and Communication
University Constantine 2, Algeria
benmerzoug@gmail.com

ABSTRACT: *Cloud computing has been seen as a promising opportunity to improve enterprise's revenues. With the emergence of Cloud computing, applications are moving away from ownership-based programs to Web delivered hosted services. Integrating and outsourcing business processes to Cloud computing services necessitate a uniform description format that facilitates the design, customization, and composition. In this context, Agent Interaction Protocols (AiP) are a useful way for structuring communicative interaction among business partners, by organizing messages into relevant contexts and providing a common guide to the all parts.*

The challenges that Cloud computing poses to business processes integration, emphasize the need for addressing two major issues: (i) which integration approach should be used allowing an adequate description of interaction aspects of the composed software components? (ii) how are these interaction descriptions stored and shared to allow other software artifacts to (re)use them?

To address these issues, we proposed an AiP-based approach for reusing and aggregating existing protocols to create a new desired business application [6][2]. In this paper, we develop an agent-based architecture that supports composition and flexible scaling of services in a virtualized Cloud computing environment. The main goal of the proposed architecture is to address and tackle interoperability challenges at the Cloud application level. It solves the interoperability issues between heterogeneous Cloud services environments by offering a harmonized API. Also, it enables the deployment of applications at public, private or hybrid multi-Cloud environments.

Keywords: Agent Interaction Protocols, Hybrid Cloud, Agent Based Architecture

Received: 18 May 2013, Revised 29 June 2013, Accepted 7 July 2013

1. Introduction

Modern enterprises face a strong economical pressure to increase competitiveness, to operate on a global market, and to engage in alliances of several kinds. In order to meet the requirements and challenges of participating in such alliances, enterprises must be able to cooperate effectively and efficiently [1] [2]. In this context, service-oriented architectures (SOA) approach andWeb service technologies represent large scale abstractions and a candidate concept for enterprises application integration and collaboration.

SOA is an approach to IT solutions that is driven by the enterprise's needs. IT solutions are delivered using mix-andmatch units

called "*services*" that support the explicitly described needs of the enterprise. Enterprises typically use a single software service to accomplish a specific business task or they may compose several software services to create a value-added distributed service-based application.

Several research efforts have been proposed for enterprise application integration based on SOA. However, a serious limitation of an SOA is that it does not make any assumptions regarding service deployment and leaves it up to the discretion of the service developer to make this deployment choice, which is a daunting task and often leads to failure [26] [34]. In addition, service orchestration and choreography is realized within one individual organization, or a limited number of organizations [37].

To address these serious shortcomings it is normal to turn our attention to Cloud Computing as it describes a new supply, consumption, and delivery model for IT services based on the Internet. Cloud computing is a model for enabling convenient, on demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with a minimal management effort or service provider interaction [33].

Cloud services composition is the ability to integrate multiple services into higher-level applications. This integration necessitates a uniform description format that facilitates the design, customization, and composition. In this context, Agent Interaction Protocols (AiP) are a useful way for structuring communicative interaction among business partners, by organizing messages into relevant contexts and providing a common guide to the all parts.

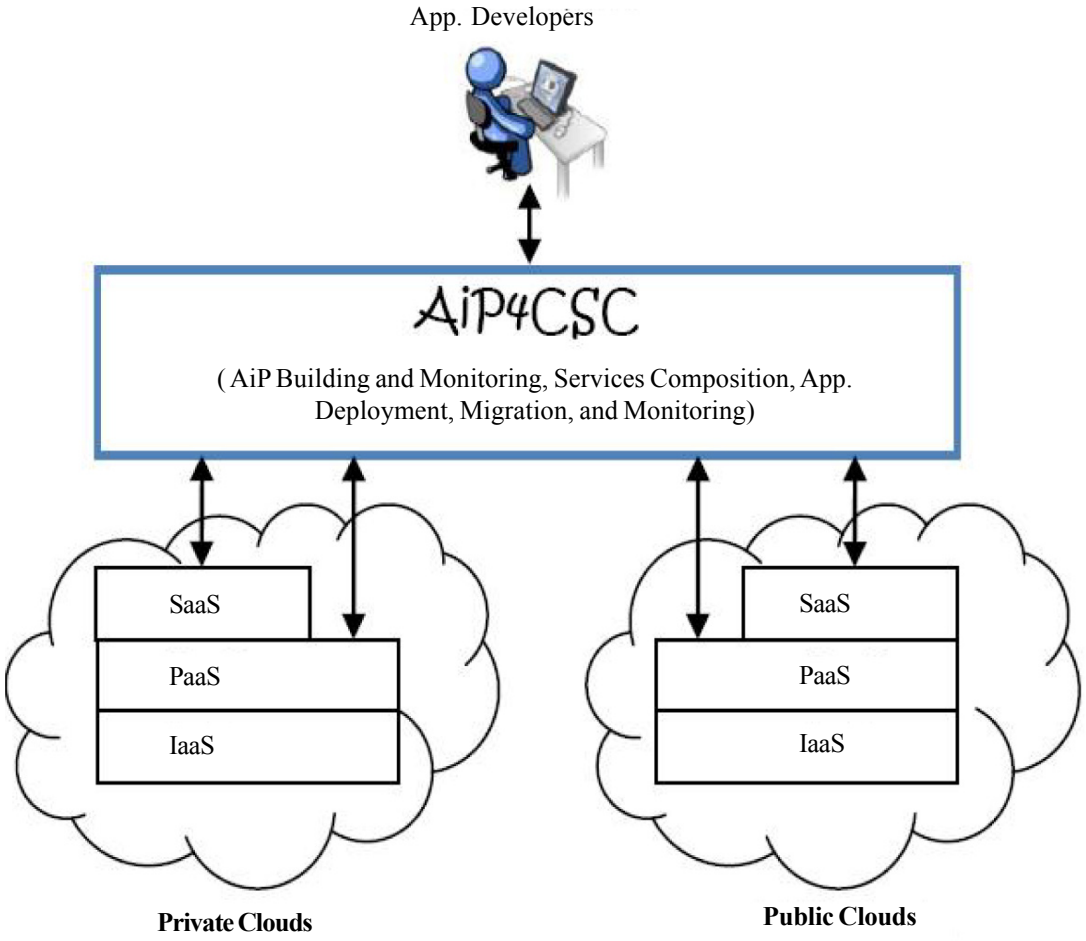

Figure 1. The AiP4CSC: Hybrid multi-Cloud Application Management

It was noted in [5] that AiP are appropriate approaches to define and manage collaborative processes in B2B relationships where the autonomy of participants is preserved. The idea of adopting AiP for managing collaborative processes was first introduced

and proposed in our previous work [5]. Whereas in [7] [8], we demonstrated the practicability of our approach by embedding it in aWeb services language for specifying business protocols, which conducive to reuse, refinement and aggregation of our business protocols. We also elaborated translation rules from interaction protocols notations used in our approach into Colored Petri Nets (CPN). These rules are implemented in IP2CPN: the tool we developed to automatically generate Petri nets from protocols specifications. Resulting Petri nets can be analyzed by dedicated tools to detect errors as early as possible.

To address the collaboration and interaction issues in modern enterprises, it is normal to turn our attention to Cloud Computing as it aims to provide both the economies of scale of a shared infrastructure and a flexible delivery model. In [6], we presented the idea of Cloud Business Protocol, which is a useful way for structuring interaction among Cloud consumers and service providers. In [2], we proposed a basis for a theoretical approach for aggregating protocols to create a new desired business application. The proposed approach provides the underpinnings of aggregation abstractions for protocols, including a set of operators that allows the creation of new value-added protocols using existing ones as building blocks. This research is among the earliest efforts, to the best of the authors' knowledge, in adopting an AiP-based approach for supporting Cloud services composition.

The challenges that Cloud computing poses to services composition, emphasizes the need for addressing two major issues: (i) which composition approach should be used allowing an adequate description of interaction aspects of a composed service ? (ii) how are these descriptions stored and shared to allow other software artifacts to (re)use them ? Driven by the motivation of reuse, we considered AiP as modular components, potentially composed into additional protocols, and applied in a variety of business processes. By maintaining repositories of commonly used, generic, and modular protocols, we can facilitate the reuse of a variety of well-defined, well-understood and validated protocols. In this present research, an agent-based approach for Cloud services composition is proposed.

Our main contribution is the AiP4CSC (AiP for Cloud Services Composition) middleware layer, between the Cloud service offerings and the Cloud-based applications developers. The approach introduces an agent-based architecture (see figure 1) whose main goal is to address and tackle interoperability challenges at the Cloud application level. The AiP4CSC architecture provides the necessary services to handle public, private and hybrid deployment models. One advantage of the AiP4CSC architecture is that it is based on the idea of reducing the complexity involved when developing a composite application. In AiP4CSC, we can maintain a protocols repository to facilitate reusing of previously used protocols and composition routines in the future. The proposed architecture is largely based on our previous work [2] [6].

The remainder of the paper is organized as follows: Section 2 introduces some key concepts and terminology. Section 3 overviews some related work. In Section 4, we present an agent-based architecture for dynamic Cloud services composition. Finally, Section 5 concludes this paper and presents future directions.

2. Terminology

2.1 Cloud Computing Models

According to the intended access methods and availability of Cloud computing environments, three major types of Cloud deployments are known: public Clouds, private Clouds, and hybrid Clouds [29].

Private Cloud: In this model, the Cloud infrastructure is exclusively used by a specific organization. The Cloud may be local or remote, and managed by the enterprise itself or by a third party.

Public Cloud: Infrastructure is made available to the public at large and can be accessed by any user that knows the service location.

Hybrid Cloud: Involves the composition of two or more Clouds. These can be private or public Clouds which are linked by a proprietary technology that provides portability of data and applications among the composing Clouds.

The increasing adoption rate of Cloud computing is currently driving developers, integrators and hosting enterprises to take Cloud computing into account. However, enterprises are driven by different reasons to maintain their own data center, such as legislation of storing data in-house, investments in the current infrastructure, or the extra latency and performance requirements. This drive is supported by the fact that enterprises have already invested heavily in their own private server equipment and software [19].

Consequently, we believe that a hybrid approach makes more sense for enterprises. This approach allows one to use the internal infrastructure combined with public Cloud resources to build a hybrid Cloud. For example, on one hand, critical applications can run on the infrastructure/platform of the private data center, and, on the other hand, the public Cloud can be used as a solution to manage peak demands or for disaster recovery.

2.2 Agent Interaction Protocols

Agent Interaction Protocols (AiP) are specific, often standard, constraints on the behaviours of the autonomous agents in a multiagent system. AiP are essential to the functioning of open business systems, such as those that arise in most interesting Web based application. Applied to Cloud computing, we propose the following variation: *An AiP is two or more business parties linked by the provision of Cloud services and related information*.

In fact, business parties in the Cloud computing area are interconnected by the AiP. These parties are involved in the end-to-end provision of products and services from the Cloud service provider for end Cloud customers.Because protocols address different business goals, they often need to be composed to be put to good use. For example, a process for purchasing goods may involve protocols for ordering, shipping, and paying for goods.

Driven by the motivation of reuse, we would like to treat protocols as modular components, potentially composed into additional protocols, and applied in a variety of business processes. By maintaining repositories of commonly used, generic, and modular protocols, we can facilitate the reuse of a variety of well-defined, well-understood, and validated protocols.

In this work, we present an agent-based architecture that supports our previous work [6] [2]. The proposed architecture has been designed to enable interoperability and cross-Cloud application management.

3. RelatedWork

This section reviews the literature related to Cloud services management and composition. Specifically, we reviewed the existing Cloud computing architectures that deal with Cloud computing interoperability. Moreover, we reviewed some agent-based approaches for enabling enterprise application integration and collaboration.

3.1 Cloud Computing Based Approaches

Today, large technology vendors as well as open-source software projects both address the hybrid Cloud market and are developing virtual infrastructure management tools to set-up and manage hybrid Clouds [19].

The Cafe project [22] provides a relevant approach for Cloud-based SaaS development, which offers an ad-hoc composition technique for application components and Cloud resources following the service component architecture.

However, this approach requires SaaS developers to possess deep technical knowledge of the application architecture and the physical Cloud deployment environment to select and compose the right application components and Cloud resources.

In [21], a systematic process for developing high-quality Cloud SaaS has been proposed, taking into considerations the key design criteria for SaaS and the essential commonality/variability analysis to maximize the reusability.

In [36], authors use a proof-of-concept hybrid multi-Cloud scenario to demonstrate the multi-PaaS application management solution developed by the Cloud4SOA project. Cloud4SOA introduces a broker-based architecture whose main goal is to address and tackle semantic interoperability challenges at the PaaS layer. The architecture is equipped with management and monitoring services providing the appropriate flexibility to handle public, private and hybrid deployment models.

The Reservoir architecture [30] aims to satisfy the vision of service oriented computing by distinguishing and addressing the needs of service providers and infrastructure providers. Service providers interact with the end-users, understand and address their needs. They do not own the computational resources; instead, they lease them from infrastructure providers which interoperate with each other creating a seamlessly infinitive pool of IT resources.

The VMware workstation [10] offers live migration of virtual appliances and machines between data centers and allows service providers to offer IaaS while maintaining compatibility with internal VMware deployments.

HP [14] provides three offerings for hybrid Cloud computing: HP Operations Orchestration for provisioning, HP Cloud Assure for cost control, and HP Communications as a Service for service providers to offer small businesses on-demand solutions. The Cloud-based HP Aggregation Platform for SaaS streamlines operations for both the service provider and the businesses customer by automating processes such as provisioning, activation, mediation charging, revenue settlement and service assurance.

Model-driven approaches are also employed for the purpose of automating the deployment of complex IaaS services on Cloud infrastructure. For instance, in [23], authors propose the mOSAIC Ontology and the MetaMORP(h)OSY methodology for enabling model driven engineering of Cloud services. The methodology uses model driven engineering and model transformation techniques to analyse services. Due to the complexity of the systems to analyse, the mOSAIC Ontology is used in order to build modelling profiles in MetaMORP(h)OSY able to address Cloud domainrelated properties.

When examining the Cloud based approaches we observe a recurrent theme. They do not allow for easy extensibility or customization options. Better ways are necessary for Cloud service consumers to orchestrate a cohesive Cloud computing solution and provision Cloud stack services that range across networking, computing, storage resources, and applications from diverse Cloud providers.

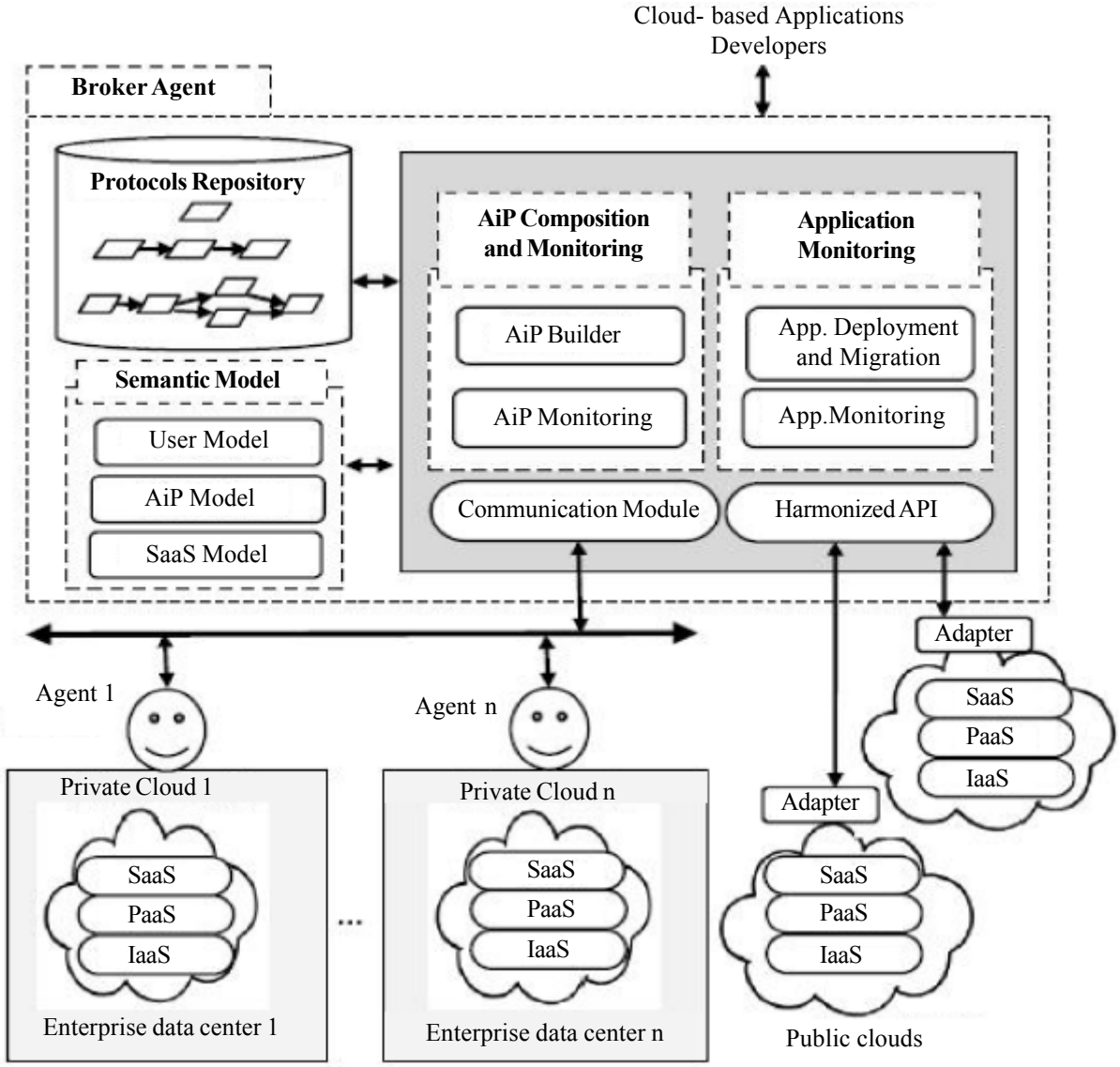

Figure 2. The Proposed Agent-Based Architecture

The work presented in this paper is considered as a first step toward AiPaaS (AiP as a Service). The AiPaaS approach is based on the idea of reducing the complexity involved when developing a composite application. In our work, the knowledge obtained during Cloud services composition is stored, shared, and reused. In fact, we proposed a basis for a theoretical approach for reusing and aggregating existing protocols to create a new desired business application. The proposed approach provides the underpinnings of aggregation abstractions for protocols. Our approach provides a set of operators that allows the creation of new value-added protocols using existing ones as building blocks.

3.2 Agent Based Approaches

Multiagent systems are a very active area of research and development. In fact, several researchers are working at the intersection of agents and collaborative enterprise systems.

For example, Buhler et al. [11] summarize the relationship between agents and Web services with the aphorism Adaptive Workflow Engines = Web Services + Agents: namely, Web services provide the computational resources and agents provide the coordination framework. They propose the use of the BPEL4WS language as a specification language for expressing the initial social order of the multi-agent system. [11] does not provide any design issues to ensure the correctness of their interaction protocols.

In [18], J. Octavio et al. proposed an agent-based approach to compose services in multi-Cloud environments for different types of Cloud services. Agents are endowed with a semi-recursive contract net protocol and service capability tables (information catalogs about Cloud participants) to compose services based on consumer requirements. However, the agent collaboration is limited to that of the contract net protocol.

Driven by the motivation for reuse of interaction protocols, [35] and [16] consider protocols as a modular abstractions that capture patterns of interaction among agents. In these approaches, composite protocols can be specified with a Protocol Description Language (such as: CPDL[1] or MAD-P[2]). Although formal, [35] and [16] do not not provide any step for the verification of the composite protocols.

Agent-oriented software methodologies aim to apply software engineering principles in the agent context e.g. Tropos, AMCIS, Amoeba, and Gaia. Tropos [9] [28] and AMCIS [4] [3] differ from these in that they include an early requirements stage in the process. Amoeba [15] is a methodology for business processes that is based on business protocols. Protocols capture the business meaning of interactions among autonomous parties via commitments. Gaia [13] differs from others in that it describes roles in the software system being developed and identifies processes in which they are involved as well as safety and liveness conditions for the processes. It incorporates protocols under the interaction model and can be used with commitment protocols.

Our methodology differs from these in that it is aimed at achieving protocol re-usability by separation of protocols and business rules. It advocates and enables reuse of protocols as building blocks of business processes. Protocols can be composed and refined to yield more robust protocols.

4. An Agent Based Architecture for Cloud Services Composition

Service composition in multi-Cloud environments must coordinate self-interested participants, automate service selection, (re)configure distributed services [18], store and share the composite services to allow other software artifacts to (re)use them.

The new challenges that Cloud computing poses to service composition, emphasize the need for the agent paradigm [2] [31] [32] [18]. Multi-agent systems represent a distributed computing paradigm based on multiple interacting agents that are capable of intelligent behavior.

In order for enterprises collaborate to fulfill their requirements, it is important to consider that more than one type of Cloud can be used. However, enterprises are driven by different reasons to maintain their own data center, such as legislation of storing data in-house, investments in the current infrastructure, or the extra latency and performance requirements. This drive is

[1]CPDL: Communication Protocol Description Language
[2]MAD-P: Modular Action Description for Protocols

supported by the fact that enterprises have already invested heavily in their own private server equipment and software [19].

Consequently, we defined two types of agent, namely, the Enterprise Agent representing an individual enterprise, and the Broker Agent, which facilitates the Cloud based application developers in searching for, deploying and governing their business applications on the SaaS offerings that best match their needs (see Figure 2).

4.1 Description of the Broker Agent

The main roles of the Broker agent, which implements the core functionalities offered by the architecture, are the creation, monitoring, and control of AiP life cycle. Its architecture features the following modules:

• **Application Monitoring:** supports the efficient deployment and governance of applications. The developers can manage the life-cycle of their applications as a homogenized way independently of the specific platform offering the application is deployed.

• **AiP Composition and Monitoring:** orchestrate AiP and control the access to them. It receives requests to resolve requirements from applications developers. Then, it handles the requests via their associated AiP. It also provides operations for monitoring interaction (i.e., creating and deleting instances).

• **Communication Module:** contains all the processes required to handle agent to agent communication, such as: reception, filtering, and translation of incoming messages, and formulation and sending of the outgoing messages. Agent to agent communication occurs via FIPA Agent Communication Language [17], where XML is used for the description of the content of the message.

• **Harmonized API:** provides the necessary tools, which enable the management of applications across different Cloud offerings.

• **Protocols Repository:** maintains repository of commonly used and generic protocols. It facilitates the reuse of a variety of well-defined, well-understood and validated protocols. It provides the AiP specification that describes the functionality, input and output of protocols. An AiP is described by the requirement it resolves, and the parameters of the requirement correspond to the input of the AiP. The AiP output is a set of parameters that results from resolving the requirement.

• **Semantic Model:** is the backbone of the architecture and spans the entire architecture, resolving Interaction conflicts by providing a common basis for publishing and searching different SaaS offerings.

4.2 Reusing Historical Composition Experiences

Using a number of various components such as services or types of Clouds can cause the AiP to be complex. Depending on enterprise requirements, one Cloud may not be able to offer the complete service they have requested. Current techniques suggest that enterprises will acquire related tools and perform integration activities locally. The knowledge and expertise to perform composition activities is hard to attain and equally difficult to maintain. Thus, the knowledge obtained during Cloud services composition is not stored, reused, or shared. To be competitive, enterprises must be able to transfer and reuse knowledge attained after each composition scenario.

Consequently, we have proposed the concept of *Agent Interaction Protocols as a Service* (AiPaaS), which aims at reducing the need for a composition infrastructure and allows to compose and deploy existing AiP.

An AiPaaS capability could acquire the previously mentioned protocols as input and suggest a specific protocol to achieve a new specific need. Enterprises can maintain a protocols repository to facilitate reusing previously used protocols and composition routines in the future. Furthermore, the AiPaaS approach can learn from historical composition information to augment future recommendations.

To deal with a AiP composition in an automatic way (i.e. to have mechanisms that automate the AiP building, monitoring and execution), the *AiP Composition and Monitoring module* (Figure 2) offers three main functionalities that enable building and execution of AiP, as well as recover from an AiP violations (Figure 3):

• **AiP Building:** Allows the automatic creation of AiP, based on previous AiP (from the protocols repository) and the semantic description of requirements specified by the Application Developer.

• **AiP Execution:** Maintains a global monitor of the AiP execution.

• **AiP Violation Recovery:** When the AiP Composition and Monitoring module detects an error of the AiP execution (the execution of the business application does not satisfy the AiP), the protocol execution is stopped, and the error is replicated to the application developer.

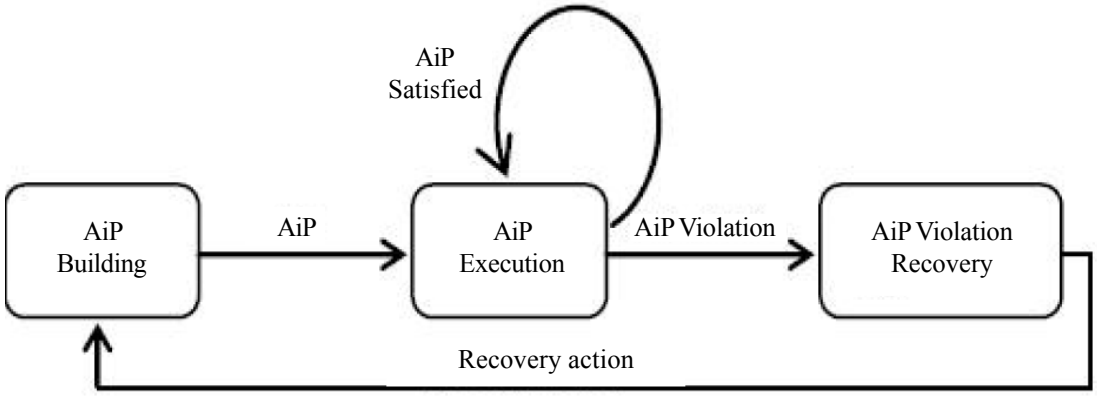

Figure 3. AiP Building - Execution - Recovery Process

4.3 Discussion

We developed a first prototype of the AiP4CSC architecture utilizing different advanced tools like: Java agent development framework [12], WSBPEL[3][20], Windows Azure Cloud platform [24], Windows Communication Foundation [25], and Windows Workflow Foundation [27].

A first test shows that AiP4CSC can provide benefits to developers willing to adopt a hybrid approach, since it allows deploying, governing and monitoring both parts of a hybrid Cloud from the same single place. It provides useful information about the performance of the applications and the fulfillment of AiP and allows bursting an application in case of AiP violation.

Further, the implementation makes us realize that the basic components needed to address service composition in the Cloud are similar to the components in conventional service composition. However, AiP4CSC architecture gives the Cloud-based application developers more chances to get computational services and provide on-demand dynamic service composition. Also, services composition is specified as modular AiP, which conducive to reuse, refinement and aggregation of business protocols. Consequently, the AiP4CSC acquires the previously mentioned protocols as input and suggest a specific protocol to achieve a new specific need. It maintains a protocols repository to facilitate reusing of previously used protocols and composition routines in the future.

5. Conclusion and Future Work

The present research highlights the synergies between Cloud computing and agent paradigm. In such environments, the complexity that matters is not so much in the size of the code through which such entities are programmed but on the number and dynamicity of the interactions in which they will be involved. In this context, AiP are a useful way for structuring communicative interaction among business partners, by organizing messages into relevant contexts and providing a common guide to the all parts.

In this paper we presented an agent based approach for Cloud services composition. First, we presented the idea of AiP, which is a useful way for structuring interaction among business partners. Second, we proposed AiP4CSC : an agentbased architecture for Cloud services composition. The main goal of the proposed architecture is to address and tackle interoperability challenges at the Cloud application level. It solves the interoperability issues between heterogeneous Cloud services environments by offering a harmonized API. Also, it enables the deployment of applications at public, private or hybrid multi-Cloud environments.

[3]WSBPEL: Business Process Execution Language for Web Services

We are currently setting up a testbed that supports Cloud services composition.This will allow us to thoroughly evaluate and tune our agent-based architecture to demonstrate an efficient composition.

References

[1] Antonia Albani, Jan L. G. Dietz. (2009). Current trends in modeling inter-organizational cooperation. *Journal of Enterprise Information Management*, 22 (3) 275–297.

[2] Djamel Benmerzoug. (2013). An Agent-Based Approach for Hybrid Multi-Cloud Applications. Scalable Computing: *Practice and Experience*, 14 (2) 95 – 109.

[3] Djamel Benmerzoug, Mahmoud Boufaida, Zizette Boufaida. (2004). Developing Cooperative Information Agent-Based Systems with the AMCIS Methodology. *In*: IEEE International Conference on Advances in Intelligent Systems: *Theories and Application, Luxembourg*, November. IEEE press.

[4] Djamel Benmerzoug, Mahmoud Boufaida, Zizette Boufaida. (2004). From the Analysis of Cooperation Within Organizational Environments to the Design of Cooperative Information Systems: An Agent-Based Approach. *In*: OTM Workshops, 3292 of LNCS, p. 495–506, Larnaca, Chypre, October. Springer.

[5] Djamel Benmerzoug, Mahmoud Boufaida, Fabrice Kordon. (2007). A Specification and Validation Approach for Business Process Integration based onWeb Services and Agents. *In*: Proceedings of the 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, MSVVEIS-2007, *In conjunction with ICEIS*, p. 163–168. NSTIIC press.

[6] Djamel Benmerzoug, Mohamed Gharzouli, Mounira Zerari. (2013). Agent interaction protocols in support of cloud services composition. *In*: 6th International Conference on Industrial Applications of Holonic and Multi-Agent Systems, V. 8062 of LNAI, p. 293 – 304, Prague, Czech Republic, August. Springer.

[7] Djamel Benmerzoug, Fabrice Kordon, Mahmoud Boufaida. (2008). A Petri-Net based Formalisation of Interaction Protocols applied to Business Process Integration. *In*: Advances in Enterprise Engineering I, 4th International Workshop on Enterprise & Organizational Modeling and Simulation (EOMAS'08), V. 10 of LNBIP, p. 78–92, Montpellier, France, June. Springer.

[8] Djamel Benmerzoug, Fabrice Kordon, Mahmoud Boufaida. (2008). Formalisation and Verification of Interaction Protocols for Business Process Integration: a Petri net Approach. *International Journal of Simulation and Process Modelling*, 4 (3–4) 195–204.

[9] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos. (2004). Tropos: An Agent- Oriented Software Development Methodology. *International journal of Autonomous Agents and Multi-Agent Systems*, 8 (3) 203–236.

[10] Edouard Bugnion, Scott Devine, Mendel Rosenblum, Jeremy Sugerman, Edward Y. Wang. (2012). Bringing virtualization to the x86 architecture with the original vmware workstation. *ACM Trans. Comput. Syst*., 30 (4) 12:1– 12:51, November.

[11] Paul A. Buhler, José M. Vidal. (2005). Towards adaptive workflow enactment using multiagent systems. *International Journal On Information Technology and Management*, 6, 61–87.

[12] Huang, C., Trappey, C., Trappey, A., Ku, C. (2009). The design of a JADE-based autonomous workflow management system for collaborative SoC design. Expert Syst. Appl., 36 (2) 2659–2669.

[13] Luca Cernuzzi, Ambra Molesini, Andrea Omicini, Franco Zambonelli. (2011). Adaptable multi-agent systems: the case of the gaia methodology. *International Journal of Software Engineering and Knowledge Engineering*, 21 (4) 491–521.

[14] David Collins. (2009). Communications as a service for midsize businesses.

[15] Nirmit Desai, Amit K. Chopra, Munindar P. Singh. (2009). Amoeba: A Methodology for Fodeling and Evolving Cross-Organizational Business Processes. *Journal of ACM Trans. Softw. Eng. Methodol*., 19 (2).

[16] Nirmit Desai, Munindar P. Singh. (2007). A modular action description language for protocol composition. *In*: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, p. 962–967. AAAI Press.

[17] FIPA-ACL. (2001). FIPA Communicative Act Library Specification. Technical Report http://www.fipa.org/specs/XC00037/, FIPA - Foundation for Intelligent Physical Agents.

[18] Octavio Gutierrez-Garcia, J., Kwang Mong Sim. (2012). Agent-based cloud service composition. Applied Intelligence, The International *Journal of Artificial Intelligence*, Neural Networks, and Complex Problem-Solving Technologies, 22 (2), September.

[19] Sofie Van Hoecke, Tom Waterbley, Jan Devos, Tijl Deneut, Johan De Gelas. (2011). Efficient management of hybrid clouds. *In*: The Second International Conference on Cloud Computing, GRIDs, and Virtualization, p. 167–172, Rome, Italy, September.

[20] SAP-Siebel Systems IBM, Microsoft. (2003). Business process execution language for web services version 1.1. *Technical report*.

[21] Hyun Jung La, Soo Dong Kim. (2009). A systematic process for developing high quality saas cloud services. *In*: Proceedings of the 1st CloudCom '09, p. 278 Ý U289. Springer-Verlag.

[22] Ralph Mietzner. (2010). A method and implementation to define and provision variable composite applications, and its usage in cloud computing. Dissertation, Universit´lat Stuttgart, Germany.

[23] Francesco Moscato, Beniamino Di Martino, Rocco Aversa. (2012). Enabling Model Driven Engineering of Cloud Services by using mOSAIC Ontology. Scalable Computing: *Practice and Experience*, 13 (1) 29–44.

[24] Mackenzie Neil. (2011). Microsoft windows azure development cookbook. Packt Publishing.

[25] Cibraro Pablo, Claeys Kurt, Cozzolino Fabio, Grabner Johann. (2010). Professional WCF 4: Windows Communication Foundation with .NET 4. Wrox Publishing.

[26] Mike P. Papazoglou, Klaus Pohl, Michael Parkin, Andreas Metzger, editors. (2010). Service Research Challenges and Solutions for the Future Internet - S-Cube - Towards Engineering, Managing and Adapting Service-Based Systems, V. 6500 of Lecture Notes in Computer Science. Springer.

[27] Arumugam Paventhan, Kenji Takeda, Simon J. Cox, Denis A. Nicole. (2006). LeveragingWindowsWorkflow Foundation for Scientific Workflows in Wind Tunnel Applications. *In*: Proceedings of the 22nd International Conference on Data Engineering Workshops, ICDE 2006, p. 65. *IEEE Computer Society*.

[28] Loris Penserini, Tsvi Kuflik, Paolo Busetta, Paolo Bresciani. (2010). Agent-based organizational structures for ambient intelligence scenarios. *Ambient Intelligence and Smart Environments*, 2 (4) 409–433.

[29] Peter Mell, Tim Grance. (2009). The NIST Definition of Cloud Computing.

[30] Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I. M., Montero, R., Wolfsthal, Y., Elmroth, E., Cáceres, E., Ben-Yehuda, M., Emmerich, W., Galán, F. (2009). The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53 (4) 535–545, July.

[31] Wang, S., Shen, W., Hao, Q. (2006). An agent-based web service workflow model for inter-enterprise collaboration. Expert Syst Appl, 31 (4) 787-799.

[32] Kwang Mong Sim. (2011). Agent-based cloud computing. *IEEE Trans Serv Comput*.

[33] Subashini, S., Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *J. Network and Computer Applications*, 34 (1) 1–11.

[34] Yehia Taher, Dinh Khoa Nguyen, Francesco Lelli, Willem-Jan van den Heuvel, Mike P. Papazoglou. (2012). On Engineering Cloud Applications - State of the Art, Shortcomings Analysis, and Approach. Scalable Computing: Practice and Experience, 13 (3) 215-231.

[35] Vitteau, B., Huget, M.-P. (2004). Modularity in interaction protocols. In Advances in Agent Communication, V. 2922 of LNCS, p. 291-U309. Springer.

[36] Dimitris Zeginis, Francesco D'Andria, Stefano Bocconi, Jesus Gorronogoitia Cruz, Oriol Collell Martin, Panagiotis Gouvas, Giannis Ledakis, Konstantinos, A. Tarabanis. (2013). A user-centric multi-PaaS application management solution for hybrid multi-Cloud scenarios. *Scalable Computing: Practice and Experience*, 14 (1) 17–32.

[37] Jiehan Zhou, Kumaripaba Athukorala, Ekaterina Gilman, Jukka Riekki, Mika Ylianttila. (2012). Cloud architecture for dynamic service composition. *IJGHPC*, 4 (2) 17–31.