Inf Syst E-Bus Manage (2017) 15:489–534
DOI 10.1007/s10257-016-0330-y

ORIGINAL ARTICLE

A novel programming framework for architecting next generation enterprise scale information systems

D. Venkatesan[1] · **S. Sridhar**[2]

Received: 2 June 2015 / Revised: 26 August 2016 / Accepted: 29 August 2016 /
Published online: 17 September 2016

Abstract The increasing popularity and usage of internet based services makes design of software system complex and their power unlimited in scale. These systems often involve heterogeneous environment and platforms. They continuously evolve in order to meet the changes in technology and business practices. Emergence of information system architecture based on disparate externally provided internet based services warrant incorporation of novel design primitives in the application design. Traditionally popular application design primitives includes separation, abstraction, compression, uniform composition, replication and resource sharing that has evolved and came to vogue based on past experience and expert practice. This work examines and evaluates approaches to incorporation of new and novel unit operations around which applications can be architected, designed and analyzed for the internet computing and big data era. Incidentally it is found that *portions* of agent technology provides several unit operations such as migration of code and speech-act based responsibility delegation/application composition as application architectural primitives. This work determines extensions to Web Services-Business Process Execution Language (WS-BPEL) programming framework—called as ACtive Internet Application Framework (ACIAF) and is suited for incorporation of novel unit operations such as migration of code and speech-act based component orchestration/application composition to enable construction of next generation information systems. An analysis of merits and demerits of using

✉ D. Venkatesan
d.venkatesan@aubit.edu.in

S. Sridhar
ssridhar@annauniv.edu

[1] Department of Information Technology, Anna University, BIT Campus, Tiruchirapalli 620024, Tamilnadu, India

[2] Department of Information Science & Technology, Anna University, Chennai, Tamilnadu, India

WS-BPEL technology to realize ACIAF is also carried out here. Code mobility feature of ACIAF is demonstrated using a use case while loosely coupled application design using ACIAF is theoretically discussed. Formal analyzability of mobility behavior of ACIAF application is highlighted based on its conformance to Petri-net formalism. Further formal logical analyzability/interpolatability of individual actions performed by collaborating ACIAF applications is also highlighted. It is argued here that ACIAF shall provide necessary foundation, guidance and motivation for further detailed technical work on several ideas presented to realize next generation information systems. This work firmly establishes extension to WS-BPEL platform as a most viable approach to realize novel architectural primitives to the design and implementation of next generation information systems.

Keywords Application architecture · Work flow · Migratable web service · Data processing · Loosely coupled system design · Programming framework · WS-BPEL

1 Introduction

Modern information system architects tend to adopt paths that facilitate leveraging ever increasing list of externally provided services using workflow metaphor (to specify the flow of business functionality) and remain platform independent. The heterogeneous and distributed nature of this environment warrant architects to incorporate necessary quality attributes such as performance, dependability and scalability into the information system design from very early phases of its conception (Barbacci et al. 1995). This will ensure satisfactory performance of the resulting software applications. Smith (2015) argues even today much work need to be done to incorporate principles and patterns for designing responsive systems. He quotes over the past 40 years (1979, 1995, and 2015) success rate of our ability to build responsive systems were very low. It implies huge wastage of human effort and resources. Hence there is a need to find suitable principles and patterns for designing responsive high performance systems.

Traditionally software systems are designed based on six unit operations namely separation, abstraction, compression, uniform composition, replication and resource sharing (Kazman and Bass 1994). These unit operations formed architectural basis in the design of information systems that has desirable quality attributes such as scalability, separation, modifiability, integrability, portability, performance, reliability, ease of creation and reusability (Barbacci et al. 1995). Architectural primitives themselves is not of much use if the designers fail to leverage and use it appropriately. In addition underlying technology should enable seamless incorporation of architectural primitives in system design. After a long journey, software industry has reached at web services technology that provides technical means (through Web Services technology) to impart several quality attributes to systems (such as separation of implementation from definition to offer loose coupling, platform-independence and inter-operability). Web services technology also enables architect applications to easily incorporate architectural primitives in system design. This lead to realization of several important system quality attributes (Milanovic 2011). Web services technology provides necessary

abstraction for platform independent composition of underlying application capability. It enables creation of information systems and applications by composing individual discrete services running on disparate platforms. A popular web services composition framework is WS-BPEL (WS-BPEL 2007; Havey 2005). WS-BPEL is an industrially successful work flow metaphor based application composition language/framework that depends entirely on web services and extensible markup language (XML) technology. It transparently adopt to rapid changes in underlying computing environment and technology due to embracement of XML technology. WS-BPEL based applications have sound formalism based on petri-net/process calculus and are visually and graphically specifiable. It remains structurally similar to traditional work flow based information system design paradigm and very intuitive. Traditional business process models can be easily translated into WS-BPEL programs and verification and validation is smooth due its similarity with real life business flow. These merits make WS-BPEL a suitable candidate for further enhancement and innovation to adapt it to rapid changes in computing environment.

As WS-BPEL technology solved several traditional problems in application development, it still has a few issues to tackle with. This is mainly due to the changes taking place to its environments such as emergence of social media that lead to data explosion and necessity of applications to deal-with an ever increasing list of externally provided services that are nascent. This makes reliable orchestration and partnership with externally provided services and consuming large quantity of data difficult.

In the past vast majority of the information systems generally deal with small quantity of data that could be easily transported to the application site and processed over the network. Due to changes in the computing environment and its scale, applications architects and designers faces new challenges in designing modern information systems. Architects/Designers are forced to search for novel unit operations that enable design and architecture of next generation of modern information systems in the changed environment (Barbacci et al. 1995; Kazman and Bass 1994). Hints are already evident: do not move around large volume of data but instead move the application, choose and associate with partner services on the fly at run time rather than statically at the design time. This makes sound business sense in meeting constraints such as price, performance and quality requirements. The search for novel unit operations to compose system is constrained by the fact that they should not deviate from traditional work-flow based application architecture that remain similar to business activity model. Another problem faced by application architects in composing application based on externally provided services is its complete dependence on WSDL (Web Services Description Language) files at the design time. It can break the applications at run time when the referenced WSDL based services are no longer available due to reasons such as version promotion, changes to business or migration of service location/address. This gives rise to an important problem of web services substitution in orchestrated applications. Speech act (FIPA-ACL 1998; Covington 1997) based services composition (Venkatesan and Sridhar 2016)–that uses simplest of communication primitives such as “ask” and “tell”, obviates direct dependence on WSDL files and introduces message based indirection where any compatible web services can be invoked by a middleware responsible for message

routing (Okouya et al. 2013; Venkatesan and Sridhar 2016). Thus speech act based services composition provides an important solution to services discovery and substitution making resulting applications robust. Hence it is proposed to imbibe these two unit operation (i.e. code migration and separation via loose coupling) capabilities into WS-BPEL framework. The resulting method and programming language technology is proposed to be added to WS-BPEL language as an extension and is called as Active Internet Application Framework (ACIAF) here. It may be noted that several such operations can be added to WS-BPEL framework and made part of ACIAF. ACIAF is called as a framework because it captures some important design decisions (and their implementation) that are common to a group of similar applications and permit its reuse in creating several other such applications (Gamma et al. 1995). The design constructs supplied by ACIAF can be reused by designers and programmers to realize solutions applicable in their problem context. ACIAF as a framework (for now) consists of a proposal for implementation of a system of design patterns namely migration and speech act based composition.

The search for novel unit operations and a sound underlying theory of them lead us to agent technology. Agent technology (AT) provides several unit operations such as migration of code, speech act based loose coupling, autonomy, and rationality that can be used to architect new class of information systems (Weiss 2012; Bradshaw 1997). One crucial technical difference between traditional application technologies (such as modular or component or service oriented) and agent technology is in how they treat and invoke collaborating partners interfaces (i.e. delegation of capability). *AT uses indirect - speech act- message based coupling of capabilities across collaborating components* (i.e. *they do not hard code the collaborating component's interface*) *whereas traditional applications uses direct specification and invocation of interfaces of collaborating components.* Thus traditional agent applications are inherently and genuinely loosely coupled. Traditional agent technology has strong roots in artificial intelligence and in fact positioned as a means of realizing many goals of artificial intelligence (Russell and Norvig 2003). Agent technology is very vast and its progress is stymied by its attempt to tackle several issues at the same time (Shehory 2014; Logan 2015). A sound wisdom taught by the discipline of software engineering is to use incremental approach in system construction: introduce new notions one by one until they are fully imbibed into the main stream software technology. Such examples include introduction of modular, object oriented and service oriented system development approaches that are very narrow and focused but yet produced remarkable results. Wooldridge (2009) has stressed at several of his writings that agent technology is 99 % about software engineering and 1 % about artificial intelligence. *Taking cue from this sound wisdom, this work takes up principles and ideas from a very narrow part of agent technology that deals with two well understood conceptually clear principles namely migration of code and speech act based application composition*. These two capabilities have a clear road to design and programmatic realization in information systems (as discussed here). Though this work is primarily about creating a web services based internet application programming framework, it refers to main stream agent technology literature because there several of its design primitives are well researched and documented over the past three decades. It helps us to quickly launch and assimilate well-understood and documented design principles though in a web services programming context.

Web services based application composition should tackle quality attribute issues (Milanovic 2011) that are non-functional properties but nevertheless very important for the success of the information systems. To realize quality attributes like scalability, separation, modifiability, integrability, portability, performance, reliability, ease of creation and reusability in the context of emergence of web services technology, one need to introduce technical constructs (i.e. "tactics" or "design option" as per Bass et al. (2003)) in the underlying technology. It is better if these technical constructs are made available for everyone's use rather than each programmer custom develop it in their own application context. Software Frameworks are one of the way to enable reuse. Hence this work suggests ACIAF as reusable software framework that help architects to achieve desirable quality attributes in their applications. Performance is an important quality attribute that improve system response. To impart this quality attribute it is essential to incorporate necessary architectural primitives and tactics (Bass et al. 2003) to the underlying technology. Mobility of applications reduces network traffic and improves performance due to proximity of code and data. In the modern context "mobility" can be added to the six unit operations presented in (Kazman and Bass 1994). Thus mobility operation adds an invaluable architectural tactic for achieving the performance quality attribute of resulting application (Bass et al. 2003). In the same manner, separation, modifiability and reliability quality attributes can be realized by orchestrating dependent components in the loosest possible manner using speech acts. Thus simple orchestration of web services using invoke operation in WS-BPEL can be extended to incorporate speech act message based invocation of participating web service interfaces without any direct reference to a partner link. This mechanism requires existence of a middleware for message routing and interface invocation. This message based orchestration is also proposed to be a part of ACIAF. Thus ACIAF incorporates mobility and speech-act based partner link interface invocation as two important new unit operations that designers can use (or reuse) for architecting information systems. It also offers ready path for implementation using WS-BPEL and underlying web services technology.

ACIAF conforms to Service Oriented Architecture (SOA) (Erl 2005; Newcomer and Lomow 2005). SOA consists of inter-operability and infrastructure based system integration and structuring principles. WS-BPEL (WS-BPEL (2007)) consists of WS-* (Weerawarana et al. 2005) standard compliant, platform neutral, industrially popular, business process modeling and orchestration and programming technology. Web 2.0 (Murugesan 2007) adds user generated content over the plain web data (such as web pages and services) through blogs, comments, ratings, tags, feedback, syndication and the like. Web 2.0 adds new method of generating highly relevant data to the organization (Murugesan 2009). There exist several works analyzing domain specific usages of Web 2.0 (Boulos and Wheeler 2007; Constantinides and Fountain 2008; Bughin and Chui 2010) that illustrates how industry got benefited from Web 2.0 technologies in managing their resources efficiently, achieve business goals and increase customer satisfaction. Due to rapid adoption of Web 2.0 and business process automation solutions modern business enterprises generate massive quantity of data situated on geographically separated locations. SOA has enabled provisioning of vast number of service based

applications that can be readily leveraged by third party applications. Traditional applications and information system environments are unable to cope with this rapid developments in software technology and are struggling to keep pace. Hence novel information processing solutions are required. Thus one of the research problem taken up in this work is to arrive at a programming language framework that leverages best of the popular pervasive technologies to enable creation of loosely coupled code mobility based business information systems to efficiently process data scattered across distributed sites. It can be seen that ACIAF applications readily provides solution to this problem.

Applications developed using ACIAF result in SOA compliant services. These applications run on WS-BPEL containers (extended to support "mobility" –that is service migration- tag) that are typically run as part of J2EE application servers such as Tomcat or Jboss. A popular open source WS-BPEL programming and run time environment is Apache Orchestration Director Engine (ODE) that is used here to realize ACIAF. ACIAF applications remain 100 % backward compatible with existing business process automation infrastructure protecting the investment made. However at the same time it permits WS-BPEL applications to incorporate newer loose coupling and mobility constructs wherever it result in performance advantage.

Traditional agent platforms (Sudeikat et al. 2004) includes JaCaMo (Boissiera et al. 2013), 3APL, JADE/WADE (Bellifemine et al. 2005, Banzi et al. 2008), Jadex (Pokahr et al. 2003; JadexTool 2016), Retsina (Sycara et al. 2003). They provides promising application level architectural primitives such as migration of code, speech act based loose coupling, deliberation, means-ends reasoning, rationality and autonomy and uses reasoning strategy using Belief-Desire-Intension framework. *However ACIAF applications differs from traditional agent technology applications as it is striped and devoid of any kind of artificial intelligence or knowledge engineering goals that are typically associated with traditional AT –such as autonomous, rational and deliberative capabilities* (Bradshaw 1997; Weiss 2012; Wooldridge 2009). In addition ACIAF applications are out growth of web services and business process automation technologies and hence are already widely used and continue to use familiar tools and techniques. The benefits are not available for traditional agent technology environments as they are vastly different (in system modeling, programming paradigm, run time environment, test/debug) and do not generally use industrially popular tools and techniques (Dastani et al. 2010). Since this work derived several ideas from AT and in future several more ideas may also be derived from it (given the revolution taking place in sematic web and readiness of ACAIF to imbibe it—to impart applications logical inferential capabilities that are also part of traditional agent technology), it is worthwhile to study ACIAF side-by-side with traditional AT.

The primary goal of this work is to illustrate ACIAF features that are useful in architecture of next generation information systems and outline how to realize them. It does explain technicalities of ACIAF itself in brief. In that it forms a foundational, motivational and unifying role for further detailed and focused technical work. This work argues and establishes WS-BPEL as the platform of choice for introducing novel unit operations (i.e. architectural primitives) and argues no other technology is likely to possess these merits and hence is a promising candidate for realizing next generation information systems.

This paper is organized as follows: Sect. 2 discusses related works such as services, work flow, web technologies to create a new programming framework that has new application architectural primitives. Current literature is reviewed to get fresh ideas and to leverage the benefits offered by them. Section 3 introduces mobility unit operation and discusses architectural issues in composing standard compliant mobility applications. It introduces ACIAF framework. It also highlights details of ACIAF application software environment. Section 4 presents an approach to loose coupling and integration of services using speech act like messaging using WS-Addressing technology. Section 5 presents a use case employing ACIAF to demonstrate code mobility based application. This helps us to experimentally analyze the outcome of various solution architectures available to build similar application and highlights impact of ACIAF. Section 6 evaluates ACIAF with respect to well know criteria on modeling, programming and technological aspects. Section 7 discusses certain formal properties enjoyed by ACIAF applications such as "verifiably safe migration" and logical/inferential interpolation of individual ACIAF application behavior in a group of ACIAF applications. Section 8 ends with some concluding remarks and a brief plan of further work.

2 Literature review

2.1 SOA and web services technology

Web services can be used to encapsulate underlying executable code while exposing their interfaces through Web Services Description Language (WSDL). It offers interaction via Simple Object Access Protocol (SOAP) and by other means. Most proprietary technology vendors expose their software capabilities using WSDL as it offers unlimited agility for composition to the underlying software components (Weerawarana et al. 2005). Hence universally these days components are written as Web services and exposed and accessed via WSDL/SOAP. Typically internet based services are provided as web services and architecting information systems using those services involve usage of Service Oriented Architecture (SOA) principles (Newcomer and Lomow 2005; Huhns and Singh 2005a). SOA is backed by supporting technology standards (Brown et al. 2008; Erl 2005). Its features include capabilities such as service re-use, message monitoring/control/transformation/security, complex event processing, service composition/discovery, legacy asset wrapping & provisioning and virtualization to address scalability. Its infrastructure include capabilities such as service repository, service registry, messaging/queuing infrastructure, activity monitoring/logging, encoding and data transformations, encryption engine, event processor, composition engine support, and model-based implementation environment (Weerawarana et al. 2005). Imperative programming paradigm is widely used to realize SOA because most developers are familiar with it and large number of industry quality tools is available. However adaptation of SOA process and methods have several challenges (more than 413 issues identified by Gu and Lago 2009) to face, most of which are being effectively addressed by software industry.

2.2 Sematic web technology

The web content or data can be encoded using standard ontology (Kotis and Vouros 2006; GeneOntology 2001) and annotated using approaches such Web Ontology Language (OWL) (Yu 2011). This enable web resources to be machine processable using query, discovery and selection (Noy 2004; Yu 2011). Thus web 3.0 capability fully depends on ontology based XML encoding of web resources, and it is a necessity for intelligent querying and processing of web resources. Hence enterprise information system architect should design information systems/application in such a way that it keeps open the path ways for future adaptation of web 3.0 tools and technologies in their application. Typically given the cost of the Information System (IS) development, they are usually used for several decades as the past enterprise IT experience shows (Gable et al. 2008). There is a huge scope for integration of sematic web services technologies to business information system application as it has been shown by García (2008). Thus enterprise information system should adopt XML based programming and integration technology to fight obsolescence, protect investment made and be future ready.

2.3 Business process modeling

SOA is an approach to build systems, while sematic web is to enable systems to make use resources intelligently and in machine processable manner without ambiguity. But neither of them address modeling of actual business processes and applications. That aspect is addressed by workflow paradigm. Workflow Management Coalition (WfMC) is a standard body and promoter of workflow paradigm. It has produced a workflow reference model (Hollingsworth 1995) that forms the basis of most of the Business Process Modeling (BPM) initiatives today. It consists of standards for process definition, interfaces, information interchange, inter-operability and control in the workflow enactment. BPM is about defining flow of data among business processes in a specified order using workflow metaphor. It is concerned with representation of the process flow, the actors within it, alerts, notifications, escalations, standard operating procedures, service level agreements, and task hand-over mechanisms (Havey 2005). It is gaining popularity due its ability to integrate disparate processes (implemented using different technologies) using modern EAI techniques such as XML, SOA, SOAP, ESB and Web Services. Different programming languages have evolved for specifying BPM such as Business Process Execution Language (BPEL), Business Process Modeling Language (BPML), BPSS and MDA (Harvey 2005). Of them industrially popular languages are BPEL (WS-BPEL 2007) and BPMN (BPMN 2016). Business rules have been used by BPM systems to provide definitions for governing behavior, and a business rule engine can be used to drive process execution and resolution. Business rules (specified visually as part of Computation Independent Model (CIM) of the process) permit dynamic processing such as intelligent routing, validation of policies within process and constraint checks. It also permits business analysts to create ad-hoc workflow based on changes to business practices and policies that help one to alter original flow in process model created such as (1) Policy based task

assignment (2) Various escalation policies and (3) Load balancing of tasks. Typically a BPEL process script is a sequence of XML tag based statements invoking web service interfaces in a specified order with data being retained and passed across different invocation by the BPEL process script. WS-BPEL programming is imperative in nature and uses direct interface invocations of partner links. Partner link invocations can be easily routed through enterprise service bus technology (ESB) (Schmidt et al. 2005; Maréchaux 2006) in a standard compliant manner. WS-BPEL is industrially accepted and prevalent technology though is not declarative in nature. Financial products Markup Language, Conversation Markup Language are some examples of XML dialects of business messages that work on ESB and WS-BPEL to implement real world systems purely based on messaging. This work shall use BPM technology to specify the logic of our mobile agents because it maps in a natural manner real life business processes and their details to software abstraction model. Juric et al. (2009) provides a method for incorporating versioning to WS-BPEL process scripts. It inspires us to find similar extensions to WS-BPEL process scripts and modeling technology so that novel architectural level unit operations can be built into BPEL processes.

2.4 Popular agent environments

While workflow technology provides scheme of modeling and building business system, it leaves out identifying inherent reusable architectural patterns that can be culled out and made part of application server engines/middleware to enable infrastructure based reuse. That aspect is addressed by agent technology. Agent technology (ResourcesAgent 2016; FIPA 2016) has both imperative (Bellifemine et al. 2005; Banzi et al. 2008) as well as declarative programming (Bordini et al. 2007, 2009; Lloyd and Ng 2011) language based programming environment. Imperative programming languages such as Java, C++ and WS-BPEL is popular in enterprise computing and enterprise software architecture (Sessions 2007). Hence this work adopts the path of programming system using imperative languages and will not discuss about declarative agent programming languages and environments further. Some popular imperative programming based agent environments are JADE, ABLE and Retsina. Java Agent Development Environment (JADE/WADE/Wolf) (Bellifemine et al. 2005; Banzi et al. 2008) is an environment/framework that provides developers infrastructural facilities by abstracting protocols and classes for agent communication, transport, messaging and workflow behavior using Java programming language technology. RETSINA (Sycara et al. 2003) is a popular agent environment consists of four basic agent types namely interface agents (for user input/output), task agents (perform user tasks/plans/workflows/collude with other software agents to accomplish work), information agents (provide intelligent access to a heterogeneous collection of information sources), middle agents (for multi-agent capability matching). It has mechanism to enable requester agents to find provider agents in environments with heterogeneous and dynamic information services, using middle agents, which serve to increase agent interoperability. It provide infrastructure for multi-agent interoperation, matchmaking, agent location resolution, failure monitoring, logger/activity visualization/launcher, agent specification in terms of

ontology and agent discovery and message transfer. ABLE (Bigus et al. 2002) is a Java framework, component library, and productivity toolkit for autonomic computing utilizing machine learning and reasoning. It uses extensively Java beans technology and developer needs to write native java programs and developed for Java platform only in mind. Jadex (JadexTool 2016) is a realization of Belief-Desire-Intention processing logic framework (Rao and George 1995) add on to JADE platform. It shows how to model business processes using goals, plans, and rules. But as per perception of experts of agent technology themselves the discipline has not seen much industry adoption (Shehory and Sturm 2014; Logan 2015).

2.5 Formal modeling of WS-BPEL

A robust business process modeling paradigm and highly capable middleware platform addresses only one aspect in real world software development. There should be a mechanism to test and verify systems built out of these technologies. Thus formal modeling, specification and verification of programming languages play a critical role. Programming languages (such as Pascal, C, C++) and their formal specification in terms of Backs-Naur form and usage of Context free/sensitive/regular grammars simplify translation and compilation of programs written in those languages. Incorporating such formal modeling notation to analyze and translate programs facilitate proper usage of languages supporting concurrency/distributed computing constructs. WS-BPEL (2007) is an industrially popular orchestration language. There are automatic verifiers (Breugel and Koshkina 2006; Lohmann 2007; Lohmann et al. 2008; Müller 2010) which can take WS-BPEL process scripts and convert it into formal models (Aalst 2005) such as Petri nets (Lohmann et al. 2009a, b), guarded concurrent automata and process algebra and prove its ability to satisfy certain set of preset service properties. These properties of WS-BPEL can be used to determine graceful completion of concurrently running tasks and absolutely determine current state of execution of WS-BPEL scripts. This formal maturity of BPEL scripts makes it an ideal choice for next generation system composition and integration programming platform.

2.6 Agent programming research

Nwana and Ndumu (1999) presents a dated assessment of the status of agent research that nevertheless relevant even today. It sets out important evaluation parameters that any agent platform must meet to be relevant to the business environment. Kamngar et al. (2005) work gives a detailed account on the design, data structures and implementation details of a mobile agent server and container. It specifies role of components such as agent loader, agent router, ACL communication, agent representation, agent state representation like resource table, service table, route table in realizing the system. It is based on third generation programming environment and does not embrace services technology. Cooney and Roe (2003) presents a model for implementing information systems using concepts of Web services and mobile agents where agents are free to move between cooperating Web servers/services to implement the application service functionality. Chunlin et al. (2003) proposes a hybrid agent platform (IMAP) composed

of several cooperating intelligent and mobile agents. It uses Java to implement the framework of the system (for communication between agents and mobility) while Prolog is used to impart reasoning and intelligence. Xu and Pears (2006) provide a fault tolerant agent middleware platform (that could survive agent server crashes) to carryout large scale distributed information retrieval applications and collaboration processing. Aberg et al. (2005) provides details of composing web services into a work flow systems to carry out business applications and deploys agent technology to discover and compose web services (using ontology based web service discovery). However the architecture fails to incorporate mobile code as part of application architecture and it essentially presents centralized coordinating software orchestrating distributed application that is of limited interest. Chi and Song (2007) proposes a method for e-business process composition using a chain of services that are executed in on-the-fly orchestrated manner that leverages adaptable agent metaphor and interoperable, extensible web services technology. Wang and Wang (2005) discusses technique of specifying and managing work flow configuration dynamically and uses agents to carry out decision making and business process orchestration. Several of the workflow related features attributed to the agents in this paper were used in realizing modern business process application software architecture in Venkatesan (2010). Wieland et al. (2008) discuss integrating context information into workflow. The supply of contextual information and provisioning of an integration layer into workflow semantics at appropriate place shall make work flow smart. The context information such as agent's status, error condition and exception are gathered and supplied to agent container engine (through an integration layer) to intelligently handle agent related management/administrative tasks. Guan et al. (2004) discusses issues faced by a system that employ mobile code in securing such agents and agent hosts. Some issues are like data confidentiality, non-repudiability, forward privacy, forward integrity of code and data (with ability for the user to verify the same), code insertion defense, code truncation defense, itinerary confidentiality and malicious action on the host. Lettmann (2011) provides an extensible definition of agent and multi-agent systems that users can customise and design agent based system. Considerable research is going on in providing security for mobile agents based on this work. Huhns and Singh (1998) argues agents are the best metaphor to realize business workflows. Huhns and Singh (2005b) claims several SOA principles are deeply influenced by MAS research and describe a 15-year roadmap for service-oriented multi-agent system research. But many of the stated research agenda is still awaiting success and have seen limited progress. It is anticipating inevitable evolution of services standard complaint MAS environment to carry out large scale cooperative computing. Huhns has published several works in the past 25 years about integrating workflow, agents, services technology and autonomic computing. In those works it is argued that there exist strong relationships between these disciplines, but till date not many applications exist that corroborate the importance of the relationship. Brazier et al. (2009) argues a case for creating web scale workflows by integrating SOA and agent technology in addition to incorporating autonomic capabilities to this environment such as self-configuration, healing, optimization and protection. Okouya et al. (2013) extends this proposal one step further making this environment semantic SOAP message based and inferentially adaptive, predictable and intelligent and proposed Open Interaction Platform. It may be noted that all the

initiatives mentioned here have not resulted in creation of any industry grade solutions (Shehory and Sturm 2014; Logan 2015). Most ideas remain on paper or isolates to their research labs and till date there exist no SOA compliant XML based MAS environment. It may be interesting to note that this work may be making a significant first step envisaged in creating SOA compliant MAS platform.

3 ACIAF and "CODE MOBILITY" operation

3.1 Adding code mobility to WS-BPEL scripts

Business application can be modeled, specified and programmed in process scripting language like Web Services-Business Process Execution Language (WS-BPEL) to codify business flow logic to interface with computing node's processing capability in an interoperable and decentralized manner Venkatesan (2010) presents a method for arriving at novel software environment that integrate few important orthogonal technologies. Building upon the ideas presented there ACtive Internet Application Framework (ACIAF) proposes to fuse Workflow (Aalst and van Hee 2004), Web 2.0/3.0 (Murugesan 2007; García 2008; Yu 2009), Service Oriented Architecture (SOA) (Newcomer and Lomow 2005; Brown et al. 2008) and WS-BPEL (2016) and some features of Agent technology to create a new distributed computing environment. The benefits provided individually by these technologies are already discussed in Sect. 2. Fusion of these technologies using an architectural/programming framework permits creation of new class of internet applications that can migrate to partner link computing nodes to leverage the services running on it. In this manner computation is carried out in proximity of data and results are accumulated in the state variables of migrating application. This mobility enabled business process application after taking a round trip across various partner link nodes, can arrive at invoker/launcher of the process with processed results. This model of computation increases performance of resulting applications. WS-BPEL is used to codify the process logic of the mobility capable applications (mobile agents) and it natively has services computing metaphor built into it. ACIAF extends WS-BPEL by adding new XML tag for mobility that permit the script to suspends its execution and migrates to specified node and recover context information and proceeds further with execution. There are several similar efforts in architecting systems (either in isolation or by fusion more than one approach) using SOA (Huhns 2002) or agents (Luck et al. 2005), or work flow (Savarimuthu et al. 2005; Havey 2005) or Web 2.0/Web 3.0 (Katasonov and Terziyan 2008; Brazier et al. 2009). ACIAF assimilate, combine, synthesis and advance their ideas. In this paper, using ACIAF a case study is described that justifies need of ACIAF and show richness of framework in composing efficient internet application. Building upon ideas expressed in Aversa et al. (2009), Banzi et al. (2008), Trione et al. (2009) and Luck et al. (2005) this work proposes a software development environment that support development/deployment of applications based on ACIAF that is inherently workflow based, business aligned, formal, industry standard and SOA compliant. Here ACIAF engine is realized as a services container server. It is clear that these

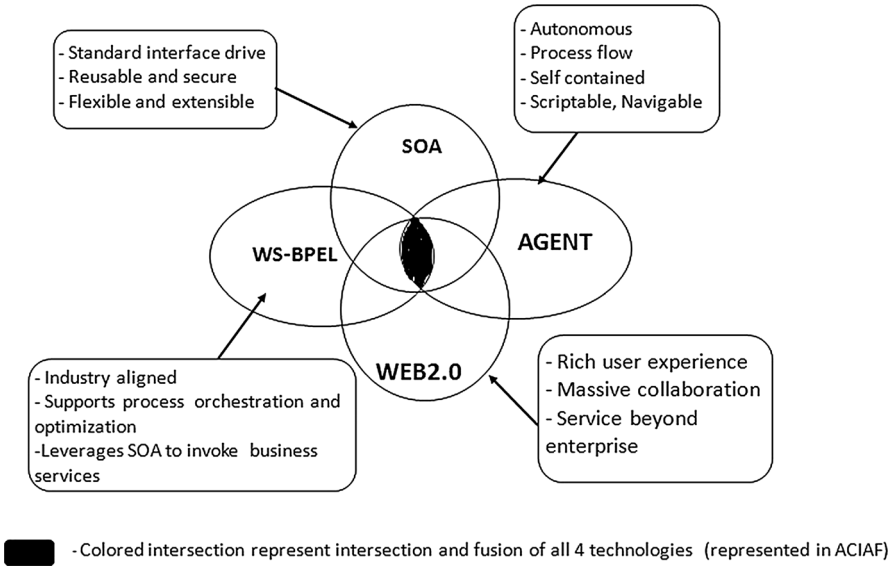

Fig. 1 Convergence of BPEL, SOA, Web 2.0 and mobile agent technologies

applications are standard compliant, easily leverages available infrastructure and are reusable in comparison to similar agent platform mentioned before.

Though multi-faceted software architectural proposals/approaches were made having the same or similar goal of fusing multiple technologies (which were reviewed in Sect. 2.6) none actually proposed a standard compliant framework. Thus ACIAF effort is matured for the following reasons:

1. It uses industry standard WS-BPEL process scripting language to provide mobility to applications.
2. It extends WS-BPEL to provide additional markup directives for script termination/archival at selective point, migration and re-initialization.
3. It uses WS-BPEL engine itself as agent container and deployment environment.
4. It extends the WS-BPEL engine to have additional capability to receive, selectively execute related portions and migrate it to other BPEL engines.
5. WS-BPEL permits through a ESB intermediary, partner link interface invocations semantics which can be extended to support "speech act based" interaction through a sematic SOAP messaging infrastructure leading to open interaction system infrastructure as envisaged in Okouya et al. (2013).

This results in richer application architecture to allow us to compose sophisticated applications. WSBPEL process scripts that migrate from node to node, are called for short as "mobBEPL" Script. Figure 1 below depicts convergence of important orthogonal components of distributed computing to create architecture for e-commerce or any kind of mobility based internet application. This is called as ACIAF.

3.2 The ACIAF system environment

Architectural vision and business need decides on the capabilities built into any software system. Agent computing tend to incorporate most ambitious features due to its conceptual strength (Brazier et al. 2009; Huebscher and McCann 2008; Mazeiar and Ladan 2009) from neural network, fuzzy reasoning, self-organizing, autonomous, artificial intelligence and so on. Typically the environment should allow for flexible design, building and operation of assortment of agents with mundane features like heterogeneity in communication, coordination, functionality, platform, semantic description. Agent societies be developed/deployed to adopt future evolution, integration in a standard compliant manner. This infrastructure need to be domain independent and reusable substrate on which actual agents, its components, services live, communicate, interact and interoperate. ACIAF due to its services computing standard compliance meets several of these needs making it a powerful business processing environment.

Despite several examples of deployed agent systems, agent technology face several barriers to their large scale adoption (McKean et al. 2008; Shehory and Sturm 2014). A fundamental obstacle in agent technology is lack of mature software development methodologies, lack of common standards, insufficient tool support, failure to enable software reuse, SOA ignorance, non-standard operating environment, un-realistic assumption of agent as a piece of self-contained code (minimal reuse of standard infrastructure), platform dependency and failure to interface with existing software infrastructure in a natural manner. However an industrially popular business process automation environment like ACIAF addresses these pressing issues as shown in Fig. 2.

The core container engine that implements ACIAF as an infrastructure constitute of extended WS-BPEL server. We call it Agent Server Web Server (ASWS). It implements code mobility capability to process scripts through new <serialize>, <loadFromArchive>, <startExcuteAt> structuring activities tag. It introduces new <migrateTo> communication activity tag related to structuring control flow of BPEL process. A sample mobBPEL process script is given in Listing 6 (Sect. 5). This work refers to a mobBPEL script also as an agent as it shares many features typically attached with a traditional agent.

ACIAF has a visual process editor/composition tool that is derived from standard Eclipse BPEL designer editor. It is a kind of Rich Client Platform (RCP) application. Its source code is modified to add ability to model, specify and generate

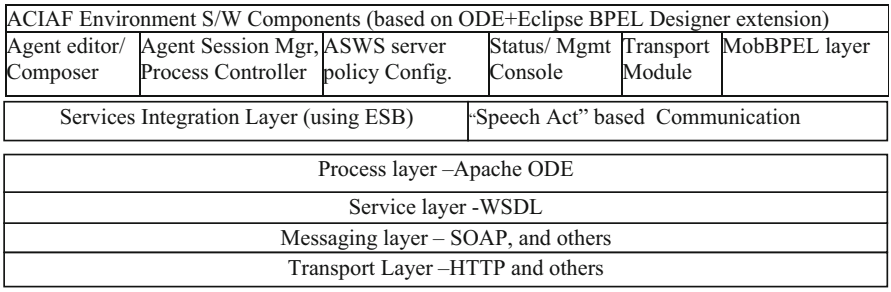

Fig. 2 Components of ACIAF server (ASWS application stack)

additional tags of mobBPEL script to make it ACIAF agent business logic coding/composition tool. Instead the mobility tag can be inserted by hand to the existing WS-BPEL script at appropriate places and submitted to bpel compiler.

ACIAF has status reporting and management module is similar to JBoss Web Services/JMX console and it lists status of all agents at a partner link. It is derived by customization of ODE-process monitoring web tool interface. mobBPEL scripts are managed from this console. ACIAF has a server policy module that governs execution of client mobBPEL scripts. It provides necessary security for partner link nodes against resource hogging, vicious scripts, Denial of service attacks by client scripts. It places certain ceiling on how much system memory, execution time, network band-width a client script can take on the partner link. This information is set by the administrators of the ASWS node. "Client process controller" uses this for client process management. ACIAF has a "Agent session manager" and it takes ACIAF agent configuration parameters that govern mobBPEL script execution. This information is set by the developers of the ACIAF script (at client side) and is part of the migrating agent.

3.3 Architectural elements of ACIAF

The architectural elements of ACIAF application server is depicted in Fig. 3. It shows software components running on a PartnerLink node and their interactions. Architectural elements of a software is expressed in terms of components, bus, system property elements (Grunbacher et al. 2003). Following this tradition AICAF system elements are identified. It includes an editor capable of inserting mobility tag in a BPEL script, a compiler extension that can process WS-BPEL scripts with code mobility tag extension, a middleware platform that is an extension of standard WS-

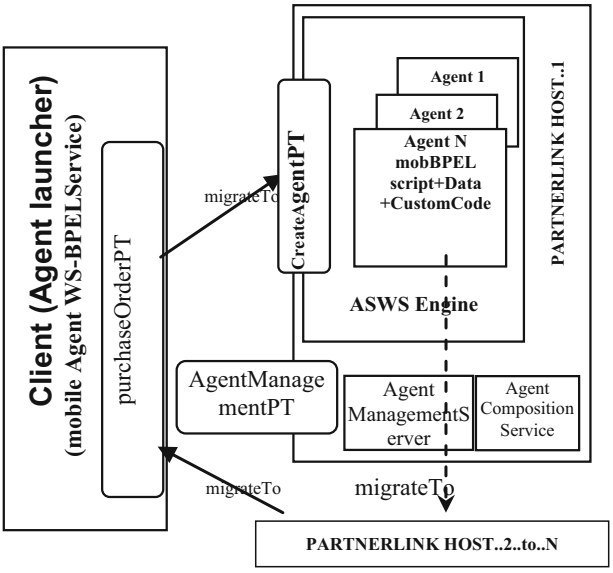

Fig. 3 Schematic of client, partner link node interaction in ACIAF

BPEL engine (with support for mobility). All these participating components and their deployment scenario is illustrated in Fig. 3.

A behavioral description of ACIAF application may be in place here. It also shows relevance of agent metaphor in combination with service. Take the case of a purchase order BPEL application. The client wants to place a purchaseOrder with a partner who in turn interact with one or more partners to determine price, packaging, shipping and invoicing, credit card payment authorization actions. Most of these services are available with third part service providers and the purchase order application must orchestrate them to realize its business functionality. These services are typically available as portTypes or WSDL interfaces. In a typical WS-BPEL scenario all these actions will be executed in sequence from the client PurchaseOrderService in a client–server mode.

But in the case of mobBPEL version of PurchaseOrderService application mobility is inserted at appropriate places. To make this application robust, proposed extensions and design approach is essential. purchaseOrder BPEL process migrates to partnerLink node (say PARTNERLINK HOST "1") to computer price of the order by interacting with local itemPriceInfoService and then migrate and interact with packagingInfoService to determine special packaging requirements and its price (which may be running on the same node or different node (say PARTNERLINK_HOST "2") and workflow cycle continues until generated invoice/pay acknowledgement is submitted to the initiating client. In ACIAF application, client can just initiate the order placement action and wait for the arrival of finished invoice (as a message in asynchronous mode) unlike client–server model where it orchestrates each and every step. However the process specification along with node migration instructions needs to be composed by the client as usual as part of WSBPEL process specification (for static case). Section 5 illustrates a more understandable and simpler use case development using ACIAF.

3.4 ACIAF system implementation description

A research prototype version is developed on Apache-ODE 1.3.5 (ODE 2016)/ Axis2.x environment by changing BPEL engine code to add new tag functionalities. The development and testing is carried out on a $\times$86 64 bit machine running CentOS/MS-Windows operating system. Apache Orchestration Director Engine (ODE) has several unique capabilities that make it ideal choice for our agent environment. In ODE the process management API (ODEAPI 2016) is defined as a Web service interface. In doing so it offers SOAP access to the service, and also create Java interfaces for SOAP access and JMX, depending on needs. All messages are defined as XML elements, mapping to WSDL doc/literal. This offers immeasurable benefits for process management (such as migration that is relevant to us). The process definition management interface of ODE defines six operations namely, list (Returns information about all, or some process definitions), details (Returns detailed information about the specified process definition), set-properties (Changes properties associated with the process definition), activate (Activates the process definition) and retire

(Retires the process definition). The process instance management interface of ODE defines following operations namely, suspend (Suspends the process instance), resume (Resumes the process instance), terminate (Terminates the process instance), fault (Faults the process instance) and delete (Deletes all or some completed process instances). These sets of APIs (ODEAPI 2016) are suitably modified to accomplish WS-BPEL process retiring, migration and reloading as and when we encounter BPEL extension tags in the process definition file. From a programming point view, we accomplish migration and reloading capability by suitably modifying and extending ODE JaCOb (ODEJaCOb 2016) implementation to archive process variables and reload them at different partner link node at process creation time (similar to suspend and resume activity implementation). Changes were also made to related packages and dependencies of DeploymentService, InstanceManagement, ProcessManagement interfaces/classes in ODE 1.3.x code base to realize several mobility related sub-functionality in ODE engine.

3.5 The ACIAF agent scripting environment

The code mobility capability for ACIAF is derived by adding mobility tag to WSBPEL script. WS-Script can be composed using a modified BPEL editor that supports insertion of mobility tag at appropriate places in the composition or mobility tag can be manually inserted into a generated WS-BPEL script. By adding <migrateTo …/> tag (a sample pseudo code is given in Sect. 5, Listing 6) it remains to be shown that it does not interfere or alter the regular data and event flow of original BPEL script. BPEL programs as soon as finish executing a tag, completely reflects the effect on the state of the process variables. No action will be kept pending or will be posted by the engine at a later time. Hence the BPEL program execution has local semantics like in a flat shell scripting program. This property can be used to analyze the effect of <migrateTo…/> tag on overall WS-BPEL program flow.

Like in any other programming language, correctness of logic and flow of program is the responsibility of the programmers. Tools can only check the syntactic correctness of used programmatic constructs. It is programmers responsibility to use <migrateTo…/> tag at suitable places in WS-BPEL programs where no other activity is ongoing or pending such as in a loop (between looping tag or flow tag, migration tag must not be used). Due to the underlying formal rigor (using Petri net theory) enjoyed by WSBPEL, its compiler can detect scripts flow anomalies due to erroneous usage of the migration tag (through static analysis). In such cases at the compilation time itself error is generated about inappropriate usage of migration tag and compilation is aborted.

The Listing 1 below shows the algorithm for safe migration of the ACIAF agent script. This algorithm assumes that the WS-BPEL script is not having any syntax error (such as missing tags, tag pairs, misspelling, variable declaration errors) and is a well formed XML document in the first place before introduction of “migrateTo” tag.

Listing 1 Algorithm for sane migration of ACIAF agent scripts

```
//assumption:BPEL script is a well formed formula before introduction of mobility tag
Algorithm checkSafeMigrationOfACIAFAgentScript (input mobBPELScrpt)
Returns flag;
Enum scriptWFFflag = {SAFE,UNSAFE}
MIGRATE_TAG = "migrateTo"
Int  pendingMatchingBpelTagCount=0;
String bpelTag=NULL;
scriptWFFflag=UNSAFE;
While (bpelTag=getNextBPELTag(mobBPELScrpt) != NULL)
{
    If  bpelTag== MIGRATE_TAG && pndngMatchingEndTagCount!=0
        scriptWFFflag = scriptWFFflag | UNSAFE; //migrate tag overlaps regular BPEL tag –error!
    else if isBpelStartPartTag(bpelTag)==TRUE
        pndngMatchingEndTagCount++; //encountering new BPEL opening tag
    else if isBpelMatchingClosePartTag(bpelTag)==TRUE
        pndngMatchingEndTagCount--;//encountered a matching BPEL close tag
}  //reached end of input script file; check if any migrate tag overlapped other tag construct
Return scriptWFFflag;
```

In this manner ACIAF engine ensure safe insertion of mobility tag at appropriate places of the mobility enabled ACIAF applications.

4 ACIAF and "Loose Coupling" operation

Like incorporating "mobility of code" as unit operation and provisioning it to system designers in a popular software environment solves few issues, there is a need to incorporate "loose coupling" (a kind of separation operation) of external services used in orchestration in ACIAF. Emergence of applications composed of externally provided services gives raise to new kind of challenges and problems that architects and programmers must address. At the architectural level the externally provided services may evolve or their location may move away or simply may not meet emerging quality/performance requirements. At the programmer level these issues must be tackled at the code level. Design time picture may fade away at run time as the typical production application life time is more than a decade. It is challenging to get reliable external partnership for an extended period of time. The requirements of consuming applications changes fast and provider must keep updating their services to meet the expectations of the consuming applications. This leads to versioning problem as different consumers wants different things from the same service over a period of time. Since WS-BPEL framework is predominantly used to compose and orchestrate services based applications, it must be capable of facing this situation and resolve it.

This problem is well known to application software developers on Microsoft windows operating system (and also developers of Unix *.so files that are sharable dynamic loadable libraries across several applications) during 1990s. It is known as "DLL hell" (Anderson 2000). Microsoft windows developers encounter problems with dynamic link libraries (DLLs)—especially after numerous applications have

been installed and uninstalled on a system. The difficulties include having many unnecessary DLL copies, conflicts between DLL versions and difficulty in obtaining required DLLs. Changes to the classes in the DLL such as adding a new member variable or virtual function would break already running application and require recompilation of it. Recent versions of Microsoft windows operating systems (as part of its .NET framework) treats DLLs as assemblies and permits side-by-side assembly sharing to enable multiple versions of a Component Object Model (COM) or Win32 assembly to run simultaneously on the system (Pratschner 2001). This technology permits instead of a single version of an assembly that assumes backward compatibility with all applications, permits coexistence of several versions of DLLs in isolated manner by appropriately relating to referenced application at run time without version mismatch.

To put it simply operating system provides infrastructure support to relate appropriate version of DLL (or external component/contract) to their connected applications; Different applications can use different versions of the same DLL without conflict. In addition to this externally provided component/function library may use different programming language technology and to integrate them one need support from the underlying operating system; this is known as automation enabled classes or function that permit cross language reuse of functionality. Indeed Microsoft operating system provided support for marshaling of data across components developed using different programming languages and also supported versioning of components through its registry entries.

Thus there is an established case that industry has accepted infrastructure offering solution to resolve version mismatch and locate/maintain appropriate version of components and their related applications (i.e. provide-consumer pairing information) as a workable approach. Hence this work suggests similar line of solution using ACIAF infrastructure to resolve "Orchestration Hell". The SOA applications are likely to face the same fate Win32 applications faced during 1990s in connecting with DLLs. If the present state of affairs in SOA technology continues, it will sure to lead to a situation that developers faced in 1990s in Microsoft Windows operating system environment. This situation may be called as "Orchestration hell". Developers will stuck with most inhospitable situation of unable to extend existing services for the fear of breaking existing orchestrations or choose the hard task of asking all clients to constantly upgrade themselves to match with their own service evolution (which is clearly unworkable). So the only way out in services technology is to introduce infrastructure based innovation to tackle services invocation problem; a functionality must be introduced in WS-BPEL framework (as it is established as the most prominent orchestration environment) to offer infrastructure based solution to orchestration applications to overcome "Orchestration Hell". Instead of permitting applications to use hard coded WSDL, message based orchestration of external services must be introduced. Incidentally this solution can be tweaked to solve supplementary problems faced by applications relating to issues like QoS, scalability, reliability and cost issues. Thus messaging based integration of external services, not only provide reliable connectivity, can also be used to meet business rules, partnership constraints at run time. Venkatesan and Sridhar (2016) offers a workable solution to this problem. Building upon the ideas presented there,

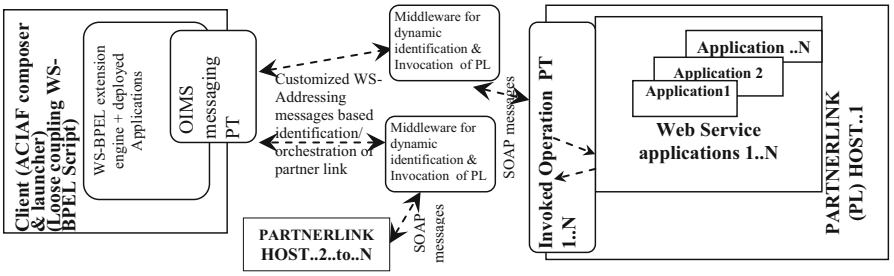

Fig. 4 Method of "Loose coupling" in ACIAF: schematic of client application, middleware for PL/ endpoint determination (that is part of ACAIAF engine) and WS-Provider nodes

Fig. 4 depicts architectural model of the proposed solution for loose coupling in ACIAF.

Traditional applications that consume WS service providers will simply fail if these "Provider" services (participants of orchestration logic) are not available for reasons such as service address migration, evolution or unsatisfactory QoS. Hence the designer should have a mechanism to specify—using underlying infrastructure capabilities—that should not be part of application logic- how the application behaves. One possible line of solution is to specify alternatives if one partnership fails then infrastructure can invoke alternatives automatically in case of failures. This would be a client side or middleware based solution rather than "Provider" based solution as in Ebrahim et al. (2015). Ebrahim et al. (2015) suggests a provider side (i.e. server-side) infrastructure based solution to cater to this situation purely based on design time contract metadata. However in the case of catastrophic failure at "Provider" side none of the suggested methods will be applicable. In such situation a client infrastructure based or middleware based fault tolerance solution is desirable. This work takes client side approach to tackle this problem. This approach may be proved more robust than traditional approach of keeping client application passive to any kind of failure and making "Provider" side active attempting remedial solutions as suggested by Ebrahim et al. (2015) and related works. Client side based error resilience solution has not been attempted so far as applications are architected predominantly in a client–server style flavor so far. Presently there exist no middleware to negotiate establishment of partners at run time.

Consumers and providers of services connect traditionally using URI. For enterprise scale application that require higher level service non-functional capabilities such as scalability, it is suggested to leverage WS-Addressing standard based connectivity between consumers and providers. WS-Addressing (2006) standard offers transport neutral mechanism to connect with instance specific service provider end points in a scalable manner. It removes dependency on transport layer for service address resolution and provide ability to use multiple transports as the data is encapsulated within SOAP envelop. So transport specific header messages are dispensed with to increase interoperability. Loose coupling with partner link invocation interfaces depends on identification of syntactically exact end point references at run time. Open Interaction Middleware Service (OIMS) suggestion by

Venkatesan and Sridhar (2016) presents an idea to leverage WS-Addressing infrastructure support to solve problems such as "Orchestration hell" and also provide "loose coupling" at the same time. OIMS suggests replacing traditional web services end point address (supplied at design time) in <wsa:To> and <wsa:Action> fields of WS-Addressing message structure with a speech act message that helps middleware to compute EPA at run time. This work integrates the ideas presented there into ACIAF and further elaborates it technically. This goal can be achieved using two different strategies that have different level of impact on "Client/Consumer" applications and "Provider/Server" applications. Subsequent paragraphs describe these two approaches separately in detail.

4.1 Loose coupling method affecting "Consumer" environment alone

Typically ACIAF applications are designed by specifying the business logic as a process flow. In the simplest case business logic may consists of orchestrating several business partner capabilities in a sequence. The business logic is converted into WS-BPEL program consisting a sequence construct and within it each business partner's capability is invoked (using the details provided in imported WSDL files of the partner link–Provider–services). Thus BPEL invoke operations are inseparably tied to the port types and operations of the partner link WSDL files leading the problem described in the outset of this section. This issue is at the application level. Even at the infrastructure level connecting to the service operations (i.e. through port type) is based on the URI and the transport of the SOAP message to the service end point depends upon the transport layer. To overcome these kind of issues WS-Addressing (referred to as WS-A for short) standard was introduced. As discussed WS-A standard consists of method of communicating EPR (end point references) to clients and message information from client to server within SOAP envelop.

WS-Addressing (2006) cater to enterprise level solutions using web services that need more than simple URI to specify the end point. Applications need more capability to convey information while invoking service end points and it just cannot delegate application level information (such as instance of web service, session id., profile info.) communication to transport layer. These information should be kept within the SOAP envelop. This requirement is fulfilled by world wide web consortium's WS-Addressing standard. It also supports extensibility by permitting end users to tailor message information header by adding custom fields in addition to regular fields.

Though WS-A technology removed direct dependence on transport layer for information communication between provider and consumer of services, it still retains client–server flavor and dependence on WSDL files and location specific service provider information. If a (SOAP message based) indirection can be introduced to overcome this direct (i.e. hard coded) dependence, it will increase reliability of orchestration/application. Venkatesan and Sridhar (2016) suggest a middleware mediated separation that uses simplest form of speech-acts between consumer and service provider. It suggest to use a message template provisioning over WSDL based on which consumers can orchestrate providers using ACL based WS-A. This work elaborates on the core idea presented there. There are several methods of realizing loose coupling at consumer end and the following sections describes few.

4.1.1 Loose coupling using syntactic method

It is illustrative to explain how loose coupling design activity is carried out at consumer side with code snippets from an actual use case. So let us take the use case presented in Sect. 5. In that the client orchestrates three "Providers" (node1, node2, node3) that provide mathematical calculation services namely "Addition", "Subtraction" and "Multiplication" respectively. In the normal case the "Consumer" (namely WS-BPEL) application refers to provider WSDL files at design time and carryout orchestration using URIs (namely http://www.cn1.com/AdditionApp/AdditionService, http://www.cn2.com/SubtractionApp/SubtractionService and http://www.cn3.com/MutiplicationApp/MultiplicationService).

A typical WS-A message transferred between consumer and provider (from application running on "client node" to WS running on compute node1 "CN1") in the use case is listed in Listing 2. This snippet highlights information content of various fields in unmodified WS-A message. The <wsa:To> element carries data on URI of the web service to which the SOAP message should be forwarded and <wsa:Action> element carries data on which operation to be executed on the server side.

Listing 2 A traditional WS-A message from client to compute node1 (CN1) providing "Addition" Service

```
(01) <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"  xmlns:wsa=http://www.w3.org/2005/08/addressing
xmlns:end="http://endpoint/" >
(02)  <S:Header>
(03)   <wsa:MessageID>http://client.com/123456ABCDEFG-XX11-1111-ABCD-00DD010662DA</wsa:MessageID>
(04)   <wsa:ReplyTo>
(05)    <wsa:Address>http://client.com/aciafmathtutor/tutorService</wsa:Address>
(06)   </wsa:ReplyTo>
(07)   <wsa:To>http://cn1.com/AdditionApp/AdditionService </wsa:To>
(08)   <wsa:Action>http:// cn1.com/Addition/addRequest</wsa:Action>
(09)  </S:Header>
(10)  <S:Body>  <add>  <arg0>1</arg0>  <arg1>1</arg1>  <arg0>2</arg0>  <arg1>2</arg1> </add> . </S:Body>
(11) </S:Envelope>
```

Listing 3 OrchestrateRegistry.xsd: configuration file to specify application preferences metadata and choice of Provider services and its metadata.

```
<?xml version="1.0"?> <xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema  xmlns="http://www.aciaf.com" >
<xs:element name="application"">
  <xs:complexType>
    <xs:element name="applicationName" type="xs:string"/>
    <xs:sequence>
        <xs:element name="partnerServiceId" type="xs:string">
        <xs:element name="partnerServiceName" type="xs:string"/>
        <xs:element name="primaryURI" type="xs:string" />
        <xs:element name="versionId  type="xs:string />

        <xs:element name=" alternateURIs"">
        <xs:complexType>
        <xs:sequence>   <xs:element name="alternateURI" type="xs:string" MinOccurs=0 MaxOccurs=unbounded />   </xs:sequence>
        </xs:complexType>
        </xs:element>  #end of alternateURIs

        <xs:element name="QoSindex" type="xs:float"/>
        <xs:element name="MaxNetworkLatency" type="xs:float"/>
        <xs:element name="MaxServerLoad" type="xs:float"/>
        </xs:element>  #end of "partnerServiceId"
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

As discussed in this orchestration the information carried by the tags <wsa:To> and <wsa:Action> are hard coded at design time. This information is rather

restrictive and suffers ills that are already described. Now the client orchestration environment (Apache ODE based ACIAF) should be enabled to permit designers to specify several alternatives to this <wsa:To> and <wsa:Action> information. This information is proposed to be stored as part of application registry file residing with ACIAF ODE engine (i.e. at client/consumer side). The structure of this configuration file—namely OrchestrateRegistry.xml—defined as a XSD file—is shown in Listing 3. This structure can be extended to hold additional information by adding new tags and appropriate handlers for them to be provided on ACIAF orchestration engine (while invoking partners). From the structure of this schema one can easily guess kind of designer level activity performed during application design using external services. Listing 4 gives an instance of this configuration file as a result of conclusion of orchestration design activity—this sample instance of configuration file for math tutor case study in Sect. 5 illustrates designer's choice of three more alternate servers (i.e. providers of addition mathematical operation contract) namely cn1a.com, cn1b.com, cn1c.com in case primary partner link cn1.com is not available or cannot meet quality parameter thresholds indicated in the configuration file. In such cases, at run time, middleware will make a suitable decision of substituting alternate partner link to execute external service contract. This way the ACIAF is more robust and active to failure resolution and enforcing meta-data based contract execution at run time. Traditional environments (Ebrahim et al. 2015) do not offer this kind of robustness using client side approach as in ACIAF.

Listing 4 A sample instance of OrchestrateRegistry.xsd snippet: for Math tutor application

```
<?xml version="1.0"?>
<application xmlns="http://www.aciaf.com"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.aciaf.com OrchestrateRegistry.xsd">
<applicationName>mathTutor</applicationName>
<partnerServiceId>addService1</partnerServiceId>
<partnerServiceName>AdditionService</partnerServiceName>
<primaryURI>http://www.cn1.com/AdditionApp/AdditionService<primaryURI>
<versionId>v1.0 </versionId>
<alternateURIs>
<alternateURI>http://www.cn1a.com/AdditionApp/AdditionService</alternateURI>
<alternateURI>http://www.cn1b.com/AdditionApp/AdditionService</alternateURI>
<alternateURI>http://www.cn1c.com/AdditionApp/AdditionService</alternateURI>
</alternateURIs>
<QoSindex>90</QoSindex>
<MaxNetworkLatency>100</MaxNetworkLatency>
<MaxServerLoad>80</MaxServerLoad>
</application>
```

The above configuration file based web service substitution method is experimented using a proof of concept research prototype application in Apache Axis-2 server and Java JAX-WS 2.x compliant Java client environment. However incorporating it in ACIAF container (that uses Apache ODE) is much more complex. The reasons are several fold; BPEL scripts are stateful and can embedded complex constructs around partner "Invoke" operations so loose coupling in WS-BPEL should not break on going interactions in the middle, Apache ODE uses extensively Axis 2 web services programming technology and integrating WS-A customization may prove to be technically challenging. This study is a work in progress.

Dynamic partnership determination in ACIAF can be enabled by adopting Java XML-Based Web Services (JAX-WS) technology. JAX-WS makes it easier for

developers to write stateful web services using inbuilt WS-A technology. The JAX-WS API provides this by adding explicit support for Web Services Addressing (WS-Addressing) in it. JAX-WS 2.x provides several ways to create endpoint references in server applications. Its API includes one subclass of EndpointReference class, called W3CEndpointReference that represent endpoint references in accordance with the WS-Addressing 1.0 Core specification. The W3CEndpointReferenceBuilder allows the developer the ability to specify any of the properties of an endpoint reference. In WS-A messages one can just specify an address and a reference parameter, however, it is possible to specify a Web Services Description Language (WSDL) service name and endpoint (port) name, instead of an address. The run time will use the service and endpoint name combination to identify the endpoint for which a reference is required, and it will return a W3CEndpointReference with the correct address filled in.

Endpoint references created in a server application returned over the wire to a consumer application. The client application can then use that endpoint reference to invoke the endpoint, as necessary. As per the WS-Addressing specifications any reference parameters included in the endpoint reference will be automatically be added as headers to the SOAP envelope that is sent to the client application. JAX-WS 2.x also provides a facility to allow a Web service application to retrieve these reference parameters. Based on this scalable, stateful and fault tolerant applications can be implemented.

The solution offered to "OrchestrationHell" using approach described in Listing 4 can be criticized (for finding alternate partner links providing same functionality) as yet another kind of hardcoding. However the enforcement of contract meta-data – such as QoS, latency enforcement may not subject to such criticism. So we need to search for alternatives. One promising approach is instead of hard coding these partner link URIs, they can be determined at runtime using semantic web query languages such as SPARQL or specified using content languages such KIF (knowledge interchange format). In either case it requires the entire ACIAF environment—and all its participants—to embrace some kind of understandable uniform ontology—which is not unrealistic to assume. In addition to this content part should be embedded in a richer communication language—as SOAP is simply syntactical. Approaches such as using FIPA-ACL (Foundation of Intelligent Physical Agents—Agent communication language) (FIPA-ACL 1998) is promising. OIMS suggested by Venkatesan and Sridhar (2016) and adopted here in ACIAF does that. The content language in the ACL can use OWL-DL (Schiemann and Schreiber 2006) or knowledge interchange format (Covington 1997).

4.1.2 Loose coupling using Scripts

It is possible to use speech-acts to indirectly specify possible partner link end points using some kind of logical assertions in its content field, shall replace traditional information in <wsa:To> and <wsa:Action>. Listing 5 lists snippet of WS-A message with a speech-act encoded inside <wsa:To> sub-filed to identify partner link dynamically. In the simplest case there are no changes whatsoever to provider side (they simply continue to provide the same WSDL file) limiting the innovation to the composition engine/framework alone making the diffusion/adaptation of the

technology very easy. It may be noted that there is some overlap between WS-A message structure and FIPA-ACL message structure such as sa-msg-id and in-reply-to. These message Ids should be used by server and clients to build suitably chained business logic in one-to-one or one-to-many basis. Provider WSDL files and their operations can be automatically encoded into message templates of the kind described in Listing 5 by consumers (i.e. client orchestration applications) –along with other non-functional or business constraints- and maintain it in their application deployment engine registry. This script based approach added one level of determining synonymous provider partner links to which consumer can connect to get the business contract fulfilled. It can be seen this approach is purely syntactical.

Listing 5 Speech act embedded WS-A messages from client

```
(01) <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"  xmlns:wsa=http://www.w3.org/2005/08/addressing
xmlns:end="http://endpoint/" xmlns:aciaf="http://aciaf.com/aciaf" >
(02)  <S:Header> ..............
(03)  <wsa:To>
(04)  <WSAACLQuery xmlns="http://www.w3schools.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.aicaf.com WSAACLquery.xsd" Language=FIPA_ACL>
            <speechact>ask</speechact>
            <sender> <aciaf:SenderAddress ID=mt.com/aciafmathtutor/tutorService instanceID=”ABCD-1234” Transport=http/> </sender>
            <receiver> <aciaf:ReceiverAddress ID=http://192.168.*.*  Transport=http/> </receiver>
            <content> /*Service endpoint descripton in OWL-DL or KIF) */ </content>
            <ontology>literal-syntax-based</ontology>
            <sa-msg-id> msg-ABCD-1234-xyz-maths-addRequest</sa-msg-id>
            <in-reply-to>converse-ABCD-1234-xyz-maths-add</in-reply-to>
          </WSAACLQuery>
      </wsa:To> ..............
(05)  </S:Header> ..............
```

4.2 Loose coupling method affecting both “Provider” and “Consumer” environment

The previous sub-section explained how changes to the consumer side orchestration environment help to achieve several robust non-functional properties that are not available to the applications otherwise. To limit ripples of the changes do not cross our own boundary, and to ensure compatibility and inter-operability with existing services infrastructure no changes were suggested to the “Provider” environment. However to reap maximum benefits of loose coupling provider environment also should provide infrastructure based support. One approach is to depend upon the OWL-S annotation of WSDL services (Martin et al. 2004a, b) to select partner links. The OWL-S coalition (Martin et al. 2007) in their original proposal had submitted automatic service discovery and invocation as one of its goal. Hence OWL-S infrastructure will sure to evolve to meet this goal (Talantikitea et al. 2009; Maleshkova et al. 2010). In this case <aciaf:WSAACLQuery> tag shall embed a semantic web query languages such as SPARQL that return equivalent service end points that carryout same contact as the design time end point. The ACIAF engine shall interpret the results based on the language attribute of this tag (in this case it would be OWL-S and SPARQL result processor plugin of ACIAF loose coupler module). Since OWL-S infrastructure is very generic for the purpose of each operation of the port type may have to be encoded into a message template and provided as part of infrastructure (Venkatesan and Sridhar 2016). As the details

of this technique is very elaborate, it is proposed to be documented as a separate work at a later date.

The practicality of the loose coupling described here is simulated in Axis2 environment using Java programming language based research prototype. Based on the experience gained in using research prototype, this capability is being incorporated in ACIAF engine.

5 Case study based evaluation of ACIAF mobility construct

Problems requiring orchestration of external services based solutions include: interconnecting and inter-operation of multiple, autonomous, 'self-interested' existing legacy systems e.g. expert systems or decision support systems, problems whose solutions draw from distributed autonomous experts, e.g. in health care provisioning or electricity distribution, problems that are inherently distributed, e.g. geographically as in distributed sensor networks or air-traffic control and problems whose solutions require the collation and fusion of information, knowledge or data from distributed, autonomous and 'selfish' information sources, e.g. personal travel assistance prototype application. Multi-agent solutions at the very least require problems that tackle distributed resource allocation problems. The case study illustrated below demonstrate one of the key ingredient of an agent system namely mobility and reduction of processing overhead at compute node due to proximity of data and code. To evaluate the feasibility of carrying out the ACIAF proposal for developing mobility based internet application and its performance with respect to other methods of realizing this functionality a case study was carried out.

5.1 Description of MATHTUTOR case study

MathTutor (say designated as compute node MT) intend to conduct an online quiz for a group of students say G. The quiz question paper has a HTML form with two elements f1, f2 to show system generated numbers and three answer sections namely S1, S2, and S3. S1 deals with "addition" mathematical operation on f1,f2 (which can be carried out by CN1 only), S2 deals with "subtraction" mathematical operation on f1,f2 (which can be carried out by CN2 only), S3 deals with "multiplication" mathematical operation (carried out by CN3). Let the quiz Q is about presenting a pair of numbers (a HTML form with elements f1, f2 of type integer) and asking students to find its "addition" (S1 part captured by HTML form element SA1), "subtraction" (S2 part captured by HTML form element SA2) and "multiplication" (S3 part captured by HTML form element SA3) values. MT on its own does not have capability to carry out the functionality carried out by CN1 or CN2 or CN3. In other words CN1, CN2, and CN3 are the one with specialized capabilities that must be leveraged by the MATHTUTOR node "MT" to carry out its functionality This situation and all the stake holders involved are depicted in Fig. 5. Assume MT, CN1, CN2, and CN3 nodes capabilities are exposed as web services namely MathTutor, AdditionWS, SubtractionWS, and MultiplicationWS.

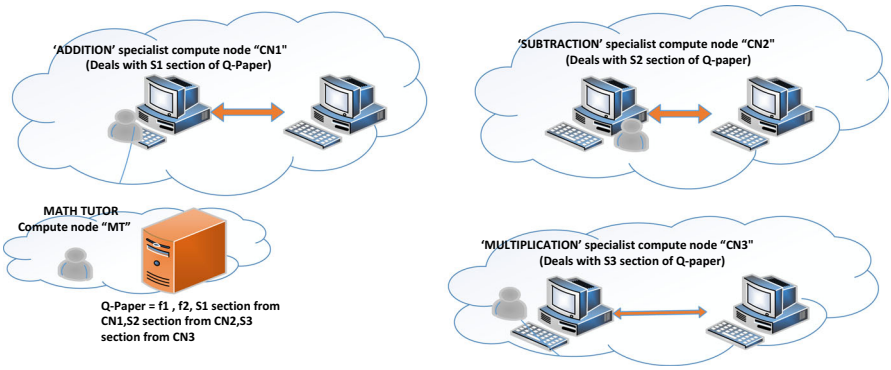

Fig. 5 MATHTUTOR examination server involving compute nodes CN1, CN2 and CN3

The number of G may vary according to the student batch size. For simulating load on application server and evaluating server computation time and network overhead, we assume three (test) situations of examination that involve a group size of 100, 1000, 10,000 students. This case study was physically carried out on actual software platforms used in real life such as web service application servers (Tomcat), web services, Java web service clients, WS-BPEL engine (Apache ODE) and mobBPEL engine. However to simplify the test environment setup, geographic separation of nodes CN1,CN2,CN3 and MT could not be enforced and simulated in a LAN (local area network). Testing and measurement effected here indicate feasibility of carrying out the actual implementation and its response to various load conditions could be ascertained.

5.2 Description of MATHTUTOR system implementation cases

This application can be realized using three distinct approach but all using WS technology. First approach (CASE1) involve a purely client–server architecture with MT as client (java web service client) and CNs as web application servers hosting addition, subtraction and multiplication services (depicted in Fig. 6a). Second approach (CASE2) is also a client –server architecture solution just like first test setup but CNs instead of directly exposing web services to clients like MT, expose

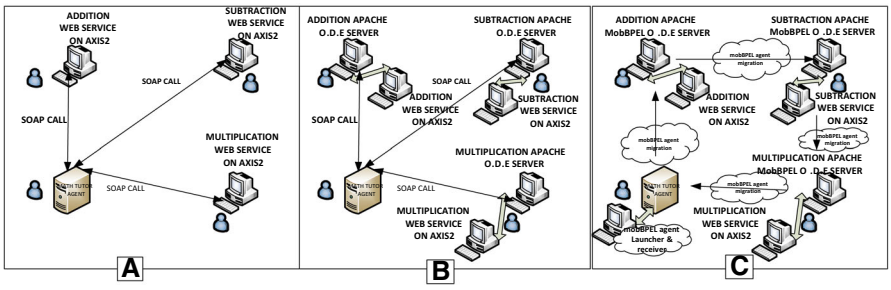

Fig. 6 MATHTUTOR test scenarios: **a** WS-based based, **b** ODE based, **c** mobBPEL based

themselves to Apache ODE 1.3.x server that orchestrate them to create end user applications for consumption (that are also exposed as web services). Since WS are stateless, adding ODE process layer over this plain WS add persistence, state and other benefits to WS. But the purpose of this test case is to measure the overhead of adding ODE process layer over and above plain WS (depicted in Fig. 6b).

Third approach (CASE3) involve usage of ACIAF to create mobility script encoded in tag extended WS-BPEL. One most important difference from the view point of MT is in the previous two cases it act as a simple web service client to CNs and invoke them as a java web service client while in this case MT will create a mobBPEL script using ACIAF paradigm codifying the application logic with mobility inserted at appropriate places. Typically WS-BPEL scripts has one receive and response pair only exposed to outside world through WSDL. Inside that script it can invoke any number of partner links that it orchestrates. The setup of various services and nodes interaction in this test case is illustrated in Fig. 6c. The logic of MATHTUTOR case study encoded using ACIAF is given below as pseudo-code in Listing 6.

Listing 6 ACIAF based MATHTUTOR mobBPEL script implementation logic – pseudo code.

```
Precondition: math-tutor invoke this mobBPEL script("MP") with quiz data.
    1) Receive quiz question & answers array data of all students, call it [D]
    2) Migrate to partner link CN1- //a mobBPEL ODE engine that exposes "additionWS"
    3) compute [CA1]:addition test result using local "additionWS" //to verify SA1
    4) update mobBPEL script state data; [D] = [D] || [CA1]
    5) Migrate to partner link CN2- //a mobBPEL ODE engine that exposes "subtractWS"
    6) compute [CA2]:subtraction test result using local "subtractWS" //to verify SA2
    7) update mobBPEL script state data; [D] = [D] || [CA2]
    8) Migrate to partner link CN3- //a mobBPEL ODE engine that exposes "multiplyWS"
    9) compute [CA3]:multiply test result using local "multiplyWS" //to verify SA3
    10) update mobBPEL script state data; [D] = [D] || [CA3]
    11) Migrate to partner link MT //business case completed; invoker analyze result
Postcondition: math-tutor receives computed answers & award mark to quiz participants
```

5.3 Test situation description

"MT" presents a quiz to all virtual students (for workload simulation) in G (having 100, 1000, 10,000 virtual students as three test runs noted in row 2, 3 and 4 of

Table 1 Test results

Number of user-objects Transmitted at a time (G size)* [C1]	CASE1: SOAP based Client–server interaction+ [C2]	CASE2: Apache ODE based orchestration+ [C3]	CASE3: ACIAF mobility enhanced script+ (mobbpel based) [C4]	JADE Agent+ [C5]
100	265	865	400	47
1000	1046	2085	968	93
10,000	7834	12,657	4423	312

* Transmitted from MT to a single server node CN. Transmitted data contains array of complex data type object that contains 2 integers and one string, apart other placeholder fields CA1, CA2 and CA3 to be filled by CNs

+ All times are in measured milli-seconds for each batch of user data processed 'per client-to-server node' interaction

Table 1) and they solve S1, S2 and S3 parts (time factor for this task is not measured or accounted in experimentation). Once the quiz is answered by all students in G, MT must package and dispatch question (HTML elements f1, f2) and S1 section answers (HTML element SA1) to CN1, questions and S2 section answers to CN2 and questions and S3 section answers to CN3. This involve sending "Array" of "Array" complex data type of integers having four attributes f1, f2 and SA1 or SA2 or SA3 and CA (of type integer) of array size 100 using SOAP protocol to CN1,CN2 and CN3 to compute correct answers "CA". For example CN1 replies back to MT by sending back the same array of array with one more field 'CA' filled out that indicate the correct answers computed by CN1. Upon receipt of reply from compute nodes, MT can compare SA1 with CA (computed by CN1), SA2 with CA (from CN2) and SA3 with CA (from CN3) and proceed with declaration of results. This way client workload, network data transfer, server side processing is simulated.

The parameter measured here is batch size of data sent from MT to CN1/CN2/CN3 and its computation and turnaround time to get back to MT with an answer. The time is measured and tabulated "per client to a server interaction" in CASE1 and CASE2. Full MATHTUTOR case study application execution time that spans CN1, CN2 and CN3 from MT need not be measured for CASE1, CASE2 as it do not offer any additional insight (However this assumes after the agent. If overall execution time is indeed required, it can be arrived at by arithmetic addition of each MT-CN interactions in the case study. The "per client to a server interaction" time is calculated from the moment data is submitted (as SOAP message) from MT to CNs and receipt of the reply from CNs and it is tabulated for each group size and test scenarios CASE1 and CASE2. In case of CASE3 per node time is computed based on time taken to migrate from one server node to another server node and time spent on a given compute node carrying out computation. Thus all measurements made and reported here are indicative of underlying computational issues and may not extrapolate to system level behavioral issues.

In CASE3 math tutor partner link node just initiate the mobility script by submitting quiz answer data packets and wait for the arrival of finished computation card (as a message in asynchronous mode) unlike client–server model where it orchestrates each steps as in typical web-based online examinations. The observed elapsed time durations are noted and reported in Table 1.

This experiment is useful to validate the "code mobility" aspects of ACIAF application. However it can be seen that this test environment can be extended to test "loose coupling" aspects of ACIAF application as well. This can be done by providing clustered environment on CN1, CN2 and CN3 and by providing different versions of addition, subtraction and multiplication services with same functionality in each of these clusters so that the client orchestrator (middleware) can choose any one of the service (using speech acts) at run time based on service meta data such as QoS, Cost and performance.

There exist other software development environments that also can claim to provide mechanism and framework to realize mobility based applications. Traditional agent technology belongs to that category. This work chooses Java Agent Development Environment (JADE) (Bellifemine et al. 2005) for qualitative comparison. JADE is a popular traditional agent platform. Using JADE agents MT, CN1, CN2 and CN3 functionality are realized on a single machine and CASE3 situation is enacted on G

size of 100, 1000 and 10,000. The time duration is measured and tabulated in column C5 of Table 1. While ACIAF modeling, development and testing environment used services standard tools and technologies, JADE environment used a more traditional third generation development, testing and monitoring environment. The overall development experience with ACIAF was of full satisfaction as it used industry standard SOA compliant tools for design, testing and monitoring.

6 Architectural and technological assessment of ACIAF

6.1 Criteria for choosing modeling paradigms and evaluation of their software engineering properties

Efforts to arrive at consensus on orchestration languages is elaborate (Aalst 2003). WS-BPEL is industrially popular business orchestration language and is compatible with Business Process Modeling Notation (BPMN) (BPMN 2016). Typically any suggestions to enhance these languages needed to be meticulously presented and analyzed to ensure absolute necessity of the extension (Aalst 2003). Architectural

Table 2 Comparison of architectural properties of software modeling paradigms

Architectural properties (C1)	Client–server Java/ C++ (C2)	Web services/ BPEL (C3)	Traditional agent technology (C4)	Proposed fusion-ACIAF (C5)
Autonomy	−(very poor)	−	++(very good)	++
Conceptual Depth	−	−	++	++
Delegation	−	−	++	++
Layering	−	+	+	++
Maintainability	−	+	−	++
Modelability	+	+	−	++
Modifiability	−	+	−	++
Performance	−+	+	+	++
Portability	−	++	−	++
Reliability	−	+	+	++
Scalability	−	+	−	++
Security	−	++	−	++
Simplicity	−	+	−	++
Specifiability	+	+	−	++
Visibility	−	+	−	++
Workflow	−	−+	+	++
Modeling notation	+	++	−	++
Software engineering tools	+	++	−	++
Openness and fluidity	+	++	−	++
Interoperability	+	++	−	++
Easy accessibility	++	++	−	++
Maturity	++	++	−	+−

properties (Sudeikat et al. 2004; Medvidovic and Taylor 2000) of the technologies govern the nature of the end product realized, its quality and their fundamental behavior. Table 2 lists some important architectural properties desired of modern information system. They originate chiefly based on stakeholder's experience and felt need. Attributes like autonomy, conceptual depth and delegation originates from the need expressed by agent technologists while others originate based on ISO/IEEE standard on Systems and software Quality Requirements (ISO25010). The quality attributes of a system depict the degree to which the system satisfies the stated and implied needs of its various stakeholders. A technology possessing very strong (++) architectural primitives for autonomy shall allow the application to adapt to plan for reaching the goal at run time based on the perceived environment. This aspect is known as planning in Artificial Intelligence (AI). Conceptual depth permit architectural constructs to capture mental actions and capacities on system design such as concurrency, speech acts, compartmentalization, migration and transformation of code and data. Delegation permit divide and conquer of a large task and facilitate composition. Maintainability results from hierarchical composition of participating components, encapsulation of related item together leading to lesser coupling and improved cohesion. Modelability deals with pictorial specification of user requirements using standardized stereotypes and step-wise refinement up to implementation and deployment phase. More importantly usage of same entity/property/role names permit easy stepwise refinement, back-and-forth refinements, cross verification, validation and iterative enhancement across life cycle deliverables.

Many criteria were chosen in Table 2 for architectural reasons (Medvidovic and Taylor 2000) and while some others were chosen to be in conformance with latest software quality standards (ISO25010 2011). The ISO/IEC quality attributes are derived based on quality in use and static and dynamic software product quality criteria. These defined characteristics of software systems form a consistent terminology for specifying, measuring, and evaluating software product quality. Some other criteria were chosen on the basis of authors vast experience in building information systems, agent-based applications and in analyzing interoperability approaches. They include the following:

- Software engineering: This criteria emphasis support for life cycle stages of the application development such as analysis and modeling, design, implementation and deployment, monitoring and testing using industrially popular CASE tools and technologies.
- Notation: This criteria emphasis the availability of industry standard notations to describe the models and deliverables that help the architects, designer and developers in designing and understanding systems. Furthermore, applicability, availability and use of a formal notation to allow a deeper study of the system, with automatic tools (such as based on Petri net or automata), ability to discover and define dependencies and collaboration requirements.
- Openness and fluidity: This criteria emphasis possibility to successfully use it in environments and scenarios where only few dependencies and integration issues

are known at the design time. For example full adoption of XML technology shall satisfy this requirement.

- Interoperability: This criteria emphasis the capability to exploit different technology platforms and hardware/software environments.
- Easy accessibility: This criteria emphasis how convenient and easy for the users (such as developers and end users) to purchase, deploy and use in real life. Certain factors include professional support, bug fixes, continuous development and integration of emerging new technologies defines ease of accessibility. Popularity of programming language, regular maintenance and up gradation of underlying technology, industry feedback also provides a measure easy accessibility.
- Maturity: This criteria includes models, interaction, coordination, and documentation. The technology should be simple to set-up, program, deploy and test. It should meet industry quality standards and should have well-known theoretical models behind them. Online download availability, details on installation procedure, multi-platform support and good documentation also provide a measure of maturity.

To explain the above table, current agent technologies (such as Jason AgentSpeak, 3APL, JADE/WADE, Jadex, Retsina) perform poorly in technological aspects due to their arcane programming/design/implementation environment. They do not have necessary software engineering properties that can be scaled up to realize the proposals given in (Brazier et al. 2009; Okouya et al. 2013). To list few aspects take the case of: Maintainability (hardwired to ports, configuration parameters like DBMS, binary level machine specific messaging), Modelability (no step-wise life-cycle wide refinement), Modifiability (cannot run seamlessly multiple versions), Performance (cannot scale to several nodes on demand on load factor), Portability (platform specific data/message formats), Scalability (on demand addition of several computing node), Security (no standard compliant authentication, encryption, non-repudiation), Simplicity (low level programming constructs, no XML messaging, Service Adaptor bus, Business rule enforcement at message level, queue integration, no versioning), Specifiability (not possible to capture enterprise business environment, draw a business workflow logic and refine down to a working system), Visibility (what transpires at wire level between working components cannot be logged and monitored in a generic fashion). This results in ignoring vast majority of technological infrastructures developed that are widely adopted by industry. However primitives of agent technology like design using mental (speech) acts gives it richer conceptual representational/implementation power of realized system, delegation and autonomous action based on situation awareness that is built into the architectural framework itself obviating need for programmers to plumb in ad-hoc solutions into their code.

Similar line of reasoning can be applied to web services technologies and their pros and cons can be understood. One distinguishing aspect is its standard compliance and platform neutrality retaining all other traits. Web service technology separates component representation and implementation. Message based invocation of their interfaces simplifies complexity of connector semantics

(invocation) and permits scalability and deployability of components in server environment. This kind of layering permits intermediaries—proxies, gateways, firewalls, host replicators and address substitutions —to be introduced transparently to improve performance via large-scale shared caching. From the above discussion it is clear that proposed ACIAF provides desirable architectural modeling primitives for novel data information system applications.

Evaluation of architectural properties of ACIAF (column C5 of Table 2) shows software engineering characteristics of applications developed in ACIAF architecture. It leads to major gain and improvement to software engineering of mobility based internet applications in comparison with the existing state of the art AOSE technologies (Sudeikat et al. 2004). Pluses stands for having the desired property on the stated constraint by the software architecture under evaluation while minus stands for the absence of the same. Considering mobile code as a web service makes it scalable by leveraging Enterprise service bus technology (Schmidt et al. 2005) for providing fault tolerance, transaction support, process migration, load balancing (multiple instances to share load) and easy monitoring. It is also simple to query agent capability (as they are exposed as port types/WSDL), interoperable and platform neutral because it sits above web services stack that offer state of the art interoperability, scalability and security solutions. However at times (for large volume, highly message based application) performance WS technology may suffer as it has to cross all these interoperability software layers to perform business logic whereas native java technologies score/perform efficiently here.

Coding work flow logic of the agent using WS-BPEL makes the agent itself as a web service leading to all the positives we just discussed above. Also WS-BPEL has been widely accepted at the enterprise scale application environments and has hugely installed base compared to other agent environments. This would facilitate seamless integration of new ACIAF technology with the existing environment with negligible additional expenses on software, hardware, reuse, maintenance, training and deployment expenses. Writing an efficient new generation real world end-user internet-applications using ACIAF such as social networking based opinion pools (like who is the most famous pop musicians), Travel booking software applications that accomplish complex multiple itinerary fixing, hotel/car reservation etc. based on user constraints are possible. This new breed of ACIAF applications has desirable properties that present day applications do not possess.

It should be noted that ACIAF as a programming language framework provides a ready path to realize applications that support process migration to heterogeneous platforms. It enables programmers to design network efficient compute node proximate business data processing applications. However ACIAF capability is very restricted in comparison to the capability of a full-fledged traditional agent environment. Theoretically proposed and anticipated agent environments have much richer features and functionality (Brazier et al. 2009; Okouya et al. 2013). *Thus ACIAF application is NOT presented here as a full-fledged agent application but as a novel mobility enabled internet application.* However what is argued here is that ACIAF has all necessary ingredients for extensibility to a full-fledged multi-agent platform at a later date. Thus based on the criteria and evaluation presented here it is concluded ACIAF technology offers better solution.

6.2 An evaluation of basis of ACIAF programing language paradigm

A skeptical reader or a super specialist may object to the choice of programming paradigm adopted in realization of AICAF framework in the first place. For example Jade and Jadex developers and users might contest that Java (OOAD approach) based development and programming environment is the best suited for agent metaphor, while JaCaMo developers may argue why their declarative agent environment is most suited. Hence an evaluation and comparison of programming language technologies for adoption to realize the framework and platform is taken up and an assessment is given in Table 3. It may be noted that today modular, object oriented, aspect oriented, concurrent and distributed, service oriented and agent oriented programming approaches are popular.

From the above comparison, it is clear that benefits offered by agent oriented programming environment in enormous. So to realize robust information systems it is essential to leverage its principles in structuring and design of information systems. Hence we embrace principles of agent oriented programming in combination with service technology to realize ACIAF technology.

6.3 Technical comparison of ACIAF applications with other mobility aware agent environments

This part undertake to compare on the hypothetical case of traditional agent platforms trying to provide loosely coupled, mobility based application technology for business data processing in competition with ACIAF applications. In such case they will be directly competing with ACIAF capabilities. In such case to bring out the best qualities of these environments, a comparison needed to be made. For this comparison some traditional agent platforms needed to be taken up. Purely research and exploratory platforms that treats agents as a discipline of Artificial Intelligence were not considered in this comparison. Given this criteria, this work select three platforms: Jade, Retsina, Able. Jadex though very popular like Jade is an add on layer over Jade and hence share all its core properties and hence not chosen for comparison. The list of agent capabilities on the first column is selected based on our own industrial experience of developing business information systems. In our opinion these criteria should be fulfilled in the most technologically savvy manner to make the environment usable in business deployment. The Table 4 depicts how different agent environments address and fulfill these criteria. It depicts how applications developed using ACIAF has several desirable traits that are very hard to put into other agent environments. Thus it demonstrates ACIAF as a new progression/generation in the software development paradigm that cannot be substituted/taken over by other agent environments even after additions to their capabilities. Since Jadex is simply an add-on layer over JADE platform it inherits all its strength and weaknesses. Jadex adds a Belief-Desire-Intension rational agency layer over JADE platform. Hence it is not separately shown in the comparison as most of the observations made on JADE platform shall be applicable to it as well.

Table 3 A comparison of attributes of modern programming language technologies

Dimension addressed	Modular programming	Object oriented programming	Aspect oriented Programming	Concurrent and distributed Programming	Service oriented programming	Agent oriented programming (existing and proposed)
Structural unit	Functions	Classes	Aspects (cross cutting across class hierarchy)	Threads, processes	Services	Agents
Structural constructs used in weaving and metrics	Coupling and cohesion	Abstract classes, encapsulation inheritance, polymorphic functions	Joinpoint, pointcut, aspect and mixin	Mutex, semaphore, shared memory, message queues	Queued messages, service flow, substitution	Cognitive messages, initiatives, group belief sharing
Communication unit	Function arguments	Messages and events	Injection	Events and messages	Messages, flows	Speech acts
Composition method	Coupling, cohesion	Association and inheritance	Interception	Embedded into program design style	Orchestration and choreography	No explicit programmatic composition needed, coalitions emerge based on commitments, speech acts and normative behavior
Integration method	Compile time or run time–binary level	Compile time or run time–binary level	Compile time or run time injection–binary level	Messages or shared memory	Enterprise service bus based–message based	Cognitive message based–zero dependency
Viewpoints addressed	Programmer	Actors, programmers, software architect	Programmer and software designer	Architect, designer, programmer, developer	Actor, consumer, provider, broker, integrator, deployer	Entire society–every participant viewpoints are addressed
Delivery/packaging unit	As library	As components	As cross cutting layer functionality realized as classes	Re-entrant code block or class	As services	As agents

Table 3 continued

Dimension addressed	Modular programming	Object oriented programming	Aspect oriented Programming	Concurrent and distributed Programming	Service oriented programming	Agent oriented programming (existing and proposed)
Solution units	As Library or applications	As reusable patterns	As features (class library)	Thread library	As reusable service choreographies	As agent societies

Table 4 A comparison of ACIAF with some traditional agent environments

Agent capability	Agent environment			
	ACIAF	JADE/WADE/ Wolf	RETSINA	ABLE
Agent coding language	WS-BPEL + its extensions	Java	Java/C/C ++	Java
Communication/ message transport	SOAP based + ESB support + speech acts	ACL	KQML	Bean events, Proprietary messages
Workflow infrastructure	WS-BPEL	Custom coding	Custom coding	Custom coding
Multi-agent capability	SOAP & WS-BPEL & speech act	Through ACL	Speech act based environment	Nil
Monitoring	Normal WS monitoring tool	Proprietary tool required	Proprietary tool required	Proprietary tool required
Agent debugging	WS-*/WS-BPEL debugger	Native Java IDE debugging	Native IDE debugging	Native Java debugging
Tracing/logging	HTTP/SOAP/TCP logger	Proprietary tool	Proprietary tool	Proprietary tool
Directory/content language	W3C standard OWL-S	Proprietary	Proprietary	Proprietary
Agent deployment environment	WS-BPEL extended engine	Java + agent infrastructure	Java + agent infrastructure	Java + agent infrastructure
		Proprietary	Proprietary	Proprietary
Services standard compliance	Full. Leverage new development in Services technology	Proprietary. Hard to leverage new developments	Proprietary env., hard to leverage new developments	Proprietary env., hard to leverage new things
Security support	In Built (WS-Security,WS-Encryption)	Proprietary	Proprietary	Proprietary
Versioning support	Full/Built-in	Proprietary	Proprietary	Proprietary
Formal concurrency model of application business logic/ business interaction	Elegant petri-net, automata models exist; formal proof possible	None. There can be no direct translators possible	None. There can be no direct translators possible	None. There can be no direct translators possible
Formal behavior verification	Since based entirely based on XML technology fully formally specifiable/ verifiable and adds to infrastructure	Very hard. May be done at code level explicitly by programmer added assertions	Very hard. May be done at code level explicitly by programmer added assertions	Very hard. May be done at code level explicitly by programmer added assertions
Autonomy/ decoupling	None in ACIAF; good in ACIAF-with ESB integration	Good	Good	Poor

Table 4 continued

Agent capability	Agent environment			
	ACIAF	JADE/WADE/ Wolf	RETSINA	ABLE
Declarative composition	No	No	No	No
Inferential capability	Yes through OWL-S annotation of in ACIAF agents	Jadex has	Yes	Yes
Temporal continuity	Very good; standards based	Proprietary	Proprietary	Proprietary
Capability advertisement	Very good (through WS-inspection)	Proprietary	Proprietary	Proprietary
Future ready? (semantic web)	Very good (through OWL-S)	Not ready	Not ready	Not ready
Business rule integration as part of middleware and application CIM	Yes	No	No	No

7 Formal analysability of ACIAF application mobility behaviour

Advantages of using formal models of a business processes is to reduce room for vagueness (and interpretation) and thereby limiting their erroneous behavior/usage. Variants of Petri nets (Open workflow nets), have proper formal semantics and, hence, one can check relevant and interesting properties on corresponding business models through model checking (Fahland et al. 2011; Müller 2010). Open workflow nets can capture control-flow details, of advanced constructs such as cancelation regions and OR-joins of WS-BPEL and offers faithful complete translation of logic of WS-BPEL/mobBPEL agent code and also data flow dependency (Müller 2010; Awad et al. 2009; Lohmann et al. 2010). We refer to the literature on the transformation and formal analysis of WS-BPEL code for desirable properties and for their having deadlock free property based on patterns (Lohmann 2007, 2008; Fahland et al. 2011). There are tools for formal soundness checking of WS-BPEL and we refer to them without elaboration. For example Rachel tool (Lohmann et al. 2008) detects, repairs deadlocking WS-BPEL service choreographies. BPEL2oWFN tool (Lohmann 2007; Müller 2010) translates WS-BPEL processes into a Petri net model. It implements a feature-complete Petri net semantics of WS-BPEL 2.0 that in turn can be verified for control flow properties. Here we limit our task to show the extensions provided to WS-BPEL to provide it to have agent-like-behavior does not violate its formal semantics and still offers a formal means of soundness/liveness/safety property verification of the underlying business process code. This leads us to have formal faith in our suggested extension to WS-BPEL as it has mechanism for formal analysis/interpretation/translation and testing. This provides us an algorithm for informing mobBPEL designer/programmer if their

intended mobility construct is well formed within the context of BPEL application syntax (but not from that of business logic semantics). Otherwise designer can be informed through error messages at the design time itself if their models are erroneous (such as premature migration to partner link before completing all pending activities at the node).

As there is a set working software tools for WS-BPEL flow verification using Petri-nets, this work propose to adopt them and extend to verify proper timely migration of ACIAF agents without any control/data flow anomalies. This verification to be incorporated at the static time verification itself (in ODE BPEL compiler at design time) and dynamically in partner links before reloading of the ACIAF agent (to check if it indeed received an agent that does not show erroneous mobility property). The formal details of this approach needs to presented separately in another work.

7.1 Formal Verifiability of ACIAF application mobility behavior (verifiable mobility theorms)

To simplify the model and analysis, a restricted set of ACIAF programs that use only WS-BPEL synchronous calls without compensation and transaction is taken up here for discussion. Graceful termination and reloading of ACIAF scripts needs to be mathematically modeled and proved. Tools developed for Petri-net API (Lohmann 2010) aid in this verification.

We state the basic ACIAF program mobility (or migration) theorem for load and reload cases without proof or elaboration here. The mathematical proof of them shall be presented elsewhere.

Theorem 1 (Sync-Migrate) *Given a synchronous call only ACIAF mobBPEL script, it is always possible to mathematically determine from script model itself about the graceful mobility of the script without any data flow/control flow anomalies.*

Theorem 2 (Reload-Sync-Migrate) *Given a migrated ACIAF mobBPEL script, it is always possible to determine mathematically based on script model alone about the absence of data flow/control flow anomalies in it and if it is fit for reload and re-activation.*

These theorems can be extended to agent cases with asynchronous calls, transaction and compensation and other complex flow patterns, but is far more complex and need an elaborate mathematical thesis based on Petri-net models and automata theory.

7.2 Fundamental theorem on behavioural analyzability of cooperative computing environment

This theorem is about formal behavioral analyzability of individual applications in a group of ACIAF applications. ACIAF programs deals with data types that have rigorous XML syntax such as XSD. Elements/Attributes/Properties of XSD types

constitute basic ontology of the application. These XSD types can be related to each other using RDF triples. As each WS can be sematically annotated using OWL-S technology, so do all BPEL scripts and AICAF scripts. Any logical composition of these annotations of a group of such operations that may span across several programs in a particular order (such as union, intersection, negation and projection) can be reasoned using ontology reasoning/inference engines such as Pellet/Fact++ (Koutsomitropoulos et al. 2008).

It is already argued that real life enterprise software applications are composed of collaborating externally provided services. Let an ACIAF program (simply called as "program" hence forth) community called as **C1** is composed of independent programs/applications/agents named **A1, A2, A3**… up to…**An**. Let each of the programs/applications provide a unique non-overlapping business functionality. This set of programs collaborate together to provide a business functionality. Example of such cooperative computation includes travel itinerary framing that involves air ticket, hotel and car bookings each of which carried out separate applications but depend on each other's results. The set of these programs all belong to the same community in the sense that all well-formed sentences of those programs that describe capabilities of individual programs belong to the same first order logic (or OWL-DL) based language L (i.e. all the programs shares same alphabets and interpretation; they all are defined by the same model). Let these **A1, A2, A3**… up to… **An** are provisioned as independent web services each having different URIs.

Assume all these programs carryout non-overlapping responsibilities (i.e. programs are non-superfluous)—that is they are orthogonal. In other words each program carry out unique work not possible by other programs. An enterprise application must divide a work among its collaborating partner links from say **A1** to **Ar** programs as it is not possible to carry out all work by itself alone.

Assume each program carryout **m** number of responsibilities (called conversation) **r1, r2,**… up to r**m**. Each responsibility **r1** or **r2**… or **rm** is further divided into say **i** number of speech acts (i.e. loosely coupled composition operation in orchestration) **sa1, sa2**… **sai**. The action performed by each speech acts can be stated with mathematical rigor. Vieira et al. (2007) has showed formal semantics enjoyed by several speech acts in a typical multi-agent programming environment (in ACIAF case it uses only "ask" and "tell"). It may be noted that each responsibility of a program (i.e. web service) corresponds to an operation. Martin et al. (2004a, b) discusses about encoding each operation (referred to as Atomic process) of a web service in terms of logical equations using inputs, outputs, preconditions, and effects (IOPEs). To achieve this goal experience states that a combination of OWL-S, OWL-DL and sematic web rule language technology is required. So it can be assumed that each operation of the port types is available as a logical equation amenable for inference.

Thus a program can be specified symbolically as **A1 {r1 {(sa1, sa2,**…, **sai)},**…**to**… **rm {(sa1, sa2,**…, **sai)}}** up to **An** {…} where **A1, A2,**… up to **An** constitute individual autonomous non-superfluous ACIAF applications or agents.

Assume that each capability carried out by *each* operations in a web service port type and its functionality can be coded by **n** number of first order logic/OWL-DL

statements in the language **L** in the form **pre1 ^ stmnt1 ^ stmnt2**… **^ stmnt-x -> post1** (in the case of operation 1 of web service A1).

Thus all the speech acts can be translated to first order logic/OWL-DL statements and each program's responsibilities also can be translated into a first order logic/OWL-DL statement in **L**. The following theorem can be stated and proved about the behavioural property of this community C1.

Theorem *Assume that this community **C1** reports a statement **Sx** then as per theory of first order logic (and hence that of OWL-DL + SWRL) and Craig's interpolation theorem for first order logic (and its extensions to Modal logics by Johan Hans Van Bentham), it is possible to trace which programs were responsible for collaboration and in which order they collaborated and arrived at **Sx**.*

The above theorem follows from the work of Keisler and Keisler (2014). This property has paramount importance in cooperative computation for security reasons (in cases like electronic commerce). This capability permits reconstruction of reasoning chain of collaborating partner services. The semantic web services technology (ontology based OWL-DL markup of WS capabilities) and WS-BPEL (and ACIAF) that enjoys necessary formal properties for composition and distributed processing makes this kind of formal attributability and verifiability of action possible. Other agent programming languages (or distributed computing platforms in general) such as Jade/Jadex, JaCaMo/Jason AgentSpeak, 3APL and will not be able to meet this requirement and demonstrate this capability in such an elegant and infrastructure leveraging manner. Another interesting property of this capability is each such cooperative group of application in turn contribute to evolution of the infrastructure itself instead of being simple beneficiary. The formal details of this theorem in ACIAF environment needs to presented separately in another work.

8 Conclusion and future work

ACIAF framework demonstrates its ability to provide a better architectural basis to construct loosely coupled mobility based internet applications that is standard based, efficient, extensible, modular, easily composed, offer better user experience, increase service efficiency, reduce service complexity and network traffic and result in simpler/elegant software model. The dynamic WS-BPEL process composition (based on ontology markups meta data) with migration capability is certainly possible and issues relating to it needs a separate study by using literature in dynamic semantic web service selection and composition. The loose coupling technique incorporated in ACIAF will have a significant impact in internet of things-IOT-domain where an ACIAF server can loosely partner with an unlimited number of clients (though manipulating destination addressees to point to many class of network addresses in destination filed of WS-A messages) and carryout monitoring and control of them. This capability needs to be demonstrated separately. The agent script can be translated into a Petri-net model and verified statically to determine its safety/liveness property. Cases where asynchronous

partner link execution is carried out, long running transactions, compensation are not addressed in this work. Effect of those properties on migration needed to be studied separately. This model of computation has challenges to address due to trust and security factors. Many organizations see it as a potential risk to permit third party code to run on their computing nodes without necessary testing due to the fear of unknown; however by using a suitable trust and security certificate regime trusted partner code can be permitted to execute in their compute nodes. Incorporating this security dimension into framework remains to be studied in future. This environment is well suited for implementing Big Data processing workflow. A suitable case study and its demonstration needs to be undertaken in near future. This work demonstrated only incorporation of mobility tag to WS-BPEL scripts and work is underway to incorporate loose coupling in Apache ODE engine. Subsequently it needed to be applied to a real life situation through a case study and to be recorded as a separate work. This work suggested loose coupling using some limited kind of open interaction infrastructure similar to the one suggested by Okouya et al. This aspect needs to be studied in detail in future works. Since WS-BPEL applications are fully XML compliant, a semantic processing layer (using OWL-S) can be added over it and it remains to be shown what kind of semantic business use cases can be developed using ACIAF framework. The integration of semantic web services capability in content language of ACL embedded within WS-A messages, and using it for orchestration is a possibility that helps ACIAF middleware to act as a logical reasoning based open interaction infrastructure to determine service partner links at run time. This is a great possibility but the technology is now ripe to practically realize it and ACIAF has the technical means to show case it practically. This needed to be demonstrated in a separate work in future.

References

Aalst WMP (2003) Don't go with the flow: Web services composition standards exposed. IEEE Intell Syst 18(1):72–76

Aalst WMP (2005) Pi calculus versus Petri nets: let us eat humble pie rather than further inflate the Pi Hype. BPTrends 3(5):1–11

Aalst WMP, van Hee K (2004) Workflow management: models, methods, and systems. MIT Press, Cambridge. ISBN 978-0262720465

Aberg C, Lambrix P, Shahmehri N (2005) An agent-based framework for integrating workflows and web services. WETICE'05 IEEE. doi:10.1109/WETICE.2005.17

Anderson R (2000) The end of DLL Hell. MSDN Magazine, San Francisco

Aversa R, Di Martino B, Venticinque S (2009) Integration of mobile agents technology and globus grid. IEEE Int Conf Comput Sci Eng. doi:10.1109/CSE.2009.121

Awad A, Decker G, Lohmann N (2009) Diagnosing and repairing data anomalies in process models. In: Rinderle-Ma S, Sadiq S (eds) BPM 2009 international workshops. Springer, Ulm

Banzi M, Caire G, Gotta D (2008) WADE: a software platform to develop mission critical applications exploiting agents and workflows. AAMAS 2008 Industry Track

Barbacci M, Longstaff TH, Klein MH, Weinstock CB (1995) Quality attributes. Technical report, CMU/SEI-95-TR-021, ESC-TR-95-021

Bass L, Clements P, Kazman R (2003) Software architecture in practice, 2nd edn. Pearson education, Upper Saddle River. ISBN 81-77589962
Bellifemine F et al (2005) Jade: a java agent development framework, multi-agent programming. Springer, Vol 15, II, pp 125–147. doi:10.1007/0-387-26350-0_5
Bigus JP et al (2002) ABLE: a toolkit for building multiagent autonomic systems. IBM Systems Journal 41(3):350–371. doi:10.1147/sj.413.0350
Boissiera O, Bordini RH, Hübnerc JF, Riccid A, Santid A (2013) Multi-agent oriented programming with JaCaMo. Sci Comput Program 78(6):747–761. doi:10.1016/j.scico.2011.10.004
Bordini RH, Hübner JF, Wooldridge M (2007) Programming multi-agent systems in agentspeak using Jason. Wiley-Blackwell, Hoboken. ISBN 0470029005
Bordini RH et al (2009) Multi-agent programming: languages, tools and applications. Springer, New York. ISBN 0387892982
Bpel (2016) Open source BPEL engine. http://ode.apache.org/
BPMN (2016) Business process modeling notation v 2.0. http://www.omg.org/spec/BPMN/2.0/PDF/ (also at: http://www.bpmn.org)
Bradshaw JM (ed) (1997) Software agents. AAAI Press, Cambridge, pp 1–46. ISBN 978-0262522342
Brazier FMT, Kephart JO, Parunak HVD, Huhns MN (2009) Agents and service-oriented computing for autonomic computing: a research agenda. Internet Comput IEEE 13(3):82–87. doi:10.1109/MIC.2009.5
Breugel F, Koshkina M (2006) Models and verification of BPEL. http://www.cse.yorku.ca/~franck/research/drafts/tutorial.pdf
Brown WA, Laird R, Gee C, Mitra T (2008) SOA governance. IBM Press, Indianapolis. ISBN 0137147465
Bughin J, Chui M (2010) The rise of the networked enterprise: Web 2.0 finds its payday. McKinsey Quarterly, Seattle
Chi J, Song J (2007) Intelligent-agent and web-service based service composition for E-business. CCECE. doi:10.1109/CCECE.2007.215
Chunlin L, Zhengding L, Layuan L (2003) Design and implementation of a hybrid agent platform. Programm Comput Softw 29(1):28–42. doi:10.1023/A:1021915913509
Constantinides E, Fountain SJ (2008) Web 2.0: conceptual foundations and marketing issues. J Direct Data Digit Mark Pract 9:231–244. doi:10.1057/palgrave.dddmp.4350098
Cooney D, Roe P (2003) Mobile agents make for flexible web services. http://ausweb.scu.edu.au/aw03/papers/cooney/paper.html
Covington MA (1997) Speech acts in electronic communication with special reference to KQML and ANSI X12. In: 13th conference on system sciences. ISBN 0-8186-7862-3/97
Dastani M et al (2010) Specification and verification of multi-agent systems. Springer, New York. ISBN 978-1-4419-6984-2
Ebrahim N, Iyer SR, Punathil G, Reghunath R (2015) Identification of critical web services and their dynamic optimal relocation. Patent numbers: US 8990388 B2, US20120124193, US20130013774, USA Patent, 2015
Erl T (2005) Service-oriented Architecture. Prentice Hall PTR, Upper Saddle River. ISBN 0-13-185858-0
Fahland D, Favre C, Koehler J, Lohmann N, Völzer H, Wolf K (2011) Analysis on demand: instantaneous soundness checking of industrial business process models. Data Knowl Eng 70(5):448–466
FIPA (2016) Foundation for intelligent physical agents. http://www.fipa.org/
FIPA-ACL (1998) FIPA specification-agent communication langugae: Part 2. FIPA, 1998. www.fipa.org/repository/aclspecs.html
Gable GG, Sedera D, Chan T (2008) Re-conceptualizing information system success: the IS-Impact Measurement Model. J Assoc Inf Syst 9(7):377–408
Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns. Addison-Wesley, Boston. ISBN 0201633612
García R (ed.) (2008) Semantic web for business: cases and applications. Information Science Reference, 2008. ISBN 978-1-60566-067-7
GeneOntology (2001) The gene ontology consortium, creating the gene ontology resource: design and implementation. Genome Res 11:1425–1433. doi:10.1101/gr.180801
Grunbacher P, Egyed A, Medvidovic N (2003) Reconciling software requirements and architectures with intermediate models. Softw Syst Model. doi:10.1007/s10270-003-0038-6
Gu Q, Lago P (2009) Exploring service-oriented system engineering challenges: a systematic literature review. SOCA 3:171–188. doi:10.1007/s11761-009-0046-7

Guan S-U, Guan S-U, Tan SL, Hua F (2004) A modularized electronic payment system for agent-based E-commerce. J Res Pract Inf Technol 36(2):67–87
Havey M (2005) Essential business process modeling. O'Reilly, Cambridge. ISBN 0596008430
Hollingsworth D (1995) The workflow reference model (Workflow Management Coalition (WfMC)), Document Number TC00-1003, Issue 1.1, 1995. http://www.wfmc.org/standards/docs/tc003v11.pdf
Huebscher MC, McCann JA (2008) A survey of autonomic computing. ACM Comput Surv 40(3):7
Huhns MN (2002) Agent as web services. IEEE Internet Comput 6(4):93–95. doi:10.1109/MIC.2002.1020332
Huhns MN, Singh MP (1998) Workflow agents. Internet Comput IEEE 2(4):94–96. doi:10.1109/4236.707813
Huhns MN, Singh MP (2005a) Service-oriented computing: key concepts and principles. Internet Comput IEEE 9(1):75–81. doi:10.1109/MIC.2005.21
Huhns MN, Singh Munindar P (2005b) Research directions for service-oriented multiagent systems. Internet Comput IEEE 9(6):65–70. doi:10.1109/MIC.2005.132
ISO25010 (2011) ISO/IEC 25010:2011 standard: Systems and software engineering—systems and software quality requirements and evaluation (SQuaRE)—System and software quality models, 2011
JadexTool (2016) Jadex agent programming environment. http://www.activecomponents.org/
Juric MB, Sasa A, Rozman I (2009) WS-BPEL extensions for versioning. Inf Softw Technol 51(8):1261–1274. doi:10.1016/j.infsof.2009.03.003
Kamel Boulos MN, Wheeler S (2007) The emerging Web 20 social software: an enabling suite of sociable technologies in health and health care education. Health Information & Libraries Journal 24(1):2–23. doi:10.1111/j.1471-1842.2007.00701.x
Kamngar F, Levine D, Zaruba GV, Thomas R (2005) Mobile agent connection establishment and management (CEMA)—message EXCHANGE for pervasive computing environments. J Supercomput 31:79–99. doi:10.1023/B:SUPE.0000049326.25067.80
Katasonov A, Terziyan V (2008) Semantic agent programming language (S-APL): a middleware platform for the semantic web. 2008 IEEE Int Conf Seman Comput. doi:10.1109/ICSC.2008.82
Kazman R, Bass L (1994) Toward deriving software architectures from quality attributes. Technical Report CMU/SEI-94-TR-10
Keisler HJ, Keisler JM (2014) Observing, reporting and deciding in networks of sentences. Ann Pure Appl Logic 165(3):812–836. doi:10.1016/j.apal.2013.10.008
Kotis K, Vouros GA (2006) Human-centered ontology engineering: the HCOME methodology. Knowl Inf Syst 10(1):109–131. doi:10.1007/s10115-005-0227-4
Koutsomitropoulos DA, Meidanis DP, Kandili AN, Papatheodorou TS (2008) Establishing the semantic web reasoning infrastructure on description logic inference engines. In: Manolopoulos Y et al. (Eds.) ICEIS 2006. LNBIP 3. Springer, pp 351–362
Lettmann T et al (2011) Modeling agents and agent systems. Trans Comput Collect Intell 5:157–181
Lloyd JW, Ng KS (2011) Declarative programming for agent applications. Auton Agent Multi Agent Syst 23:224–272. doi:10.1007/s10458-010-9138-1
Logan B (2015) A future for agent programming. In: Baldoni M et al. (ed.) Engineering multi-agent systems. LNAI 9318, Springer. doi:10.1007/978-3-319-26184-3_1
Lohmann N (2007) A feature-complete Petri net semantics for WS-BPEL 2.0. In: Heckel MDR (eds) Web services and formal methods, WS-FM. Proceedings, LNCS. Springer
Lohmann N (2008) Correcting deadlocking service choreographies using a simulation-based graph edit distance. In: Dumas M et al (Eds) Business Process Management, BPM 2008, Milan, Sep 1–4, 2008, vol. 5240 of LNCS, pp. 132–147, Springer
Lohmann N, Massuthe P, Stahl C, Weinberg D (2008) Analyzing interacting WS-BPEL processes using flexible model geneneration. Data Knowl Eng 64(1):38–54
Lohmann N, Verbeek HMW, Ouyang C, Stahl C (2009a) Comparing and Evaluating Petri Net Semantics for BPEL. Int J Business Process Integration and Management 4(1):60–73
Lohmann N, Verbeek HMW, Dijkman R (2009b) Petri net transformations for business processes—a survey. Transactions on Petri Nets and Other Models of Concurrency II, Springer 2009:46–63
Lohmann N, Mennicke S, Sura C (2010) The Petri Net API: A collection of Petri net-related functions. In: Schwarick M, Heiner M (eds) Proceedings of the 17th German Workshop on Algorithms and Tools for Petri Nets (AWPN 2010), Cottbus, Germany, October 7–8, 2010, volume 643 of CEUR Workshop, pp 148–155

Luck M, McBurney P, Shehory O, Willmott S (2005) Agent technology: computing as interaction (A Roadmap for Agent Based Computing), AgentLink, 2005. ISBN 085432 845 9. http://www.agentlink.org/roadmap/index.html

Maleshkova M, Pedrinaci C, Domingue J (2010) Semantic annotation of Web APIs with SWEET. In: 7th ESWC2010

Maréchaux JL (2006) Combining service-oriented architecture and event-driven architecture using an enterprise service bus, IBM developer works

Martin D et al (2004a) Bringing semantics to web services: The OWL-S Approach. In: Cardoso J, Sheth A (eds) SWSWPC 2004. LNCS 3387, Springer, pp 26-42

Martin D et al (2004b) OWL-S: semantic markup for web services. W3C Member Submission. https://www.w3.org/Submission/OWL-S/

Martin D et al (2007) Bringing semantics to web services: the OWL-S Approach. World Wide Web 10(3):243–277. doi:10.1007/s11280-007-0033-x

Mazeiar S, Ladan T (2009) Self-adaptive software: landscape and research challenges. ACM Trans Autonomous & Adaptive Systems 4(2):14

McKean J, Shorter H, Luck M, McBurney P, Willmott S (2008) Technology diffusion: analysing the diffusion of agent technologies. Auton Agent Multi-Agent Syst 17(3):372–396. doi:10.1007/s10458-008-9052-y

Medvidovic N, Taylor RN (2000) A Classification and Comparison Framework for Software Architecture Description Languages. IEEE Trans Software Eng 26(1):70–93

Milanovic N (2011) Non-functional properties in service oriented Architecture:Requirements, Models and Methods. Model Labs, Berlin. ISBN 978-1-60566-795-9

Müller R (2010) Formal characterisation of partners of an open net. Humboldt-Universität zu Berlin (Thesis report), Institut für Informatik

Murugesan S (2007) Understanding Web 2.0. IT Professional. doi:10.1109/MITP.2007.78

Murugesan S (2009) Handbook of research on Web 2.0, 3.0, and X.0. Information Science Reference, Hershey. ISBN 978-1605663845

Newcomer E, Lomow G (2005) Understanding SOA with web services. Addison Wesley, Boston. ISBN 0-321-18086-0

Noy NF (2004) Semantic integration: a survey of ontology-based approaches. SIGMOD Record 33(4):65–70. doi:10.1145/1041410.1041421

Nwana HS, Ndumu DT (1999) A perspective on software agents research. Knowl Eng Rev 14(2):1–18

ODE (2016) Apache ODE (Orchestration Director Engine) software (WS-BPEL v 2.0 compliant container). http://ode.apache.org/

ODEAPI (2016) BPEL-ODE Process Management API specification. http://ode.apache.org/bpel-management-api-specification.html

ODEJaCOb (2016) Apache Orchestration director engine – Java concurrency object layer. http://ode.apache.org/developerguide/jacob.html

Okouya D, Fornara N, Colombetti M (2013) An infrastructure for the design and development of open interaction systems. In: Cossentino M et al (Eds) Engineering multi-agent systems, EMAS 2013, LNCS 8245, 2013, pp 215-234. doi:10.1007/978-3-642-45343-4

Pokahr A, Braubach L, Lamersdorf W (2003) Jadex: A BDI Reasoning Engine. In: Bordini RH et al. (Ed) Multi-agent programming. Springer, pp 149–174. ISBN: 978-0-387-26350-2. doi:10.1007/0-387-26350-0_6

Pratschner S (2001) Simplifying deployment and solving DLL Hell with the.NET framework. MSDN Magazine, San Francisco

Rao AS, George MP (1995) BDI agents from theory to practice. Proceedings of the First International Conference on Multi Agent Systems, ICMAS-95, San Francisco

ResourcesAgent (2016) Historical resources on agent technology. http://agents.umbc.edu/introduction

Russell S, Norvig P (2003) Artificial intelligence, 3rd edn. Prentice-Hall, Upper Saddle River

Savarimuthu BTR et al (2005) Integrating web services with agent based workflow management system (WfMS), Web Intelligence, 2005 IEEE/WIC/ACM. International Conference. doi:10.1109/WI.2005.81

Schiemann B, Schreiber U (2006) OWL-DL as FIPA-ACL content language. In: Formal ontology for communicating agents, Malaga, Spain

Schmidt MT, Hutchison B, Lambros P, Phippen R (2005) The enterprise service bus: making service-oriented architecture real. IBM Systems Journal 44(4):781–797. doi:10.1147/sj.444.0781

Sessions R (2007) A comparison of the top four enterprise-architecture methodologies. https://msdn.microsoft.com/en-us/library/bb466232.aspx
Shehory O, Sturm A (eds) (2014) Agent-Oriented Software Engineering. Springer, New York. doi:10.1007/978-3-642-54432-3
Smith CU (2015) Software performance engineering then and now: a position paper. WOSP-C' 15:2015. doi:10.1145/2693561.2693567
Sudeikat J et al (2004) Evaluation of agent—oriented software methodologies—examination of the gap between modeling and platform. AOSE'04. Springer, pp. 126–141. doi:10.1007/978-3-540-30578-1_9
Sycara K et al. (2003) The RETSINA MAS, a case study. LNCS, 2003, Springer, Vol 2603/2003, pp.103–119. doi:10.1007/3-540-35828-5_15
Talantikitea HN et al (2009) Semantic annotations for web services discovery and composition. Computer Standards & Interfaces 31(6):1108–1117
Trione L, Long D, Gotta D, Sacchi G, WeMash W (2009) WADE: unleash the power of collective intelligence. In: 8th AAMAS 2009, pp 53–60
Venkatesan D (2010) Development of a novel software architecture for active internet applications. ICWS2010. doi:10.1109/ICWS.2010.86
Venkatesan D, Sridhar S (2016) A novel method and environment for scalable web service orchestration. In: Proceedings of IEEE 12th 2016 world congress on services computing (SERVICES 2016), San Francisco, pp 128–129. doi:10.1109/SERVICES.2016.27
Vieira R et al (2007) On the formal semantics of speech-act based communication in an agent-oriented programming language. J Artif Intell Res 29:221–267. doi:10.1613/jair.2221
Wang M, Wang H (2005) Intelligent agent supported business process management. In: Proceedings of the 38th Hawaii international conference on system sciences. doi:10.1109/HICSS.2005.332
Weerawarana S, Curbera F, Leymann F, Storey T, Ferguson DF (2005) Web services platform architecture: SOAP, WSDL, WS-Policy, WS-addressing, WSBPEL, WS-reliable messaging and more. Prentice Hall PTR, Upper Saddle River
Weiss G (2012) Multi-agent technology, 2nd edn. MIT Press, Cambridge
Wieland M et al (2008) Context integration for smart workflows. In: 6th IEEE International Conference 2008, pp 239–242. doi:10.1109/PERCOM.2008.27
Wooldridge M (2009) Introduction to multi-agent systems, 2nd edn. Wiley, New York. ISBN 978-0-470-51946-2
WS-Addressing (2006) Web Services Addressing 1.0 – Core standard, http://www.w3.org/TR/2006/PR-ws-addr-core-20060321
WS-BPEL (2007) OASIS Web Services Business Process Execution Language v 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
Xu J, Pears S (2006) A dynamic shadow approach to fault-tolerant mobile agents in an autonomic environment. Real-Time Systems 32(2006):235–252. doi:10.1007/s11241-005-4682-5
Yu L (2009) A developer's guide to the semantic web, Springer. e-ISBN 978-3-642-15970-1. doi:10.1007/978-3-642-15970-1