Ontology Driven Meta-Modelling of Service Oriented Architecture

Shreya Banerjee #1, Shruti Bajpai #2, Anirban Sarkar #3, Takaaki Goto*4, Narayan C Debnath*5

#1Department of Computer Applications, National Institute of Technology, Durgapur, India
#2bajpai.shruti1705@gmail.com
#3sarkar.anirban@gmail.com
*4Ryutsu Keizai University, Japan
tg@gotolab.net
*5Department of Computer Science, Winona State University, MN, USA
ndeb Nath@winona.edu

Abstract—Effective modelling has helped in explain, formalize and understand Service Oriented Architecture (SOA) that is a complex architecture style inherently. Yet, a serious gap is still exist in modelling inter dependency between structural and behavioural characteristics of SOA. Beside, lack of precise semantics and formalization in modelling of SOA has made serious challenges in checking consistency over SOA models. In order to address these challenges, in this paper, an ontology driven meta-model has been proposed for SOA. The proposed meta-model is conformed towards an ontology driven meta-meta model called, Generalized Ontology Modelling (GOM). It can be further restricted towards distinct models of SOA based applications. The proposed meta-model is implemented using ontology editorial tool Protégé and illustrated using suitable case study.

Keywords-Service Oriented Architecture, Ontology, Meta-Model, Upper Ontology, Meta-Meta Model

I. INTRODUCTION

The services represent reusable business functionalities. Service consumers compose applications using such services through standard interfaces [1]. Business functionalities, which are delivered by Service Oriented Architecture (SOA), are self-describing and independent of any domain [2]. SOA can consists of several artefacts such as service, participants, and relations. Thus, SOA is very complex architectural style [3]. However, modelling of SOA has several challenges. The first challenge is that a suitable model for SOA should be both business-understandable and executable. Further, services in SOA has provided business logic and data, interfaces through which, functionality of services are explored, and a service contract specifying operations and pre or post conditions [4]. Thus, the second challenge is that an appropriate model for SOA should cover both structural and behavioral characteristics of design artefacts in Service Oriented Systems (SOS). Thirdly, establishing connections and checking consistency between those characteristics is also a prime requisite. Through this the right level dependency between these distinct characteristics as well as the realization of service oriented architecture is facilitated in full fledge [5]. Towards addressing the first challenge, a model for SOA can be devised through methodology of domain specific modelling that may be both business-understandable (by high level abstraction in meta-model level) and executable (by low level abstraction in model level). Domain specific modelling languages are tailored to certain domains and is consisting of one abstract syntax, one or more concrete syntax, mapping between abstract and concrete syntax and semantics [6]. Thus, a domain model is conformed towards a meta-model (which has provided abstract syntax towards domain models) and meta-model is further conformed towards a meta-meta model (which has provided abstract syntax towards meta-models) [7]. However, mere domain specific modelling is not sufficient for adequate representations of structural, behavioral characteristics (Second Challenge), inter dependency and consistency between them (Third Challenge). Because, descriptions of domain specific modelling languages are consisting of suitable syntaxes (both abstract and concrete) but lack of precise abstract and concrete semantics. To overcome this problem, mixing of Ontology with domain specific modelling can be a better solution. Ontology is defined as an explicit specification of shared conceptualization in terms of concepts, relationships present between those concepts and related axioms [9]. Axioms has enabled ontology to provide enriched and formal semantics. Formal semantics has facilitated in checking consistency between distinct design concepts of SOA. However, a suitable framework is required that may mix ontology with domain specific modelling and provide formal, generic and domain specific syntaxes as well as precise semantics towards models of SOA.

Aiming to address aforementioned challenges and related drawbacks of domain specific modelling, in this paper, a framework described in [8] is applied. Based on this framework, an ontology driven specification of SOA conceptualization (Meta-Model level) is proposed in this paper. This proposed specification is conformed towards an upper level ontology known as Generalized Ontology Modelling (GOM) [8] that is realized as meta-meta model level of domain specific modelling. Further, the proposed ontology specification for SOA can be instantiated towards a case scenario based on SOA conceptualization (domain model level). Thus, the proposed ontology is a middle level ontology that can be realized as meta-model level in domain specific modelling for SOA. Fig. 1 has
described the relation between GOM, proposed ontology driven specification of SOA and instantiated models based on proposed specification.

The novelty of the proposed work is that it has provided generic formal syntaxes and semantics (represented in mathematical logic) towards SOA conceptualizations in meta-model level by incorporating ontology with domain specific modelling. Further, the proposed conceptualization can be facilitated in deriving consistent models those can be built for different domains based on SOA conceptualization. GOM [8] has provided abstract syntaxes towards the proposed conceptualization. On the other hand, the derived model level vocabularies are the concrete syntaxes of the proposed meta-model level conceptualization. In this way, the proposed meta-model has facilitated in both high and low level conceptualization. In this way, the proposed meta-model level by incorporating ontology with domain specific mathematical logic) towards SOA conceptualizations in meta-

Further, the proposed conceptualization can be facilitated in deriving consistent models those can be built for different domains based on SOA conceptualization. GOM [8] has provided abstract syntaxes towards the proposed conceptualization. On the other hand, the derived model level vocabularies are the concrete syntaxes of the proposed meta-model level conceptualization. In this way, the proposed meta-model level by incorporating ontology with domain specific mathematical logic) towards SOA conceptualizations in meta-

On the other hand, several approaches are exist in second direction too. In [16], The Open Group mission of Boundary-less has potential contributions towards ontology driven model
driven implementations in order to enhance the common
driven implementations in order to enhance the common
driven implementations in order to enhance the common

II. RELATED WORK

Related existing approaches can be organized in two directions, (i) approaches those have applied domain specific modelling towards SOA and (ii) approaches those have used only ontology or mixed ontology with domain specific modelling and applied them towards modelling of SOA.

A large number of researches are exist in the first direction. Major approaches have specified SOA design artefacts in Service Oriented Architecture Modelling Languages (SoaML) and using Model Driven Development (MDD). In [11, 12, 13, 14], business processes are modeled using BPMN or UML Activity diagram. Then, from these specifications, SOA is modeled using SoaML. All of these approaches aimed at achieving integration between business modelling and service-based system design. However, SoaML do not provide generic and formal explicit semantic towards SOA. Since it is based on Unified Modelling Language (UML 2.0) which is a semiformal language and unable to provide precise semantics. In [15], reference model for SOA is described all concepts related to SOA are represented in useful way. Although, it lacks from proper formal foundations those are required to check consistency between SOA concepts.

On the other hand, several approaches are exist in second direction too. In [16], The Open Group mission of Boundary-less has potential contributions towards ontology driven model
driven implementations in order to enhance the common

TABLE I DISTINCT CONCEPTS AND RELATIONSHIPS OF GOM [8]

<table>
<thead>
<tr>
<th>GOM Facets</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptualization (C)</td>
<td>Consists of concepts, relationships between concepts and set of world states.</td>
</tr>
<tr>
<td>Concepts (D)</td>
<td>Representative elements of any conceptualization (universe of discourse).</td>
</tr>
<tr>
<td>World States (W)</td>
<td>World states are defined as particular states of conceptualization.</td>
</tr>
<tr>
<td>Structural Element Conceptualization (CSE)</td>
<td>Representative elements of CSE and have several properties P.</td>
</tr>
<tr>
<td>Activity Element Conceptualization (CAE)</td>
<td>Representative elements of CAE and related with several basic tasks T.</td>
</tr>
<tr>
<td>Event Element Conceptualization (CEE)</td>
<td>Representative elements of CEE and related with several constraints.</td>
</tr>
<tr>
<td>Artefact Element Conceptualization (CAE)</td>
<td>Representative elements of CAE and produce more consistent knowledge regarding SE; AE and EE in useful and comprehensive way.</td>
</tr>
<tr>
<td>Intra Concept kind relationship (R_{ik})</td>
<td>This is the relation, through which a structure element concept derive another structure element concept.</td>
</tr>
<tr>
<td>Intra Inheritance (IH_{ik})</td>
<td>It is present between two structural elements (SE) based on roles of them in a particular context.</td>
</tr>
<tr>
<td>Intra Collaborator (CR)</td>
<td>This relation assists in encapsulation of one concept within similar kinds of concept.</td>
</tr>
<tr>
<td>Intra Containment (CTS)</td>
<td>This relationship convey messages between AE and interacts with each other by passing data in the form of SE to accomplish their specific functionality.</td>
</tr>
<tr>
<td>Intra Data flow (RD_{ik})</td>
<td>It denotes transition from one state of SE to another state. SE triggers SE. Subsequently, SE participates in AE and changes its state.</td>
</tr>
<tr>
<td>Inter Concept kind relationship (R_{ik})</td>
<td>This relation exist when one group of concept encapsulates another group of concept.</td>
</tr>
<tr>
<td>Inter Containment (CTD)</td>
<td>It is communication between AE and SE through which SE executes one or multiple low level tasks of AE or pass messages towards AE.</td>
</tr>
<tr>
<td>Inter Interaction (IR)</td>
<td>These Relations are connections of various artefacts with SE, AE and EE. Example of this relationship may be Has Attribute (HA) and Inverse Has Attribute (IHA) which is between SE and related Properties (P) and vice-versa.</td>
</tr>
<tr>
<td>Inter Artefacts Relations and their Inverse relations</td>
<td>Has Concept (HCON) and Inverse of Has Concept (HCON) exist between conceptualization and concepts where Has World (HW) and Inverse of Has World (HWW) exist between conceptualization and world states.</td>
</tr>
</tbody>
</table>
understanding of SOA concepts in the business and technical communities. In [17], authors have developed service oriented architecture modelling by Model Driven Architecture (MDA) approach. They have developed Computation Independent Model (CIM) using domain ontology, which is transformed towards Platform Independent Model (PIM) through UML profile and further towards Platform Specific Model (PSM) described by OWL-S. In [18], an autonomic architecture has been developed based on decision models built on ontologies and aimed at self-configuring and self-adapting service-oriented and event-driven distributed systems. However, majority of these approaches are incompetent to explore behavioral characteristic of SOA such as temporal relationships, interaction between distinct concepts, message passing, sequencing, event triggering etc. Further, those have not provided formal foundations in order to aid in checking consistency on modelling of distinct concepts of SOA. The proposed meta-model in this paper is different from these aforementioned approaches as it conceptualize both structural and behavioral features of SOA, their inter-dependency and is based on mathematical logic.

III. PROPOSED ONTOLOGY DRIVEN META-MODEL FOR SOA

Under the domains of different ownerships, Service Oriented Architecture (SOA) paradigm facilitates organizing and utilizing distributed capabilities. In general, Service Oriented Architecture is consisting of a numbers of concepts such as Service, Service Description, Service Interface, Real World Effect, Contract & Policy, Execution Context and Visibility [15]. The proposed ontology driven meta-model is aimed to provide unifying semantics and syntaxes towards these distinct concepts of SOA. The proposed conceptualization is specified in mathematical logic based on axioms of GOM (Generalized Ontology Modelling) described in [8]. GOM is capable to specify domain independent formal semantics and syntaxes since it is an upper level ontology and can be placed in meta-meta model level of domain specific modelling hierarchy. Table I has specified distinct facets of GOM.

A. Conceptualizations and Concepts of Proposed Meta-Model

Based on formal conceptualization of GOM, distinct concepts of SOA is specified below. The proposed ontology driven meta-model of SOA is consisting of three key conceptualizations – Information Model (IM), Behavioral Model (BM) and Action Model (AM). Distinct concepts of SOA and their in between dependencies or relationships are included to these three models.

Information Model (IM): It is defined as a conceptualization of structural concepts included in SOA such as Service (S), Data (DA), Service Interface (SI), Actor (AC), Service Description (SD), Service Contract (SC), Registry (REG) and System (SYS). These concepts are structural since these have certain properties. Thus, Information Model (IM) can be instantiated from Structural Element Conceptualization (CS\textsubscript{SE}) of GOM. The formalization of Information Model is as follows.

\[
F1: (C_{\text{SE}}(IM) \leftrightarrow ((SE(S) \land SE(DA) \land SE(SI)) \land SE(AC) \land SE(SC) \land SE(REG) \land SE(SYS)) \land R_{\text{SE}}(r) \land W(SOA))
\]

Explanation: Since, \textit{IM} has instantiated Conceptualization of Structural elements \textit{C}\textsubscript{SE}, it is consisting of \textit{Structural Elements} (SE). Thus, Service (S), Data (DA), Service Interface (SI), Actor (AC), Service Description (SD), Service Contract (SC), Registry (REG) and System (SYS) – all are instances of the Predicate \textit{Structural Elements} (SE). \textit{R}\textsubscript{SE} is a predicate represented corresponding relationships related to all SE belongs to \textit{IM}. \textit{W(SOA)} is a predicate which has specified the particular world state SOA since this conceptualization is related to SOA. In this way, \textit{IM} is an instance of the predicate \textit{C}\textsubscript{SE}.

Behavioral Model (BM): It is defined as a conceptualization of functional and related relationships included in SOA such as operations offered by services. Functionality offered by services are activity since those functions are related with distinct tasks. Thus, \textit{Action Model (AM)} can be instantiated from \textit{Activity Element Conceptualization} (C\textsubscript{AE}) of GOM. Activity Elements (AE) included in this model is \textit{Action (ACN)}. Formalization of \textit{Action Model (AM)} is as follows.

\[
F2: (C_{\text{AE}}(AM) \leftrightarrow ((AE(ACN) \land R_{\text{AE}}(r) \land W(SOA))
\]

Behavioral Model (BM): It is defined as a conceptualization of achieving effects and responding towards surrounding environment by services. Real world effects are events since achieving effects may be related with distinct constrains. Thus, Behavioral Model (BM) can be instantiated from \textit{Event Element Conceptualization} (C\textsubscript{EE}) of GOM. Event Elements (EE) included in this model can be \textit{Effect(Eff)}. The formalization of \textit{Behavioral Model (BM)} is as follows.

\[
F3: (C_{\text{EE}}(BM) \leftrightarrow ((EE(Eff) \land R_{\text{EE}}(r) \land W(SOA))
\]

Description, explanation and formalism of Services\textit{(S)} is specified below. Specification of other distinct concepts included in these three models are summarized in Table II. However, comprehensive formalization of all concepts cannot be specified in the paper due to page limitations. Among the specified concepts in Table II, \textit{Flag} is a new concept denoting visibility of a service and can be added towards Artefact Elements (AFE) category of GOM since AFE is an open group. Visibility refers to the capacity for those with needs (Service Consumer) and those with capabilities (Service Provider) to be able to see each other [15].

Service (S): Service is a package of closely related business functionalities, which are called repeatedly in a similar fashion [2]. The concept of service is included in Information Model of proposed conceptualization since services are structural units of SOA and may have distinct properties such as Service Version, Service Identification Number etc. Every service has an interface. Through this interface, services has explored capabilities. Further, services have contract, which influence the description of services. Services can be associated with particular time stamps when those are invoked. Beside, services can have real world effect and have played roles such as consumer and producer. Thus, services have collaboration relationships with other services or other actors. Likewise,
services should be visible towards service consumers [15]. The formal conceptualization of service is specified below.

F4: (SE(S) ↔ ((P(Service_ID) ∧ P(Version_No.)) ∧ (R_S(HCA) ∧ R_S(HSI) ∧ R_S(HSC) ∧ R_S(CR) ∧ R_S(CTD) ∧ R_S(HR) ∧ R_S(HTM) ∧ R_S(HF) ∧ R_S(HA))))

Explanation: SE is a predicate representing that Service (S) is a Structural Element. Service ID and Version No. are instances of predicates P which has denoted properties of service. Predicate R_S is represented as relationships related to Service (S). Instances of R_S are as follows - Has Capability (HCA) denoted relationships between service and its capability; Has Service Interface (HSI) denoted relationships between service and its interface; Inter Containment (CTD) denoted relationships between service and its effects; Has Service Contract (HSC) denoted relationships between service and its contracts; Has Role (HR) denoted relationships between services and roles played by them such as consumer or producer; Has Collaboration (CR) denoted relationships between two or more services and between service and other actors when they have played roles of consumers or producers; Has Time (HTM) denoted relationships between services and time when those are invoked; Has Flag (HF) denoted relationships between service and its visibility and Has Attributes (HA) denoted relationships between service and its properties. Details of these relationships and relations of GOM from which these relationships are instantiated has explained in sub section B.

B. Relationships of Proposed Meta-Model

In proposed conceptualization of SOA, different concepts are related with each other using distinct relationships. Those relationships are instantiated from relationships of GOM [8]. Further, two different relationships (Association and Intra Interaction) are added to GOM through this proposed conceptualization, since several relationships in SOA can be instantiated from these two relationships. Both relationships are of Intra Concept kind Relationship (RC). Definition of these two relationships are specified below.

Association (AS): Association relationships attach similar types of Structural Elements (SE), which are connected with each other in the purpose of accomplishment of several objectives jointly. Formalization of AS is as follows.

F5: ∀x∃y3z3v3u(AS(x) ↔ (SE(y) ∧ SE(z) ∧ x(y, z) ∧ AE(v) ∧ Perform(v, y, z)))

Explanation: This axiom has specified that Association relationship (AS) between instances of Structural Element (SE), has been accomplished when those have participated in performing a common objective which is an instance of Activity Element (AE). y and z are instances of Activity Element (SE) and v is an instance of Activity Element (AE). Perform() is a predicate implying that y and z has performed v in together.

Intra Interaction (IRS): This relationship has connected similar types of concepts when those are connected with each other through the means of message passing. This relationship can be between two SE or AE. Formalization of IRS is as follows.

F6: ∀x∃y3z3v3k3m3l3q3r3l3s3t3a(IRS(x) ↔ (Message(v) ∧ ((SE(y) ∧ SE(z) ∧ x(y, z) ∧ AE(k) ∧ AE(m) ∧ CTD_SE−AE(y, k) ∧ CTD_SE−AE(y, m) ∧ pass(y, k, v, m, z)) ∨ (AE(k) ∧ AE(m) ∧ x(k, m) ∧ send(k, v) ∧ receive(y, m))))

Explanation: This axiom has specified that Intra Interaction (IRS) relationship can be present between instances of:

<p>| TABLE II SUMMARIZATION OF DISTINCT CONCEPTS AND CONCEPTUALIZATION OF PROPOSED SOA META-MODEL |</p>
<table>
<thead>
<tr>
<th>Conceptualization</th>
<th>Concepts</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Model (IM)</td>
<td>Service (S)</td>
<td>A package of closely related business functionalities which are called repeatedly in a similar fashion.</td>
</tr>
<tr>
<td></td>
<td>Data (DA)</td>
<td>Input and output parameters of operations which constitute a service.</td>
</tr>
<tr>
<td></td>
<td>Actor (AC)</td>
<td>Elements which have several properties and play the roles of consumer, provider or performer.</td>
</tr>
<tr>
<td></td>
<td>Service Description (SD)</td>
<td>Descriptions of functional and nonfunctional properties of services, service interfaces, and legal and technical constraints for usage of services.</td>
</tr>
<tr>
<td></td>
<td>Service Contract (SC)</td>
<td>Comprised of one or more service description documents.</td>
</tr>
<tr>
<td></td>
<td>Registry (REG)</td>
<td>An information catalog that is constantly updated with information about different services.</td>
</tr>
<tr>
<td></td>
<td>System (SYS)</td>
<td>A system is an organized collection of services. It can be of two types - Choreography System (SYSCH) and Orchestration System (SYSOR).</td>
</tr>
<tr>
<td></td>
<td>Choreography System (SYSCH)</td>
<td>This type of System is organized through the means of choreography among distinct services.</td>
</tr>
<tr>
<td></td>
<td>Orchestration System (SYSOR)</td>
<td>This type of System is organized through the means of orchestration among distinct services.</td>
</tr>
<tr>
<td></td>
<td>Service Interface (SI)</td>
<td>Service Interface defines the way in which other elements may interact and exchange information with a service [15]. It can be of two types - Interface-in-General and Interface-Orchestrator.</td>
</tr>
<tr>
<td></td>
<td>Interface-in-General</td>
<td>Interface-in-General are Service Interfaces those have not played the role of coordinator when several services have organized a system through the way of orchestration.</td>
</tr>
<tr>
<td></td>
<td>Interface-Orchestrator</td>
<td>Interface-Orchestrator are interfaces those have played the role of coordinator when several services have organized a system through the way of orchestration.</td>
</tr>
<tr>
<td>Action Model (AM)</td>
<td>Action (ACN)</td>
<td>Action has represented the activities or operations through which functionality of a service is accomplished. Action can be of two types – Composite Action (CACN) and Simple Action (SACN).</td>
</tr>
<tr>
<td></td>
<td>Composite Action (CACN)</td>
<td>This is the composition of simple actions.</td>
</tr>
<tr>
<td></td>
<td>Simple Action (SACN)</td>
<td>This is represented as atomic activity which have several decomposable tasks.</td>
</tr>
<tr>
<td>Behavioral Model (BM)</td>
<td>Effect (E)</td>
<td>Effect has denoted the outcome after invoking a service.</td>
</tr>
<tr>
<td></td>
<td>Properties (P)</td>
<td>Properties has denoted the attributes of concepts included in Information Model (IM).</td>
</tr>
<tr>
<td></td>
<td>Messages (MSG)</td>
<td>Messages has denoted the send and receive messages through Service Interfaces.</td>
</tr>
<tr>
<td></td>
<td>Constraint (CO)</td>
<td>Constraint has denoted conditions related to effects.</td>
</tr>
<tr>
<td></td>
<td>Role (RL)</td>
<td>Role is separation of concern of actors.</td>
</tr>
<tr>
<td></td>
<td>Time (TM)</td>
<td>This has specified existing time of all concepts of the proposed meta-model.</td>
</tr>
<tr>
<td></td>
<td>Task (T)</td>
<td>Tasks are decomposable work done by simple activities.</td>
</tr>
<tr>
<td></td>
<td>Flag (Flg)</td>
<td>Flags has denoted visibility of Services.</td>
</tr>
<tr>
<td>Relationships in Proposed Conceptualization</td>
<td>Description</td>
<td>Equivalent Relationships In GOM</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Has Interface</td>
<td>Relationship between concepts of Service and Service Interface</td>
<td></td>
</tr>
<tr>
<td>Has Consumed</td>
<td>Relationship between concepts of Service Consumer and Service</td>
<td>Association (AS)</td>
</tr>
<tr>
<td>Has Contract</td>
<td>Connection between concepts of Service and Service Contract</td>
<td></td>
</tr>
<tr>
<td>Has Performed</td>
<td>Connection between concept of Actor and Service</td>
<td></td>
</tr>
<tr>
<td>Has Provided</td>
<td>Relationship between concept of Service Provider and Service</td>
<td></td>
</tr>
<tr>
<td>Has Bind</td>
<td>Connection between two Service Interfaces</td>
<td>Intra Interaction (IRS)</td>
</tr>
<tr>
<td>Has Find</td>
<td>Connection between Service Interface and Registry, when consumer has find the service.</td>
<td></td>
</tr>
<tr>
<td>Has Publish</td>
<td>Connection between Service Interface and Registry, when provider has published the service.</td>
<td></td>
</tr>
<tr>
<td>Has AND relationship</td>
<td>Relationship between two Actions when those are performed in parallel.</td>
<td></td>
</tr>
<tr>
<td>Has OR Relationship</td>
<td>Relationship between two Actions when those are performed by alternative choices.</td>
<td></td>
</tr>
<tr>
<td>Has Sequence</td>
<td>Relationship between two Actions when one is performed after one.</td>
<td></td>
</tr>
<tr>
<td>Has Data</td>
<td>Connection between Service Interface and Data</td>
<td>Intra Containment (CTS)</td>
</tr>
<tr>
<td>Has Composite Action</td>
<td>Encapsulation relationship between two Composite Actions</td>
<td></td>
</tr>
<tr>
<td>Has Description</td>
<td>Encapsulation relationship between Service Contract and Service Description</td>
<td></td>
</tr>
<tr>
<td>Has Simple Action</td>
<td>Encapsulation relationship between Composite Action and Simple Action</td>
<td></td>
</tr>
<tr>
<td>Has Capability</td>
<td>Encapsulation between Service and its Functionality</td>
<td>Inter Containment (CTD)</td>
</tr>
<tr>
<td>Has Operations</td>
<td>Encapsulation between Service Interface and operations</td>
<td></td>
</tr>
<tr>
<td>Has Effect</td>
<td>Encapsulation between Service and its real world effect.</td>
<td></td>
</tr>
<tr>
<td>Collaboration</td>
<td>Connection between two Services or two Actors when each have played distinct Roles.</td>
<td>Intra Collaboration (CR)</td>
</tr>
<tr>
<td>Has Role</td>
<td>Connection between Actors and distinct Roles played by them.</td>
<td>Has Role (HR)</td>
</tr>
<tr>
<td>Send Message</td>
<td>Connection between Actions and Messages send by them.</td>
<td>Send Message (SM)</td>
</tr>
<tr>
<td>Receive Message</td>
<td>Connection between Actions and Messages received by them.</td>
<td>Receive Message (RM)</td>
</tr>
<tr>
<td>Has Property</td>
<td>Connection between Concepts of Information Model and their respective Properties</td>
<td>Has Attribute (HA)</td>
</tr>
<tr>
<td>Has Time</td>
<td>Connection between all concepts and corresponding time</td>
<td>Has Time (HTM)</td>
</tr>
<tr>
<td>Inverse Relationships</td>
<td>Responsible for dynamically addition of related concepts towards Service. For example, through Intra Interaction new tasks can be added dynamically towards Action (ACN).</td>
<td>Inverse Relationships (IRL)</td>
</tr>
<tr>
<td>Has Concept</td>
<td>Connection between Conceptualization and Concept</td>
<td>Has Concept (HC)</td>
</tr>
<tr>
<td>Has Data Flow</td>
<td>Relationship when data is passed from one activity to another</td>
<td>Intra Data flow (RLLD)</td>
</tr>
</tbody>
</table>

**Structural Element (SE) - y and z or instances of Activity Element (AE) - k and m, Message(v) is a predicate denoting messages passed through interaction. pass() and CTDSE-AE () are two predicates those are used to represent IRS between two SE. pass() is a predicate implying message passing between two SE via encapsulation of AE in those SE, since SE has no direct provisioning to pass messages between each other in GOM. CTDSE-AE () – this predicate has implied the encapsulation relationship among SE and AE. Likewise, send() and receive() are two predicates those are used to represent IRS between two SE. send() has denoted the relationship when one AE has passed messages to another AE. receive() has denoted the fact when one SE has received messages from another AE.**

Distinct relationships of proposed meta-model is specified in Table III. However, comprehensive formalization of all relationships cannot be specified in the paper due to page limitations. The proposed meta-model has facilitated in representation of inter dependency between structural and behavioral features of SOA through concepts of Information, Action and Behavioral model and their in between relationships such as – Send Messages, Receive Messages, Association, Interaction, Collaboration. Further, it has aided in service composition by concepts of Interface-in-General, Interface-Orchestrator, Choreography System, Orchestration System, Composite Action and their in between relationships such as Has AND, Has OR, Has Sequence, Has Interface. Moreover, service versioning can also be supported by the proposed meta-model through Inverse Relationships those are responsible for dynamically addition of related concepts towards service and change of structural and behavioral characteristics of services.

**IV. PROTÉGÉ IMPLEMENTATION OF THE PROPOSED META-MODEL**

In this section, the proposed meta-model has been implemented using OWL (Web Ontology Language) based ontology editorial tool Protégé [10]. Protégé has facilitated in representation of formally expressed axiom set of this proposed meta-model in OWL logic. It is composed of a number of reasoners for automated inference and initial checking of consistency on ontological theory expressed in OWL logic. OWL is based on Description Logic (DL). Fig. 2 (a) has illustrated Information Model of proposed meta-model; Fig. 2 (b) has illustrated Action Model of proposed meta-model and The ontological graphs contained in these figures are obtained through OntoGraf plug-in of Protégé.

**V. CONCLUSION AND FUTURE WORK**

This paper has proposed an ontology driven meta-model for SOA. This meta-model is conformed towards an ontology driven meta-meta model GOM and can be instantiated towards models of distinct SOA based applications. The motivation of this work is twofold. Firstly, the lack of modelling in inter dependency between structural and behavioral design artefacts of SOA. Secondly, missing of precise semantics and formalization in conceptualization has created serious difficulties in checking consistency on models of SOA. The novelty of the proposed meta-model is that it is specified in mathematica logic and enriched with formal semantics for representing inter dependency between structural and behavioral characteristics of SOA. This conceptualization has
facilitated in service composition through conception of System, Service Interface, Composite Actions and relationships between them. It has also aided in representing versioning of services through inverse relationships and version number of services. Further, in future the proposed conceptualization has facilitated in checking consistency over models of distinct SOA applications.

Future work will include automatic transformation from the proposed meta-model level towards model level. Devising of a suitable method for automation of consistency checking in model level will also be an important future work.

REFERENCES


