World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:12, No:9, 2018

Consolidating Service Engineering Ontologies Building Service Ontology from SOA Modeling Language (SoaML)

Purnomo Yustianto, Robin Doss, Suhardi, Novianto Budi Kurniawan

***Abstract*—**As a term for characterizing a process of devising a service system, the term 'service engineering' is still regarded as an 'open' research challenge due to unspecified details and conflicting perspectives. This paper presents consolidated service engineering ontologies in collecting, specifying and defining relationship between components pertinent within the context of service engineering. The ontologies are built by way of literature surveys from the collected conceptual works by collating various concepts into an integrated ontology. Two ontologies are produced: general service ontology and software service ontology. The software-service ontology is drawn from the informatics domain, while the generalized ontology of a service system is built from both a business management and the information system perspective. The produced ontologies are verified by exercising conceptual operationalizations of the ontologies in adopting several service orientation features and service system patterns. The proposed ontologies are demonstrated to be sufficient to serve as a basis for a service engineering framework.

***Keywords*—**Engineering, ontology, service, SoaML.

I. Introduction

GIVEN its nature as a multidiscipline endeavour, establishing a common paradigm is quite problematic in the service science field. Varying perspectives emerged from different contributors with their own set paradigm influenced by particular academic backgrounds. Despite the research contribution over two decades, service science still experiences a lack of standard ontology for *'service'*, and *'service system'* concepts [1]. Therefore, one of the challenges in service science is to consolidate the perspectives into a shared and cohesive perspective [2].

Within the research context of *"Service Engineering Framework"* [3], a series of ontology is developed. Three ontologies are presented in this paper: (1) Service Oriented Architecture (SOA) Ontology, (2) General Service Ontology, and (3) Software Service Ontology and. The first ontology is built from the SOA stream, which is later adapted and generalized to cover non-technical perspectives originated from classic service engineering into the second ontology. The third ontology is an adjustment and refinement of the first ontology as a special case for software service.

The term 'ontology' is defined as a set of structured (abstract) concepts and relationship between concepts within a defined domain [4]. Ontology serves a basis for a language, to be used as a communication tool to share an understanding regarding a specific domain. Therefore, ontology definition is often linked with a modelling activity.

A *'model'* is defined as representation of a reality within a definite purpose. To facilitate a common understanding between multiple parties, a model is usually built based on a specific modelling language, i.e. a metamodel. The metamodel specifies a palette of concepts and constraint rules for a valid model for a specific modelling language [5].

Fig. 1 shows the relation between ontology and model. Ontology is an explicit and formal specification of a shared conceptualization, in both model and metamodel level. Two types of ontology are involved: (1) ontology of meta models, and (2) ontology of problem domain, or a *'domain ontology'* [6]. A model is an instantiation of a 'meta model' and similarly, a *'domain ontology'* is an instantiation of a 'meta model ontology'. Both are semantically interpreted by their respective ontology. Ontologies presented in this paper are in the category of 'metamodel ontology'.

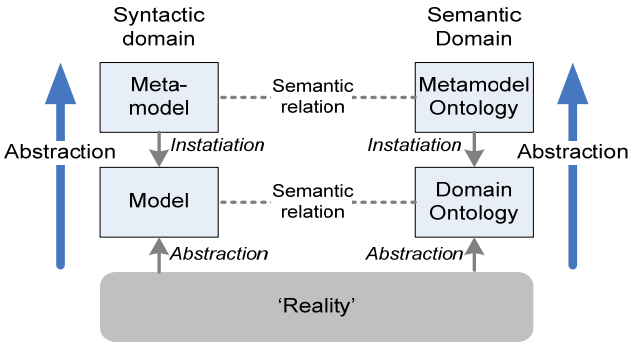

Fig. 1 Model, metamodel, and ontologies [6]

II. Related Work

Several ontology propositions of *'service'* emerged from IS/IT contributors. One of early attempts in defining a service ontology is found as a '*service system metamodel*' within the context of a SOA methodology [7]. While the focus is on software-service, it already relates to one non-IT concept: '*business process*'.

Purnomo Yustianto is with the School of Information Technology - Deakin University, Geelong, VIC, Australia (corresponding author, e-mail: pyustian@deakin.edu.au).

Robin Doss is with the School of Information Technology Deakin University, Geelong, VIC, Australia (e-mail: robin.doss@deakin.edu.au).

Suhardi is with the School of Electrical Engineering and Informatics-Institut Teknologi Bandung, Indonesia (e-mail:suhardi@stei.itb.ac.id).

Novianto B. Kurniawan is with the School of Electrical Engineering and Informatics - ITB, Indonesia (e-mail: noviantobudik@students.itb.ac.id).

As visualized in Fig. 2, service is differentiated based on participant type: (1) For *service consumers*, a service is a unit of expected functionality with a service level agreement as *'Target Service'*. (2) For *service providers*, a service is a unit of deployed functionality as *'Publishable Service'*. *'Service Interface'* serves as a front-end for *'Service Component'*, which can be in an atomic or a composite form.

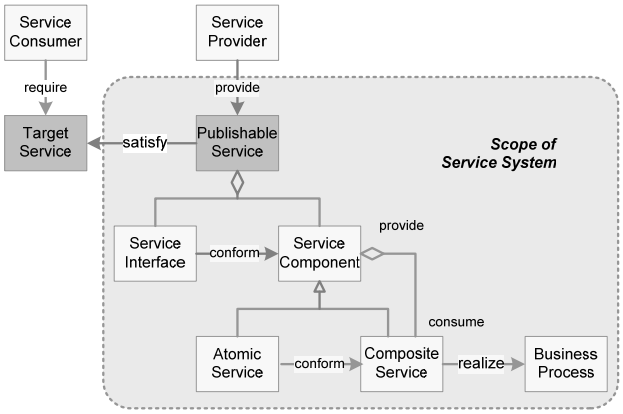

Fig. 2 Metamodel of service system [7]

Another ontology proposition from software service-orientation perspective is found in [8]. It defines three successive abstractions of a service: (1) single interaction, (2) multiple interactions (*choreography*), and (3) multi-provider (*orchestration*). Five overlapping aspects of a service model were also defined: (1) *structure*, (2) *behaviour*, (3) *information*, (4) *goal,* and (5) *quality*. The structural aspect of a service is conceptualized as a metamodel in Fig. 3, covering 12 concepts entirely from a SOA perspective.

As an attempt to consolidate the non-orthogonality of competing SOA concepts, a literature survey on SOA concepts is performed in [1]. Nine core identifiers which characterize a service-orientation were extracted: (1) *architecture*, (2) *binding*, (3) *capability*, (4) *composition*, (5) *contract*, (6) *delivery*, (7) *distributed sources*, (8) *identity*, and (9) *interoperability*.

From these previous works, selected concepts that can be considered as a potential component for targeted ontology are: (1) *architecture*, mostly for software level abstraction, (2) *binding*, related to *'role'* concept, (3) *capability*, as user perspective of business function, (4) *composition*, related to atomicity or composite nature of service, and (5) *contract,* as terms and conditions agreement of a service. Additionally, [9] suggests that service structural model is consisted of: (1) *service operation*, (2) *service component*, and (3) *service interface*.

To produce an integrative perspective, a more practical approach is hence taken to use collaborative SOA conception as the source for ontology building. Over the years, several standard groups have produced SOA open standards: *OASIS*, *The Open Group*, *The International Organization for Standardization* (ISO/IEC) and *Object Management Group* (OMG).

The standards published are not always compatible to each other, and actually competing in its overlapping terminology [10]. As illustrated in Fig. 4, two products can be considered as the state-of-the-art: (1) *OMG's SoaML* [11], as the definitive SOA metamodel originated from OASIS stream, and (2) *ISO/IEC's SOA Reference Architecture* [12], as the definitive SOA ontology definition originated from The Open Group stream. Unfortunately, these two are not semantically related.

III. SOA Ontology

SoaML is one of many specifications produced by OMG. SoaML was first formalized in 2009, with minor updates later in 2012 [11]. SoaML is conceived based on the limitations of UML in representing SOA concepts [13]. While its popularity in the industry is very limited, SoaML is consistently referenced in academic publication as the definitive metamodel for SOA. Table I lists SOA concepts covered by SoaML.

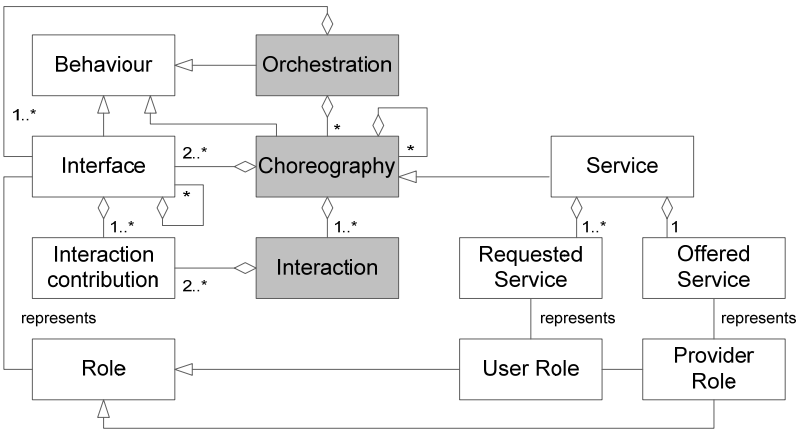

Fig. 3 Metamodel of service concept [8]

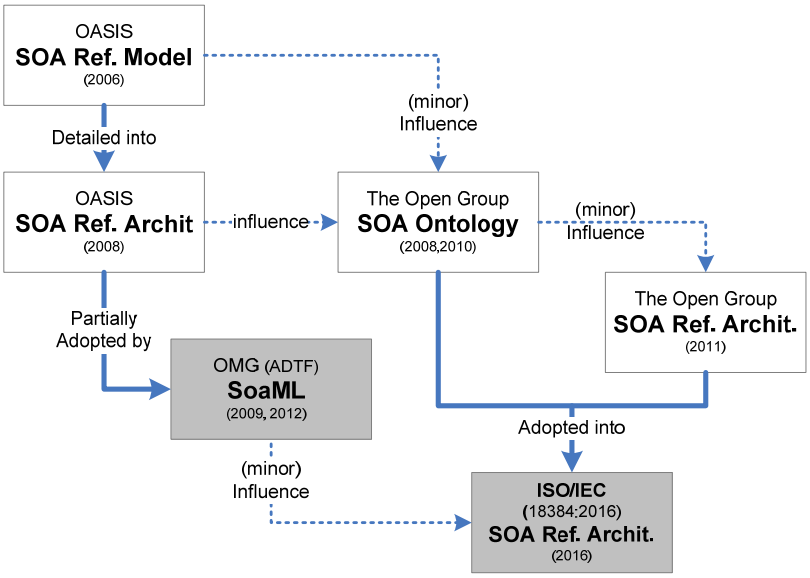

Fig. 4 Succession of SOA open standard [10]

TABLE I
SOAML CONCEPTUAL COMPONENTS [11]

	Concept	Description
1	Participant	Entities (physical or software) that provide or use services
2	Port	Participant's service interaction points in providing or consuming services
3	(UML) Interface	A type of service interaction description for synchronous unidirectional interaction
4	Service Interface	A type of service interaction description for (multiple) asynchronous interaction
5	Service Contract.	A type of service description based on roles and rules as an agreement for multi-party interaction
6	Capability	Ability owned, or required, by participant to affect some changes.
7	Role	A specific functionality assumed by participant in an instance of service interaction
8	(Role) Binding	A pairing instance of a participant with a role within a specific service interaction context
9	Interaction protocol	*Sequential arrangement of operation invocation between role/interface that may involve rules*
9	Operation	*An atomic invokable software behaviour with input-output message passing feature*
10	Message type	Data values that can be sent between participants
11	Service Architecture	High level description of connection between participants through service contracts within a specific service community
12	Method	Owned behaviour of a participant

While providing a formal specification of its stereotyping extension from the original UML specification, SoaML specification document is surprisingly lacks an ontological definition. The specification actually offers two types of service modelling approaches: (1) *Interface*-based and, (2) *Contract*-based [11], and therefore multiple forms of service abstraction is permissible [14]-[16].

A peculiar feature of SoaML is the absence of specific abstraction for '*service*'. Three abstractions are offered superimposed to service description components [11], as:

1. *Interface*, accommodating atomic services containing only self-contained operations.
2. *Service Contract*, accommodating atomic services and composite services by combining interfaces into a service contract.
3. *Service Interface*, accommodating atomic services and composite services by combining interfaces

Consequently, there are three versions of an ontological structure that can be inferred from the specification: (1) *Interface*-based, (2) *Service Interfaced*-based, and (3) *Contract*-based. These versions are elaborated in the following paragraph.

To highlight the differences between these versions, a special visualization technique is employed where: (1) Compositional relationship is visualized in the form of a Venn diagram, in which member components are placed inside a container representing its compositional parent, (2) a service abstraction is symbolized inside a thick border around the superimposed service description components.

Fig. 5 represents the first version, in which the *'service'* abstraction is superimposed to the (UML) *'interface'* concept. The approach is used for simple SOA where the whole architecture is composed of flat atomic services without the possibility of a service composition. Each interaction is synchronous and involving exactly two participants with an interface embedded with a specific role; either as a service requester in a consumer role or as a service responder in a provider role.

Fig. 6 shows the second ontological version where a *'service'* is abstracted with the *'service contract'* concept. Service contract is equipped with an interaction protocol as an internal logic that governs invocation sequences, both synchronous and asynchronous, between participants. The approach is able to accommodate a service interaction with more than two participants. Service composition is possible in the form of a composite participant or compound service contract [11], [15].

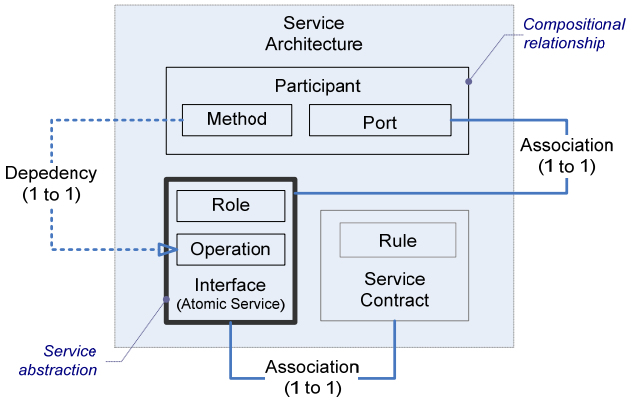

Fig. 5 *Interface*-based SoaML ontology

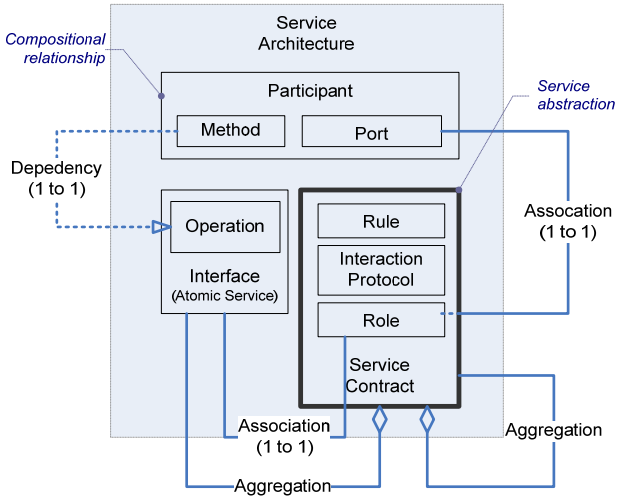

Fig. 6 *Service Contract*-based SoaML ontology

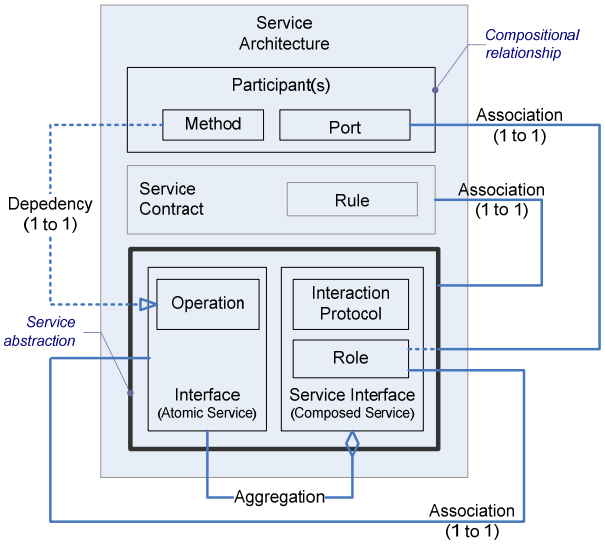

Fig. 7 *Service Interface*-based SoaML ontology

Fig. 7 shows the third version in which a *'service'* is abstracted as both *'interface'* (for atomic service) and *'service interface'* (for composite service). Service interface has an interaction protocol as an internal logic to govern invocation sequencing in both synchronous and asynchronous invocation between participants. Service interface can also accommodate an interaction service with more than two participants.

The ontology structure in Fig. 8 is built based on the third version of the ontology (*service interface*-based). It sufficiently covers all of the components emerged in the related works section. Most of the components are an abstract design concept, except for three components denoted with darker shade in Fig. 8: (1) *participant*, to become a software component, (2) *simple interface*, and (3) *service interface* to be implemented as an atomic or a composite software service, e.g. web service and WSDL interface.

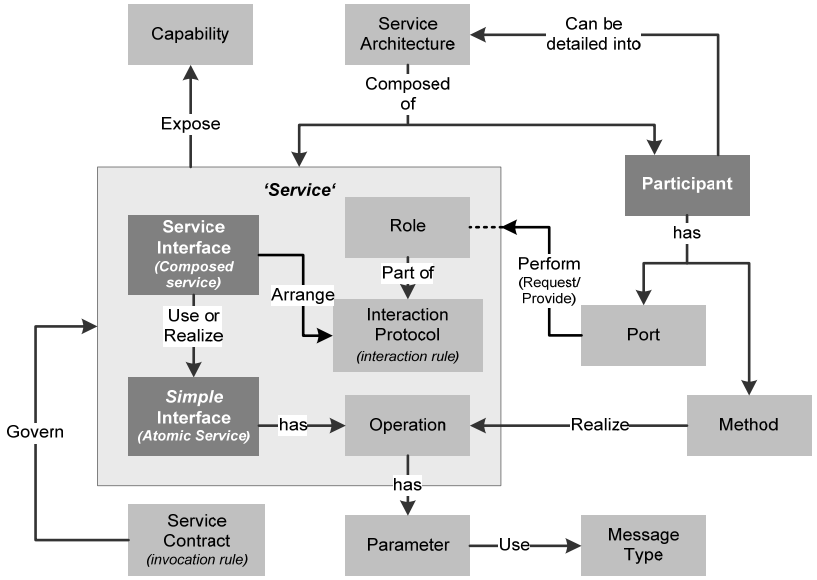

Fig. 8 SoaML ontological structure

ISO/IEC 18384:2016 is based on the products of The Open Group: SOA Ontology (2010), and SOA Reference Architecture (2011). The ontology produced is quite simple, and its coverage is superficial if compared to SoaML ontology [12]. The coverage is lacking in a detailed level of (software) service abstraction, i.e. port, operation, role, and interaction protocol. The ontology is also missing some of the concepts defined in the terminology part, i.e. Capability, Service Architecture, and Role.

Despite these differences, correlation with SoaML ontology occurs in: (1) Service Interface, (2) Service Contract, (3) Information Type, and (4) Element (Participant in SoaML). The relationships are also consistent:

- *Element* perform *Service* (via *Port* and *Role* in SoaML)
- *Service* has *Service Contract* and *Service Interface* as service description
- *Service Interface* has *Information Type* as attribute (via *Operation* and *Parameter* in SoaML)

It can be observed that the produced SoaML ontology is relatively consistent with the other software service ontologies. It is also a superior ontology compared to the ISO/IEC 18384:2016's ontology, in coverage and level of detail. The produced SoaML ontology is therefore established as an intermediary for a true Software Service Ontology.

IV. General Service Ontology

Despite the fact that the adoption of service-oriented concepts is not apparent beyond the technological sphere, its conception always strives to provide a generalized abstraction covering both IT-based and non-IT-based systems. While in reality the distinction for IT and non-IT context of a service system needs to be made, the generality feature of SOA conceptions is useful to build the General Service Ontology.

The goal is to generalize and enlarge the coverage of the Software Service Ontology. The generalization is achieved by applying the concept from a software-service context to the general context of a service system, which includes the physical and the manual system. The objective of the enlargement is to cover concepts included in the classic service engineering context but missing in informatics service engineering, such as (1) *'value'*, (2) *'business process'*, (3) *'business model'* and (4) *'capability*'. These four concepts are the target components to be integrated with concepts already covered within the Software Service Ontology.

The general ontology is built based on the available standard documents published by ISO/IEC and OMG. Sixteen standard documents were identified to cover the definition of targeted concepts. The identified documents are originated from both business and technical domain, including IT domain.

Some of the source documents contain a partial ontological view of concepts covered in the document, i.e. ISO 18384 (SOA-RA), ISO 19505 (OMG-UML), ISO 19510 (OMG-BPMN) and OMG-BMM. In these cases, the targeted concept definition is extracted, along with the available defined relation between them. For other documents, only the concept definitions were extracted. If available, the definitions were captured from the formal terminology definition section. In the other cases, the implied definition is extracted from the descriptive narration. Table II lists the source document for building the consolidated ontology.

TABLE II
Source Documents of Concept Definitions

	Name	Description
1	ISO 2382	IT Vocabulary
2	ISO 9000	Quality Management Systems
3	ISO 14662	Open Electronic Data Interchange (EDI)
4	ISO 15288	System Life Cycle
5	ISO 15944	Business Operational View
6	ISO 16500	Digital Audio Visual (AVI) System
7	ISO 12207	Software Life Cycle
8	ISO 18384	SOA Reference Architecture
9	ISO 19505	Unified Modeling Language (OMG's UML)
10	ISO 19510	Business Process Model & Notation (BPMN)
11	ISO 30102	Distributed Application Platforms & Services
12	ISO 90003	Software Engineering
13	ISO 14813	Intelligent Transport System
14	OMG - VDML	Value Definition Modeling Language
15	OMG - BMM	Business Motivation Model
16	OMG - SoaML	Service Oriented Archit. Modeling Language

A total of 74 concept definitions are identified. These concepts are arranged and grouped based on similarity. Concepts observed to be covering similar idea are merged into a single representative label. Table III collects the merged concepts into a hierarchy of 17 concepts as the components of the ontology.

The ontological relation between each merged concept is visualized in Fig. 9. For ease of reference, the numbering in the figure is correlated with the number in Table III.

As defined in Table III, service is a container for *Interface* and its *Operation*. In Fig. 9, these service components are also aggregated with the underlying process component, i.e. *Activity* and *Task*, to form a larger abstraction of *Service*, between the front-end interface and the back-end supporting activities. This also reflects a SoaML perspective of the relation between *'process'* and *'service'* as different views of a similar object. 'Process' view focuses on the how and why of the whole interaction, while 'service' focuses on participant activities in provision and consumption of services [11].

A shaded background is introduced in Fig. 6 to define an ownership boundary. Three components are extended beyond the boundary: (1) *Collaboration*, as an abstraction of atomic or composite interaction, (2) *Choreography*, where the arrangement of interaction sequence is an agreement with outside entity, and (3) *Message*, which is exchanged with party outside ownership boundary.

A pair of concepts is merged in the ontology visual: *Choreography* (7b) and *Contract* (7c). This merger is not only implemented for visual simplification, but also to show the strong intersection between the two related to an interaction arrangement. To be precise, choreography refers to the

sequencing aspect, while contract refers to the rule and constraint.

TABLE III
HIERARCHY OF MERGED CONCEPT DEFINITIONS

Concept	Definition
1. Entity	Individual *element* in a system which can act as a service provider or consumer.
1a. Interaction point	Location at which information is received (or sent) to invoke an interaction
2. Business model	A description of value creation, delivery, and capturing to fulfill the motivations
3. Process	A set of business activities as steps to achieve a business objective
3a. Activity	A collection of related (cohesive) tasks of a process
3b. Task	An atomic activity that is included within a process, accomplishes a defined result.
3c. Rule	A structured, discrete, specific, and practicable guidance for business process
4. Service	A set of accessible *'activities'* of a capabilities through an interface and constraints.
4a. Interface	Shared boundary between two units, characterized by operations
4b. Operation	A definition of messages consumed and, optionally, produced when called.
5. Capability	Participant ability to act and produce an outcome
6. Value	A measurable factor of benefit to a recipient, in association with a business item
7. Collaboration	Predefined set of activities and/or processes initiated to accomplish a shared goal
7a. Role	A defined set of activities assigned to an entity to performs a specific function
7b. Choreography	An ordered sequence of message exchanges between two or more entity
7c. Contract	Explicitly stated rule, that prescribes, limits, governs or specifies transaction
7d. Message	The contents of a communication between two participants

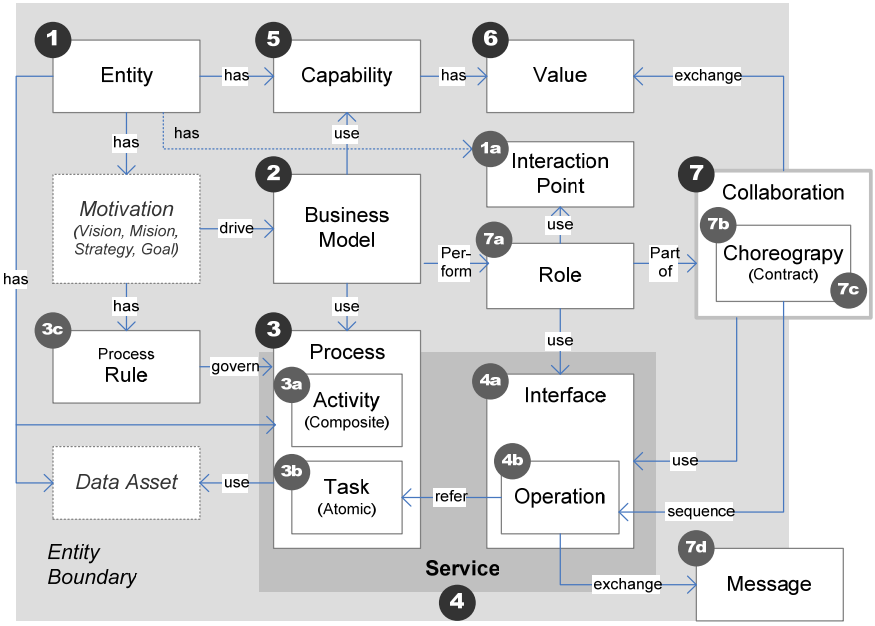

Fig. 9 General Service Engineering Ontology

V. SOFTWARE SERVICE ONTOLOGY

The general service ontology combines concepts from a software-service context with concepts from classic service engineering, i.e. (1) *'value'*, (2) *'business process'*, (3) *'business model'* and (4) *'capability'*. In practical engineering level, different treatments are performed for each context: IT-context and non-IT context.

The type of service encounter is a factor in differentiating software and non-software service. Two types of service encounters are defined in [17]: (1) *Physical*, and (2) *Virtual*. If an interaction of an encounter is mediated by technical devices, the encounter is categorized as a virtual encounter. In this typology, technology may facilitate the encounter in various forms, e.g. from e-mail to website. The software-services context resides in a specific situation, in which a software component offers service consumables by other software components.

To focus on the software context, the ontology is trimmed to only include software related concepts. Excluding the three business analysis level components, i.e. (1) *values*, (2) *capability*, and (3) *business model*, the 17 components in the general context of Table III are reduced to 14 components for software-service context, as listed in Table IV.

To achieve a uniformity and as a form of triangulation, SoaML ontology (Fig. 8) is juxtaposed and adapted with the

structure of the consolidated general-service ontology (Fig. 9) into the resulting ontology in Fig. 10.

The term *'service'* represents an abstraction of externally accessible software components in the software service context. Therefore, it covers both, the underlying software component behaviour (3a and 3b) originated from the *'process'* context, and the related published description (4a and 4b). Similarly, the term *'service contract'* merges the two aspects of: internal process rule (3c), with the externally shared and agreed rule (7c). These characteristics are derived from SoaML's feature in superimposing a service component with its description. This merging is also coherent with SoaML perspective that views *'process'* and *'services'* as different perspective of the same object [11].

Two additional SoaML components are not represented in this context: (1) *Capability*, and (2) *Service Architecture*. The two are considered to be residing in the business analysis level of service engineering.

TABLE IV
CONCEPTS FOR SOFTWARE SERVICE CONTEXT

General Context	Software-service context	SoaML Label
1. Entity	1. Software component	1. Participant
1a. Interaction point	1a. Port	2. Port
3. Process	3. Software service*	-
3a. Activity	3a. Composite service	3. Service Interface
3b. Task	3b. Atomic service	4. (UML) Interface
3c. Rule	3c. Service contract **	5. Service Contract ^
4. Service	4. Software service *	-
4a. Interface	4a. Service interface	3. (UML) Interface 4. Service Interface
4b. Operation	4b. Service operation	9. Operation
7. Collaboration	7. Software interaction	9. Interaction Protocol ^^
7a. Role	7a. Component role	7. Role
7b. Choreography	7b. Interaction Protocol	9. Interaction Protocol ^^
7c. Contract	7c. Service contract **	5. Service Contract ^
7d. Message	7d. Message	10. Message type

Merged concepts are marked with pairs of *, **, ^ and ^^ symbols

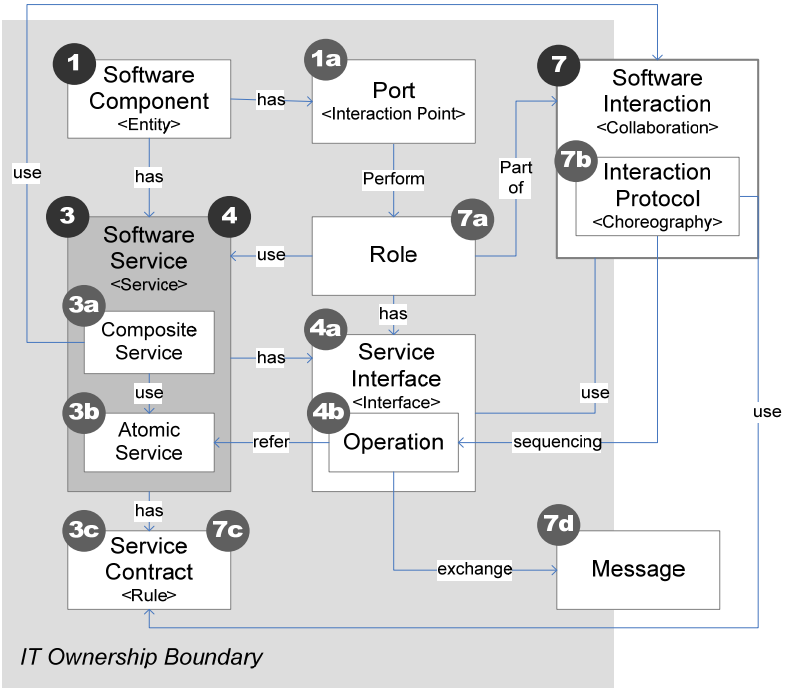

Fig. 10 Software service ontology

VI. ONTOLOGY VERIFICATION

Components decoupling and its *binding* mechanism is an important principal in SOA conception [18]. It is also the underlying motive in introducing SoaML over UML limitation [13]. Conceptual exercises for decoupling, binding and service consumption is narrated in this section to demonstrate and evaluate the capability of the software service ontology (Fig. 10) in covering basic SOA concepts.

A *'service decoupling'* is implemented as a separation between a published service description (component 4a and 4b) and its underlying supporting *behaviours* (component 3a and 3b). *Service behaviours* (component 3) are actually a part of a specific *software component* (component 1).

'Binding', or more precisely *'role-binding'*, is an execution time instance when a *software component* (component 1) assumes a *role* (component 7a) within a context of specific *software interaction* (component 7), using a *service interface* (component 4a) as the guidance in invoking its internal behaviours (component 3a and 3b), via its defined *port* (component 1a) as the location address, for *message* (component 7d) passing *operations* (component 4b).

The ontology visualization structure is not only describing a service providing software component. In a case where a component requires services from other component within its own composite behaviour (component 3a), it follows the previously described role binding mechanism. The difference

is that the component (component 1) assumes consumer *role* (component 7a) and adheres to a *collaboration* mechanism (component 7), which is implemented by services (component 3a and 3b) in the providing components.

To demonstrate the feasibility of the general service ontology (Fig. 6), several patterns of a service interaction are applied to it. The patterns of service interaction are implied in three abstraction level of service modelling: (1) simple interaction, (2) business-to-business (B2B) interaction, and (3) complex interaction [15]. These exercises can be seen as a proto-operationalization of the metamodel ontology toward domain ontology (Fig. 1).

In the first pattern, a simple interaction is occurred between a service provider and a service user, e.g. individual end-user consumer. Here, the whole ontology is positioned as the service-providing entity. To illustrate the first pattern, the ontology is paired with the existence of a simple consumer outside the entity boundary in Fig. 11.

The resulting pattern covers the concepts of: (1) capability offered by the service provider, (2) value offered and requested by the consumer, (3) value brought by the consumer (e.g. in the form of monetary asset), (4) choreographed activity between the two entities, and value exchanged during the transaction.

In the second pattern, choreography level of abstraction, a service is modelled as a process with multiple interactions between two entities. A B2B arrangement between a company and its supplier is an example of this pattern. The pattern is formed by pairing two ontology sets as two interacting entities.

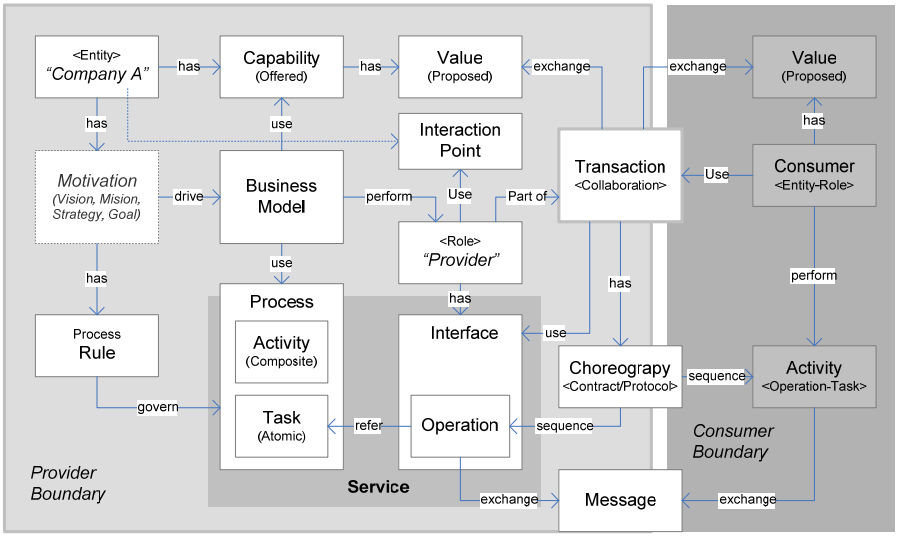

Fig. 11 Model of a Simple Service Pattern

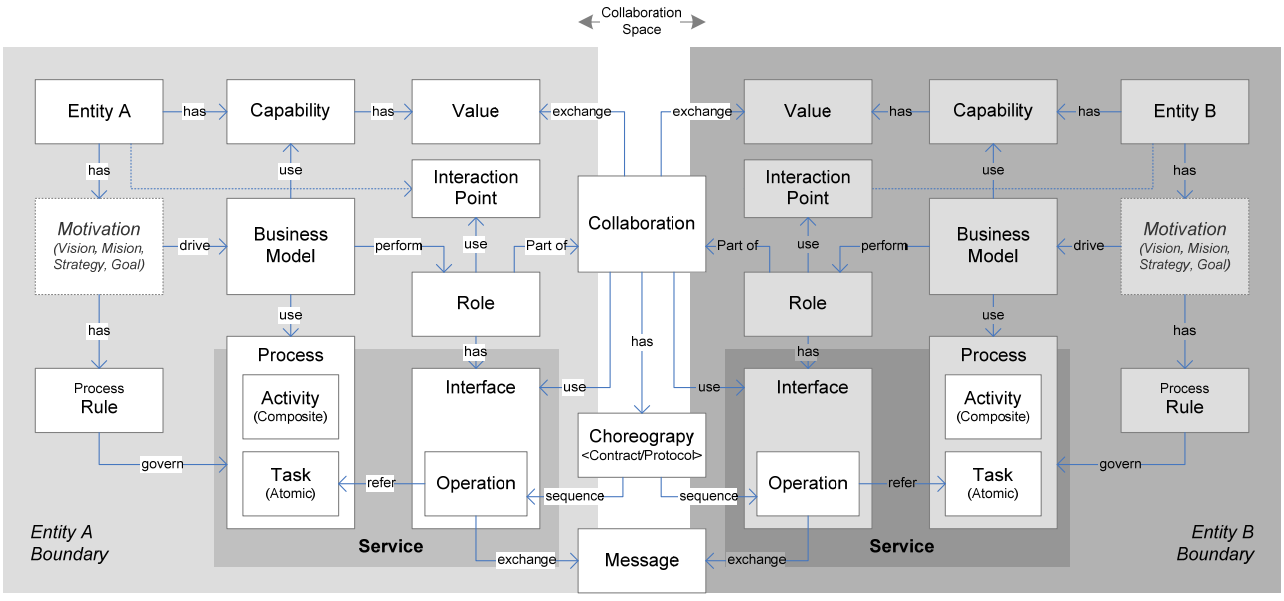

Fig. 12 Model of a B2B Service Interaction Pattern

Fig. 12 shows a model of the second pattern which visualizes pairs of external behaviour requested and offered by each participant in the pattern. This pattern specifies and analyses interoperability between two service participants. The model structure also introduces the concept of '*collaboration space*' in which the interactions take place. It may reside (i.e. owned) within one of the participant boundaries, or in independent third-party location. In the software-service context, the 'collaboration space' relates to the operator and controller of software-service repository, i.e. service registry and service publication.

In the third pattern, orchestration abstraction, an offered service is modelled as a composition of other services. Fig. 10 illustrates this pattern by combining the first and second pattern approaches with the introduction of both a simple customer, and a partnering service co-provider.

The third pattern is related with indirect type of service encounter in the typology of service encounter [17], where an external party is involved in the service process, as co-provider or intermediary, and may make a direct contact to the service-consumer. In a more complex pattern, multiple co-providers may form service architecture over a set of services. Consequently, this model can be used to analyse and specify possible implementation of the offered service.

The model also raises the issue of 'collaboration spaces'. It relates to the existence of a service coordinator, with the central role of interacting and orchestrating other providers. While the arrangement can be made to be in equal term (distributed and federated), each particular of collaboration tends to require a dominant participant role as the main operator. Other types of service system patterns and combinations may exist, related to elaboration of provider role and components of collaboration space, but the three illustrated patterns adequately demonstrate the feasibility and flexibility of the produced ontology in covering various types of service system.

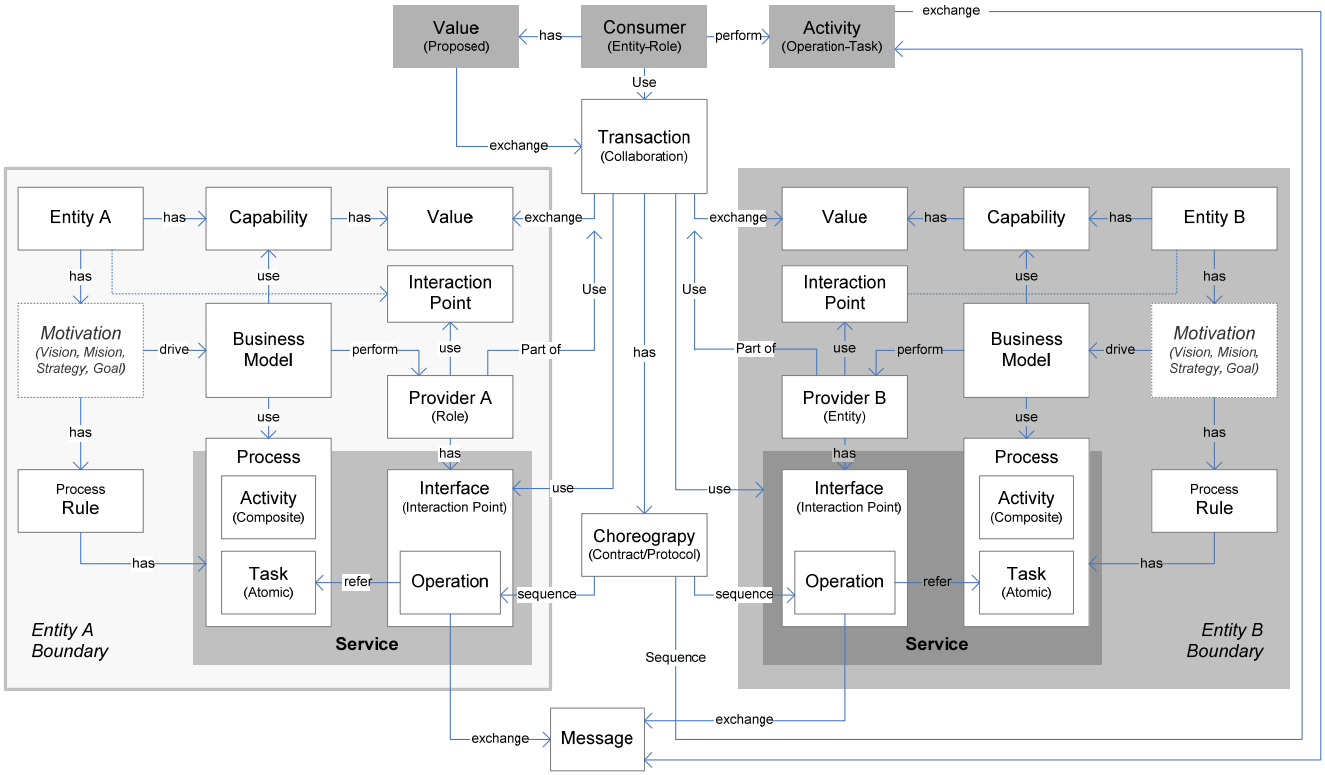

Fig. 13 Model of a multi-provider service pattern

VII. Conclusion

This paper introduces an ontological basis for '*Service Engineering*' [19] by describing a process of ontology building of a series of ontologies; service-oriented architecture ontology, general service ontology, and software-service ontology. Two final sets of ontologies were produced: general service ontology (Fig. 9) and software-service ontology (Fig. 10). The two are correlated in which the software service ontology is a specialization of general service ontology.

As conceptual models are composed from literature study approach, the ontologies were verified with its ability to represent features of service concepts. In another part of the research, these defined ontologies are to be used to assess the completeness of a service engineering framework in covering on the aspects of a service system. The mapping of framework artefacts with the ontology structure also serves as a bridge to characterize the artefacts format.

References

[1] M. Anjum and D. Budgen, "A mapping study of the definitions for service oriented architecture," in 16th International Conference on

Evaluation & Assessment in Software Engineering (EASE 2012), 2012, pp. 57–61.
[2] S. E. Sampson, "The unified service theory," in Handbook of service science, Springer, 2010, pp. 107–131.
[3] P. Yustianto, R. Doss, and Suhardi, "Activities and Artifacts in Service Engineering Case Studies Report of Service Engineering Framework," Inf. Technol. Syst. Innov. (ICITSI), 2017 Int. Conf., no. 1, pp. 373–377, 2017.
[4] G. Guizzardi, "On ontology, ontologies, conceptualizations, modeling languages, and (meta) models," Front. Artif. Intell. Appl., vol. 155, p. 18, 2007.
[5] U. Aßmann, S. Zschaler, and G. Wagner, "Ontologies, meta-models, and the model-driven paradigm," in Ontologies for software engineering and software technology, Springer, 2006, pp. 249–273.
[6] M. Saeki and H. Kaiya, "On relationships among models, meta models and ontologies," in Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling (DSM 2006), 2006.
[7] S. H. Chang, "A systematic analysis and design approach to develop adaptable services in service oriented computing," in Services, 2007 IEEE Congress on, 2007, no. 2, pp. 375–378.
[8] D. A. C. Quartel, M. W. A. Steen, S. Pokraev, and M. J. Van Sinderen, "COSMO: A conceptual framework for service modelling and refinement," Inf. Syst. Front., vol. 9, no. 2–3, pp. 225–244, 2007.
[9] M. Anjum and D. Budgen, "An investigation of modelling and design for software service applications," PLoS One, vol. 12, no. 5, p. e0176936, 2017.
[10] H. Kreger and J. Estefan, "Navigating the soa open standards landscape around architecture," Jt. Pap. Open Group, OASIS, OMG, 2009.
[11] OMG, "Service oriented architecture Modeling Language (SoaML) Specification," Object Management Group (OMG) Specification, 2012. (Online). Available: http://www.omg.org/spec/SoaML/. (Accessed: 07-Jun-2017).
[12] International Organization for Standardization (ISO), "ISO/IEC 18384:2016 - Reference Architecture for Service Oriented Architecture (SOA RA)." (Online). Available: https://www.iso.org/standard/63104.html. (Accessed: 20-Apr-2018).
[13] I. Todoran, Z. Hussain, and N. Gromov, "SOA integration modeling: An evaluation of how SoaML completes UML modeling," in Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011 15th IEEE International, 2011, pp. 57–66.
[14] B. Elvesæter, A.-J. Berre, and A. Sadovykh, "Specifying Services using the Service Oriented Architecture Modeling Language (SoaML)-A Baseline for Specification of Cloud-based Services.," in CLOSER, 2011, pp. 276–285.
[15] J. Amsden, "Modeling with soaml, the service-oriented architecture modeling language," IBM, Tech. Artic. Ser. January-February, pp. 1–22, 2010.
[16] C. Casanave, "Service oriented architecture using the omg soaml standard," 2009.
[17] R. G. Qiu, Service science: The foundations of service engineering and management. John Wiley & Sons, 2014.
[18] T. Erl, Service-Oriented Architecture: Concepts, Technology, and Design. Pearson Education, 2005.
[19] P. Yustianto, R. Doss, and Suhardi, "Consolidating service enginering perspectives," in Information Technology Systems and Innovation (ICITSI), 2015 International Conference on, 2015, pp. 1–7.