Constructing A Creative Service Software with Semantic Web

Pei-Shu Huang[a], Faisal Fahmi[b,c], Feng-Jian Wang[a], and Hongji Yang[d]

[a]Dept. of Computer Science, National Yang-Ming Chiao-Tung University, Hsinchu City, Taiwan {pshuang, fjwang}@cs.nctu.edu.tw

[b]EECS Int'l Graduate Program, National Yang-Ming Chiao-Tung University, Hsinchu City, Taiwan faisalfahmiy@gmail.com

[c]Information and Library Science, Airlangga University, Surabaya City, Indonesia faisalfahmiy@gmail.com

[d]Department of Informatics, University of Leicester, Leicester, England hongji.yang@leicester.ac.uk

***Abstract*—In software development, Service Oriented Architecture (SOA) and creative computing can be adopted to utilize multiple-domain knowledges to construct service software possessing creative properties, i.e., novel, useful, and surprising. In the past, several theoretical evaluation metrics have been proposed to measure creativity of a software system. However, a systematic practical method to construct creative service software is rarely considered in current researches. In this paper, we propose a model for creative service software development based on semantic web, which is applied in two phases: domain-creative requirement specification and semantic-based service design. The model can reduce communication work between domain experts and software engineers, improve traceability of the specifications, and improve machine readability during the generation of creativity. After the model of service design is validated for completeness and consistency, the creative service software is well-designed and can be implemented and reused effectively without losing of creativity.**

Keywords: creative computing; semantic web; service-oriented architecture; software engineering

I. INTRODUCTION

The software applying creative computing to generate a novel, astonish, and useful specification or computation can be defined as a creative software [1]. Theoretically, the scope of creativity in creative computing has been claimed to be more powerful than that of current Artificial Intelligence (AI) techniques [2-4], where an AI-based software is commonly applied on a specific domain only, and a creative computing is applied to create a new domain which is the combination of two different domains, or the transformation or exploration of existing domains. There are many theoretical results for creative computing [5-7]. However, it is still lack of a systematic construction method based on current software techniques such as object-oriented languages and Service-Oriented Architecture (SOA).

Recently, SOA is one of the most acceptable software development techniques, due to the enormous open services or microservices and better integration capability [8, 9]. On the other hand, semantic web is one of the best knowledge representations for machine reading and (thus) supports automatic data processing. However, a current semantic web may contain many knowledges that cannot be utilized directly for software development.

In this paper, we developed a model, with specific ontologies and frameworks based on specialized semantic web, to help constructing a creative software with SOA during requirement specification and design phases. There are a series of software development methods adopted from [8, 10] and organized to work in requirement specification and design phases. Each phase contains part of activities from these methods and their expected results are demonstrated using a real-world creative application adopted from [7]. Besides, the result of each phase is validated for consistency and completeness by the respected development participants.

However, our work does not concern software implementation, testing, and maintenance parts still. Our next work is to introduce the detailed implementation with SOA, in order to find the development pattern and characteristics of service software developed based on our model. There are several issues can be discussed further. For example, a static anomaly detection method [11] can be applied on the specification to help remove the redundancy.

The rest of this paper is organized as follows: In Section 2, the background introduces creative computing, semantic web, and SOA. In Section 3, we introduce the models based on semantic web. In Section 4, we present the series of work that need to be done and their expected results in the requirement specification and design phases respectively. Finally, Section 5 gives a conclusion and future work.

II. BACKGROUND

A. *Creative Computing*

Creative computing can be defined as the properties of software, where its computing has all factors of creativity: fresh, surprising, and useful [1]. Creative computing attempts to consolidate the objectivity of computer system with the subjectivity of human creativity resulting a creative algorithm that solve practical problems. The software construction for creative computing has been discussed more widely in recent years, targeted to produce innovative software [12].

It is observed that new knowledge created through creative ways can be used to innovate new products or services in various domains. For some products or services in a domain, an innovative product or service tends to be more successful in the competitive market than the others [7]. Besides, there are three creative ways to build new knowledge: exploratory, combinational, and transformational [2]. For the exploratory way, new knowledge can be found by researching an existing conceptual space. For the transformational way, new knowledge can be created by transforming an existing conceptual space into another. For the combinational way,

2767-6684/21/$31.00 ©2021 IEEE
DOI 10.1109/DSA52907.2021.00074

new knowledge can be created by combining similar ideas, thoughts, or knowledge. However, the current approaches of creative computing [6, 7] are only applied on regular application development which is not concerned with the reuse of existing services or microservices to improve the software development and its performance.

B. *Semantic Web*

The web, i.e., World Wide Web (WWW), is an information system where documents and other web resources are identified by Uniform Resource Locators (URLs, such as https://example.com/resource) and accessible over the Internet [13]. The Semantic Web [14-16] is proposed to extend and improve the representation of data relation in conventional web which consists of a bunch of Web of Documents linked by various approaches, e.g., hyperlinks. The Semantic Web provides a common framework allowing data and the link(s) between data to be shared and reused across application, enterprise, and community boundaries through a web. By applying Semantic Web, the Internet data becomes machine-readable, and thus they can support better discovery, data integration, navigation, and automation of tasks.

The structure of Semantic Web contains seven layers, which are identifier and character set layer, syntax layer, data interchange layer, ontology and rule layer, logic layer, proof layer, and trust layer from bottom to top. This structure allows a Semantic Web to represent various knowledges using existing ontologies, e.g., there are more than 700 ontologies with more than 1,300 vocabularies defined in Schema.org. There are two core standards to construct a document of Semantic Web: Resource Description Framework (RDF) in data interchange layer and ontologies representation language, e.g., Web Ontology Language (OWL) and Resource Description Framework Schema (RDFS) in ontology, query, and rule layer.

Using Semantic Web, the knowledge is represented as RDF triples (subject, predicate, object) based on vocabularies in specified ontologies, where a subject represents a resource, an object represents an attribute of the resource, and a predicate represents the relationship between the resource and its attribute. Fig. 1 shows an example of knowledge modified from [7] and represented in Semantic Web using Protégé. However, such a knowledge representation is too complex to be applied on the software development directly because it contains too many vocabularies unrelated to the development. In this paper, we present a simple model which can be used to help the development of creative service-based software and utilize any knowledges represented with Semantic Web.

C. *Service-Oriented Architecture*

Service-Oriented Architecture (SOA) promotes the reuse or creation of services which are applied intra- and inter-enterprise to improve enterprise agility, i.e., ability to respond to a change (industry change). The changes can include: to re-compose existing services or functions to meet the change of customer requirements, to develop new services or functions rapidly, to scale the existing systems to meet different levels of demand, etc.

Services designed with SOA commonly have the following characteristics: interoperable, loosely coupled, reusable, composable, and autonomy [8]. Although SOA technology provides several benefits in service design, the implementation and maintenance are complex for small scale organizations. Microservice technologies are further introduced to improve the performance and maintainability of an application, where a microservice has additional characteristics: bounded by context, smaller in size, and

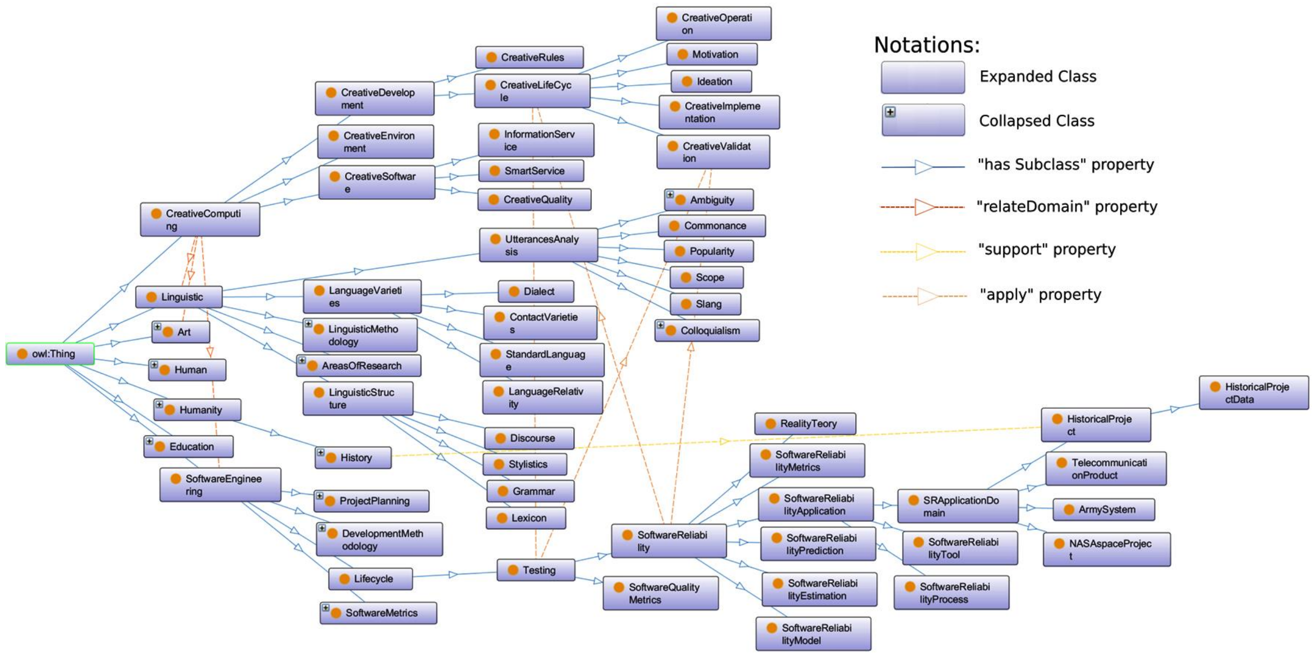


Fig. 1: An example of knowledge represented in Semantic Web using Protégé

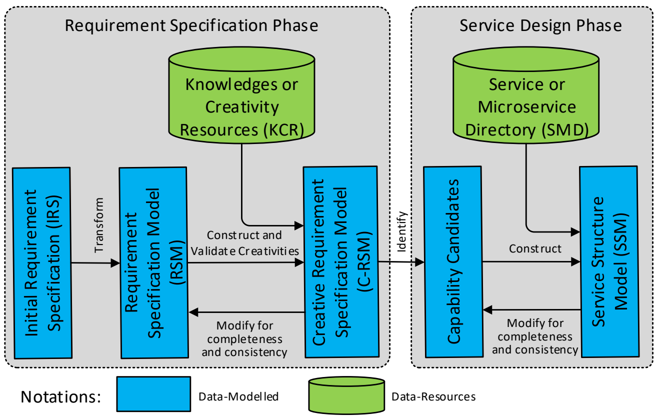

Requirement Specification Phase
Service Design Phase
Knowledges or Creativity Resources (KCR)
Service or Microservice Directory (SMD)
Initial Requirement Specification (IRS)
Transform
Requirement Specification Model (RSM)
Construct and Validate Creativities
Modify for completeness and consistency
Creative Requirement Specification Model (C-RSM)
Identify
Capability Candidates
Construct
Modify for completeness and consistency
Service Structure Model (SSM)
Notations:
Data-Modelled
Data-Resources

Fig. 2: Structure of DMCS

operationally independent. To construct a service-based application, two main steps are to identify the services/microservices and integrate them to collaborate.

However, the models of service structure used in current software development methodologies with SOA [8, 9, 17, 18] commonly separate the service description and service composition (or integration) which enable an automatic engine for service discovery to be developed but introduce complexity. Besides, it also becomes more complex to review the consistency and completeness if both descriptions are separated.

III. Data Model Based on Semantic Web to Develop Creative Service Software

In this paper, we develop a model for the development of creative service software, where the model contains several sub-models of two types: data-modelled and data-resources. The model, called Development Model of Creative Service Software (DMCS) and shown in Fig. 2, is divided into two parts used in requirement specification and service design phases separately. The detailed contents for each part and their usage are described in Subsections 3.1 and 3.2, respectively.

A. *Semantic-based Requirement Specifications*

An IRS is constructed by software engineers and domain experts firstly and transformed into RSM. C-RSM can be constructed automatically or semi-automatically by adopting the machine of creative computing algorithms to recursively merge the similar concepts in RSM and those in KCR.

1) Initial Requirement Specification (IRS)

An IRS can be constructed in natural language or common graph representation such as Use Case. Conventionally, requirement specifications contain a series of features to be provided by the software. Each system feature can be decomposed into functional requirements and non-functional requirements. The functional requirements contain the interactions between the system and the user(s) or external system(s) describing a behavior between each pair of input and output on the system. The non-functional requirement specifications include the aspects concerned by the users, but not related to functionality directly, e.g., the constraints on the performance of the system and its quality factors.

2) Requirement Specification Model (RSM)

Recent studies [3-6] show that a semantic web can be used to simplify the knowledge acquiring process inside creative computing algorithms when adding creativity into the system. In this paper, we propose a model containing directed graph and RDF/XML structure based on specialized semantic web with specific ontology and RDF, called RSM, where the ontology adopted from [19] covers all vocabularies used commonly in scenario-based modeling [10] and the RDF describes the properties of better requirement specifications based on that modeling.

The directed graph and RDF/XML structure of RSM can be transformed to each other since they are constructed based on the same ontology and RDF. The directed graph of RSM, shown in Fig. 3, can reduce the communication work between domain experts and software engineers and the RDF/XML structure of RSM can help the machine of creative computing to read and understand the requirement. Fig. 4 shows an example RDF/XML structure of RSM describing parts of directed graph in Fig. 3.

3) Knowledge and Creativity Resources (KCR)

KCR contains various existing ontology-based knowledges represented in semantic web for the information resources of the computation in creative computing. Each of them can contain several concepts, property of that concept (i.e., attributes or instances), and the relationships between the concepts and their properties. KCR can be constructed based

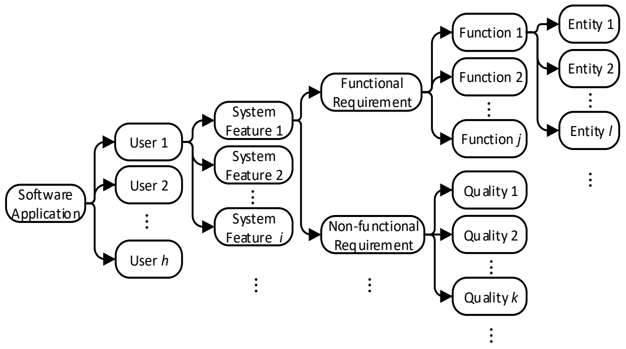


Fig. 3: The directed graph of RSM

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:ars="ApplicationRequirementSpecifications.owl">
<ars:SystemUser rdf:about="User 1">
   <ars:SystemFeature>Feature 1</ars:SystemFeature>
   <ars:SystemFeature>Feature 2</ars:SystemFeature>
   … (continue for SystemFeature)
   <ars:SystemFeature>Feature i</ars:SystemFeature>
</ars:SystemUser>
<ars:SystemUser rdf:about="User 2">
   … (SystemFeature accessed by User 2)
</ars:SystemUser>
… (continue for SystemUser)
<ars:SystemUser rdf:about="User h">
   … (SystemFeature accessed by User h)
</ars:SystemUser>
</rdf:RDF>
```

Fig. 4: RDF/XML of RSM describing part of requirement specifications represented in Fig. 2

on personal knowledge or public information. An example of KCR is shown in Fig. 1.

4) Creative Requirement Specification Model (C-RSM)

C-RSM is an RSM model with additional creativity generated by a machine of creative computing automatically or semi-automatically. The machine of creative computing can be constructed based on inference engines or Machine Learning algorithms applying explorational, transformational, and/or combinational creativities employed from [2], where the algorithms are Self-Organizing Map (SOM), Principle Component Analysis (PCA), and association rules learning (Apriori), respectively.

B. Semantic-based Service Structures

Capability Candidates are identified from the C-RSM constructed in the first phase. SSM is constructed by recursively processing the Capability Candidates and SMD.

1) Capability Candidates

A service can be defined as an independent software program that makes its functionalities available to be accessed through a message mechanism. These functionalities are also called as service capabilities [8], where capability candidates can be derived from system features and functional requirements in C-RSM.

2) Service or Microservice Directory (SMD)

To speed up the development, a service capability can be constructed as a composition of other capabilities from the existing services. SMD contains the description of service and its associated contract containing the access method and quality assurance. SMD also can be used to identify capability candidates which can be replaced with those provided by existing services.

3) Service Structures Model (SSM)

Jolie [20] is claimed as a service-oriented programming language which combines service description and its implementation to simplify the development. However, a service modeled in Jolie can increase the complexities since the service also contains its implementation details. In this paper, we proposed a model, similar to Jolie but without implementation details, containing directed graph and RDF/XML structure based on specialized semantic web, called SSM, to model a layered service structure, where each service contains its capabilities and the service composition and quality requirement associated with these capabilities (if any). The ontology for SSM is called Service Structure and adopted from [21], where the ontology contains all vocabularies of common service structures and communication mechanism between services derived from [18], e.g. communication protocols and methods, endpoint address, etc. The RDF of SSM is used to describe the properties of a layered services model, e.g., layer of service and service composition.

The model is constructed based on the services in Domain-aware Layered Service Model (DLSM) [22] which separates the services of an application into three layers to reduce the management complexities of reusability. The layers of services in DLSM include Task, Domain-Specific, and Domain-General Service Layers from top to bottom, where a service can be composed of services in bottom layer(s) and/or external services (if any). Task Service Layer contains services with non-reusable context that corresponds to single-purpose business process logic. A task service usually encapsulates service compositions that invoke other services to complete its capabilities. Domain-Specific Service Layer contains reusable services applied in a specific domain only. Domain-General Service Layer contains reusable services that can be requested from different domains. It usually encapsulates low-level technology-centric functions, such as notification, logging, and security processing.

Fig. 5 shows a directed graph of SSM to be easily understood by the software engineers. Similar to RSM, the directed graph and RDF/XML structure of SSM can be transformed to each other since they constructed based on the same ontology and RDF. To simplify the directed graph of SSM, the input and output messages for a service capability are omitted and only show the name of the capability. An example RDF/XML describing parts of service structure in Fig. 5 is shown in Fig. 6.

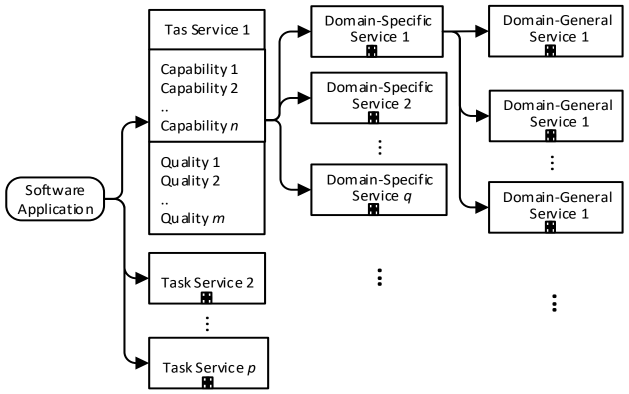
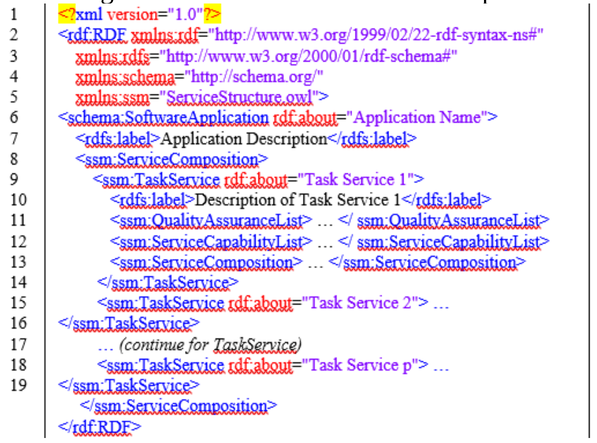


Fig. 5: A directed graph structure of SSM

IV. A Method to Construct Creative Service Software

In this section, we present a method to construct creative requirement and service design specifications for service software using DMCS. The method contains two phases:



```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:schema="http://schema.org/"
   xmlns:ssm="ServiceStructure.owl">
<schema:SoftwareApplication rdf:about="Application Name">
   <rdfs:label>Application Description</rdfs:label>
   <ssm:ServiceComposition>
      <ssm:TaskService rdf:about="Task Service 1">
         <rdfs:label>Description of Task Service 1</rdfs:label>
         <ssm:QualityAssuranceList> … </ ssm:QualityAssuranceList>
         <ssm:ServiceCapabilityList> … </ ssm:ServiceCapabilityList>
         <ssm:ServiceComposition> … </ssm:ServiceComposition>
        </ssm:TaskService>
        <ssm:TaskService rdf:about="Task Service 2"> …
</ssm:TaskService>
        … (continue for TaskService)
        <ssm:TaskService rdf:about="Task Service p"> …
</ssm:TaskService>
    </ssm:ServiceComposition>
</rdf:RDF>
```

Fig. 6: An example RDF/XML describing service structure in Fig. 5

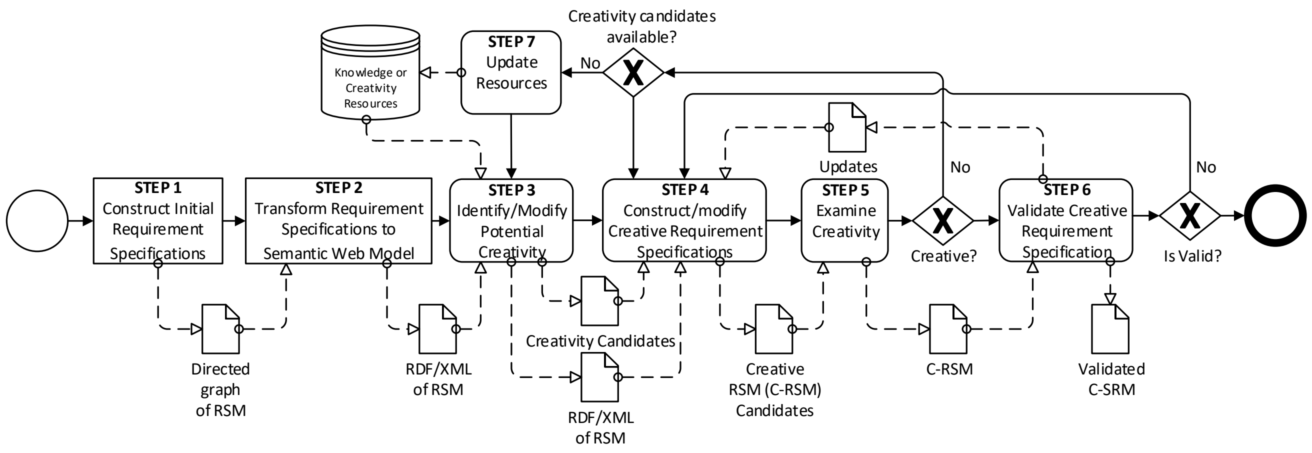

Creativity candidates available?
Knowledge or Creativity Resources
STEP 7 Update Resources
No
STEP 1 Construct Initial Requirement Specifications
STEP 2 Transform Requirement Specifications to Semantic Web Model
STEP 3 Identify/Modify Potential Creativity
STEP 4 Construct/modify Creative Requirement Specifications
Updates
STEP 5 Examine Creativity
No
Creative?
STEP 6 Validate Creative Requirement Specification
No
Is Valid?
Directed graph of RSM
RDF/XML of RSM
Creativity Candidates
RDF/XML of RSM
Creative RSM (C-RSM) Candidates
C-RSM
Validated C-SRM

Fig. 7: Activities of domain-creative requirement specifications

(1) domain-creative requirement specification and (2) semantic-based service design. In each phase, there is a distinct method designed to derive the specification, which is checked the consistency and completeness then. If any of inconsistency or incompleteness for the specifications exists, the work will move to the corresponding step for modification and improvement.

A. *Domain-Creative Requirement Specification*

The work in the first phase include two parts: (1) identify primary requirement specifications (non-creative) and (2) an iterative work to construct creative requirement specifications, with semantic web. The specification constructed in this phase contains users, system features, functional requirements and entity(ies) accessed or manipulated, and non-functional requirements. Fig. 7 shows activities of this phase, where Steps 1 and 2 (rectangle) contain the activities in Part (1) and iterations of Steps 3~7 (rounded rectangle) describe the activities in Part (2).

Step 1: Construct Initial Requirement Specification.

Input: -

Output: directed graph of RSM

Activities: The requirement specifications are constructed by software engineers based on the viewpoints of users or domain experts. To construct better requirement specification, software engineers need to discuss with users or domain experts to understand the application domain and analyze documentations of the corresponding legacy system (if any).

Step 2: Transform Requirement Specifications to Semantic Web Model.

Input: directed graph of RSM

Output: RDF/XML of RSM

Activities: To enable a machine of creative computing understand the requirement specification quickly and effectively, the directed graph inputted is transformed to RDF/XML of RSM. An algorithm from [21] is employed for the transformation.

Step 3: Identify or Modify Creativity Candidates.

Input: RDF/XML of RSM and KCR

Output: Creativity Candidates

Activities: KCR may contain some knowledges that are not appropriate when used as creativity resources, according to the context and constraint of the application domain. The set of creativity candidates is a subset of KCR that can be used to add creativity to RSM. In this step, Creativity Candidates are discovered from the knowledges represented in KCR based on syntactic and semantic similarities [5] between the concepts in KCR and those derived from system features and functional requirements in the RSM inputted, where syntactic similarity computes the distance between two input strings and semantic similarity computes the similarity of the meaning between two input strings.

Step 4: Construct or Modify Candidates of Creative Requirement Specifications.

Input: RDF/XML of RSM and Creativity Candidates

Output: C-RSM candidates

Activities: This step works to construct C-RSM candidates by iteratively merging the functional requirements inside the input of RSM with the selected properties of concepts inside the input of Creativity Candidates, where the merging options are discovered by the machine of creative computing applying explorational, transformational, and/or combinational creativities. For each iteration, the C-RSM candidate is modified based on different merging options.

Step 5: Examine Creativity.

Input: C-RSM candidates

Output: C-RSM

Activities: To examine the creativity of C-RSM candidate inputted, the creativity scores are measured based on the creativity factors: novel, surprising, and useful [5]. The C-RSM candidate with the highest creativity score and above the user defined threshold is selected to be the C-RSM.

Step 6: Validate Creative Requirement Specifications.

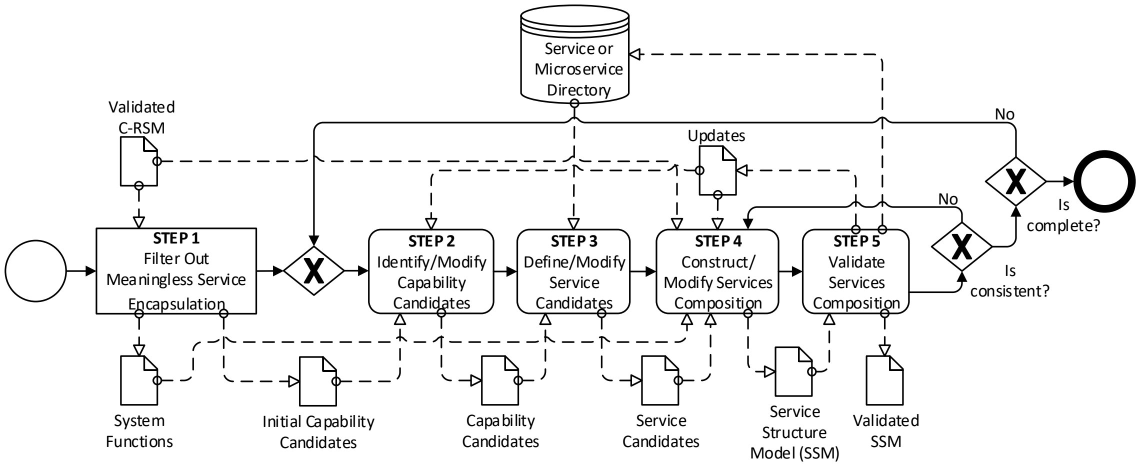
Input: C-RSM

Output: Validated C-RSM

Activities: In this step, domain experts and software engineers check the completeness and consistency of the input of C-RSM. If any error is found, go back to Step 4, perform necessary updates, and continue.

Step 7: Update Resources (if necessary).

Input: -



Service or Microservice Directory
Validated C-RSM
Updates
No
No
STEP 1 Filter Out Meaningless Service Encapsulation
STEP 2 Identify/Modify Capability Candidates
STEP 3 Define/Modify Service Candidates
STEP 4 Construct/ Modify Services Composition
STEP 5 Validate Services Composition
Is complete?
Is consistent?
System Functions
Initial Capability Candidates
Capability Candidates
Service Candidates
Service Structure Model (SSM)
Validated SSM

Fig. 8: Activities of semantic-based service design

Output: -

Activities: If all C-RSM candidates cannot pass the examination of creativity in Step 5, KCR needs to be updated with new knowledge extracted from textual documents [2], service descriptions [23], or web documents [24].

B. Semantic-Based Service Design

The work in the second phase is done by software engineers and includes two main part: the filtering out system features and functional requirements that will not be implemented as services and an iterative service design. The specification derived in this phase contains services definition and composition in SSM model, where each service definition contains its service capabilities and quality requirements. At the end of the phase, the specification is validated for consistency and completeness. The activities of this phase are shown in Fig. 8, where Step 1 (rectangle) is an activity of the first part and iterations of Steps 2~5 (rounded rectangle) contains the activities of the second Part.

Step 1: Filter Out Meaningless Service Encapsulation.

Input: validated C-RSM

Output: Initial Capability Candidates and System Functions

Activities: The system features and functional requirements in the input of validated C-RSM are divided into two categories: 1) Those that will be implemented as services are categorized into Initial Capability Candidates and 2) the rest are named as System Functions. Such a categorization is calculated based on its benefits and drawbacks toward the goals of the desired software [25]. For example, if the goal includes performance, the functional requirement whose implementation needs low latency real-time communication, it is meaningless to be encapsulated within a service.

Step 2: Identify or Modify Capability Candidates.

Input: Initial Capability Candidates and Updates (generated in Step 5 of this phase)

Output: Capability Candidates

Activities: The Initial Capability Candidates are defined as Capability Candidates directly. The modification of Capability Candidates is based on the Updates containing the System Functions to be accessed for service composition and the capability candidates used to fix the error of completeness, if required. System Functions existing in the Updates indicate to adopt one or more capability candidates to handle the communication to these System Functions.

Step 3: Define or Modify Service Candidates.

Input: Capability Candidates

Output: Service Candidates

Activities: In this step, the elements in the input of Capability Candidates are grouped based on one or more logical contexts defined in service layers of DLSM, where each group represents a service candidate. For example, the layer of task services requires that a service has a context specific to a business process of an enterprise or organization. The Service Candidates are modified for each iteration if the input is updated. The capability of services replaced with those provided by the existing services stored in SMD are also identified in this step to increase the reusability.

Step 4: Construct or Modify Services Composition.

Input: Service Candidates, Validated C-RSM, System Function, and Updates

Output: SSM

Activities: In this step, the relationships among system features and functional requirements inside the input of Validated C-RSM are used to identify the interactions among the capabilities inside the input of Service Candidates, including those provided by existing services (if any), and System Functions. All interactions discovered are then generalized to construct Service Composition. The Updates from Step 5 are used to modify Service Composition, where the Updates contain the redundant or conflicted Capability Candidates (i.e., consistency error) to be fixed. Similar to Service Candidates, the Service Composition is modified for each iteration if the input is updated. Finally, the Service Composition constructed is used to derive SSM.

Step 5: Validate Service Composition.

Input: SSM

Output: Validated SSM

Activities: In this step, software engineers validate the completeness and consistency of the services definition and

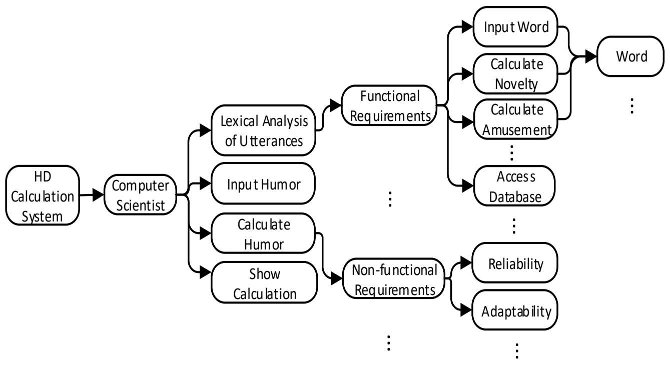
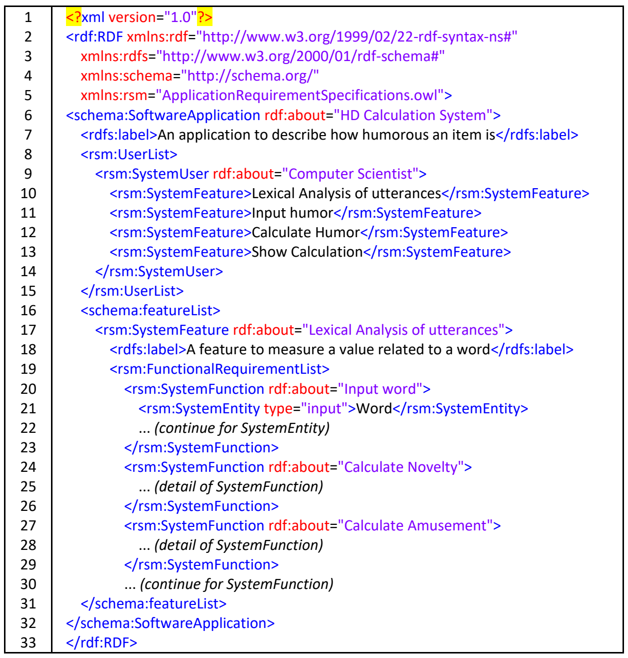


Fig. 9: An example directed graph of RSM for HD Calculation System

composition in DLSM from the input of SSM. If any error is found, go back to the corresponding Step, perform necessary updates, and continue. For example, in the service composition, if a capability of task or domain-specific service has a conflict with one or more capabilities of domain-general services, the former capability is removed to increase reusability. After validation succeeds, the validated SSM is stored into SMD to help the development of creative service software and increase reusability in the future without losing the creativity generated in the first phase.

V. Practical Example Aplling The Method

A real-world creative application, called Humor Degree (HD) Calculation System [26], is adopted to demonstrate the feasibility of the presented method. Due to the space limitation, all requirement specifications in the example are implemented as services and there is no problem of consistency and completeness s found during construction process. HD Calculation System is an application to measure the level of humor associated with an input of verbal joke containing no more than two utterances. The users of the system are computer scientists which attempt to provide a quantitative description of humor during the humor recognition using

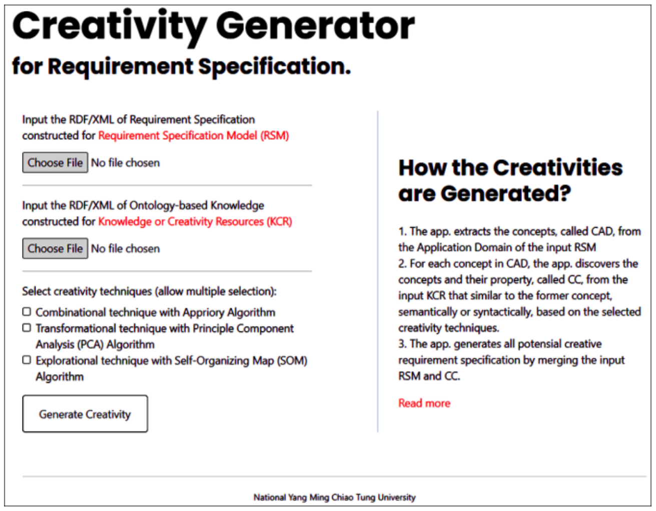


```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:schema="http://schema.org/"
   xmlns:rsm="ApplicationRequirementSpecifications.owl">
<schema:SoftwareApplication rdf:about="HD Calculation System">
   <rdfs:label>An application to describe how humorous an item is</rdfs:label>
   <rsm:UserList>
      <rsm:SystemUser rdf:about="Computer Scientist">
         <rsm:SystemFeature>Lexical Analysis of utterances</rsm:SystemFeature>
         <rsm:SystemFeature>Input humor</rsm:SystemFeature>
         <rsm:SystemFeature>Calculate Humor</rsm:SystemFeature>
         <rsm:SystemFeature>Show Calculation</rsm:SystemFeature>
      </rsm:SystemUser>
   </rsm:UserList>
   <schema:featureList>
      <rsm:SystemFeature rdf:about="Lexical Analysis of utterances">
         <rdfs:label>A feature to measure a value related to a word</rdfs:label>
         <rsm:FunctionalRequirementList>
            <rsm:SystemFunction rdf:about="Input word">
               <rsm:SystemEntity type="input">Word</rsm:SystemEntity>
               ... (continue for SystemEntity)
            </rsm:SystemFunction>
            <rsm:SystemFunction rdf:about="Calculate Novelty">
               ... (detail of SystemFunction)
            </rsm:SystemFunction>
            <rsm:SystemFunction rdf:about="Calculate Amusement">
               ... (detail of SystemFunction)
            </rsm:SystemFunction>
            ... (continue for SystemFunction)
   </schema:featureList>
</schema:SoftwareApplication>
</rdf:RDF>
```

Fig. 10: An example RDF/XML of RSM for HD

computer techniques. The users also expect the HD calculation in the system can be reliable and adaptable for humor exploration.

In the first phase, the domain experts and software engineers work on Step 1 to construct a directed graph of SRM for the requirement specifications. Here, a lexical analysis is selected to calculate the HD of utterances derived from the input, where the utterances consist of words. Two general features of humor, i.e., novelty and amusement, are selected as the computation elements, where each of them can be assigned proper values according to their contribution to the humor. To perform lexical analysis, the approach adopts a way which allows the users to store words with associated novelty and amusement values in order to calculate HD of the words derived from the inputs of humor. The result of a directed graph of SRM is shown in Fig. 9.



Creativity Generator

for Requirement Specification.

Input the RDF/XML of Requirement Specification constructed for Requirement Specification Model (RSM)

Choose File No file chosen

Input the RDF/XML of Ontology-based Knowledge constructed for Knowledge or Creativity Resources (KCR)

Choose File No file chosen

Select creativity techniques (allow multiple selection):

☐ Combinational technique with Appriory Algorithm
☐ Transformational technique with Principle Component Analysis (PCA) Algorithm
☐ Explorational technique with Self-Organizing Map (SOM) Algorithm

Generate Creativity

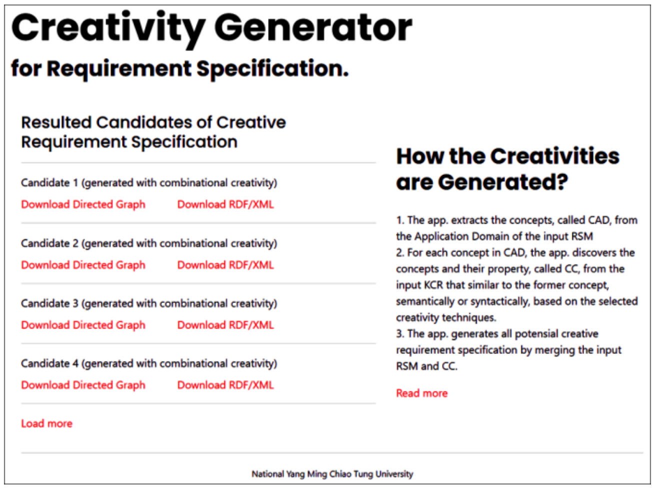
How the Creativities are Generated?

1. The app. extracts the concepts, called CAD, from the Application Domain of the input RSM
2. For each concept in CAD, the app. discovers the concepts and their property, called CC, from the input KCR that similar to the former concept, semantically or syntactically, based on the selected creativity techniques.
3. The app. generates all potensial creative requirement specification by merging the input RSM and CC.

Read more

National Yang Ming Chiao Tung University

Fig. 11: A web interface of CG for input



Creativity Generator

for Requirement Specification.

Resulted Candidates of Creative Requirement Specification

Candidate 1 (generated with combinational creativity)
Download Directed Graph Download RDF/XML

Candidate 2 (generated with combinational creativity)
Download Directed Graph Download RDF/XML

Candidate 3 (generated with combinational creativity)
Download Directed Graph Download RDF/XML

Candidate 4 (generated with combinational creativity)
Download Directed Graph Download RDF/XML

Load more

How the Creativities are Generated?

1. The app. extracts the concepts, called CAD, from the Application Domain of the input RSM
2. For each concept in CAD, the app. discovers the concepts and their property, called CC, from the input KCR that similar to the former concept, semantically or syntactically, based on the selected creativity techniques.
3. The app. generates all potensial creative requirement specification by merging the input RSM and CC.

Read more

National Yang Ming Chiao Tung University

Fig. 12: The candidates of C-RSM generated by CG

```
…   …
24      <rsm:SystemFunction rdf:about="Calculate Ambiguity">
25          … (detail of SystemFunction)
26      </rsm:SystemFunction>
27      <rsm:SystemFunction rdf:about="Calculate Popularity">
28          … (detail of SystemFunction)
29      </rsm:SystemFunction>
…   …
```

Fig. 13: Part of C-RSM for HD Calculation System

In Step 2, the directed graph is transformed into RDF/XML of RSM, shown in Fig. 10. We developed a web interface applying algorithm in [19] to work for Steps 3 and 4, called Creativity Generator (CG), where Step 3 discovers the Creativity Candidates from KCR shown in Fig. 1 and Step 4 merges the Creativity Candidates discovered with the RDF/XML of RSM. Fig 11 shows the interface to input the RDF/XML of RSM and KCR, and preferred creativity technique selection in CG. Fig. 12 shows the candidates of C-RSM generated from the execution of Steps 3 and 4 in CG sequentially. Lines 24~29 of a C-RSM candidate in Fig. 13 show the result of merging between the functional requirements for *Lexical Analysis of Utterances* containing *Calculate Novelty* and *Calculate Amusement*, shown in Lines 24~29 of Fig. 10, and properties of *Utterances Analysis* concept containing *Ambiguity*, *Slang*, *Colloquialism*, *Commonance*, *Popularity*, and *Scope*, discovered from KCR in Fig. 1. The generated C-RSM candidates are then examined for creativity in Step 5 and validated for consistency and completeness in Step 6.

In the second phase, the software engineers extract the requirement specifications from the validated C-SRM to identify Initial Capability Candidates in Steps 1, which are defined as Capability Candidate in Step 2 and shown in Fig. 14. In Step 3, these Capability Candidates are grouped based on the contexts defined in the layer of services in DLSM. For example, the Capability Candidates of *Calculate Commonance*, *Calculate Ambiguity*, *Calculate Popularity*, …, and so on, are grouped as *Utterances Analysis* service candidate in domain-specific layer since the capabilities are reusable for a specific domain only. Step 4 identifies the Service Composition for Service Candidates, resulted in Step 3, to construct the SSM model. Fig. 15 shows an example of service definition, containing a set of capabilities and quality requirements (if any), and service composition, where the blue color indicates the reusable existing service adopted. Fig. 16

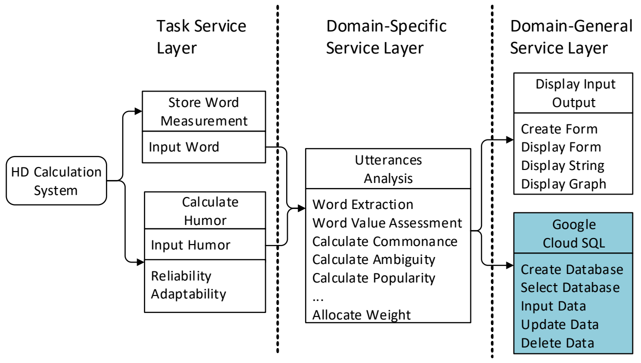


Fig. 15: An example service composition of HD Calculation System

- Store word measurement
- Input Humor
- Calculate Humor
- Show Calculation
- Input Word
- Calculate Commonance
- Calculate Ambiguity
- Calculate Popularity
- Access Database
- Word Value Assignment
- Allocate Weights
- Word extractions

…

…

Fig. 14: Example capability candidates of HD Calculation System

shows an example of SSM model based on Fig. 15. After the SSM is validated in Step 5, the creative service software is well-constructed and can be implemented effectively without losing the creativities generated in Phase 1.

The development activities inside the example above show the feasibility of our presented method and its associated specification models to construct a creative service software. The model constructed during the first phase can be used to generate necessary creativity successfully. In the second phase, the model of services in SSM can organize the services with no reusability, reusability in a specific domain, and reusability in multi domain, into different layers of service. Besides, the validated SSM can also be stored into SMD to help the development of creative service software in the future. For example, if SMD contains necessary SSM models, a machine of service discovery can automatically and effectively discover the available existing services to be reused from an input of C-SRM.

VI. Discussion And Concluding Remarks

In this paper, we presented a novel specification model based on semantic web that are applied on a systematic method to construct creative software with SOA. The method

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:swss="ServiceStructure.owl">
<swss:DomainSpecificService rdf:about="Utterances Analysis">
   <rdfs:label>An analysis of a group of words</rdfs:label>
   <swss:ServiceCapabilityList>
      <swss:ServiceCapability rdf:about="Word Extraction">
         <rdfs:label>Exstract each word from an input</rdfs:label>
         <swss:MessageMechanism>REST</swss:MessageMechanism>
         <swss:ServiceEndPoint>http://example.com/utterance/extract
</swss:ServiceEndPoint>
         <swss:MessageInput>
            <xs:element name="Humor" type="xs:String"/>
            … (continue for xs:element)
         </swss:MessageInput>
         <swss:ServiceComposition>
            <swss:GenericService rdf:about="Display Input/Output">
               <rdfs:label>A service to make and display input/output user
interface</rdfs:label>
               … (continue for ServiceCapability)
            </swss:GenericService>
         </swss:ServiceComposition>
         <swss:MessageOutput>
            <xs:element name="Status" type="xs:String"/>
            … (continue for xs:element)
         </swss:MessageOutput>
      </swss:ServiceCapability>
      <swss:ServiceCapability rdf:about="Word Value Assessment">
         … (details of ServiceCapability)
      </swss:ServiceCapability>
      … (continue for ServiceCapability)
   </swss:ServiceCapabilityList>
</swss:DomainSpecificService>
</rdf:RDF>
```

Fig. 16: An example SSM of HD Calculation System

is composed of two phases, requirement specification and service design, where each phase adopts a novel specification model containing directed graph and RDF/XML structure, and each pair of input and output for each step is defined clearly. The service software constructed based on the proposed method can contain not only the characteristics of services in SOA, but also the creativity properties. We further provide a practical example to demonstrate the feasibility of this method and its associated specification model. However, there are many non-functional aspects not considered inside the method. For example, empirical evaluation for the reusability of the services constructed, communications speed between services in DLSM, and patterns to better utilize the creativity resulted. Besides, it neither considers the development issues during the implementation, testing, and maintenance phases.

REFERENCES

[1] L. Zhang and H. Yang, "Definition, research scope and challenges of creative computing," in *Proceedings of 19th International Conference on Automation and Computing*, London, UK, pp. 1-6, 2013.

[2] D. Jing and H. Yang, "Domain-Specific 'Ideation': Real Possibility or Just Another Utopia?", *Applied Science Journal*, pp. 68-99, 2015.

[3] L. Zhang, H. Yang, C. Zhang and N. Li, "A New Way of Being Smart? Creative Computing and Its Applications in Tourism," in *Proceedings of 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Tokyo, Japan , pp. 45-50, 2018.

[4] C. Y. Huang, M. C. Yang, and C. Yu. Huang, "An Empirical Study on Factors Influencing Consumer Adoption Intention of an AI-Powered Chatbot for Health and Weight Management." *International Journal of Performability Engineering*, vol. 17, no. 5. pp. 422-432, 2021.

[5] L. Zhang, L. Zou, D. Jing and H. Yang, "An Approach to Constructing a General Framework for Creative Computing: Incorporating Semantic Web," in *Proceedings of 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, Oxford, UK, pp. 297-306, 2016.

[6] D. Jing, H. Yang, L. Xu and F. Ma, "Developing a Creative Idea Generation System for Innovative Software Reliability Research," in *Proceedings of 2015 Second International Conference on Trustworthy Systems and Their Applications*, Hualien, Taiwan, pp. 71-80, 2015.

[7] H. Yang, D. Jing and L. Zhang, "Creative Computing: An Approach to Knowledge Combination for Creativity?," in *Proceedings of 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, Oxford, UK, pp. 407-414, 2016.

[8] T. Erl, *Service-Oriented Architecture: Analysis and Design for Services and Microservices*, Prentice Hall, 2016.

[9] F. J. Wang and F. Fahmi, "Constructing a Service Software with Microservices", in *Proceedings of 2018 IEEE World Congress on Services (SERVICES)*, San Francisco, CA, USA, pp. 43-44, 2018.

[10] B. Brueggle and A.H. Dutoit, *Object-Oriented Software Engineering: Using UML Patterns and Java*, Prentice Hall, 2004.

[11] P.S. Huang, F. Fahmi and F.J. Wang, "Improving the Detection of Artifact Anomalies in a Workflow Analysis," *IEEE Transactions on Reliability*, doi: 10.1109/TR.2020.3048612, 2021.

[12] A. Hugill and H. Yang, "The Creative Turn: New Challenges for Computing", *International Journal of Creative Computing*, vol. 1, no. 1, pp. 4–19, 2013.

[13] Wikipedia, "World Wide Web", https://en.wikipedia.org/wiki/World_Wide_Web

[14] World Wide Web Consortium (W3C), "Semantic Web", https://www.w3.org/standards/semanticweb/

[15] B.-L. Tim. (1994). Plenary at WWW Geneva 94. http://www.w3.org/Talks/WWW94Tim/

[16] W. Hall and K. O'Hara, "Semantic Web," *Encyclopedia of Complexity and Systems Science*, R. A. Meyers, Ed., ed New York, NY: Springer New York, pp. 8084-8104, 2009.

[17] T. Erl, *SOA: Principles of Service Design*, Prentice Hall, 2008.

[18] S. Newman, *Building Microservice: Designing Fine-Grained Systems*, O'Reilly Media, 2015.

[19] F. Fahmi, P. S. Huang, F. J. Wang, and H. Yang, "Constructing a Creative Software with Services," Under submission review.

[20] Jolie: The Service-Oriented Programing Language, http://www.jolie-lang.org/.

[21] Z. Wu, et al., "Implementing an Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle," in *Proceedings of 2008 IEEE 24th International Conference on Data Engineering*, Cancun, Mexico, pp. 1239-1248, 2008.

[22] P. S. Huang, F. Fahmi, F. J. Wang, "A Model to Helping the Construction of Creative Service-Based Software", Accepted in *45th IEEE Annual Computer Software and Applications Conference (COMPSAC)*, 2021.

[23] N. Zhang, J. Wang and Y. Ma, "Mining Domain Knowledge on Service Goals from Textual Service Descriptions," *IEEE Transactions on Services Computing*, vol. 13, no. 3, pp. 488-502, 2020.

[24] H. Alani et al., "Automatic ontology-based knowledge extraction from Web documents," *IEEE Intelligent Systems*, vol. 18, no. 1, pp. 14-21, 2003.

[25] S. Newman, *Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith*, O'Reilly Media, 2019.

[26] T. Liu, H. Yang, and F. J. Wang, "A Creative Approach to Humour Degree Calculation for Utterances," in *Proceedings of 20th IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Macau, China, pp. 650-656, 2020.