Available online at JDR Website: http://journal.unublitar.ac.id/jdr

Journal Of Development Research, 3(1), May 2019, Page 1-6

DOI: https://doi.org/10.28926/jdr.v3i1.59

# The User Interface (UI) Discovery Application to Measure Query Accuracy on Interface Repository

**Shofiyatun Najah[1], Muhammad Ainul Yaqin[2], Linda Salma Angreani[3]**
**Abd. Charis Fauzan[4]**

[1,2,3]*Department of Informatics Engineering, Universitas Islam Negeri Maulana Malik Ibrahim, Indonesia*
[4]*Department of Computer Science, Universitas Nahdlatul Ulama Blitar, Indonesia*

*Email: [1]shofiyatun.najah11@gmail.com, [2]yaqinov@ti.uin-malang.ac.id, [3]linda@uin-malang.ac.id, [4]abdcharis@unublitar.ac.id*

Received: 6 November 2018; Revised: 26 April 2019; Accepted: 5 May 2019

## Abstract

SaaS (Software as a Service) application has many architecture layers, those are: data layer, service layer, process layer, and UI layer. UI layer is a GUI layer which has function to prepare the interface both system and user receiving input from user and show back to user. This research tells about how to build user interface (UI) discovery as a part of software based on SaaS. The aim of this application is to make easier user for finding user interface which appropriate needed configuration. This UI discovery application is developed by using ontology theory, those are RDF map and query SPARQL. The working way of UI discovery is entering the keywords which contain form name, or component details needed. UI discovery will search based on the entering keywords automatically. The trials will use two search models, those are data search and detail search. The calculation result of those both trials are from recall score 95%, accuracy score 100%, and precision score with different percentage. The percentage of data search model is 95%, while on detail search model is 90%. Those differences found because of some of irrelevant document found on detail search and netted query. Therefore, on data search shows that the result is more relevant and shows the better of system's prosperity.

**Keywords**: software as a service; discovery; query accuracy; interface repository

## Introduction

SaaS (Software as a Service) is a model where applications are offered to clients as a service. If a service is presented to the client, the client does not need to treat and update the application, But conversely, if the provider will update the application, then the client only following the provider, it was explained by previous research on similar topics (Najah, Ainul Yaqin, Angreani, & Fauzan, 2019). SaaS has a variety of architectural layers to fill the functional requirements in an application. SaaS is one of the service delivery models where of software as a service will change the way people build, sell, buy and use software (Kumar, 2014). The purpose of this research is finding the user interface of program become more accessible. The step to overcome these problems are by using an ontology approach (Fauzan, Sarno, & Ariyani, 2018).

An ontology is an explicit, machine readable specification of a shared conceptualization (Gómez-pérez, Corcho, & Madrid, 2002). A development tool for Semantic Web applications should provide services to access, visualize, edit, and use ontologies (Knublauch, Fergerson, Noy, & Musen, 2004). With the growth of ontologies over the Web, one of the challenging tasks is to search for the desired ontologies (Patel, Supekar, Lee, & Park, 2003). Ontology is used for

the search process. Then, the keyword are searched are not only the keywords entered by user but those keywords have synonims and related with the words are semantically arranged in the ontology. The GUI layer, UI ontology is built to give wider meaning and make computer systems work easier. Ontology is a model of lists the types of objects, a relation can connect with each other, and a way to combine objects and relations. SaaS has layers which consists of data layer, service layer, process layer, UI layer respectively. The data layer and the service layer are the foundation and they establish the data structure and operations for applications. The process layer manages collaboration mechanisms, organizes services into the process to achieve complex tasks. The UI layer provides the interface between systems and end users, accepts input from users and returns results back to users (Shao, 2011).

UI layer makes users easy to change and configure their experience, including adding/ deleting icons, colors, fonts, titles, menus. The step to developed a user interface is to become an important role in system, namely by User Interface Management System (UIMS). User Interface Management System (UIMS) is a mechanism for the process of separating the Graphic User Interface (GUI) code in a computer program. The purpose of UIMS is creating how to get a consistent interface has a display of different applications but in the same system. The workshop was held in Seattle, in 1982. The workshop explained about development of graphical user interfaces, where interactive input was handled by the application. The result of the workshop was user interface is implemented by using tools of programming and separate from the application code. User Interface Management Systems (UlMS) are the mediators between the user and the application programs (Enderle & Duce, n.d.). Interactive application should have software as a support in controlling applications (external control) than application code (internal control) (Barber & Lucas, 1983). The Seeheim architecture calls for three UIMS components (Dan R. Olsen, 1992).

The primary component is the dialog control. The dialogue control component manages the dialogue between the user and the application. This structure is then converted into a sequence of input tokens sent to the application interface model in order to execute the command (Green, 1985). Presentation model is the component responsible for the interface that includes input and output available to the user. RDF (Resource Description Framework) is used as a representation of semantic web technology components. RDF is a metadata used to describe the address of a resource on the web. SPARQL is a component used as a query to retrieve data stored in RDF. UI Discovery is an important part of web service architecture. UDDI (Universal Description, Discovery, and Integration) provides a standardization and discovery of UI ontology for keyword searches. The discovery process cannot be known on the UI web service if the keyword is different but has the same meaning. So, the search can be resolved by discovery system to match the query on web service using ontology. Measurement of business process similarity is searching of distance between two business processes were compared (Fauzan, Sarno, & Yaqin, 2018) and it can be done in various ways, that it can use a label matching similarity , structural similarity, or behavioral similarity (Fauzan, Sarno, Yaqin, & Jamal, 2017) and (Yaqin, Sarno, & Fauzan, 2017).

This research proposed approaches annotation interface repository using RDF Map as ontology with UI discovery. So that, UI discovery be smart and automated then use SPARQL query.

## Material and Methods

The data used was obtained from previous research that is about web services in ERP boarding schools in academic. The data is processed by the PHP file that contains an interface on a form. Then, the file parsed to take the interface components. Figure 1 shows the diagram block of process UI discovery.

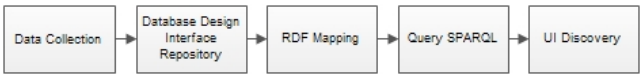


Figure 1. Diagram Block of Process UI Discovery

### Data Collection

The data is used namely a PHP document. Figure 2. Shows the PHP document contains interface with various HTML tags.

```
<form method='POST' id='form-add-
angkatan'>
<a style='cursor:pointer;' id='referensi'>
<font size='1' color='#000000'>
<b>Referensi</b></font></a><a
style='cursor:pointer;' id='kembali' >
<font ><b>Angkatan</b>
</font></a> >
<font size='1' color='#000000'><b>$status
Angkatan</b></font>
<fieldset>
<legend><span id='judul'>$status</span>
Angkatan</legend>
```

```
<dl class='inline'>
<dt><label>Departemen :</label></dt>
<dd><form>$select</form></dd>
<font size='1' fontfami-
ly='Tahoma,Verdana,Arial'>
Nama Departemen Tidak Boleh dari 50 Karak-
ter </font>--></dd>
<dt><label>Angkatan :</label></dt>
<dd><input type=text name='nm-angkatan'
id='nmangkatan' placeholder='Angkatan'
maxlength=50 size=30><br><font size='1'
font family='Tahoma,Verdana,Arial'><t>*
Tidak boleh lebih dari 50 karakter</t></
font></br></dd>
<dt><label>Keterangan :</label></dt>
<dd><textarea name='keterangan'
id='keterangan' rows='3' cols='45'></
textarea></dd></dl>
<div class='buttons'>
<input class='button blue' type=submit
value=$status id='$status'>
<input class='button red' type=reset val-
ue=reset>
</div>
</fieldset></form> ?>
```

Figure 2. The Interface Document

**Database Design Interface Repository**

The data is processed by the PHP file that contains an interface on a form. Then, the document parsed to take the interface components. Figure 3. Shows the design database of interface repository, the name of tables are form and component. The function of form table is to save the identify that are name, location, and source code or contents of the form. While, the functon component table is to save data atributes of interface elements are the name of components, and component IDs.

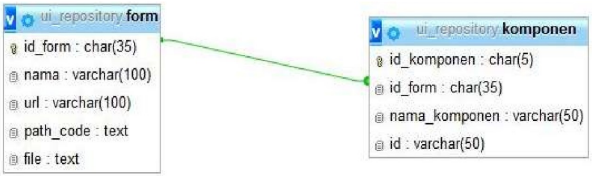


Figure 3. Database Interface Repository

## RDF Map

Nowadays, the most formatted data is stored in relational databases. To be able to use this data in a semantic context, it has to be mapped to RDF, the data format of the Semantic Web. RDF Map is an ontology of web semantic as an intermediary in accessing data in a repository. Figure 4. shows RDF Map from the ui_repository database saved in turtle format.

```
@prefix map: <#> .
@prefix db: <> .
@prefix vocab: <vocab/> .
@prefix rdf: <http://www.w3.org/1999/02/22
-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/
rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/
XMLSchema#> .
@prefix d2rq: <http://www.wiwiss.fu-
berlin.de/suhl/bizer/D2RQ/0.1#> .
@prefix jdbc: <http://d2rq.org/terms/jdbc/
> . map:database a d2rq:Database;
 d2rq:jdbcDriver "com.mysql.jdbc.Driver";
 d2rq:jdbcDSN "jdbc:mysql:///
ui_repository";
 d2rq:username "root";
jdbc:autoReconnect "true";
jdbc:zeroDateTimeBehavior
"convertToNull";.
# Table form
map:form a d2rq:ClassMap;
 d2rq:dataStorage map:database;
      d2rq:uriPattern "form/
@@form.id_form|urlify@@";
d2rq:class vocab:form;
d2rq:classDefinitionLabel "form";.
map:form__label a d2rq:PropertyBridge;
 d2rq:belongsToClassMap map:form;
 d2rq:property rdfs:label;
 d2rq:pattern"form #@@form.id_form@@";
 map:form_id_form a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:form;
d2rq:property    vo-
cab:form_id_form;
d2rq:propertyDefinitionLabe
l    "form    id_form";
d2rq:column "form.id_form";
map:form_nama a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:form;
d2rq:property vocab:form_nama;
d2rq:propertyDefinitionLabel "form nama";
d2rq:column "form.nama";.
map:form_url a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:form;
d2rq:property vocab:form_url;
d2rq:propertyDefinitionLabel "form url";
d2rq:column "form.url";.
map:form_path_code a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:form;
d2rq:property vocab:form_path_code;
d2rq:propertyDefinitionLabel "form
path_code";  d2rq:column
"form.path_code";. map:form_file a
d2rq:PropertyBridge;
d2rq:belongsToClassMap map:form;
d2rq:property vocab:form_file;
d2rq:propertyDefinitionLabel "form file";
d2rq:column "form.file"; .
# Table komponen
map:komponen a d2rq:ClassMap;
d2rq:dataStorage map:database;
d2rq:uriPattern "komponen/
  @@kompo-
  nen.id_komponen|urlify@@";
d2rq:class vocab:komponen;
d2rq:classDefinitionLabel
        "komponen";.
map:komponen__label
d2rq:PropertyBridge;
d2rq:belongsToClassMap map:komponen;
d2rq:property rdfs:label;
d2rq:pattern "komponen
#@@komponen.id_komponen@@";   .
map:komponen_id_komponen    a
d2rq:PropertyBridge;
d2rq:belongsToClassMap map:komponen;
```

```
d2rq:property vocab:komponen_id_komponen;
d2rq:propertyDefinitionLabel "komponen
id_komponen";  d2rq:column
"komponen.id_komponen"; .
 map:komponen_id_form a
 d2rq:PropertyBridge;
 d2rq:belongsToClassMap map:komponen;
 d2rq:property vocab:komponen_id_form;
 d2rq:propertyDefinitionLabel "komponen
 id_form";  d2rq:column
 "komponen.id_form"; .
map:komponen_nama_komponen a
d2rq:PropertyBridge;
d2rq:belongsToClassMap map:komponen;
d2rq:property vo-
cab:komponen_nama_komponen;
d2rq:propertyDefinitionLabel "komponen
nama_komponen";  d2rq:column
"komponen.nama_komponen";
      .
map:komponen_id a
d2rq:PropertyBridge;
d2rq:belongsToClassMap map:komponen;
d2rq:property vocab:komponen_id;
d2rq:propertyDefinitionLabel "komponen
        id";
d2rq:column "komponen.id";.
```

Figure 4. The Result of RDF *Mapping*

The Mapping of this research created by template of D2R Mapping Language. Map: The database is used to connect the ui_repository database, on the mapping file there is a map: database. Map: ClassMap serves to describe the columns in the database. The each classMap has a Property Bridge. The functions of Property Bridge is to initialize each column in the table. Vocab: is a part of the Property Bridge, that is used as a variable in the SPARQL query.

The result of process mapping is a RDF file in Turtle (.ttl) format. Then, the database of SQL can be accessed by SPARQL queries using D2RQ tool. Figure 5. Describes about UI discovery approach with RDF mapping. After that, the user can access the converted data from relational model to RDF using D2R server (Chen, Zhao, & Zhang, 2013).

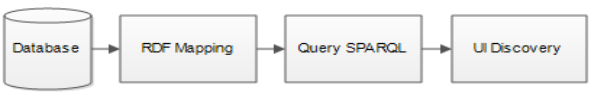


Figure 5. The Approach of RDF Mapping

## SPARQL

SPARQL is a query language used for RDF data. In 2004 the RDF Data Access Working Group (part of the Semantic Web Activity) released a first public working draft of a query language for RDF, called SPARQL (Consortium., 2003). The syntax of query SPARQL are select, where, and from. This research used three variables are name of form, URL, and code. The variable begins with question mark symbol (?) on select syntax. The "where" syntax is used to select the field of object discovery.The "from" syntax is used to determine the table which the data will be retrieved. In Figure 8 we implement the SPARQL query on the process of discovery.

```
SELECT DISTINCT ?Judul ?Link ?code
WHERE {
?as vocab:form_nama?Judul.
?as vocab:form_file?code.
?as vocab:form_url?Link

  Filter(regex(?Judul,'$kkunci','i')||
regex(?code,'$kkunci','i')||
         regex(?Link,'$kkunci','i'))
       }
```

Figure 8. SPARQL Query

## Results and Discussion

The scenario of this research is the repository interface data search process in data search menu. It is used form name as keywords. In this step, data search process only uses name form search which saves in interface repository, and the result is a name form which contains word element as user's keyword. The next step is detail search process. In this step, the searching is based on name's form or metadata which saves in repository.'

The trial of this system includes data query search trial in interface repository. Precission score and recall will indicate document relevance score which is found from discovery process. The trial uses 30 form data from academic sector of Islamic boarding school information systems. The result of recall, precission, and accuracy calculation trial on 20 queries which has been tried on data search and detail search step has average result as in Table 1.

Table 1. Test Results

| | *Recall* | *Precision* | *Accuracy* |
|---|---|---|---|
| Data Search | 95% | 95% | 100% |
| Detail Search | 90% | 95% | 100% |

The trial result which has done based on the previous scenario known that there was a difference on string-matching word search which was input by user on UI discovery. By using 20 keywords, the trial of data search and detail search on interface repository has showed ratio accurate score.

The difference of accurate score is caused of there were some of irrelevant document found on

detail search and netted query. Therefore, on data search shows that the result is more relevant and shows the better of system's prosperity.

## Conclusion

Based on the trial, on user interface discovery application is used RDF map as ontology approach and as mediator to access the data in database relational. The differences of data search and detail search are seen from the difference of entering query in searching process. On interface data search step, the netted query is based on name form only. Whereas on detail search step, the netted query is not only name form but also all saved query in database.

Based on the trial of two searching model, data search and detail, by entering 30 data and 20 queries which has tried in each searching models. On data search step got recall score 95%, precission 95%, dan accuracy 100%, whereas got recall score 95%, precission 90%, dan accuracy 100%. Those differences found because of some of irrelevant document found on detail search and netted query. Therefore, on data search shows that the result is more relevant and shows the better of system's prosperity.

## Suggestion

This paper, there are a fiew things need to be developed, namely adding the stage of workmanship system with indexing method on search result using weghting techniqes to get more relevant search results, and use OWL standardization in writing descriptions and word equations, so that discovery processes become easier.

## Acknowledgement

Authors would like to thank the Informatics Engineering Department, Faculty of Science and Technology. State Islamic University of Maulana Malik Ibrahim Malang.

## References

Barber, R. E., & Lucas, H. C. (1983). System response time operator productivity and job satisfaction. *Communications of the ACM*, *26*(11), 972–986. https://doi.org/10.1145/182.358464

Chen, Y., Zhao, X., & Zhang, S. (2013). Publishing rdf from relational database based on d2R improvement. *WSEAS Transactions on Information Science and Applications*, *10*(8), 241–248.

Consortium., A. J. U. (2003). *The Unicode standard*. Addison-Wesley 2003 pp: 1462.

Dan R. Olsen, J. (1992). *User Interface Management Systems: Models And Algorithms*. San Mateo, California: Morgan Kaufmann Publishers.

Enderle, G., & Duce, D. (n.d.). *User Interface Management Systems* (G. E. Pfaff, ed.). The European Association for Computer Graphics.

Fauzan, A. C., Sarno, R., & Ariyani, N. F. (2018). Structure-based ontology matching of business process model for fraud detection. *Proceedings of the 11th International Conference on Information and Communication Technology and System, ICTS 2017*, *2018-Janua*, 221–225. https://doi.org/10.1109/ICTS.2017.8265674

Fauzan, A. C., Sarno, R., & Yaqin, M. A. (2018). Petri net arithmetic models for scalable business processes. *Proceeding - 2017 3rd International Conference on Science in Information Technology: Theory and Application of IT for Education, Industry and Society in Big Data Era, ICSITech 2017*, *2018-Janua*, 104–109. https://doi.org/10.1109/ICSITech.2017.8257093

Fauzan, A. C., Sarno, R., Yaqin, M. A., & Jamal, A. (2017). Extracting Common Fragment Based on Behavioral Similarity Using Transition Adjacency Relations For Scalable Business Processes. *International Conference on Information & Communication Technology and System (ICTS)*, 131–136.

Gómez-pérez, A., Corcho, O., & Madrid, U. P. De. (2002). *for the Semantic Web*. *4*.

Green, M. (1985). The University of Alberta user interface management system. *SIGGRAPH Comput. Graph.*, *19*(3), 205–213. https://doi.org/10.1145/325165.325286

Knublauch, H., Fergerson, R. W., Noy, N. F., & Musen, M. A. (2004). *The Protege OWL Plugin : An Open Development Environment for Semantic Web Applications*. 229–230.

Kumar, K. V. K. M. (2014). *Software as a Service for Efficient Cloud Computing*. 2319–2322.

Najah, S., Ainul Yaqin, M., Angreani, L. S., & Fauzan, A. C. (2019). Mengukur Akurasi Query Pada Interface Repository

 

Menggunakan User Interface (UI) Discovery Berbasis Software as a Service (SAAS). *BRILIANT: Jurnal Riset Dan Konseptual*, *4*(1), 206–214. https://doi.org/ http://dx.doi.org/10.28926/ briliant .v3i4.312

Patel, C., Supekar, K., Lee, Y., & Park, E. K. (2003). *OntoKhoj : A Semantic Web Portal for Ontology Searching , Ranking and Classification*. 58–61.

Shao, Q. (2011). *Towards effective and intelligent multi-tenancy SaaS*. (May).

Yaqin, M. A., Sarno, R., & Fauzan, A. C. (2017). Scalability measurement of business process model using business processes similarity and complexity. *International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, *4*(September), 306–312. https:// doi.org/10.11591/eecsi.4.1033

 