2nd International Workshop on Software Mining, 11th Nov 2013, Palo Alto, CA, USA

Interaction Traces Mining for Efficient System Responses Generation

Miao Du [§], Steve Versteeg [+], Jean-Guy Schneider [§], Jun Han [§], and John Grundy [§]

[§] Faculty of Information and Communication Technologies
Swinburne University of Technology
Hawthorn, VIC 3122, Australia
{miaodu, jschneider, jhan, jgrundy}
@swin.edu.au

[+] CA Labs
Level 2, 380 St. Kilda Rd
Melbourne, VIC 3004, Australia
steven.versteeg@ca.com

ABSTRACT

Software service emulation is an emerging technique for creating realistic executable models of server-side behaviour. It is particularly useful in quality assurance and DevOps, replicating production-like conditions for large-scale enterprise software systems. Existing approaches can automatically build client-server and server-server interaction models of complex software systems directly from analysis of service interaction trace data. However, when these interaction traces become large, searching an entire trace library to generate a run-time responses can become very slow. In this paper we describe a new technique that utilises data mining, specifically clustering algorithms, to pre-process large amounts of recorded interaction trace data. With the obtained clusters we facilitate efficient yet well-formed runtime response generation in our Enterprise System emulation environment. We evaluate our approach using two common application-layer protocols: LDAP and SOAP. Our experimental results show that by utilising clustering techniques in the pre-processing step, the response generation time can be reduced by 99% on average compared with existing approaches.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*Restructuring, reverse engineering, and reengineering*; D.2.9 [**Software Engineering**]: Management—*Software quality assurance (SQA)*

General Terms

Algorithms, Measurement, Performance

Keywords

Service emulation, Interaction emulation, Traces clustering, Automatic modelling

1. INTRODUCTION

Enterprise software systems have grown very large and complex. Many software systems provide services that in turn rely on third party services or systems to perform their functionalities. Testing such a system must be inclusive of the functionality which interacts with third party software services. However access to the real third party services during testing may be restricted, or expensive, or not available at the very large scale representative of the production environment. Having an *emulation environment* [12, 13], where realistic interactive models of the third party services are executed, is therefore useful for the purposes of quality assurance and in DevOps.

The creation of executable models is pivotal to the emulation approach. There are two common approaches to building such executable models: 1) to manually define interaction models by taking advantage of available knowledge about the underlying interaction protocol(s) and system behaviour [12], and 2) to automatically infer executable models based on mining interaction recordings (henceforth referred to as *interaction traces* in this paper) [7]. Manually defining interaction models has an advantage in defining complex sequences of request/response patterns between elements of the system including suitable parameter values. However, in some cases, this approach is not feasible due to the time required or lack of required expertise.

The core of our automatic modelling approach is to synthesize protocol conformant responses by mining interaction traces, thereby mimicking the interaction behaviour among enterprise system components. In [7], we introduced a framework consisting of two main functions: the *Distance Function* and the *Translation Function*. Given an incoming request to a modelled service, the *Distance Function* is used to search for the most similar request in the previously recorded interaction traces by computing their distances. The *Translation Function* is then used to synthesize a valid response. An approach which has to process the whole recorded interaction collection - which may become extremely large - becomes very inefficient in practice.

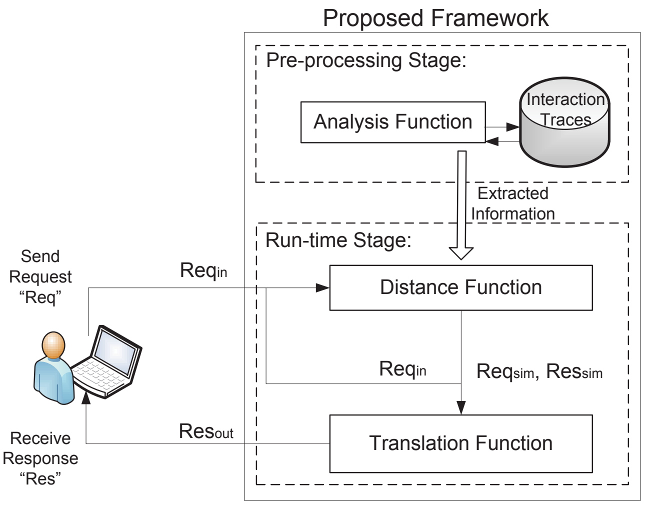


Figure 1: The Response Synthesis Framework

In order to address this problem, we describe an *analysis function* that is able to select and provide representative interaction data to the response synthesis approach, thereby accelerating response inference time at runtime. Our prior framework is split into 2 consecutive stages, a *pre-processing* stage and a *run-time* stage, as shown in Figure 1. At the pre-processing stage, we use the *Analysis Function* to summarise and derive characteristics of the interaction trace collection, such as probable message types. The *Distance Function* and the *Translation Function* work at the run-time stage.

In this paper, we introduce a clustering-based method for the trace analysis function in the pre-processing stage. Given a collection of interaction traces, our technique (i) calculates the distance between pairwise interactions and builds a distance matrix; (ii) clusters interactions; and (iii) exports the clusters and infers the cluster centres for use later in the process. In a proof of concept implementation, we use the *Needleman-Wunsch* algorithm as the distance measure and two clustering algorithms, BEA and VAT, to cluster the interactions. Furthermore, we use two methods to infer responses at the run-time stage, *Center_Only* and *Whole Cluster* (*cf.* Section 3.4). For evaluation purposes, we report the effectiveness and efficiency of our response synthesis approaches after pre-processing recorded interaction traces of two widely used application-layer protocols: LDAP [24] and SOAP [2].

The remainder of this paper is organised as follows: In Section 2, we review some related work in the areas of software emulation and data mining. The design of our approach for pre-processing interaction traces to assist the response synthesis task is described in Section 3. In Section 4, we present the results of our evaluation and discuss the relevant findings as well as identified limitations. Finally, we conclude this paper and identify some problems for future work in Section 5.

2. RELATED WORK

A major challenge when attempting to assure the quality of large enterprise systems is producing a suitable test-bed environment. A number of approaches have been proposed over years to address this need. Hardware virtualization tools, such as VMware [26] and VirtualBox [17], are capable of replicating specific facets of deployment environments, but they suffer similar scalability limitations as physical recreation of deployment environments [23]. Mock objects [8] mitigate some of these scalability concerns. However, they are often too language specific and require the re-implementation of some of an environment's functionality, resulting in testing environment configuration and maintenance problems and requiring detailed knowledge of environment components. Performance and load testing tools, such as the ones proposed in [10, 22], provide a means to emulate many thousands of software system clients with limited resources. However, they are designed to generate scalable client *load* towards a target system, rather than the opposite direction needed for our problem situation - a realistic, large-scale emulation environment for an enterprise system-under-test.

In order to overcome shortcomings with these existing enterprise system QA approaches and tools, the creation of "virtual", or emulated, deployment environments has been proposed. Ghosh and Mathur [9] state that "an emulation environment, which is able to provision representations of diverse components, enables tests of components that interact with a heterogeneous environment, while scalable models put scalability and performance testing for components into practice." In our prior work [11, 12, 13], we proposed an enterprise software environment emulator, called Kaluta. It provides a large-scale and heterogeneous emulation environment capable of simultaneously emulating thousands of endpoint systems on one or a few physical machines. We have shown this scales very well to the needs of enterprise system QA as outlined above. ITKO LISA [20] is a commercial software tool which aims to emulate the behaviour of services which a system under test interacts with in its deployment environment. In the work above, both creation of executable endpoint models and emulation rely on detailed knowledge of the transport or service protocol, which are not always available. They also must be updated whenever the target deployment environment changes.

For building system models from analysing interaction traces, Cui *et al.* [4] proposed an emulator aiming to mimic both client and server side behaviours. With the emulator, they can record/replay the interactions of web applications for checking conformance of web server behaviours. In our prior work [7], we presented a novel method of automatically building these client-server and server-server interaction models of complex software systems directly from interaction trace data, utilising longest common subsequence matching and field substitution algorithms. In above work, the accuracy of generated responses relies on processing and analysing the recorded interaction traces. Furthermore, if the size of interaction traces is large, the efficiency of this method will be jeopardised.

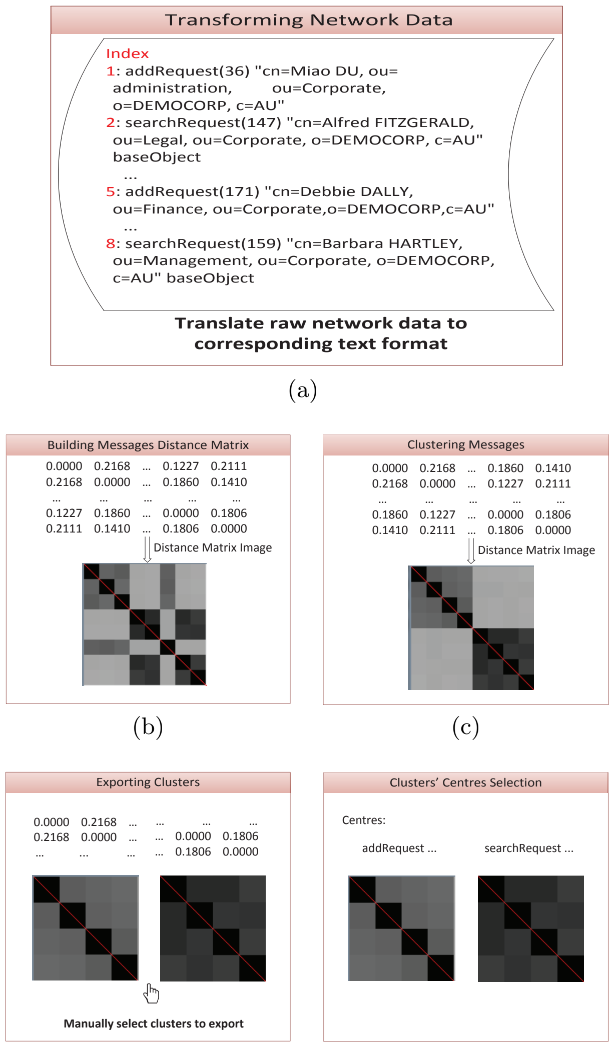
In recent years, data mining techniques have been widely utilised in software engineering area for discovering the structure of complex software systems [25], generating UML class diagram layout [14] , analysing software crash reports [5], labelling massive textual project profiles [28], predicting bug fixing time [30], and etc. In this paper, we present a new technique that utilises data mining to analyse very large recorded interaction traces. Specifically, we use two popular clustering algorithms to cluster the recorded interactions, which are the BEA (Bond Energy Algorithm) algorithm [18] and the VAT (Visual Assessment of cluster Tendency) algorithm [1]. The interaction clusters facilitate the effectiveness and efficiency of our response synthesis approach.

3. APPROACH

In an enterprise system emulation environment, requests - sent from the enterprise system under test - need to be responded to by the emulated operating environment according to the various interaction protocols between the deployment environment components. Such interaction protocols include LDAP, HTTP/HTTPS, SOAP, SMTP, SMB, etc. In an emulation environment, a request message sent from an enterprise system under test is responded to by a *generated* response message, rather than a real response message. This allows a complex, large-scale emulation environment to be provided to the system under test without the scalability and configuration limitations of other techniques. However, the technique is critically dependent on the ability of the emulation environment to generate realistic responses to the requests.

These interaction request messages and response messages contain two types of information: (i) protocol structure information (such as the operation name), used to describe software services behaviours; and (ii) payload information, used to express software systems' attributes. In general, given a collection of interactions conforming to a specific interaction protocol, the repeated occurrence of protocol structure information is very common, as only a limited number of operations are defined in the protocol specification. In contrast, payload information is usually quite diverse according to various interaction operation parameter values.

The core of our approach to analyse a very large number of interactions (thousands or even millions) is an algorithm that classifies


Transforming Network Data
Index
Translate raw network data to corresponding text format
(a)
Building Messages Distance Matrix
Distance Matrix Image
(b)
Clustering Messages
Distance Matrix Image
(c)
Exporting Clusters
Manually select clusters to export
(d)
Clusters' Centres Selection
Centres:
addRequest ...
searchRequest ...
(e)

```
1: addRequest(36) "cn=Miao DU, ou=
 administration,        ou=Corporate,
 o=DEMOCORP, c=AU"
2: searchRequest(147) "cn=Alfred FITZGERALD,
 ou=Legal, ou=Corporate, o=DEMOCORP, c=AU"
 baseObject
 ...
5: addRequest(171) "cn=Debbie DALLY,
 ou=Finance, ou=Corporate,o=DEMOCORP,c=AU"
  ...
8: searchRequest(159) "cn=Barbara HARTLEY,
 ou=Management, ou=Corporate, o=DEMOCORP,
 c=AU" baseObject
```

```
0.0000  0.2168  ...  0.1227  0.2111
0.2168  0.0000  ...  0.1860  0.1410
...     ...     ...  ...     ...
0.1227  0.1860  ...  0.0000  0.1806
0.2111  0.1410  ...  0.1806  0.0000
```

```
0.0000  0.2168  ...  0.1860  0.1410
0.2168  0.0000  ...  0.1227  0.2111
...     ...     ...  ...     ...
0.1860  0.1227  ...  0.0000  0.1806
0.1410  0.2111  ...  0.1806  0.0000
```

```
0.0000  0.2168  ...   ...     ...     ...
0.2168  0.0000  ...   ...  0.0000  0.1806
...     ...     ...   ...  0.1806  0.0000
```

Figure 2: Trace Analysis Process: (a) Transforming Network Data; (b) Building Messages Distance Matrix; (c) Clustering Messages; (d) Exporting Clusters; and, (e) Clusters' Centres Selection.

these component interactions into groups to assist in searching for the most similar recorded messages for a incoming request. At a high level, this algorithm can be viewed as a natural application of clustering and sequence alignment techniques for inferring protocol structure information. The motivation behind our approach is that if request/response pairs are less distant to some other requests/responses, then these less distant requests/responses are more likely to have the same structure information. Hence, computing the distance between each pair of messages should give us an indication of how to classify recorded interactions. These classified interactions then enable us to generate responses conforming better to the expected responses.

Our proposed solution requires that we first transform network traffic to a suitable format for further processing, shown in Figure 2a. Next, we build a distance matrix that contains the distance calculated by pairwise messages in the trace library. This process can be seen in Figure 2b. Figure 2c shows that the distance matrix provides a basis for the further clustering task. According to clustering results, we can then classify the whole interaction trace into a number of groups, shown in Figure 2d. Finally, Figure 2e shows that the *Centre Messages* are selected automatically to represent each groups.

3.1 Translation

We assume that for a given protocol under investigation, we are able to record a sufficiently large number of interactions between two (or more) deployed software endpoints (components). These recordings are assumed to be "valid", that is, that the sequence of recorded interactions are (i) correct with regards to the temporal properties of the underlying protocol and that (ii) each request and response message is well-formed. Without loss of generality, we further assume that each request is always followed by a single response. If a request does not generate a response, we insert a dedicated "no-response" message into the recorded interaction traces. If, on the other hand, a request leads to multiple responses, these are concatenated into a single response.

Given these assumptions, we write (Req, Res) to denote a single interaction, where Req represents the request, the corresponding response of Req is defined by Res. Both Req and Res are a sequence of characters describing the message structure and payload. An *interaction trace* is defined as a finite, non-empty sequence of interactions, which is denoted by $((Req_1, Res_1), (Req_2, Res_2) \ldots (Req_n, Res_n))$. Tools such as Wireshark [16] have the functionality to filter network traffic and record interactions of interest in a suitable textual format for further processing, which are in the form of $Req\#Res$, followed by a line break.

3.2 Building Request/Response Distance Matrix

Given a sufficient number of interactions in suitable formats, the next task is to calculate the distance between either requests or responses. Our chosen distance measure is based on sequence alignment [15], which was originally introduced in the area of bioinformatics. The basic idea of the sequence alignment is *align* all common subsequences of two sequences under comparison and insert *gaps* into either of the sequences when they differ.

In this work, we are using the Needleman-Wunsch algorithm [21] to find the optimal sequence alignment of the requests/responses. The following example briefly shows how our message alignment process works. Consider the following two text sequences:

- Where is my computer book?
- Where is your computer magazine?

The common subsequences are "Where is ", " computer ", and "?". (Note the spaces in around " computer ".) "my" versus "your" and "book" versus "magazine" are the two differing parts of the two sequences. The fully aligned sequences will be as follows (we use the character '$\star$' to denote an inserted gap):

- Where is my$\star\star\star$ computer book$\star\star\star\star\star\star\star\star$?
- Where is $\star$ your computer $\star\star\star\star$magazine?

Based on the alignment results, we can define our distance measure as

$$dist(\text{msg,msg'}) = \frac{N_{mismatch}}{N_{alg} + N'_{alg} - N_{mismatch}} \tag{1}$$

where N_{alg} and N'_{alg} denote the number of characters (including inserted gaps) in the sequence alignment for msg, msg', and $N_{mismatch}$ represents the total number of inserted gaps. Therefore, $16/(37 + 37 - 16) = 0.275$ is the distance for the example given above. Then, we are able to construct a $N \times N$ symmetrical distance matrix DM by iteratively computing the distance for all the candidate requests/requests, where N is the total number of requests/responses.

3.3 Grouping Similar Requests/Responses into Clusters

Once the distance matrix DM has been constructed, we are able to use it to group the requests/responses into clusters. Clustering algorithms are applied to the distance matrix, thereby producing clusters that consist of requests/responses with similar message characters sequences. Our approach focuses on the distance matrix reordering as the first step to achieve the clustering. The BEA [18] and VAT [1] clustering algorithms were selected for this purpose. These algorithms do not require us to define the number of clusters or a distance threshold value in advance, as is needed with some other clustering methods, such as K-Means.

The BEA (Bond Energy Algorithm) was proposed in 1972 [18], and is often adopted to cluster large data sets [29]. Given a distance matrix, it can group similar items along the matrix diagonal by permutating rows and columns in way of which is able to maximise the following global measure equation (2), denoted by GM,

$$GM = \sum_{i=1}^{n}\sum_{j=1}^{n}(1 - DM_{ij})(2 - DM_{i,j-1} - DM_{i,j+1}) \qquad (2)$$

where DM_{ij} denotes the distance between msg_i and msg_j. The detailed of this algorithm can be found in [27].

The VAT (Visual Analysis of Cluster Tendency) is a visual method, which works on a pairwise distance matrix D. This algorithm utilise a modified version of Prim's minimal spanning tree to reorder the rows and columns of the distance matrix (*cf.* [1] for a detailed description). The reordered matrix is then displayed as a gray-scale image.

Using either the BEA algorithm or VAT algorithm to reorder the distance matrix DM enables users to classify messages based on the matrix image rather than underlying protocol expertise.

3.4 Selecting Cluster Centres for Matching Function and Translation Function

The last step in our approach is to select a representative centre, denoted by $centre_i$, for every $cluster_i$, which is able to minimise $\sum_{k=1}^{n} dist(centre_i, msg_k)$, where n denotes the number of messages in $cluster_i$ and msg_k represents one of them. When a request is observed , the *Matching Function* is able to best decide which cluster the incoming request belongs to in the way of comparing this request with representative centres using the distance measure introduced in 3.2.

After the cluster centre is selected, we are able to search the most similar request for an incoming request with two methods: 1) *Centre_Only* and 2) *Whole Cluster*. Specifically, given an incoming request, the *Centre_Only* method directly uses the selected cluster centre to be its most similar request, while *Whole Cluster* method means that *Matching Function* needs to further searches for its most similar request in the selected cluster.

The *Translation Function* is to synthesize a response for the incoming request, exploiting the commonalities between the incoming request, its best match, as well as the associated recorded response(*cf.* [7] for a detailled description).

3.5 Implementation

We have developed a proof-of-concept realisation of this trace analysis approach, including the Needleman-Wunsch algorithm and using two clustering algorithms (BEA and VAT). We have integrated this with our prior interaction analysis and response generation prototype [7]. This prototype interaction analysis and response generation environment was implemented in the Java programming language. We used the Wireshark tool to capture network traffic and exported this into a format suitable for input into our Java implementation.

4. EVALUATION

In this section, we evaluate both the effectiveness and the efficiency of our response generation approach when it uses our new trace analysis function to preprocess a large number of interactions. We then discuss the key results from our experiments.

4.1 Experimental Setup

For evaluation purposes we used two commonly used application-layer protocols as case studies, the Simple Object Access Protocol (SOAP) [2] and the Lightweight Directory Access Protocol (LDAP) [24]. SOAP is a light-weight protocol designed for exchanging structured information in a decentralised, distributed environments whereas LDAP is widely used in large enterprises for maintaining and managing directory information. The experiment was running on a HP Z800 Workstation using Intel Xeon X5660 2.80GHz CPU, with 48GB of RAM.

The interaction trace for SOAP used for our evaluation was generated based on a recording of a banking example using the LISA tool [20]. This SOAP trace contains 6 different request types, each with a varying number of parameters, encoding "typical" transactions one would expect from a banking service. From a pre-defined set of account id's, account names etc, we then randomly generated an interaction trace containing $1,000$ request/response pairs. Among those we had 548 unique requests [1](with 25 different requests occurring multiple times to compose the rest) , 714 unique responses (the replicated ones are predominantly due to the fact that the `deleteTokenResponse` message only had true or false as possible return values), and 22 duplicated request/response pairs. For the purpose of our evaluation, we considered this a sufficiently diverse "population" of messages to work with.

The following shows one of the recorded requests:

```
<?xml version="1.0"?>
 <S:Envelope
  xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
   <ns2:getAccount xmlns:ns2="http://bank/">
    <accountId>867-957-31</accountId></ns2:getAccount>
  </S:Body>
 </S:Envelope>
```

with the following the corresponding response:

```
<?xml version="1.0"?>
 <S:Envelope
  xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
   <ns2:getAccountResponse xmlns:ns2="http://bank/">
    <return>
     <accountId>867-957-31</accountId>
     <fname>Steve</fname>
     <lname>Hine</lname>
```

[1]Unique request/response refers to either requests or responses occur once in the trace library

```
      </return>
    </ns2:getAccountResponse>
  </S:Body>
</S:Envelope>
```

This example illustrates that besides the structural SOAP information encoded in both messages, there is specific information that appears in both the SOAP request and the SOAP response, such as the account-ID in the example above.

LDAP is a binary protocol that uses an ASN.1 encoding to encode and decode text-based message information to and from its binary representation, respectively. For the purpose of our study, we used a corresponding decoder in order to translate recorded LDAP messages into a text format and an encoder to check whether the synthesized responses were well-formed (*cf.* Section 4.2).

The LDAP interaction trace used for our evaluation consisted of 1000 unique interactions containing the core LDAP operations, such as *adding*, *searching*, *modifying* etc. applied to CA's *DemoCorp* sample directory [3]. The trace did not contain any duplicated requests or responses, and the search responses contained a varying number of matching entries, ranging from zero to 12.

The following briefly illustrates the textual representation of a *search* request:

```
Message ID: 15
ProtocolOp: searchRequest
  ObjectName: cn=Juliet LEVY,ou=Administration,
              ou=Corporate,o=DEMOCORP,c=AU
  Scope: 0 ( baseObject )
```

and the corresponding response, consisting of the merge of a *search result entry* and a *search result done* message:

```
Message ID: 15
ProtocolOp: searchResEntry
  ObjectName: cn=Juliet LEVY,ou=Administration,
              ou=Corporate,o=DEMOCORP,c=AU
  Scope: 0 ( baseObject )
Message ID: 15
ProtocolOp: searchResDone
  resultCode: success
```

This example LDAP request contains a (unique) message identifier (`Message ID: 15`) and a specific object name (`ObjectName: ...`) as the root node for the search to be used. The corresponding responses use the same message identifier (to indicate the request they are in response to) and the `searchResEntry` message refers to the same object name as the request. For our approach to synthesize correct LDAP responses, the corresponding information needs to be copied across from the incoming request to the most similar response to be modified.

4.2 Cross-Validation Approach and Evaluation C-riteria

Cross-validation [6] is one of the most popular methods for assessing how the results of a statistical analysis will generalise to an independent data set. For the purpose of our evaluation, we applied the commonly used 10-fold cross-validation approach [19] to both the recorded SOAP and LDAP messages.

As shown in Figure 3, we randomly partitioned the original interaction data set into 10 groups. Of these 10 groups, group *i* (*cf.* top-left rectangle in Figure 3) is considered to be the *evaluation group* for testing our approach, and the remaining 9 groups constitute the *training set*. The process is then repeated 10 times (the same as the number of groups), so that each of the 10 groups will be used as the evaluation group once.

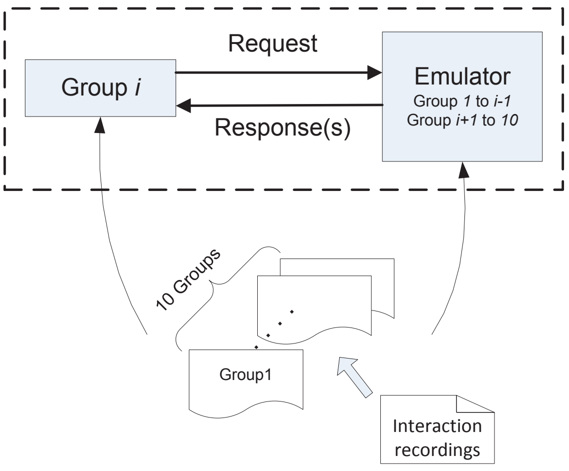


Figure 3: 10-fold Cross Validation Approach

We investigated both the effectiveness and efficiency of our proposed approach for response synthesis. For effectiveness investigation we defined the following criteria to evaluate the synthesized responses:

1. **Identical**: the synthesized response is identical to the recorded response if all characters in the synthesized response are the same as those in the recorded response

2. **Protocol Conformant**[2]: this criterion indicates that the responses conform to the temporal interaction properties of the given protocol, that is, the temporal consistency between request and response is preserved.

3. **Well-Formed**: this criterion requires that the synthesized responses correspond to the structure required for responses as defined by the underlying protocol.

4. **Ill-Formed**: Synthesized responses do not meet above criteria.

For the efficiency investigation, for each message in the evaluation group, we record how much time was taken to synthesize a response for an incoming request.

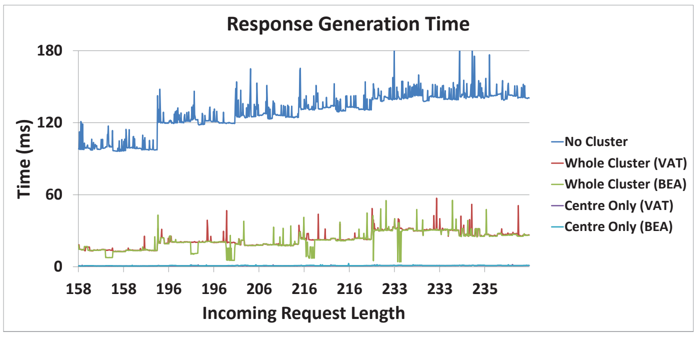
4.3 Evaluation Results

For the purpose of our evaluation, we used our prior "whole-interaction-trace-based" approach [7], referred to as the *No Cluster* method, to benchmark both the effectiveness and efficiency of the proposed cluster-based approach for synthesizing responses. In our experiment, in the pre-processing stage, we apply the BEA clustering algorithm and VAT clustering algorithm to cluster interactions; and then, at the runtime stage, we further use the *Centre Only* method and *Whole Cluster* method (*cf.* Section 3.4) to synthesize responses.

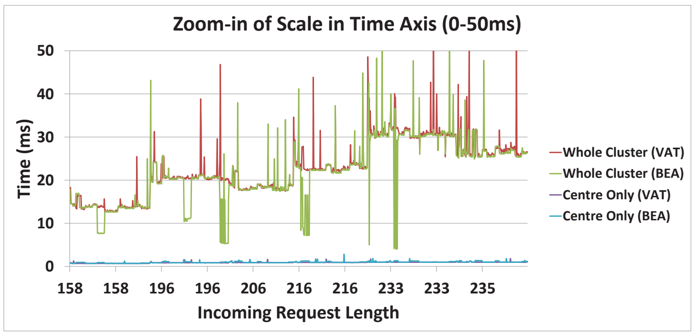
[2]We use a weaker notion of *protocol comformant* as the order in which the requests are selected from the evaluation set is *random* and, as a consequence, unlikely to conform to the temporal sequence of request-response pairs.

Experiment	Cluster Method	No.	Valid	Ident.	Conf. [1]	Well-form. [2]	Ill-form.
No Cluster	None	1,000	1,000	85	915	0	0
Whole Cluster	VAT	1,000	1,000	90	910	0	0
	BEA	1,000	1,000	82	918	0	0
Centre Only	VAT	1,000	1,000	87	913	0	0
	BEA	1,000	1,000	92	908	0	0

Table 1: Validity of Synthesized SOAP Responses



(a)



(b)

Figure 4: SOAP Responses Generation Time

In Table 1 and Table 2, we summarise the number of generated SOAP responses and LDAP responses, which were categorised according to the criteria introduced in Section 4.2. In addition, we also list the number of valid responses, that is, the sum of responses falling in the *Identical* and *Protocol-conformant* categories, for both protocol examples.

Figure 4a and Figure 5a outline the amount of time that was taken to generate responses for incoming requests of different lengths, *i.e.* the number of characters per incoming request, by using the *No Cluster* method, the *Centre Only* method and the *Whole Cluster* method. For both the *Centre Only* method and the *Whole Cluster*, we further compare the response generation times of the VAT generated clusters versus the BEA generated clusters, as shown in Figure 4b and Figure 5b.

4.3.1 Evaluation results for SOAP

Table 1 and Figure 4 compare the different outcomes of the *No Cluster* method and cluster-based approaches. From Table 1 we can see that neither the *Centre Only* method nor the *Whole Cluster* method generate ill-formed SOAP responses, which produce the same outcomes as using the *No Cluster* method. This shows that for the SOAP case study used, our cluster-based approach has the same effectiveness as the *No Cluster* method in synthesizing accurate responses.

From the Figure 4, we can see that whatever approach we use to generate a response (*i.e.* either *No Cluster* method or cluster-based methods), more time is consumed to synthesize responses for longer incoming requests. However, a significant improvement of the generation time is observed by using cluster–based approaches. Specifically, response generation time of using *Whole Cluster* methods is at least 5 times quicker than using the *No Cluster* method; using *Centre Only* is able to further accelerate response message synthesis by approximately 120 times than that of using the *No Cluster* method. By further investigating cluster-based approaches in Figure 4b, we can see that although there is more fluctuation in the response generation time when using the BEA algorithm to cluster messages, the selection of clustering approaches does not impact the response generation time too much. Based on these observations, we conclude that for the SOAP protocol, the *Centre Only* method is able to provide a good response message synthesis.

4.3.2 Evaluation results for LDAP

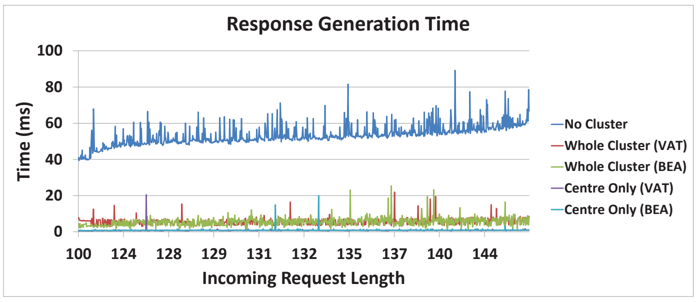
Table 2 gives a summary of the result of our LDAP experiments. For the *No Cluster* method, 451 (out of 1,000) generated response messages are identical to the the corresponding recorded responses (45.1%), and an additional 457 of the generated responses met the protocol conformant criterion (45.7%). Therefore, a total of 908 (or 90.8%) of all generated responses were considered to be valid. In contrast, Cluster-based approaches generate less valid responses, the number of which decrease by 14.7% to around 761 (out of 1000). By observing valid responses of cluster-based approaches, we identify that the VAT algorithm performs better than the BEA algorithm. However, there is no distinguishable difference between results of the *Centre Only* method and the *Whole Cluster* method.

Figure 5 also shows an increasing trend of response generation time when the length of incoming requests becomes longer. As the length of the majority of incoming requests is shorter than the SOAP incoming requests, the LDAP response generation time is shorter than SOAP response generation time, shown in Figure 4. Specifically, compared with the average response generation time of using *No Cluster* method (about 53.28ms), using *Whole Cluster* method is able to produce responses about 9 times faster (about 5.46ms), while using *Centre Only* method can further improve the generation time to around 0.79ms. As we can see from the Figure 5b, the response generation time of using *Whole cluster* method fluctuates significantly. This is because the sizes of the clusters generated by the clustering algorithms are different. Therefore, the time taken when using them to generate responses are accordingly different. In contrast, as the number of clusters is stable,

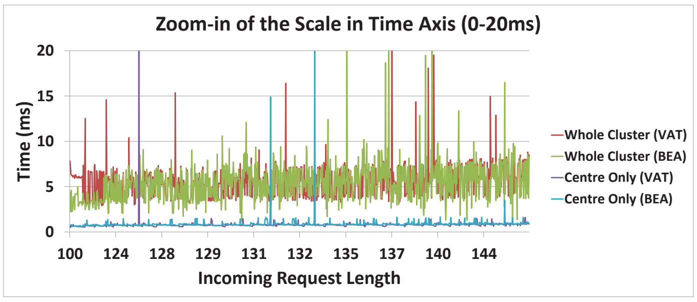
[3]We exclude the number of identical messages in the column *Conf.*
[4]We exclude the number of valid responses in the *Well-form.* column.

Experiment	Cluster Method	No.	Valid	Ident.	Conf. [3]	Well-form. [4]	Ill-form.
No Cluster	None	1,000	908	451	457	89	3
Whole Cluster	VAT	1,000	751	360	391	246	3
	BEA	1,000	753	383	370	241	6
Centre Only	VAT	1,000	751	296	455	235	14
	BEA	1,000	761	330	431	232	7

Table 2: Validity of Synthesized LDAP Responses



(a)



(b)

Figure 5: LDAP Responses Generation Time

the response generation time of using the *Centre Only* method is observed to have only a slight fluctuation.

4.4 Discussion

Based on summaries of both the SOAP and LDAP experimental results in Table 3, we can see that the cluster-based approach is able to generate valid responses much more efficiently than searching the entire trace library. However, as illustrated in the results for LDAP, the cluster-based approaches generate fewer valid responses. This is attributed to differences between the SOAP and LDAP protocols. Most application-level protocols define message structures containing some form of *operation* or service name in their request, followed by a *payload*, containing the data the service is expected to operate upon [11]. In LDAP, some messages contain much more *payload* information than *operation* information. Two LDAP messages of different operation types, but with a similar payload, may be found to be the closest matching messages. In such a case, a response of the wrong operation type may be sent back, which would not be a valid protocol conforment response.

In Table 3, we can see that the time cost of generating responses depends on the number of distance calculations. In our prototype, we use the Needleman-Wunsch algorithm as the distance function, which has a relatively high time complexity of *O(MN)*, where *M* and *N* denote the length of pairwise messages.

Experiment	Cluster Method(VAT)	Response Time(ms)	Dist. Calcs.	Accuracy
SOAP	No Cluster	128.6	899	100%
	Whole	22.92	156	100%
	Centre	0.90	6	100%
LDAP	No Cluster	53.28	899	90.8%
	Whole	5.46	96	75.1%
	Centre	0.79	11	75.1%

Table 3: Average Response Times, Number of Distance Calculations and Accuracy

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have utilised data mining techniques in a large enterprise software emulation environment to efficiently generate system responses. Specifically, we utilized two popular clustering algorithms, the BEA algorithm and the VAT algorithm, to pre-process recorded interaction traces. Using these clustered results we then facilitated the mimicking of software interaction behaviours in a runtime fashion. As our proposed approach does not require explicit knowledge of the protocols that the target software components uses to communicate, it eliminates the human effort of manually specifying interaction models. Moreover, by utilizing data mining techniques, the efficiency of response generation in the emulation environment has been greatly improved. Experimental results conducted on LDAP and SOAP protocols demonstrate that the response generation time has been reduced by 99% on average compared to our prior approach, while the accuracy of response generation (the valid response rate) was 100% for SOAP and 75% for LDAP.

Future work will omit the format transformation to build diverse interaction models. We will devise better cluster centre selection methods to automatically summarize the most common characters of all messages within a cluster, which will be further used for synthesizing representative cluster centres. Possible approaches include multiple sequence alignment algorithms in the area of bio-informatics [15], or using hierarchical clustering to group responses/requests into a tree which can help to infer the most common characters among requests/responses. In our proof of concept implementation, we use the Needleman-Wunsch algorithm as the edit distance calculation, which has relatively high time complexity. Future work will also include improving the efficiency of distance calculations by using parallel processing and using an approximation of the Needleman-Wunsch edit distance. Finally, there is a need to test our methods on much bigger trace collections and on a wider range of protocols. For example, proprietary protocols on legacy mainframe systems, which are often poorly documented, are a particularly interesting category to test.

Acknowledgements

This work is supported by ARC Linkage Project LP100100622 Large-Scale Emulation for Enterprise Software Systems.

6. REFERENCES

[1] J. C. Bezdek and R. J. Hathaway. Vat: A tool for visual assessment of (cluster) tendency. In *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN 2002)*, volume 3 of *IJCNN 2002*, pages 2225–2230. IEEE, 2002.

[2] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple Object Access Protocol (SOAP) 1.1,. W3C Note 8, W3C, May 2000. `http://www.w3.org/TR/2000/NOTE-SOAP-20000508/`.

[3] CA Technologies. *CA Directory Administration Guide (r12.0 SP11)*, 2012.

[4] W. Cui, V. Paxson, N. C. Weaver, and R. H. Katz. Protocol-independent adaptive replay of application dialog. In *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS 2006)*, Feb. 2006.

[5] Y. Dang, R. Wu, H. Zhang, D. Zhang, and P. Nobel. Rebucket: A method for clustering duplicate crash reports based on call stack similarity. In *Proceedings of the 34th International Conference on Software Engineering (ICSE 2012)*, pages 1084–1093. IEEE, 2012.

[6] P. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice/Hall International, 1982.

[7] M. Du, J.-G. Schneider, C. Hine, J. Grundy, and S. Versteeg. Generating service models by trace subsequence substitution. In *Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures (QoSA 2013)*, pages 123–132, Vancouver, British Columbia, Canada, 2013. ACM.

[8] S. Freeman, T. Mackinnon, N. Pryce, and J. Walnes. Mock roles, objects. In *Companion to the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA 2004)*, pages 236–246, New York, NY, USA, 2004.

[9] S. Ghosh and A. P. Mathur. Issues in Testing Distributed Component-Based Systems. In *Proceedings of the 1st International ICSE Workshop on Testing Distributed Component-Based Systems*, 1999.

[10] J. Grundy, Y. Cai, and A. Liu. Softarch/mte: Generating distributed system test-beds from high-level software architecture descriptions. *Automated Software Engineering*, 12(1):5–39, Jan. 2005.

[11] C. Hine. *Emulating Enterprise Software Environments*. Phd thesis, Swinburne University of Technology, Faculty of Information and Communication Technologies, 2012.

[12] C. Hine, J.-G. Schneider, J. Han, and S. Versteeg. Scalable emulation of enterprise systems. In *Proceedings of the 20th Australian Software Engineering Conference (ASWEC 2009)*, pages 142–151, Gold Coast, Australia, Apr. 2009. IEEE Computer Society Press.

[13] C. Hine, J.-G. Schneider, and S. Versteeg. Reac2o: A runtime for enterprise system models. In J. Andrews and E. Di Nitto, editors, *Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2010)*, pages 177–178, Antwerp, Belgium, Sept. 2010. ACM.

[14] H. Hu, J. Fang, Z. Lu, F. Zhao, and Z. Qin. Rank-directed layout of uml class diagrams. In *Proceedings of the First International Workshop on Software Mining*, pages 25–31. ACM, 2012.

[15] N. C. Jones and P. Pevzner. *An Introduction to Bioinformatics Algorithms*. MIT press, 2004.

[16] U. Lamping, R. Sharpe, and E. Warnicke. *Wireshark Users's Guide*, 2012.

[17] P. Li. Selecting and Using Virtualization Solutions: our Experiences with VMware and VirtualBox. *Journal of Computing Sciences in Colleges*, 25(3):11–17, Jan. 2010.

[18] W. T. McCormick, P. J. Schweitzer, and T. W. White. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20(5):993–1009, 1972.

[19] G. J. McLachlan, K.-A. Do, and C. Ambroise. *Analyzing Microarray Gene Expression Data*. Wiley-Interscience, 2004.

[20] J. Michelsen. Key Capabilities of a Service Virtualization Solution, October 2011. ITKO White Paper. Available at: `http://www.itko.com/resources/service_virtualization_capabilities.jsp`.

[21] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[22] S. Sampath, S. Sprenkle, E. Gibson, L. Pollock, and A. Souter Greenwald. Applying Concept Analysis to User-Session-Based Testing of Web Applications. *IEEE Transactions on Software Engineering*, 33(10):643–658, 2007.

[23] J. Sanchez. Squeezing virtual machines out [of] CPU cores, June 2011. VM Install.

[24] J. Sermersheim. Lightweight Directory Access Protocol (LDAP): The Protocol. RFC 4511 (Proposed Standard), June 2006.

[25] L. Šubelj and M. Bajec. Software systems through complex networks science: Review, analysis and applications. In *Proceedings of the First International Workshop on Software Mining*, pages 9–16. ACM, 2012.

[26] J. Sugerman, G. Venkitachalam, and B.-H. Lim. Virtualizing i/o devices on vmware workstation's hosted virtual machine monitor. In *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pages 1–14, Berkeley, CA, USA, 2001. USENIX Association.

[27] M. Tamer Őzsu and P. Valduriez. *Principles of Distributed Database Systems*. Springer, 2011.

[28] T. Wang, G. Yin, X. Li, and H. Wang. Labeled topic detection of open source software from mining mass textual project profiles. In *Proceedings of the First International Workshop on Software Mining*, pages 17–24. ACM, 2012.

[29] D. Yuan, Y. Yang, X. Liu, and J. Chen. A data placement strategy in scientific cloud workflows. *Future Generation Computer Systems*, 26(8):1200–1214, 2010.

[30] H. Zhang, L. Gong, and S. Versteeg. Predicting bug-fixing time: An empirical study of commercial software projects. In *Proceedings of the 35th International Conference on Software Engineering*, pages 1042–1051. IEEE, 2013.