Automatic Generation of Interaction Models for Enterprise Software Environment Emulation

Miao Du
Faculty of Information and Communication Technologies
Swinburne University of Technology
Hawthorn, VIC 3122, Australia
miaodu@swin.edu.au

***Abstract*—In modern enterprise software environments, software systems often need to cooperate with other systems to fulfil their responsibilities. A significant challenge is how to assure the quality of such software systems before deploying them in realistic software environment. Enterprise software environment emulation is an emerging technique for quality assurance. By replacing software systems with their corresponding interaction models, quality assurance engineers can conduct tests in an emulation environment with production-like conditions. In order to create interaction models, previous approaches needed to manually specify interaction behaviours using protocol knowledge, which are very time-consuming and error-prone. In this paper, we propose a framework to automate the generation of interaction models via exploiting interaction traces, which can eliminate the manual effort for interaction modelling.**

***Keywords*-automatic modelling; environment emulation; interaction traces analysis;**

I. Introduction

Modern enterprises require a variety of software systems to assist with complex day-to-day business processes. Many of these software systems have to cooperate with other systems in the operating environment to perform their functionalities. A significant and non-trivial engineering challenge is how to assure the quality of such software systems before deploying them into realistic production environments.

Enterprise software environment *emulation* [8] [9] has been proposed as a solution to provision an approximation of the operating environments and can be used by quality assurance team to assess the run-time characteristics of the enterprise software systems. It consists of two key elements: (i) *interaction models* which describe the interactive behaviour of individual enterprise software systems and (ii) an *emulator* which provides a means to execute or interpret these models, enabling communication with a real enterprise software system.

Being able to create interaction models is pivotal to the enterprise software environment emulation. The most common approach is to manually specify the interaction behaviours between enterprise software systems in the production environment. It is very time-consuming, error-prone and even unfeasible, as it require: 1) to manually define sequences of request/response patterns between communication partners including parameter values, 2) to manually identify temporal characteristics of interaction behaviours. Neither of these information are necessarily available at the required level of detail (if at all). Therefore, researchers started to investigate automatic approaches to eliminate the manual effort of specifying interaction behaviours.

For this purpose, the aim of our research is to develop a framework that allows automatic generation of interaction models via exploiting *interaction traces*. When an enterprise software system interacts with a system in its deployment environment, observable interaction behaviours are preserved by a network sniffer tool, which are referred to *interaction traces*. Due to the fact that any valid interaction must conform to a specific protocol specification, the interaction traces contain precise information in terms of both sequences of request/response pattern including parameter values and potential temporal properties. Our framework thus targets to infer enterprise system element interaction behaviours through direct operation on *interaction traces*. We have identified three requirements that we have to address in our work, which include:

- processing interaction traces in order to extract sufficient amount of inherent protocol information;
- creating interaction models based on extracted information;
- using created interaction models to communicate with the system in the production environment, thereby replacing the realistic systems for quality assurance purpose;

The rest of this paper is organised as follows: Section II discusses the motivation and some related work. Then, we describe a generic framework as well as illustrate preliminary evaluation results in Section III. Section IV outlines some identified ongoing research issues and promising approaches. Finally, we conclude this paper in Section V.

II. Related work and motivation

Over the years, a number of approaches have been proposed that aim to reverse engineer protocol specification from previous interaction recordings. Early effort in reverse engineering was for protocol determination. By analysing a large amount of packets and traces captured on networks, researchers are able to obtain structure information of the target protocol for network analysis [1] [10] and even automatically reverse engineering the state-machine model of network protocols [3] [4]. Cui *et al.*. [5] proposed an emulator aiming to mimic both the client and server side behaviours. With the emulator, they can record/replay the interactions of web applications for checking conformance of web server behaviours. Although the proposed approach deals with the emulation of interaction process, they essentially intend to test conformance of the client-side systems rather than work in the opposite way as we do.

As our approach is purely based on analysis of interaction traces without any knowledge of protocols, we need effective methods and tools for processing the recorded traces. In [6] [14], authors presented a series of trace exploration tools and techniques. However, these approaches did not consider the case of processing large amount of interaction traces, hence not efficient to be utilised in our automated model to synthesize responses at the runtime. Furthermore, these techniques require some known patterns to analyse the interaction traces, which in turns requires the assistance of human knowledge. Hence, more powerful tools are needed for exploring for our purpose.

ITKO LISA [11] is a commercial software tool which aims to emulate the behaviour of services which a system under test interacts with in its deployment environment. One of the key features of LISA is that, after recording a set of real interactions between an enterprise system and an endpoint, it uses these to produce responses to further requests, thus behaving like a 'virtual' service. However, LISA still requires the transport protocol and the service protocol to be known in advance of the recording for the modelling to be effective.

III. approach

To fulfill identified requirements in Section I, we split our targeting framework into consecutive two stages: *preprocessing* and *run-time*. At preprocessing stage, we intend to distinguish protocol information (*i.e.* message structural information defined by a particular protocol specification) from payload information (*i.e.* variables that are produced/consumed by application programs) by analysing interaction traces. At run-time stage, given a request from the system under test, we (i) search for the most "similar" recorded request, and then (ii) we synthesize a request and send it back to the system under test. Figure 1 shows an overview of our proposed framework.

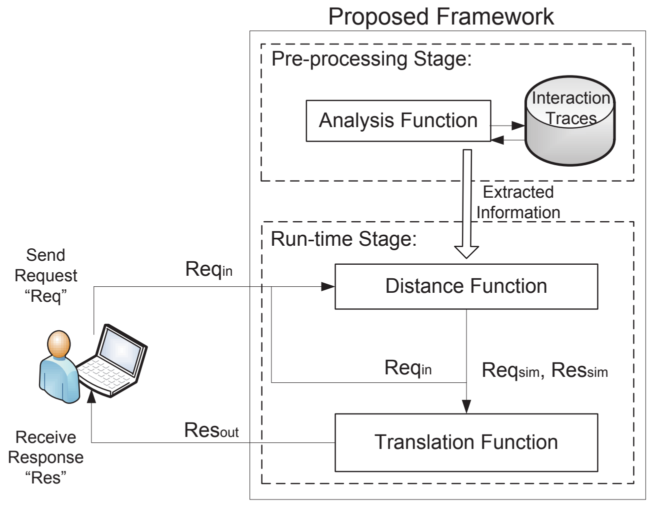

Figure 1. An Overview of the Proposed Framework

Our initial work is able to synthesize valid responses via directly manipulating original interaction traces, without performing the analyzing function. We define the interaction traces $\mathcal{I}$ as a non-empty set of request/response pair, denoted as (Req, Res). Given an incoming request, denoted Req_{in}, we can use distance function $dist$ to find its most similar request in the interaction recording, denoted as (Res_{sim}, Res_{sim}). With both Req_{in} and (Res_{sim}, Res_{sim}), we can then use *translation* function to substitute some parts of Res_{sim} in oder to synthesize a Res_{out} and send it back.

The following example will help illustrating the identification of the most similar interaction and how fields in the Res_{sim} are substituted in the response generation process. We start with an incoming LDAP search request[1]

```
Message ID: 18
ProtocolOp: searchRequest
  ObjectName: cn=Mal BAIL,ou=Administration,
              ou=Corporate,o=DEMOCORP,c=AU
  Scope: 0 ( baseObject )
```

we are looking to generate a response for. Using the distance function to search for the most similar request in the available interaction traces returns the following request

```
Message ID: 37
ProtocolOp: searchRequest
  ObjectName: cn=Miao DU,ou=Administration,
              ou=Corporate,o=DEMOCORP,c=AU
  Scope: 0 ( baseObject )
```

that is paired-up with the following response:

```
Message ID: 37
ProtocolOp: searchResEntry
  ObjectName: cn=Miao DU,ou=Administration,
              ou=Corporate,o=DEMOCORP,c=AU
  Scope: 0 ( baseObject )
```

[1]For presentation purposes, we use a more user-friendly layout of the corresponding LDAP messages. During processing, newlines, leading white spaces etc. were removed from the textual representation.

```
  Message ID: 37
  ProtocolOp: searchResDone
    resultCode: success
```

The translation function is able to identify two substrings that are identical across request and response:

```
  Message ID: 37
  ProtocolOp:
```

and

```
  ObjectName: cn=Miao DU,ou=Administration,
              ou=Corporate,o=DEMOCORP,c=AU
  Scope: 0 ( baseObject )
```

Substituting the corresponding values from the incoming request, we synthesize the following response:

```
  Message ID: 18
  ProtocolOp: searchResEntry
    ObjectName: cn=Mal BAIL,ou=Administration,
                ou=Corporate,o=DEMOCORP,c=AU
    Scope: 0 ( baseObject )
  Message ID: 18
  ProtocolOp: searchResDone
    resultCode: success
```

We have submitted a paper to report current implementation and preliminary evaluation results. In [7], we choose the longest common substring matching algorithm and field substitution algorithm to implement the distance function and the translation function, respectively. We also report evaluation results with the use of two commonly used application-layer protocols: the Simple Object Access Protocol(SOAP) [2] and the Lightweight Directory Access Protocol(LDAP) [13].

The evaluation results in [7] have demonstrated that our framework is able to automatically synthesize responses from interaction traces for mimicking interactive behaviours. However, we also have noticed that a small proportion of synthesized responses are not conformant to the temporal definitions of protocols or even do not have proper structure. Investigation of synthesized results indicate the occurrence of unexpected responses are attributed to the incorrect identification of the suitably "similar" recorded requests. This fact implies that without any knowledge of message structures and required payloads, identifying the most similar recorded request purely based on manipulating the interaction recordings cannot always work well. In future work, we intend to devise suitable methods allowing to distinguish message structure from payloads through the analysis of interaction recordings. These points will be explained in Section IV in detail.

IV. Research Issues and promising approaches

Based on what we have stated before, while preprocessing interaction traces is not necessary for inferring responses for the system under test, properly performing this activity will improve both the accuracy and efficiency of the our proposed approach. In this section, we will first discuss about some unsolved issues involved in our work, and then we will introduce promising solutions.

- **Interaction Traces Analysis**
  To establish a communication between two system elements, both of them must adhere to a particular protocol specification, which implies that observable interaction traces contain protocol knowledge. And furthermore, besides structural information, transmitted messages often deliver user data (also known as payloads) that are consumed/produced by an application using this protocol to exchange messages with other application on other host. With use of little or none of protocol knowledge, how to distinguish between protocol-related information (*i.e.* message format) from application-related information (*i.e.* payload) is the core of interaction traces analysis.
- **Distance Function Selection**
  The distance function is responsible for choosing the most suitably request for an incoming request. To this end, being able to measure the similarity is crucial to a distance function. In our work, we use a popular notion of similarity, that is, the edit distance [12] between two sequences s1 and s2, which indicate the minimum number of modifications (insertions, deletions, and substitutions) in order to obtain s2 from s1. The ideal situation is that the selection of distance measure(s) is able to best express our intention of similarity. Therefore, depending on what kind of distance function is used, a different pre-recorded request will be chosen to be the most "similar" to the incoming request.
- **Translation Function Selection**
  The translation function is used to synthesize a valid response to a incoming request. The validity of a response depends on two aspects, which are 1) both the message structure and the sequence of transmitted messages must adhere to a particular protocol specification that is used by an application one one host to exchange data with its communicating partner on other host(s) over the network and 2) synthesized payloads must can be recognised, extracted and further processed by its communicating partner.

I outline some potential approaches to deal with identified issues, which are as follows.

1) Processing interaction traces is to investigate widely-used application-layer protocols. Doing this will provide an insight into both messages structures and encoding rules of available protocols, thereby obtaining a set of heuristic rules for inference purpose. Specifically, if interaction traces inherently conform to a protocol whose message structures and encoding rules have been well defined, an approach is needed to associate interaction traces with this particular protocol automatically. If, on the other hand, interaction traces do not conform to any known protocols, this approach should also be able to automatically select

the most relevant rules and further compose a new heuristic rule set.

2) Distance function is crucial for matching the incoming request to a record interaction. In essence, the distance function is used to compute the distance between two requests, which refers to the minimum number of modifications in order to alter the incoming request to the recorded one. Depending on what kind of distance function is used, a different pre-recorded request will be chosen to be the most "similar" to the incoming request. I intend to develop a set of distance functions, where for different protocol, the suitable distance function can be selected automatically.
3) After processing interaction traces and operating the distance function, both protocol and application-related information have been prepared. The translation function is to structure messages in the expect format and fill in payload contents. For this piece of work, I also attempt to develop a function set, where a potential approach is able to automate the selection.

We are going to evaluate the proposed approach against LDAP and SOAP protocols as they were employed in our initial work [7] which can be used as a benchmark to assess feasibility and effectiveness of our proposed approach for synthesising responses.

V. Conclusion

In our prior work [7], we have demonstrated that it is feasible to infer expected interaction behaviours directly from interaction traces, without requiring explicit knowledge of the protocols which the software systems use to communicate. In our initial framework, given a request, we first utilise algorithms to search for the most "similar" recorded request in the database of interaction traces, and then manipulate its attaching recorded response to produce a response. Although our evaluation results show that our initial framework is able to infer expected interaction behaviours in most situations, its working efficiency and memory consumption are still concerns, which in turn implies that we have to incorporate a component to pre-process interaction traces.

The major contributions of our framework are the elimination of the manual effort and the reduction of the reliance on a system expert when creating interaction models. Furthermore, our framework is able to create a substantial number of interaction models adhering to different types of protocols to a new level which current approaches cannot achieve.

Acknowledgment

This work is part of the ARC Linkage Project LP100100622, supported by the Australian Research Council and CA Technologies.

References

[1] M. Beddoe. The protocol informatics project. *Toorcon*, 4:4, 2004.

[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] P. M. Comparetti, G. Wondracek, C. Kruegel, and E. Kirda. Prospex: Protocol Specification Extraction. In *Proceedings of the 30th IEEE Symposium on Security and Privacy (SP 2009)*, pages 110–125. IEEE, 2009.

[4] W. Cui, J. Kannan, and H. J. Wang. Discoverer: Automatic protocol reverse engineering from network traces. In *Proceedings of the 16th USENIX Security Symposium (Security 2007)*, number 14, pages 1–14, Boston, MA, Aug. 2007.

[5] 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*, 2006.

[6] W. De Pauw, E. Jensen, N. Mitchell, G. Sevitsky, J. Vlissides, and J. Yang. Visualizing the execution of java programs. *Software Visualization*, pages 647–650, 2002.

[7] M. Du, J.-G. Schneider, C. Hine, J. Grundy, and S. Versteeg. Generating service models by trace subsequence substitution. Accepted, 2013.

[8] 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.

[9] 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.

[10] C. Leita, K. Mermoud, and M. Dacier. Scriptgen: an automated script generation tool for honeyd. In *Proceedings of 21st Annual Computer Security Applications Conference (ACSAC 2005)*, pages 203–214. IEEE, 2005.

[11] 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.

[12] E. S. Ristad and P. N. Yianilos. Learning String-Edit Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532, May 1998.

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

[14] T. Systä, K. Koskimies, and H. Müller. Shimbałan environment for reverse engineering java software systems. *Software: Practice and Experience*, 31(4):371–394, 2001.