Abstract—Queuing network modelling is a modelling technique for capacity planning studies of computer and communication systems. Due to complexity of the technique, it is very difficult to know from the computed outputs whether the computation of the modelling software is correct. It is necessary to have an effective testing technique to address this problem. Recently, it has been noticed that metamorphic testing is an effective technique for this kind of problem. In this paper, we study the technique of metamorphic testing in testing queuing network modelling through a set of testing experiments in the Java Modelling Tool – a popular open source queuing network modelling suite.

Keywords—Queuing Network Modelling, software testing, Metamorphic Testing.

I. INTRODUCTION

Queuing theory was developed to model the behaviour of systems that attempt to provide services for randomly arising demands. It is oriented towards the application of advanced mathematical techniques in modelling a complex system. Queuing network modelling (QNM) was developed to employ a small subset of queuing theory techniques for efficiently modelling two important queuing network systems, computer and communication (C&C) systems. There are two types of QNM [1]. They are open QNM and closed QNM. Open QNM models open queuing networks which have an infinite stream of arriving customers. Customers that have been served will leave the model. Closed QNM models closed queuing networks which have fixed-size “batch” of customers circulating inside the system.

The algorithm of QNM for modelling C&C systems with multiple service stations and different types of customers involves a number of complex mathematical formulae, which make it very difficult to check the correctness of QNM implementation from its computed outputs. This problem is known as the oracle problem [2] in software testing. Recently, the technique of metamorphic testing has been proposed for testing software without the need of oracle [3, 4, 5, 6]. Instead of testing the outputs of the software, this technique identifies some necessary properties as metamorphic relations and checks them among multiple executions of the program being tested. In this paper, we study the application of metamorphic testing to alleviate the oracle problem of testing QNM software. We shall discuss a case study on the testing of a QNM suite called the Java Modelling Tool (JMT) [7].

JMT is a queuing network modelling suite developed in the Java language for capacity planning and performance evaluation of C&C systems. It is composed of six tools that support different analyses frequently used in capacity planning studies of queuing network systems. It is freely downloadable under the GNU General Public Licence. The architecture of JMT is shown in Figure 1. Details of the features of each tool can be obtained from [8]. JMVA is a tool in JMT to calculate the major outputs of the queuing network systems using the mean value analysis technique [9]. It contains two separate modules for open QNM and closed QNM. We shall report the testing of the JMVA module of JMT for open QNM in this paper.

In this paper, we outline the definition of a number of metamorphic relations for this family of problems, and conduct experiments to examine their fault detection effectiveness. We examine whether the metamorphic relations can be used to detect any existing real faults in JMVA, and examine the effectiveness of the different metamorphic relations in detecting seeded faults where no existing faults could be detected.
The rest of the paper is organized as follows. Section II gives a brief introduction to metamorphic testing. Section III explains the basic concept of QNM. In Section IV, we identify seven metamorphic relations for modelling of open queuing network. We then apply the technique of metamorphic testing to test the JMVA module. Section V concludes the paper.

II. A BRIEF INTRODUCTION TO METAMORPHIC TESTING

Metamorphic testing is a property-based testing approach for software testing. It does not check the correctness of individual outputs. Instead, it checks metamorphic relations among multiple executions of the target program. A metamorphic relation (MR) is an expected relation, which is identified from the necessary properties of application domain, over a set of distinct input data and their corresponding output values for multiple executions of the target program. In theory, a program should satisfy all the necessary properties. Any program which violates the MRs contains faults.

Let us consider a function \( f \). Suppose \( R_f \) denotes some properties of \( f \) that can be expressed as a relation among a series of the function’s inputs \( x_1, x_2, \ldots, x_n \) where \( n > 1 \), and their corresponding values \( f(x_1), f(x_2), \ldots, f(x_n) \). This relation \( R_f \) is called a metamorphic relation. For instance, consider the sine function. For any two inputs \( x_1 \) and \( x_2 \) such that \( x_1 + x_2 = \pi \), we must have \( \sin(x_1) = \sin(x_2) \). This property can be a metamorphic relation for testing the correctness of the sine function. It can be written as

\[
R_{\sin}: \text{If } x_1 + x_2 = \pi \text{ then } \sin(x_1) = \sin(x_2)
\]

To verify this relation, two executions are needed in metamorphic testing. The first input to sine function is a real number \( x_1 \), followed by a second input \( x_2 = \pi - x_1 \).

In summary, even if a testing oracle does not exist, metamorphic testing can still be applied as it checks the relations among the inputs and outputs of more than one execution of the program.

III. BASIC CONCEPT OF QUEUING NETWORK MODELLING

QNM is a particular approach to C&C system modelling in which the system is represented as a network of queues to be evaluated by either an analytical or simulation approach. In QNM, a network of queues is a collection of service stations (representing system resources) and customers (representing users, transactions, requests or jobs). The analytical approach involves using software to solve a set of equations induced by the network of queues and its parameters. The simulation approach involves the use of several probability distributions to characterise the system parameters. JMVA employs the analytical approach to model the queuing networks.

The service stations in a queuing network are either connected in series or parallel. In our testing experiment, we tested the systems with system configuration as shown in Figure 2 because they cover the serial and parallel connection of the service stations

There are two types of service stations. A delay station is a station which is allocated to one customer only. There is no competition of service for delay station. On the other hand, a non-delay station is a station which allows more than one customer. There is competition of service in non-delay station, where customers not in service are put in queue. In this paper, we will focus on testing the open QNM with non-delay stations only.

A. System inputs

The following lists the input parameters of QNM software.

1. Number of service stations

Let \( M \) be a set of service stations of a network system, \( M = \{M_1, M_2, \ldots, M_q\} \), where \( q \geq 1 \) is the number of service stations. Figure 2 shows the model with \( q \) service stations.

2. Number of servers

A service station can contain a number of servers. Let \( F_k = \{F_1, F_2, \ldots, F_k\} \) be a set of servers for a service station \( M_k \).

3. Class of customers

The customers of a queuing network model can be grouped into a number of classes based on their workload intensity and average service demands. The workload intensity and service demands are defined below. A class of customer is a set of customers that are statistically equal. Let \( n \) denote the number of customer classes in the model and \( C_i \) denote a class of customers, the model will have a union of all classes of customers denoted as

\[
C = \bigcup_{j=1}^{n} C_j
\]

4. Workload intensity

Workload intensity is the rate at which customers of a given class arrive at the queuing network system. For open queuing network models, the workload
intensity is specified by the arrival rate of the customers [10]. We can represent the workload intensity of an open queuing network model by a set of arrival rate $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the arrival rate of $C_1, C_2, \ldots C_n$, respectively.

(5) Number of visits, service time and service demands

The system behaviour should be steady over time in order to be correctly modelled. With this property in mind, many theorems have been derived about QNM. The most important one was developed by Little in the early 1960s [11]. It is known as Little’s Law. Little’s Law states that the number of service completions at that station to the proportion of time when station is busy, or, equivalently, as the average number of customers in the steady state of an open queuing network. We will use Little’s Law and some related theorems to explain the outputs of QNM software.

A. Utilization of a service station

The utilization of a service station is interpreted as the proportion of time when station is busy, or, equivalently, as the average number of customers in service there [8]. For any station $M_k$ and customer class $C_c$, the utilization is denoted as $U_{c,k}$, where

$$U_{c,k} = \lambda_c \cdot D_{c,k} \tag{1}$$

The aggregate utilization of a station $M_k$ is given by:

$$U_k = \sum_{c=1}^{n} U_{c,k} \tag{2}$$

$U_k$ is not an output parameters. It is used to calculate the residence time as defined below.

B. Throughput

Queue length for the customers of a class $C_c$ at service station $M_k$ is the average number of customers at station $M_k$ waiting in queue and receiving service [8]. We denote it as $Q_{c,k}$. By Little’s law, we have the queue length for open QNM as:

$$Q_{c,k} = \lambda_c \cdot R_{c,k} \tag{8}$$

The queue length for the customers of a class $C_c$ in the system is given by:

$$Q_c = \lambda_c \cdot R_c = \sum_{k=1}^{q} Q_{c,k} \tag{9}$$

IV. TESTING OF JMVA USING THE TECHNIQUE OF METAMORPHIC TESTING

A. Testing subject

The JMVA module of Java Modelling Tool is used as a case study in this paper to evaluate the performance of
metamorphic testing in open QNM. The version used for our study is v.0.7.3. It was released on 26 November 2007.

**B. Inputs and outputs of JMVA**

The inputs and outputs of JMVA are listed in Table I.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters Symbol</td>
<td>Parameters Symbol</td>
</tr>
<tr>
<td>Number of Stations</td>
<td>q</td>
</tr>
<tr>
<td>Number of Servers</td>
<td>(h)</td>
</tr>
<tr>
<td>Number of Classes</td>
<td>(n)</td>
</tr>
<tr>
<td>Arrival Rate</td>
<td>(\lambda)</td>
</tr>
<tr>
<td>Service Time</td>
<td>(S_c)</td>
</tr>
<tr>
<td>Number of Visits</td>
<td>(V_{c,k})</td>
</tr>
</tbody>
</table>

C. Metamorphic relations for testing of JMVA

We have identified seven MRs for open QNM testing from the properties of the queuing network models stated in Section III.

We notice that if \(\lambda_c\) is increased and all other inputs are unchanged, then

i. \(U_{c,k}\) will increase (equation (1)).

ii. \(U_i\) will increase (equation (2)).

iii. \(R_{c,k}\) will increase (equation (3) and (4)).

iv. \(R_c\) will increase (equation (5)).

v. \(X_c\) will increase (equation (6)).

vi. \(X_{c,k}\) will increase (equation (7)).

vii. \(Q_{c,k}\) will increase (equation (8)).

viii. \(Q_c\) will increase (equation (9)).

Since \(U_{c,k}\) and \(U_i\) can be very small (less than \(10^{-7}\)) in the low utilization conditions, the increase of them may not be significant enough to have obvious increase of \(R_{c,k}\) and \(R_c\) in considering the limitation of numerical rounding of floating point computation. This limitation applies to all MRs identification. Based on the properties as stated above, we can identify a metamorphic relation, denoted MR1 as follows:

**MR1:**

If \((\lambda_c < \lambda'_c)\) then

\[(X_{c,k} \leq X'_{c,k}) \land (Q_{c,k} \leq Q'_{c,k}) \land (R_{c,k} \leq R'_{c,k}) \land (U_{c,k} < U'_{c,k}) \land (X_c < X'_c) \land (R_c \leq R'_c) \land (Q_c \leq Q'_c)\]

We also notice that if we increase \(S_{c,k}\) while leaving all other inputs unchanged, then

i. \(U_{c,k}\) will increase (equation (1)).

ii. \(U_i\) will increase (equation (2)).

iii. \(R_{c,k}\) will increase (equation (3) and (4)).

iv. \(R_c\) will increase (equation (5)).

v. \(X_c\) will not change (equation (6)).

vi. \(X_{c,k}\) will not change (equation (7)).

vii. \(Q_{c,k}\) will increase (equation (8)).

viii. \(Q_c\) will increase (equation (9)).

We can identify MR2 based on the above properties as follow:

**MR2:**

If \((S_{c,k} < S'_{c,k})\) then

\[(X_{c,k} = X'_{c,k}) \land (Q_{c,k} < Q'_{c,k}) \land (R_{c,k} < R'_{c,k}) \land (U_{c,k} < U'_{c,k}) \land (X_c = X'_c) \land (R_c < R'_c) \land (Q_c < Q'_c)\]

If we increase the number of parallel service stations \(q\) and keep \(\lambda_c\) and \(S_{c,k}\) constant, then

i. \(V_{c,k}\) will decrease due to more parallel stations serving the customers.

ii. \(U_{c,k}\) will decrease (equation (1)).

iii. \(U_i\) will decrease (equation (2)).

iv. \(R_{c,k}\) will decrease (equation (3) and (4)).

v. \(R_c\) will decrease (equation (5)).

vi. \(X_c\) will not change (equation (6)).

vii. \(X_{c,k}\) will decrease (equation (7)).

viii. \(Q_{c,k}\) will decrease (equation (8)).

ix. \(Q_c\) will decrease (equation (9)).

MR3 is therefore defined as follow:

**MR3:**

If \((q < q')\) then

\[(X_{c,k} = X'_{c,k}) \land (Q_{c,k} = Q'_{c,k}) \land (R_{c,k} = R'_{c,k}) \land (U_{c,k} = U'_{c,k}) \land (X_c = X'_c) \land (R_c = R'_c) \land (Q_c > Q'_c)\]

By applying Little’s law, we can identify two more MRs, MR4 and MR5, which involve multiplying \(\lambda_c\) by a constant \(n\).

**MR4:**

If \((\lambda'_c = n \cdot \lambda_c)\) then

\[Q'_{c,k} = n \cdot \frac{R'_{c,k}}{R_{c,k}}\]

**MR5:**

If \((\lambda'_c = n \cdot \lambda_c)\) then

\[Q'_c = n \cdot \frac{R'_c}{R_c}\]

Increasing \(\lambda_c\) will cause \(R_{c,k}\) to increase except for any service stations which have low utilization. Considering the queuing network shown in Figure 2, the first service station is utilized. Hence, MR6 is defined as follows:

**MR6:**

If \((\lambda_c < \lambda'_c)\) then

\[\exists M_k \in M \text{ such that } R_{c,k} \neq R'_{c,k}\]

If we modify the service time of any service station \(M_k\) for a customer class \(C_k\) from \(S_{c,k}\) to \(S'_{c,k} = n \cdot S_{c,k}\), we will have the following relation from equation (3) for the single server service stations:
\[ \frac{R_{c,k}' \cdot (1 - U_k')}{R_{c,k} \cdot (1 - U_k)} = n \]

But this relation is not valid for the multiple server service stations as their residence time is computed by using equation (4). We can define a metamorphic relation, denoted MR7, for the computation of residence time of multiple server service stations as shown below:

\[ \text{MR7: } \begin{cases} (l_k = l_k') & \land (l_k \neq 1) \land (U_k \neq 1) \land \left( S_{c,k}' = n \cdot S_{c,k} \right) \\ R_{c,k}' \cdot (1 - U_k') \neq n \end{cases} \]

As a reminder, in the generation of the follow-up test cases, random selection would be applied if necessary. In this study, only one input parameter was changed for testing against a MR. For example, we only changed \( \lambda \) to \( \lambda' \) for MR1.

### Table II. Input Parameters Range for Open QNM

<table>
<thead>
<tr>
<th>Input Parameters</th>
<th>Original Test Cases</th>
<th>Follow-up Test Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of class: ( n )</td>
<td>[1, 8]</td>
<td>Same values</td>
</tr>
<tr>
<td>Arrival rate: ( \lambda )</td>
<td>(0, 0.4]</td>
<td>(0, 0.6]</td>
</tr>
<tr>
<td>Service time: ( S )</td>
<td>(0, 0.002]</td>
<td>(0, 0.01]</td>
</tr>
<tr>
<td>Number of station: ( q )</td>
<td>[2, 30]</td>
<td>[3, 50]</td>
</tr>
<tr>
<td>Number of visits: ( V )</td>
<td>[3, 80]</td>
<td>Same values</td>
</tr>
<tr>
<td>Number of server: ( l )</td>
<td>[1, 5]</td>
<td>[2, 10]</td>
</tr>
</tbody>
</table>

The number of servers \( l \) is used in the Erlang C function for the computation of residence time. The output of the Erlang C function will be close to zero when \( l \) is large enough (\( l = 10 \) in our case). The change in residence time is not significant if the output of the Erlang C function becomes very small (less than \( 10^{-8} \)). Hence, we set the range of \( l \) as [1, 5] for the original test cases and [2, 10] for the follow-up test cases.

As mentioned in Section III-B, a system should be steady over time to be correctly modelled. For an open queuing network model, the traffic intensity \( \rho \) is defined as \( \frac{\lambda}{\mu} \) [12], where \( \mu = 1/D \) [10]. \( \rho \) must be less than 1 for steady-state to exist. Hence, we have \( \frac{\lambda}{\mu} \cdot S \cdot V < 1 \) for a service station \( M \).

For the whole queuing network system, this requires

\[ \max_k \left\{ \sum_{c=1}^{n} \frac{\lambda_c \cdot S_{c,k} \cdot V_{c,k}}{S} \right\} < 1 \quad (10) \]

The selection of test cases is subject to the constraint specified in (10) for the testing of the open QNM tool of JMVA. If the selected inputs do not satisfy (10), JMVA will reject the inputs with an alert message.

Consider a queuing network model as shown in Figure 2. We randomly generated 100 test inputs as original test cases. We then generated 100 follow-up test cases for every MRs.
There are four mutants (Mu4, Mu11, Mu17, Mu18) which cannot be killed by any MRs in open QNM. All these four mutants are related to the computation of equation (4). The relationship (4) between $R_{c,k}$ and $\lambda_c$ with all other inputs constant is plotted in Figure 4. In Figure 4, Figure 5 and Figure 6, $R(y)$ denotes the $R_{c,k}$ of the original software while $Mu4(y)$, $Mu11(y)$, $Mu17(y)$ and $Mu18(y)$ denote the $R_{c,k}$ returned by mutants Mu4, Mu11, Mu17 and Mu18, respectively. We observe in Figure 4 that the original software, Mu4, Mu11, Mu17 and Mu18 all have the same trend of variation. Consequently, the MRs related to the change of $R_{c,k}$ and $\lambda_c$ cannot detect the fault of these mutants. Figure 5 shows the relationship of $R_{c,k}$ and $S_{c,k}$ that the original software and the four mutants have the same linear relationship. Hence, the MRs related to the change of $S_{c,k}$ are not able to detect the fault of these mutants. We plotted the relationship between $R_{c,k}$ and $q$ in Figure 6. We can observe that the original software, $Mu4$, $Mu11$, $Mu17$ and $Mu18$ all have the same trend of variation. The MRs related to the change of $R_{c,k}$ and $q$ cannot kill these mutants either.
V. CONCLUSION

QNM applies either analytical or simulation methods to evaluate the C&C systems. As mentioned in Section I, QNM is algorithmically complex, and therefore verifying testing output can be difficult or impossible. In this paper, we tested open QNM of the JMVA module of JMT as a case study of using metamorphic testing to verify the correctness of software implementation of QNM.

We identified a number of metamorphic relations for open QNM and conducted metamorphic testing of JMVA to see whether we can detect real fault in this popular open source QNM suite.

No real faults were detected in our experiments. Hence, we applied the technique of mutation analysis to investigate the effectiveness of metamorphic testing. We generated 20 mutants in open QNM module of JMVA and tested them again with seven metamorphic relations. We killed 16 out of 20 mutants by our MRs. Although we cannot kill all the mutants in open QNM, we have demonstrated the capability of metamorphic testing to reveal a high proportion of the faults in this kind of software. In practice, it is unrealistic to expect metamorphic testing to detect all mutants; indeed, some mutants may produce the correct outputs on the selected test cases so would not be detected even with a testing oracle. Our results are broadly in line with an earlier case study of metamorphic testing [13], in which MT was able to kill a high proportion of mutants. In this case study, the nature of the incorrect output of the mutants was such that the metamorphic relations would not reveal a failure, regardless of the source test cases chosen. Therefore, the only way to detect more mutants would have to be devising additional MRs.

Our results show that different mutants were killed by different MRs, even though the source test cases were the same in each case. MR1 and MR3 killed the most mutants. However, MR7, while killing only one mutant, killed a mutant that was not killed by any other metamorphic relation. MR4 and MR5 killed the same mutants, and thus – at least in our experiments - one of them could be viewed as redundant. Collectively, this indicates that, as one might expect, the best results are likely to be achieved by the broadest range of metamorphic relations.

This paper is a first step to study the effectiveness of metamorphic testing in the QNM software. Identifying the effective MRs relies on the knowledge of the software application domain. The authors are not experts in QNM, and the MRs were devised based on quite simple intuitions about the properties of QNM. Nor did the authors have detailed knowledge of the implementation of JMVA. Despite this, we were able to test the system quite effectively. However, it is reasonable to imagine that true domain experts would be able to devise additional or more effective MRs. Furthermore, it took some time to devise the MRs used in this case study, and some initial experimentation was conducted with MRs that proved to be invalid. Domain experts may well have avoided these mistakes and devised valid MRs much more quickly. Determining the extent to which domain experts can improve the effectiveness and productivity of metamorphic testing would seem an area worthy of future study.

ACKNOWLEDGMENT

This research project is supported by an Australian Research Council Discovery Grant (DP09847600) and a grant from the China 863 Project (2008AA01Z142).

REFERENCES