Hidden Markov Model Based Automated Fault Localization for Integration Testing

Ning Ge, Shin Nakajima, Marc Pantel

Hidden Markov Model Based Automated Fault Localization for Integration Testing

Ning Ge
University of Toulouse - IRIT/INPT
Toulouse, France
ning.ge@enseeiht.fr

Shin Nakajima
National Institute of Informatics
Tokyo, Japan
nkjm@nii.ac.jp

Marc Pantel
University of Toulouse - IRIT/INPT
Toulouse, France
marc.pantel@enseeiht.fr

***Abstract*—Integration testing is an expensive activity in software testing, especially for fault localization in complex systems. Model-based diagnosis (MBD) provides various benefits in terms of scalability and robustness. In this work, we propose a novel MBD approach for the automated fault localization in integration testing. Our method is based on Hidden Markov Model (HMM) which is an abstraction of system's component to simulate component's behaviour. The core of this method is a fault localization algorithm that gives out the set of suspect faulty components and a backward algorithm that calculates the matching degree between the HMM and the real system to evaluate the confidence degree of the localization conclusion. The proposed method is evaluated on a specific test bed and is applied to a simple helicopter control system case study.**

***Index Terms*—Model-Based Diagnosis; Hidden Markov Model; Automated Fault Localization; Integration Testing;**

I. Introduction

Automated integration testing is an efficient way to detect the faults caused by the interactions between components. However, detecting and locating faults in a complex system with large amount of components are expensive activities. Models, as the abstraction of real system, contain enough core information to represent the desired behaviour of a system under test. Model-based diagnosis (MBD) [1] provides various benefits in terms of scalability and robustness. Improving integration testing by using automated fault localization approach based on model is still an important subject. Fault localization algorithms follow two paradigms: cause-effect and effect-cause analyses. Cause-effect analysis [2]–[4] starts from possible causes (fault models). A simulator is used to predict system's behaviour in the presence of various faults. Then predictions are matched against observed behaviour. Effect-cause analysis [5], [6] reasons faulty localization based on observed behaviour and expected good functions. It back-traces faulty causes from the identified suspect components.

In this work, we make a tradeoff of cause-effect and effect-cause analyses and propose an Hidden Markov Model (HMM) [7] based approach for the automated localization of faulty components in integration testing. The component can be hardware device, software modules or functional blocks in the system. Let $T = \{\tau_1, \tau_2, ..., \tau_n\}$ be test cases of a faulty system S, where $\tau_k = (i_k, o_k)(k = 1...n)$ is a test case with i_k as input and o_k as expected output. The test output of τ_k is $\hat{o}_k = S(i_k)$. τ_k is a passed test case, if and only if $\hat{o}_k = o_k$, otherwise, τ_k is a failed case. We automatically locate the set of suspect faulty components, with given information (S, T). This method combines forward localization analysis and backward confidence degree evaluation. HMM, as a component's abstraction, provides statistically identical information to component's real behaviour. The core of this method is a fault localization algorithm that gives out the set of suspect faulty components and a backward algorithm that calculates the matching degree between the HMM and the real system to evaluate the confidence degree of the localization conclusion. The proposed method is evaluated on a specific test bed and then is applied to a simple helicopter control system case study.

This paper is organized as follows: Section II briefly introduces HMM; In Section III, the modeling method based on HMM and the fault localization algorithms are discussed; The experimental results in our specific test bed and in the helicopter control system case study are presented in Section IV; Section V gives some concluding remarks.

II. HMM Modeling and Analysis

An HMM is defined as a statistical model used to represent stochastic processes, where the states are not directly observed. A basic HMM can be described as follows:

- $\mathbf{N}$: number of states
- $\mathbf{M}$: number of observations
- $\mathbf{M_I}$: initial probability distribution; $\sum_{i=1}^{\mathbf{N}} \mathbf{M_I}(i) = 1$
- $\mathbf{M_T}$: probability distribution of transitions from states to states; $\sum_{j=1}^{\mathbf{N}} \mathbf{M_T}(i,j) = 1, i = 1...\mathbf{N}$
- $\mathbf{M_E}$: emission distribution for the observations associated with states; $\sum_{j=1}^{\mathbf{M}} \mathbf{M_E}(i,j) = 1, i = 1...\mathbf{N}$

HMM is statistically identical to system's real behaviour. HMM separates the concept into two conceptually independent paradigms: behaviour and observation. Behaviour refers to what the system really is; while observation relates the elements exhibited by the system that are used for its recognition. $\mathbf{M_I}, \mathbf{M_T}$ and $\mathbf{M_E}$ can be obtained by modelling or through a learning process. Once all these matrix parameters are estimated, HMM is capable to deduce the maximum likelihood

estimation of inner-state transition sequences. The trained HMM will be used to estimate system's past condition, predict system's future condition and compute a future observed sequence's occurrence probability.

Our goal is to find out how the system performs in terms of behaviour, however, the only available information source for the external world is the observation. Therefore a backward analysis is necessary to recover the behaviour from the observation. The core issue is to construct HMM representing a system with probable fault. The learning task cannot be used here. Because the learning task will train all the parameters in HMM, while in our method, the parameters in $\mathbf{M_T}$ are calculated by using the test results.

III. Automated Faults Localization Based on Hidden Markov Model

This approach is based on component analysis. Each component is mapped to an HMM. If all the input/output pair of a component can be exhaustively listed, we can get an exact distribution of how this component respects the functional constraints. This approach can be explained by using Fig. 1. An HMM, as a component's abstraction, provides statistically

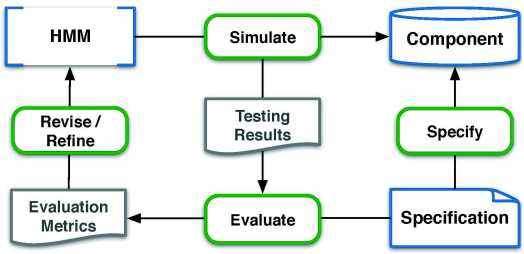

Fig. 1. HMM Local Approach

similar information to simulation by $\mathbf{M_T}$. That is equally saying, a component can be simulated by HMM if we can be sure they behave statistically in the same way. To measure whether they behave the same, we introduce an evaluation approach by using test results. The evaluation metrics returned are used to revise or refine the parameters in HMM, until it approximates the component's real behaviour.

A. System States

A component C is mapped to an HMM. HMM's states are the combination of component's faulty status and its inputs faulty status. System's behaviour is modeled by 4 states:

- PI_p: C is not faulty, C's inputs are passed
- PI_f: C is not faulty, C's inputs are failed
- FI_p: C is faulty, C's inputs are passed
- FI_f: C is faulty, C's inputs are failed

B. Observations & Observed Sequence

The observation is defined by the test result of component's outputs. There are only 2 observations:

- O_p: outputs are passed
- O_f: outputs are failed

By defining HMM states and observations in this way, the dependency between components is built. For component C, its output observation is the input (contained in the defined states) of the successor.

C. Initial Probability Distribution $\mathbf{M_I}$

Assume components' faulty probability is ω, $\mathbf{M_I}$ is defined as follow. If ω is not available, we assume each state has identical initial faulty probability, i.e. $\omega = \frac{1}{2}$

	PI_p	PI_f	FI_p	FI_f
$Init$	$\frac{1-\omega}{2}$	$\frac{1-\omega}{2}$	$\frac{\omega}{2}$	$\frac{\omega}{2}$

D. Transition Probability Matrix $\mathbf{M_T}$

$\mathbf{M_T}$ is statistical values derived from test results. If n test cases are observed, a component with m inputs corresponds to $m \cdot n$ input sequences. Using the $m \cdot n$ input sequences, we can calculate the fault probabilities $\alpha, \beta, \gamma, \delta$ respectively for the transitions $O_p \sim O_p, O_p \sim O_f, O_f \sim O_p, O_f \sim O_f$, where $\alpha + \beta + \gamma + \delta = 1$. $\mathbf{M_T}$ is then calculated as follows. If ω is not available, we assume $\omega = \frac{1}{2}$.

	PI_p	PI_f	FI_p	FI_f
PI_p	$\frac{\alpha(1-\omega)}{\alpha+\beta}$	$\frac{\beta(1-\omega)}{\alpha+\beta}$	$\frac{\alpha\omega}{\alpha+\beta}$	$\frac{\beta\omega}{\alpha+\beta}$
PI_f	$\frac{\gamma(1-\omega)}{\gamma+\delta}$	$\frac{\delta(1-\omega)}{\gamma+\delta}$	$\frac{\gamma\omega}{\gamma+\delta}$	$\frac{\delta\omega}{\gamma+\delta}$
FI_p	$\frac{\alpha(1-\omega)}{\alpha+\beta}$	$\frac{\beta(1-\omega)}{\alpha+\beta}$	$\frac{\alpha\omega}{\alpha+\beta}$	$\frac{\beta\omega}{\alpha+\beta}$
FI_f	$\frac{\gamma(1-\omega)}{\gamma+\delta}$	$\frac{\delta(1-\omega)}{\gamma+\delta}$	$\frac{\gamma\omega}{\gamma+\delta}$	$\frac{\delta\omega}{\gamma+\delta}$

E. Emission Probability Matrix $\mathbf{M_E}$

The key of this approach is evaluating $\mathbf{M_E}$. According to previous analysis, we need to measure how well an HMM statistically simulates a given component, which is represented by *Matching Level.*

Definition 1 (**Matching Level** (μ)). *Matching level evaluates how HMM simulates a component's real behaviour.*

The objective is to find an HMM with the highest matching level. This turns the problem to be an optimization problem, and many techniques can be applied, e.g. exhaustive search by minimum internal, heuristic algorithm, evolution algorithm, experimental design, etc. In this work, the matching level is derived by calculating the probability that a component's outputs pass the tests through HMM observations. However, this matching level is a local value relative to one component, making it have no meaning to compare with others. Therefore, only the forward search is not enough to guarantee the matching level. We introduce the concept of *Confidence Level* to evaluate the global confidence of matching level.

Definition 2 (**Confidence Level** (ρ)). *Confidence level evaluates the confidence of the matching level.*

Algorithm 1 ($\mathbf{M_E}$ **Evaluation Algorithm**). *The following algorithm is used to evaluate the* $\mathbf{M_E}$ *in HMM* (h) *corresponding to component* C. $T_{m,n} = \{\tau_1, \tau_2, ..., \tau_n\}_m$ *are test cases results for* m *outputs of* C. *The threshold of*

matching level μ and confidence level ρ are pre-defined.

1: ***procedure*** EVALUATE($h, \mu, \rho, s, T_{m,n}$)
2: $e \Leftarrow \mu$
3: ***while*** $e \leqslant \mu$ ***do***
4: $p \Leftarrow \rho$
5: ***while*** $p \leqslant \rho$ ***do***
6: $\mathbf{M_E} \Leftarrow$ *Generate_M_E()*
7: $p \Leftarrow \sqrt[m \cdot n]{Evaluate\_Confidence(h, T_{m,n})}$
8: ***end while***
9: *states* $\Leftarrow$ *Get_behaviour_States(h)*
10: $in_{hmm} \Leftarrow$ *Get_Input(h, states)*
11: $out_{pre} \Leftarrow$ *Get_Output(Pre(C))*
12: $e \Leftarrow$ *Evaluate_Matching(in_{hmm},out_{pre})*
13: ***end while***
14: ***end procedure***

We explain the core functions in the algorithm.

1) $\mathbf{Generate\_M_E}()$*:* randomly generates an $\mathbf{M_E}$. $\mathbf{M_E}$ is the emission probability from states (PI_p, PI_f, FI_p, FI_f) to observations (O_p, O_f). It is thus a 4-by-2 matrix with 4 variables, because sum of row is 1. Each time an $\mathbf{M_E}$ is generated, we get a complete HMM for one component.

2) $\mathbf{Evaluate\_Confidence}(h, T_{m,n})$ *:* evaluates how the generated HMM approximates the real behaviour of current component by using the output test cases results $T_{1:m,n}$, which represents real behaviour. As HMM can calculate the occurring probability of an observed sequence, this probability means how probable a component's behaviour happens in the evaluated HMM. If this probability is high enough, the evaluated HMM's similarity to real component is also high. This probability is calculated by multiplying all the probabilities on the sequence's path. The value is normalized to make the confidence level comparable to the values of other components.

3) $\mathbf{Get\_behaviour\_States}(h, T_{m,n})$*:* calculates the states behaviour of HMM. HMM can give out the most probable states behaviour once the observation sequence is given.

4) $\mathbf{Get\_Input}$*(h, states):* Since HMM states is a union of input passed results and component passed result, the input sequence can be derived from the states behaviour.

5) $\mathbf{Get\_Output}(\boldsymbol{Pre}(C))$*:* gets the result about the output of component C's predecessor, which is the real input behaviour of C.

6) $\mathbf{Evaluate\_Matching}$*($in_{hmm}$, out_{pre}):* evaluates the matching level between the input derived from test results and the input derived from HMM states. The former represents component's real behaviour, while the later represents HMM's simulation.

F. Estimating System's behaviour & Locating Fault

When the HMM with high matching level and high confidence level is confirmed, we can estimate the component's status. With the states calculated from function $\mathbf{Get\_behaviour\_States}(h, T_{m,n})$, e.g. a states behaviour sequence like

$$(FI_p \quad FI_p \quad PI_f \quad FI_p \quad ... \quad FI_f \quad FI_p \quad ...)$$

In the states behaviour sequence, we focus on the status of component, as follow. The status with higher occurrence probability is confirmed as this component's faulty status.

$$(F \quad F \quad P \quad F \quad \blacksquare \quad F \quad F \quad ...)$$

This result's confidence is guaranteed by the confidence level (ρ), and the HMM's similarity to the component is guaranteed by the matching level (μ). ρ and μ are calculated within system's topological structure, therefore we can compare all the faulty component's ρ and μ, and give a set of suspect faulty components ordered by ρ and μ.

IV. EXPERIMENTAL RESULTS

A. Test Bed

We design a specific test bed to evaluate the method's accuracy and efficiency by generating a large quantity of use cases. Each use case includes: the **system architecture** which defines the components and the ports, and their interconnections; the **failure probability** of each component; the **functional specification** corresponding to the inputs/outputs;

The method assumes that: Each component in the system has a chance to fail if it has design problem. This probability will be 0 if no design fault is presumed for this component. All functional constraints are based on input/output's value itself, and for simplification, they are all range constraint, which means they delimit only the min/max value of the input/output.

If a faulty component exists, the test bed will by chance give out an out-of-range value for this component's output. This emulates how a system fails, whatever the model is.

Each component's input and output will be allocated to a variable by the test bed. It guarantees that the interconnected ones share the same variable. The variable will be associated with a random range, which is the functional specification. If a device is more probable to fail, the test-bed-generated value for its entire output variables will be more probable to go out of the defined range.

The approach will use the generated data and the functional specification to automatically locate the faulty component. The test bed will then compare this computed conclusion with the initial context to deduce whether the method is efficient.

B. Experiments

Our test bed has generated 1000 use cases to evaluate the method's performance in terms of accuracy. The criteria that impacts the accuracy is the complexity of system's architecture. This can be measured by component number. (Fig. 2) and component's average input & output number (Fig. 3). We find out that this method is more sensible to the average input & output number, while it is more scalable to component number. This method deals with the fault localization for middle-range systems with a accuracy superior to 90%.

The computation of $\mathbf{M_E}$ parameters by iteratively searching algorithm consumes time, which is bounded by the iteration limit. This Monte-Carlo algorithm runs from several seconds

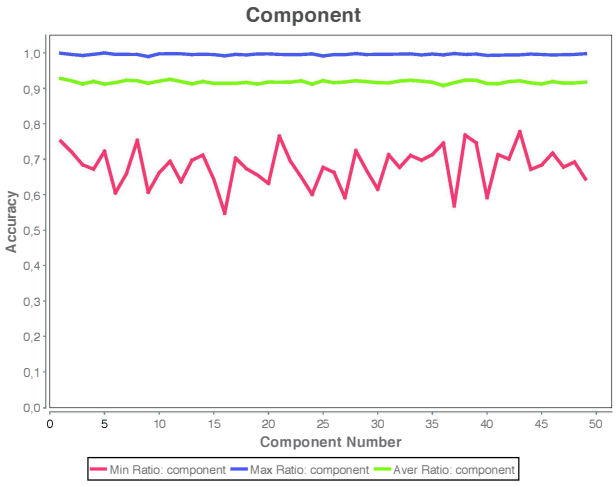

Fig. 2. Accuracy by Component Number

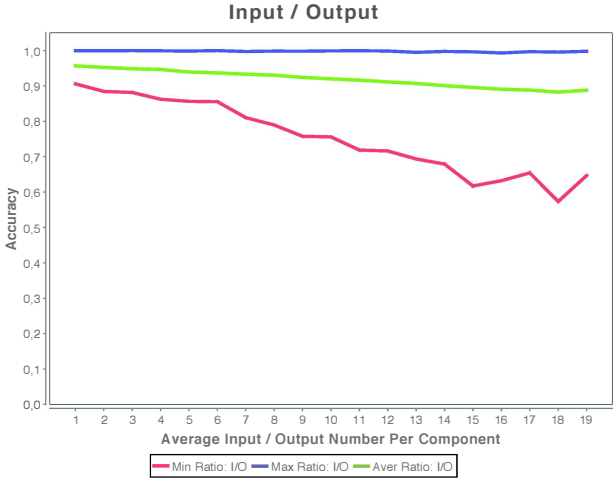

Fig. 3. Accuracy by Input Number

to several minutes. However, as the computation of each component is independent, the whole method is linearly scalable. For a large system, a parallel cluster will locate the probable design faults within minutes.

C. Helicopter Control System Case Study

After certifying statistically the method's performance using the test bed, we rely on a classical Simulink teaching model of simple helicopter control system [8] to evaluate, whether the proposed approach works also well for real system. A complete mathematical model, including propeller dynamics, forces generated by the propellers, static and dynamic friction of the bearings, etc. is an overkill for method proofing objectives, therefore a simplified version is chosen. Regarding the functional specification, this helicopter is a linear 4^{th} order system, where the 6 control matrices are pre-defined.

In order to introduce faults into the case study, we modify the Simulink model to have some components generating faulty outputs in some test cases. These components will have a computed failure probability which is based on the ratio between failure count and total test case number. The fault localization method gives a good rate of accuracy throughout the test, as shown in Table I.

TABLE I
HELICOPTER CONTROL SYSTEM CASE STUDY TEST RESULTS

Comp. ID.	Testbed Status	HMM Conclu.	Matching Level	Confidence Level	Approach Accuracy
1	P	P	1	0.024181	YES
2	P	P	1	0.024181	YES
3	P	P	1	0.996539	YES
4	P	P	1	0.996539	YES
5	F	F	0.905	0.024181	YES
6	F	P	0.975	0.024181	NO
7	P	P	1	0.99654	YES
8	P	P	1	0.024181	YES
9	P	P	1	0.996573	YES
10	P	P	1	0.99654	YES
11	F	F	0.71	0.54545	YES
12	P	P	1	0.024181	YES
13	P	P	1	0.99654	YES
14	F	F	0.675	0.024181	YES
15	F	F	0.67	0.541556	YES

V. CONCLUSION

In this paper, we proposed an automated fault localization approach based on HMM for integration testing in system engineering. Benefiting from HMM, we propose to model faulty component's behaviour. Instead of estimating HMM's parameters by classic learning process, a forward analysis method to estimate parameters using system's topological structure and tests results was introduced. To compute the matching degree between HMM and component's behaviour and to evaluate the confidence degree of the localization conclusion, a backward analysis algorithm was introduced. Experimental results shows that, this method can deal with the fault localization problem for middle-range systems with accuracy superior to 90%.

REFERENCES

[1] W. Hamscher, L. Console, and J. de Kleer, Eds., *Readings in model-based diagnosis*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992.

[2] H. Takahashi, K. Boateng, K. Saluja, and Y. Takamatsu, "On diagnosing multiple stuck-at faults using multiple and single fault simulation in combinational circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 21, no. 3, pp. 362 –368, mar 2002.

[3] A. Zeller, "Isolating cause-effect chains from computer programs," in *Proceedings of the 10th ACM SIGSOFT symposium on Foundations of software engineering*, ser. SIGSOFT '02/FSE-10. New York, NY, USA: ACM, 2002, pp. 1–10.

[4] B. Jobstmann, S. Staber, A. Griesmayer, and R. Bloem, "Finding and fixing faults," *J. Comput. Syst. Sci.*, vol. 78, no. 2, pp. 441–460, 2012.

[5] M. Abramovici and M. Breuer, "Multiple fault diagnosis in combinational circuits based on an effect-cause analysis," *Computers, IEEE Transactions on*, vol. C-29, no. 6, pp. 451 –460, june 1980.

[6] A. Smith, A. Veneris, M. Ali, and A. Viglas, "Fault diagnosis and logic debugging using boolean satisfiability," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, no. 10, pp. 1606 – 1621, oct. 2005.

[7] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257 –286, February 1989.

[8] B. Cazzolato, *Automatic Control II - 2DOF Helicopter Tutorial & Lab*.