This discussion is based on the paper:


We will discuss the paper for what it represents and later see how the approach can benefit us with respect to our “mission”

The paper discusses Markov Chains as models for software usage
- uses finite state discrete parameter Markov chain
- states of the Markov chain represent entries from the input domain of the software
- transitions (arcs) define ordering that determines the event space, or sequence, of the experiment

**Black box view of software system**

[Whi93, fig.1]
USAGE MODELS

• Markov analysis of software specifications
  • define underlying probability law for the usage of the software under consideration
  • analysis of specification done prior to design and coding
  • analysis yields irreducible Markov chain (usage Markov chain)
    • unique start state $S_0$
    • unique final state $S_F$
    • set of intermediate usage states $S_i$
    • states set $S = \{S_0, S_F\}$ union $S_i$
    • set $S$ is ordered by probabilistic transition relation

\[(S \times [0,1] \times S)\]

• next state is independent of all past states given the present states
  • Markov property (first order chain)

Usage Markov chain has two properties

• Structural Phase
  • the states and transitions of the chain are established
• Statistical Phase
  • the transition probabilities are assigned

Highest level transition diagram

[Whi93, fig. 2]
• Example: a simple window application [Whi93, fig3]

Example Window

- (pull down menu)

<table>
<thead>
<tr>
<th>Stimulus</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invocation</td>
<td>Place the window of figure 3.2 on the screen</td>
</tr>
<tr>
<td>Select ▲</td>
<td>Expand the window dimensions to cover the entire area of the screen</td>
</tr>
<tr>
<td>Select ▼</td>
<td>Remove the window and replace it with its corresponding icon</td>
</tr>
<tr>
<td>Select • and choose Move</td>
<td>Move the window as directed by the mouse input (obeying screen boundaries)</td>
</tr>
<tr>
<td>Select • and choose Size</td>
<td>Size the window as directed by the mouse input (obeying minimum and maximum limits)</td>
</tr>
<tr>
<td>Select • and choose Close</td>
<td>Remove the window from the screen</td>
</tr>
<tr>
<td>Select the icon and release</td>
<td>Remove the icon from the screen and restore the window</td>
</tr>
</tbody>
</table>

• Example
Software Specification

[Whi93, table I]
• Expansion of the top level usage diagram \([\text{Whi93, fig. 4}]\)

![Usage Model Diagram](image)

• Structural phase - Constructing the usage Markov chain

• phase is complete when usage is completely modeled \([\text{Whi93, fig. 5}]\)

![Usage Model Diagram](image)
• **Statistical Phase**
  - assignment of transition probabilities
  - different approaches to statistical phase
  - **uninformed approach**
    - assign uniform probability distribution across the exit arcs for each state
    - useful when no information is available to make more informed choice
  - **informed approach**
    - when some actual user sequences are available
    - could be captured inputs from a prototype, or profiling information
    - resulting relative frequencies can be used to estimate the transition probability in the usage chain
• Statistical Phase
  • intended approach
    • similar to informed approach but...
    • sequences are obtained by hypothesizing runs
      of the software by a careful and reasonable
      user
    • relative frequency estimates of transition
      probabilities are computed from the symbol
      transition counts as in the informed approach

• How does one rank the approaches?

• Captured or hypothesized sequences  [Whi93, table II]

1. <Invocation> <Window> <Maximize> <Window> <Close> <Termination>
2. <Invocation> <Window> <Minimize> <Icon> <Restore> <Window> <Close> <Termination>
3. <Invocation> <Window> <Move> <Drag Mouse> <Down> <Drag Mouse> <Right> <Drag Mouse> <Down> <Drag Mouse> <Window> <Close> <Termination>
4. <Invocation> <Window> <Size> <Drag Mouse> <Left> <Drag Mouse> <Up> <Drag Mouse> <Left> <Drag Mouse> <Window> <Close> <Termination>
5. <Invocation> <Window> <Move> <Drag Mouse> <Down> <Drag Mouse> <Left> <Drag Mouse> <Down> <Drag Mouse> <Window> <Close> <Termination>
6. <Invocation> <Window> <Size> <Drag Mouse> <Down> <Drag Mouse> <Right> <Drag Mouse> <Window> <Close> <Termination>
**USAGE MODELS**

<table>
<thead>
<tr>
<th>From-State</th>
<th>To-State</th>
<th>Frequency</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invocation</td>
<td>Window</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Window</td>
<td>Maximize</td>
<td>1</td>
<td>1/12</td>
</tr>
<tr>
<td>Window</td>
<td>Minimize</td>
<td>1</td>
<td>1/12</td>
</tr>
<tr>
<td>Window</td>
<td>Move</td>
<td>2</td>
<td>1/6</td>
</tr>
<tr>
<td>Window</td>
<td>Size</td>
<td>2</td>
<td>1/6</td>
</tr>
<tr>
<td>Window</td>
<td>Close</td>
<td>6</td>
<td>1/2</td>
</tr>
<tr>
<td>Maximize</td>
<td>Window</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Minimize</td>
<td>Icon</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Icon</td>
<td>Restore</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Restore</td>
<td>Window</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Move</td>
<td>Drag Mouse</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Size</td>
<td>Drag Mouse</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Drag Mouse</td>
<td>Window</td>
<td>4</td>
<td>4/15</td>
</tr>
<tr>
<td>Drag Mouse</td>
<td>Up</td>
<td>1</td>
<td>1/15</td>
</tr>
<tr>
<td>Drag Mouse</td>
<td>Down</td>
<td>5</td>
<td>1/3</td>
</tr>
<tr>
<td>Drag Mouse</td>
<td>Left</td>
<td>3</td>
<td>1/5</td>
</tr>
<tr>
<td>Drag Mouse</td>
<td>Right</td>
<td>2</td>
<td>2/15</td>
</tr>
<tr>
<td>Up</td>
<td>Drag Mouse</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Down</td>
<td>Drag Mouse</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Left</td>
<td>Drag Mouse</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Right</td>
<td>Drag Mouse</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Close</td>
<td>Termination</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Termination</td>
<td>Invocation</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

**Assigning transition probabilities**

[Whi93, table II]

(putting it on one page)
Usage Models

• Test Cases
  • Statistical Test Case
    • any connected state sequence of the usage chain begins in the start state and ends in the termination state

• Usage Distribution $\pi$
  • the structure of the usage chain induces a probability distribution on the input domain of the software
  • this distribution is called usage distribution
  • each state $S_i$ has steady-state probability $\pi_i$
    • i.e., the probability of being in state $i$ is $\pi_i$

Usage Distribution $\pi$

• usage distribution can be computed by $\pi = \pi P$

• $P$ is the transition matrix of the usage chain
  • $P$ can be encoded as a 2-D matrix ($P$ is a square matrix)
  • state labels are indices and transition probabilities are entries
  • each row sums up to one
  • each entry $\pi_i$ is the expected appearance rate of state $S_i$ in the long run
  • this tells software testers where the user spends most of its time
    • perhaps focus attention on these parts
    • there is a danger to this though, the bug may be in the less used functions
  • states can be grouped (allows comparison of subsections of software)
    • usage distributions are just summed up
    • collapsing states in a Markov chain may require adjustments to transitions
USAGE MODELS

• Other useful statistics

• Number of states necessary until $S_i$ is expected to be generated, denoted by $x_i$, is computed by

$$x_i \pi_i = 1 \quad \Rightarrow \quad x_i = \frac{1}{\pi_i}$$

• if $S_i$ is the termination state, then $x_i$ is the expected number of states until termination of the software

USAGE MODELS

• Expected number of sequences $s_i$ necessary until state $i$ occurs

$$s_i = \frac{x_i}{x_{TERM}} = \frac{\pi_{TERM}}{\pi_i}$$

• largest element of vector $s$ identifies the amount of expected testing until all usage states are encountered at least once

• note: TERM indicates termination state
usage models

• Mean first passage times $m_{jk}$

  • $m_{jk}$ is the expected number of usage states visited starting from $S_j$ until the first visit to $S_k$

  \[
  m_{jk} = 1 + \sum_{i \neq k} p_{ji} m_{ik}
  \]

  • $p_{ij}$ indicate the transition probabilities

  • indicates the extent to which $S_i$ and $S_k$ are encountered within the same sequence

  • e.g. if $m_{jk}$ is greater than the expected test case length, then

    • occurrence of $S_j$ followed by $S_k$ is expected to require multiple sequences

   • note: in figure of next slide the diagonal is vector $x$
Source entropy of usage chain

- the source entropy quantifies the uncertainty in a stochastic source

- the entropy of a random variable $f$ is the expected “surprise” of the event that $f(x) = y$

$$H = - \sum_i \pi_i \sum_j p_{ij} \log p_{ij}$$

- again $\pi_i$ is the usage distribution and $p_{ij}$ is the transition probability

- $H$ is exponentially related to the number of sequences that are “statistically typical” of the Markov chain

- a Markov chain has a set of typical sequences whose ensemble statistics closely match the statistics of the chain

[Whi93, fig.6]

The mean first passage matrix for the example usage model (entries are rounded)
• Source entropy of usage chain

• high H
  • => exponentially greater number of typical sequences
  • more sequences exist because of the uncertainty present in the model
  • => Markov chain must generate more sequences in order to accurately describe the Markov source

• source entropy serves as a comparative measure for chains with same structure but different probabilities
  • example: two chains $U_1$ and $U_2$ (chains are structurally the same)
    • transition probabilities of $U_1$ are *uninformed*
    • transition probabilities of $U_2$ are *informed*
    • Let $H_1$ and $H_2$ be the source entropies for $U_1$ and $U_2$ respectively
    • If $H_1 > H_2$ then one should expect exponentially greater number of sequences using $U_1$ than $U_2$
    • $U_1$ could serve as frame of reference
    • in previous example $H_1 = 1.0884$ and $H_2 = 0.8711$
• Conclusions
  • Usage chains are a good tool trying to answer the question
    “What is the user likely to do when using the software?”
  or
    “What is the software to be able of doing?”
  • The paper was written to aid testing of software, not with
    survivability in mind
  • We need to determine how usage models can be used to
    benefit our “survivability” cause, e.g.,
    • How can we use usage models to define normal usage of the
      system?
    • How can we reverse-engineer usage patterns?
    • How can an attacker take advantage of usage models?