KarDo: Configuration Independent Automation by Non-Experts

Nate Kushman & Dina Katabi
Users constantly run into computer tasks they don’t know how to do
Sync with iPod
Enable security on wireless router
Defragment hard drive
Tether to Blackberry
Set up VPN
Today, we repeatedly solve the same computer problems

Why?

No easy way to automate across machine configurations!
KarDo

- Observes users actions as they perform a task
- Produces solution that works across configurations
A Naïve Strawman

Collects a separate trace for every possible configuration

But ...
Space of Configurations is Huge

Different Apps

Different App configuration

Different default settings

A task has multiple steps, and multiple options per step
Unlikely to have more than a few traces for the majority of tasks

How do we handle configuration diversity with just a few traces per task?
KarDo Can Generalize Using a Few Traces

Turn on out-of-office e-mails in Outlook

Trace

1. Click Outlook icon.
2. From menu → out of office.
3. Click away Java updater.
4. Out of office.
KarDo Can Generalize Using a Few Traces

Turn on out-of-office e-mails in Outlook

Trace 1:

1. Click on the Outlook icon.
2. From menu → out of office.
3. Click away Java updater.
4. Out of office.

Automation machine?
KarDo Can Generalize Using a Few Traces

Turn on out-of-office e-mails in Outlook

Trace: click away Java updater

From menu → out of office

Different Task: change view

Automation machine

Need far fewer traces!
KarDo Can Generalize Using a Few Traces

Turn on out-of-office e-mails in Outlook

- Click Outlook icon
- From menu → out of office
- Click away Java updater
- Turn on out-of-office emails in Outlook

Automation machine

No java updates
KarDo Can Generalize Using a Few Traces

Turn on out-of-office e-mails in Outlook

From menu → out of office

Allows us to remove unnecessary dependencies
Contributions

- A system that automates across configurations, with a few traces per task
- Requires no kernel or app modifications
- Used it to automated tasks from MS Help & eHow
**E-mail**

- Turn off E-mail Read Receipts
- Automatically forward e-mail to another address
- Restore the unread mail folder
- Highlight all messages sent only to me
- Change an e-mail filtering rule
- Add an e-mail filter rule
- Make the recipient column visible in the Inbox
- Order e-mail message by sender
- Create an Outlook Search Folder
- Turn on threaded message viewing in Outlook
- Mark all messages as read
- Automatically empty deleted items folder
- Empty junk e-mail folder
- Turn off Junk e-mail filtering
- Consider people e-mailed to be safe senders
- Send an e-mail with a receipt request
- File Outlook contacts by last name
- Set Outlook to start in Calendar mode
- Add a new RSS feed
- Change the Name of an RSS feed
- Turn off Outlook Desktop Alerts
- Reduce the size of a .pst file
- Turn off notification sound
- Switch calendar view to 24-hour clock

**Networking**

- Enable firewall exceptions
- Enable Windows firewall
- Disable Windows firewall notifications
- Disable Windows firewall
- Disable IPv6 to IPv4 tunnel
- Show the current IPv4 routing table
- Show the current IPv6 routing table
- Use OpenDNS
- Stop caching DNS replies
- Use Google’s public DNS server
- Use DNS server from DHCP
- Configure system to pick routes based on link speed
- Set routing interface metric

**System**

- Analyze hard drive for errors
- Defragment hard drive
- Enable Automatic Updates
- Set Up Remote Desktop
- Hide the Outlook icon in the System tray
- Change to Classic UI
- Delete an Item from the Task Bar
- Change desktop background color
- Enable Accessibility Options
- Auto-Hide the Taskbar
- Change date to Long Format
- Set Visual Effects for Performance
- Set Outlook as default E-mail program
- Enable Password on Screen Saver

**Web**

- Browser Install Firefox
- Configure SSL proxy
- Set Default Http Proxy

**Office**

- Delete a worksheet in Excel
- Turn on AutoSave in Excel
- Disable add-ins in Word
Contributions

• A system that automates across configurations, with a few traces per task

• Requires no kernel or app modifications

• Used it to automated tasks from MS Help

• Using only 2 traces per task,
  – KarDo automates 84% of configurations
  – A baseline that tries both traces automates only 18% of configurations
KarDo

Tracing User Actions → Generalizing to a Canonical Solution → Replay
Challenge:
OS gives us only mouse clicks and keypresses which are meaningless for other machines

Can’t rely on windowing layer:
- many applications use custom widget libraries
Solution:  

Accessibility Interface

For each widget:

- Type: check box
- Value: checked
- Location: x1,y1, x2,y2
- Text: “Enable IMAP”
- Window Hierarchy: in view V, in window W,...

Uniquely identify each widget

Map mouse clicks & key-presses to GUI actions, which are meaningful across machines
KarDo

Tracing User Actions

Generalizing to a Canonical Solution

Replay
But which action is which?

**non-state modifying actions**

- **Update**
  - pending change to state
    - e.g. check a check box

- **Commit**
  - write pending updates to state
    - e.g. click “OK” button

- **Navigate**
  - make new widgets available
    - e.g. move to a new tab

**State-Modifying**
- Learn within a task

**Non-State-Modifying**
- Learn across tasks
Challenge: How do we automatically map a GUI action to Update, Commit or Navigate?
Solution: Machine Learning Classifier

Action

(Widget type
Opens window Changes state

SVM Classifier

Label

Update
Commit
Navigate
Solution:

Machine Learning Classifier

Action

(SVM Classifier)

Label

Update
Commit
Navigate

(Widget type
Opens window
Changes state
:
Text: "OK")
We can differentiate state-modifying from non-state modifying.

But, how do we generalize across configurations?
KarDo's Generalization Framework

Task 1

Task 2

Task 3
KarDo’s Generalization Framework

Task 1 -> Task 2 -> Task 3

Generalize Navigation across all tasks
KarDo’s Generalization Framework

**Generalize State Mods.**
within a task

**Generalize Navigation**
across all tasks

Automated solution

out-of-office
KarDo’s Generalization Framework

**Generalize State Mods. within a task**

**Generalize Navigation across all tasks**

How to get to this checkbox?

out-of office
KarDo's Generalization Framework

Generalize State Mods. within a task

Generalize Navigation across all tasks

How to get to this button?

out-of-office
KarDo's Generalization Framework

Generalize State Mods. within a task

Generalize Navigation across all tasks

out-of-office

OK
KarDo

- Tracing User Actions
- Generalizing to a canonical solution
- Replay
KarDo

Tracing User Actions

Generalizing Navigation Actions

Generalizing State-Modifying Actions

Replay
Generalizing Navigation
Generalizing Navigation

Global Navigation Graph

During replay, **Breadth first search** from the widget in automated solution to any widget on the screen
Generalizing State-Modifying Actions

Are all Update and Commit actions necessary?

Open Word
Copy+Paste in Word → Update
Close Word

Open Notepad
Copy+Paste into Notepad
Save in Notepad
Close Notepad

Unnecessary
Generalizing State-Modifying Actions

Are all Update and Commit actions necessary?

Open Word
Copy+Paste in Word \(\rightarrow\) Update
Close Word \(\Rightarrow\) Unnecessary

Open Notepad
Copy+Paste into Notepad \(\rightarrow\) Update
Save in Notepad \(\rightarrow\) Commit

Unnecessary Actions \(\Rightarrow\) Unnecessary Dependencies
Challenge:
How do we remove unnecessary updates and commits?

Solution Idea:
Remove any action that does not contribute to final system state.
Algorithm for Removing Unnecessary Actions

- Click **Open Dialog**
- Check **Check Box**
- Click **OK**
- Click **Open Dialog**
- Click **OK**
- Click **Open Dialog**
- UnCheck **Check Box**
- Click **Cancel**
Algorithm for Removing Unnecessary Actions

<table>
<thead>
<tr>
<th>Navigate to Dialog$_i$</th>
<th>Update (Dialog$_i$, Widget$_k$)</th>
<th>Commit (Dialog$_i$, Widget$_k$)</th>
<th>Navigate to Main</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigate to Dialog$_i$</td>
<td>Commit (Dialog$_i$, Widget$_k$)</td>
<td>Navigate to Main</td>
<td></td>
</tr>
<tr>
<td>Navigate to Dialog$_i$</td>
<td>Update (Dialog$_i$, Widget$_k$)</td>
<td>Navigate to Main</td>
<td></td>
</tr>
</tbody>
</table>
Algorithm for Removing Unnecessary Actions

Update \( (\text{Dialog}_i, \text{Widget}_k) \)
Commit \( (\text{Dialog}_i, \text{Widget}_k) \)

Commit \( (\text{Dialog}_i, \text{Widget}_k) \)
Update \( (\text{Dialog}_i, \text{Widget}_k) \)
Algorithm for Removing Unnecessary Actions

Pass 1: Unnecessary Updates
Go backwards → Eliminate updates with no commit
Removing Unnecessary Actions

Pass 1: Unnecessary Updates
Go backwards → Eliminate updates with no commit

Pass 2: Unnecessary Commits
Go forwards → Eliminate all commits w/o pending updates
Removing Unnecessary Actions

Pass 1: Unnecessary Updates
Go backwards $\rightarrow$ Eliminate updates with no commit

Pass 2: Unnecessary Commits
Go forwards $\rightarrow$ Eliminate all commits w/o pending updates

Update $(\text{Dialog}_i, \text{Widget}_k)$
Commit $(\text{Dialog}_i, \text{Widget}_k)$
Creating a Canonical Solution
A per task state-modifying graph, with if-then branches

But how do we decide which branch to take for a given configuration?
Creating a Canonical Solution

A per task state-modifying graph, with if-then branches

Dynamically evaluate branches
KarDo

Tracing User Actions

Generalizing Navigation Actions

Generalizing State-Modifying Actions

Replay
KarDo

Tracing User Actions

Generalizing Navigation Actions

Generalizing State-Modifying Actions

Replay
Replay

Global Navigation Graph

Per Task Commit/Update Graph

Navigation actions to Defragment

Navigation to next ...
Experiments
Experimental Setup

• Implemented KarDo as thin client and a server

• Tested on 57 Tasks from eHow and MS Help sites
**E-mail**
- Turn off E-mail Read Receipts
- Automatically forward e-mail to another address
- Restore the unread mail folder
- Highlight all messages sent only to me
- Change an e-mail filtering rule
- Add an e-mail filter rule
- Make the recipient column visible in the Inbox
- Order e-mail message by sender
- Create an Outlook Search Folder
- Turn on threaded message viewing in Outlook
- Mark all messages as read
- Automatically empty deleted items folder
- Empty junk e-mail folder
- Turn off Junk e-mail filtering
- Consider people e-mailed to be safe senders
- Send an e-mail with a receipt request
- File Outlook contacts by last name
- Set Outlook to start in Calendar mode
- Add a new RSS feed
- Change the Name of an RSS feed
- Turn off Outlook Desktop Alerts
- Reduce the size of a .pst file
- Turn off notification sound
- Switch calendar view to 24-hour clock

**Networking**
- Enable firewall exceptions
- Enable Windows firewall
- Disable Windows firewall notifications
- Disable Windows firewall
- Disable IPv6 to IPv4 tunnel
- Show the current IPv4 routing table
- Show the current IPv6 routing table
- Use OpenDNS
- Stop caching DNS replies
- Use Google's public DNS server
- Use DNS server from DHCP
- Configure system to pick routes based on link speed
- Set routing interface metric

**System**
- Analyze hard drive for errors
- Defragment hard drive
- Enable Automatic Updates
- Set Up Remote Desktop
- Hide the Outlook icon in the System tray
- Change to Classic UI
- Delete an Item from the Task Bar
- Change desktop background color
- Enable Accessibility Options
- Auto-Hide the Taskbar
- Change date to Long Format
- Set Visual Effects for Performance
- Set Outlook as default E-mail program
- Enable Password on Screen Saver

**Office**
- Delete a worksheet in Excel
- Turn on AutoSave in Excel
- Disable add-ins in Word

**Web**
- Browser Install Firefox
- Configure SSL proxy
- Set Default Http Proxy
Experimental Setup

• Implemented KarDo as thin client and a server
• Tested on 57 Tasks from eHow and MS Help sites
• 20 diversely configured VMs
  – 10 training VMs and 10 test VMs
• Each task performed manually on 2 training VMs
Testing

- Given 2 traces, automate using
  - KarDo
  - Baseline that runs both traces and succeeds if either automates the task

- Test each solution on the 10 test VMs
Automation Success Rate

Percentage of VM-Task Pairs

Baseline

KarDo

84%
Automation Success Rate

Baseline: 18% success
Automation Success Rate

KarDo: 84% success  
Baseline: 18% success

% Successful VMs

(Ordered) Task Id
Conclusion

• KarDo automates tasks across configurations based on just a few traces

• Using two traces it successfully automated MS and eHow tasks on 84% of configurations

• Applicable to a wide variety of problems:
  – Automated Helpdesk, Automation of repetitive tasks, Automated GUI testing, etc.