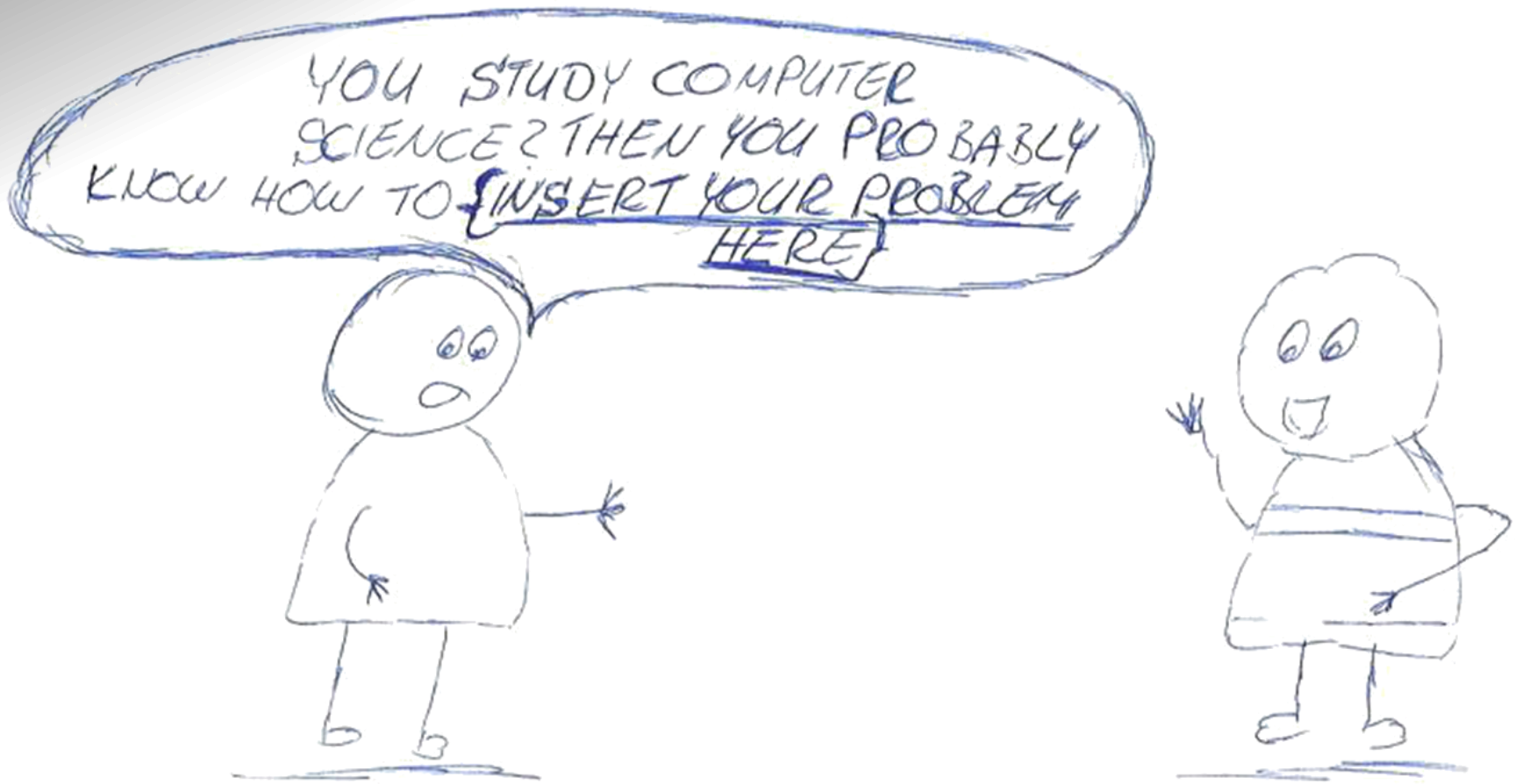
Where is that Button Again?! – Towards a Universal GUI Search Engine

Sven Hertling, Markus Schröder, Christian Jilek, Andreas Dengel

ICAART 2017

Motivation

- Typical scenario for a computer science researcher:

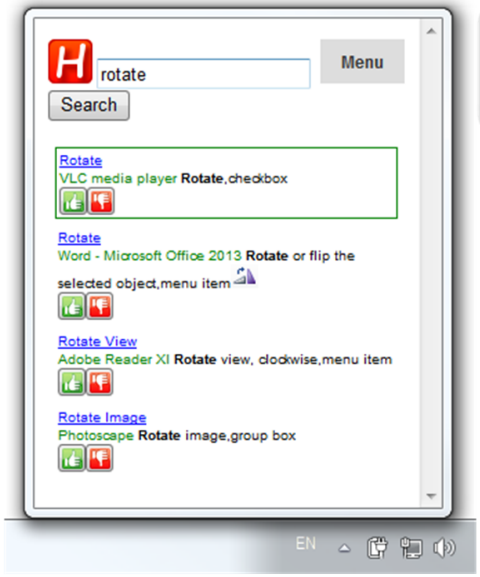


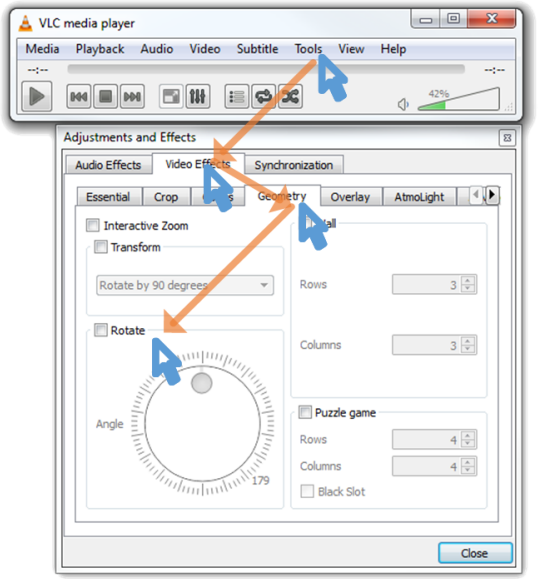
Motivation

- GUIs contain high amount of features
  - spread across
    - Menus
    - dialog windows
    - tool bars
    - etc.
- Remembering where to find each feature is usually very hard
  - especially if not regularly used

Motivation

- One possible solution:
  - Search engine for elements of GUIs
    1. user clicks on a search result
    2. appropriate mouse and keyboard actions are performed
    3. the element is visible on screen





Overview

- Approach
  - Software basis
  - GUI Analysis
  - Interpreting User Queries
  - Solution Execution
- Evaluation
- Conclusion

Software basis

- good sampling of most often used software for approach and evaluation
- analysed 11 download sites
  - out of 35 ranked by

    web traffic analyzers
    - Alexa
    - Compete
    - Semrush(en/de)
    - PageRank

amazon.com	amazon.de
cnet.com	sourceforge.net
softonic.com	zdnet.com
chip.de	computerbild.de
heise.de	netzwelt.de
filehippo.com	

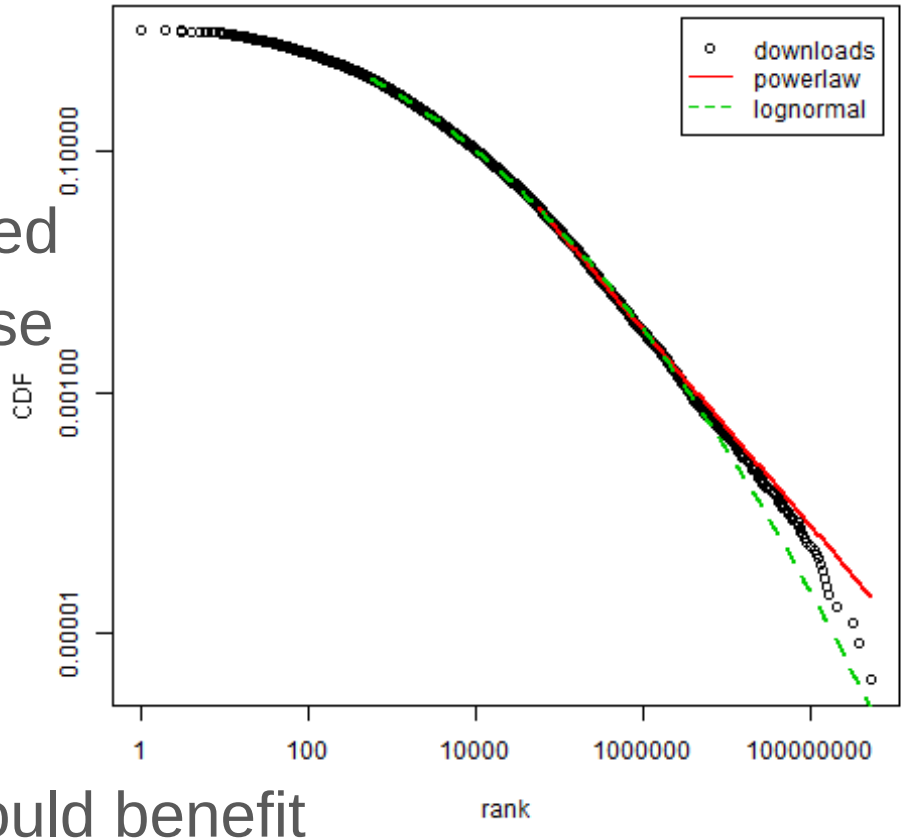
Software basis

- chose software which is among the top ten in the ranking of at least two sites
- after removal of
  - Flash Player
  - Minecraft

Word 2013 (Trial)	Excel 2013 (Trial)
PowerPoint 2013 (Trial)	Open Office Writer
Open Office Calc	Open Office Impress
VLC media player	CCleaner
Google Chrome	Mozilla Firefox
7-Zip	WinRAR
Skype	PhotoScape
AntiVir 2014 - Avira	NortonInternetSecurity
Adobe Reader	avast! Free Antivirus

Software basis

- analysis of the download count of cnet.com
- log-log plot
  - only few software tools which are often downloaded
  - and a large number of those that are demanded less frequently
- appropriate selection of applications
  - a huge amount of users could benefit

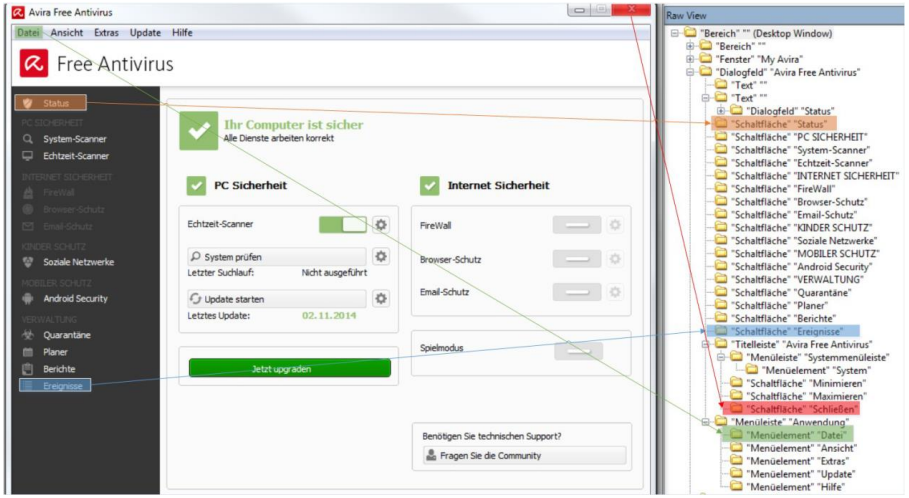


GUI Analysis

- utilize accessibility interfaces
  - typically also used by screen readers
  - Microsoft Active Accessibility (MSAA)
  - Microsoft UI Automation (UIA)
  - Not supported by all apps
    - “Norton Internet Security” and “avast! Free Antivirus
    - (2 of 18 applications or 11.11 % of the sample set)

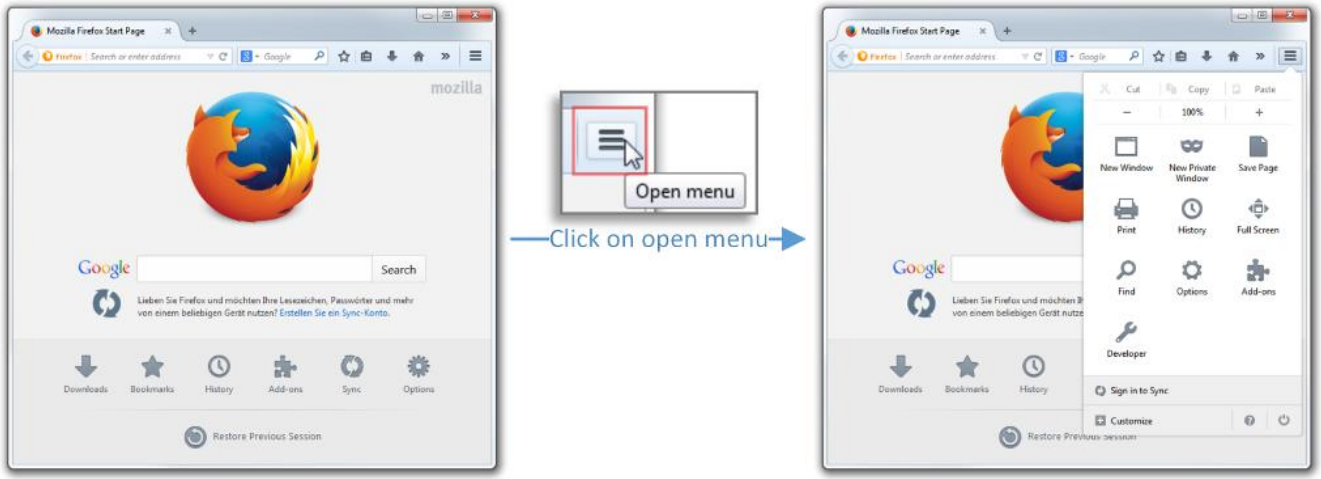
GUI Analysis

- tree of currently visible UI elements
- observe GUI interactions by
  - users (getting GUI usage behavior)
  - click monkey (automatical exploration)
    - clicking all elements (depth-limited traversal strategy)
    - more complete software model



GUI Analysis

- observer capture UI interactions with
  - the causing action
  - an application software's GUI tree
  - the graphical element corresponding to the current cursor location
- software model based on detailed interaction log



Interpreting User Queries

- searchable GUI model using names and metadata of UI elements
  - Field “UI text”:
    Name, LegacyIAccessibleDescription,HelpText and LocalizedControlType
  - ProductName, CompanyName, FileDescription of the corresponding software
- 3 approaches for searching
  - baseline
  - language
  - context

Interpreting User Queries

- Baseline: StandardAnalyzer of Lucene
- Language: should clause query of “baseline” and three “language” fields
  1. standard tokenizer, HTMLStripCharFilter, lowercase, synonyms, ASCIIFolding, stopword removal, language dependent filter like GermanNormalizationFilter, stemming
  2. standard tokenizer,lowercase, NGramTokenFilter (min:1, max:20)
  3. same as (2) but with EdgeNGramTokenFilter

  – increase recall but keep precision

Interpreting User Queries

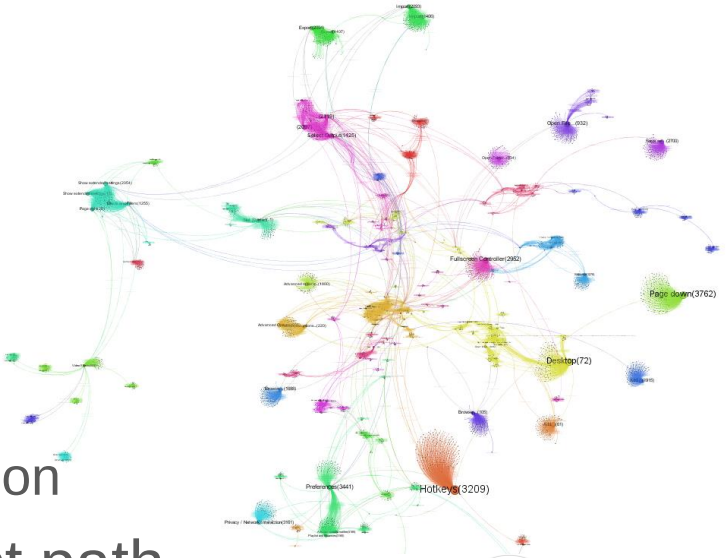
- Context: re-rank results of “Language“ considering desktop context

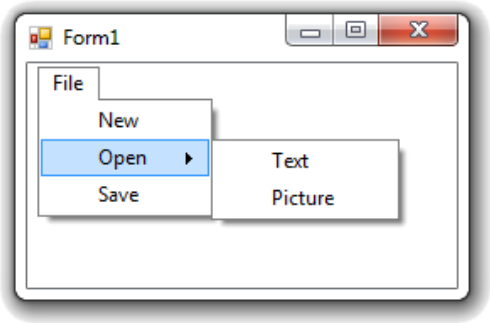
- all approaches uses TF-IDF
  - Okapi BM25 only marginally different

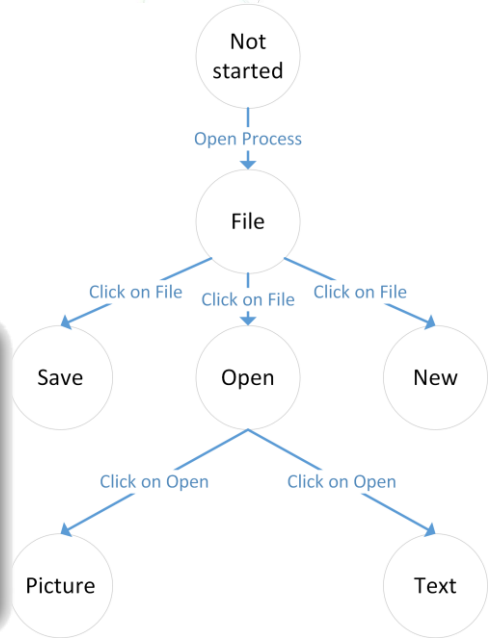
- other approaches could be also used
  - QF-graphs (Fourney et al., 2011)
  - CommandSpace (Adar et al., 2014)

Solution Execution

- graph-based software model
  - directed, weighted graph
    - vertices = UI elements
    - edges = interactions
    - edge weight = reliability of interaction
  - navigation is reduced to shortest path
  - an edge encodes what element causes another one's occurrence
    - "Show New Element Graph" (SNEG)







Solution Execution

- users still navigate though the GUI on their own

- Automatic execution needs
  - a reliable recovery of graphical elements
  - the execution of the interactions

Solution Execution

- UI element recovering:
  - all elements of start screen of 16 selected software tools
  - 714 elements which should clearly be differentiated
  - in GUI testing, identification characteristics determined at test recording time

Property name	ID	Count of equi-valence classes
~~RuntimeId~~	~~30000~~	~~583~~
~~BoundingRectangle~~	~~30001~~	~~570~~
Name ~~LegacyIAccessibleName~~	30005 ~~30092~~	299
~~ProviderDescription~~	~~30107~~	~~156~~
~~NativeWindowHandle~~	~~30020~~	~~139~~
HelpText ~~LegacyIAccessibleHelp~~	30013 ~~30097~~	100
AutomationId	30011	92
AccessKey ~~LegacyIAccessible-KeyboardShortcut~~	30007 ~~30098~~	66
ClassName	30012	62
LegacyIAccessible-Description	30094	45
~~LegacyIAccessibleState~~	~~30096~~	~~35~~
LegacyIAccessibleRole	30095	32
LocalizedControlType	30004	31
~~LegacyIAccessibleValue~~	~~30093~~	~~28~~
ControlType	30003	27
~~ValueValue~~	~~30045~~	~~25~~
~~ProcessId~~	~~30002~~	~~15~~
~~LegacyIAccessible-DefaultAction~~	~~30100~~	~~13~~
LegacyIAccessibleChildId	30091	10

Evaluation

- user study with 10 participants
  - 5 male, 5 female, age 35.2 ±16.3
  - computer usage: 3.3 ± 0.67
    - “almost no use” (1) to “very frequent use” (4)
- for each software (16 apps)
  - a fictional task
  - corresponding pictogram
  - participant is asked if the software is frequently used (expert)or if it is only rarely used (novice)
- afterwards all participants filled out a user experience questionnaire (UEQ)

Evaluation

- evaluation setup for each task
  1. task's pictogram,related program's icon and name were displayed (simulation of an intention)
  2. participant had to formulate a textual query describing the task
     a) judge randomized list of top 15 graphical elements
  3. participant searches for the graphical element on their own
  4. GUI Search Engine is used to find the UI element

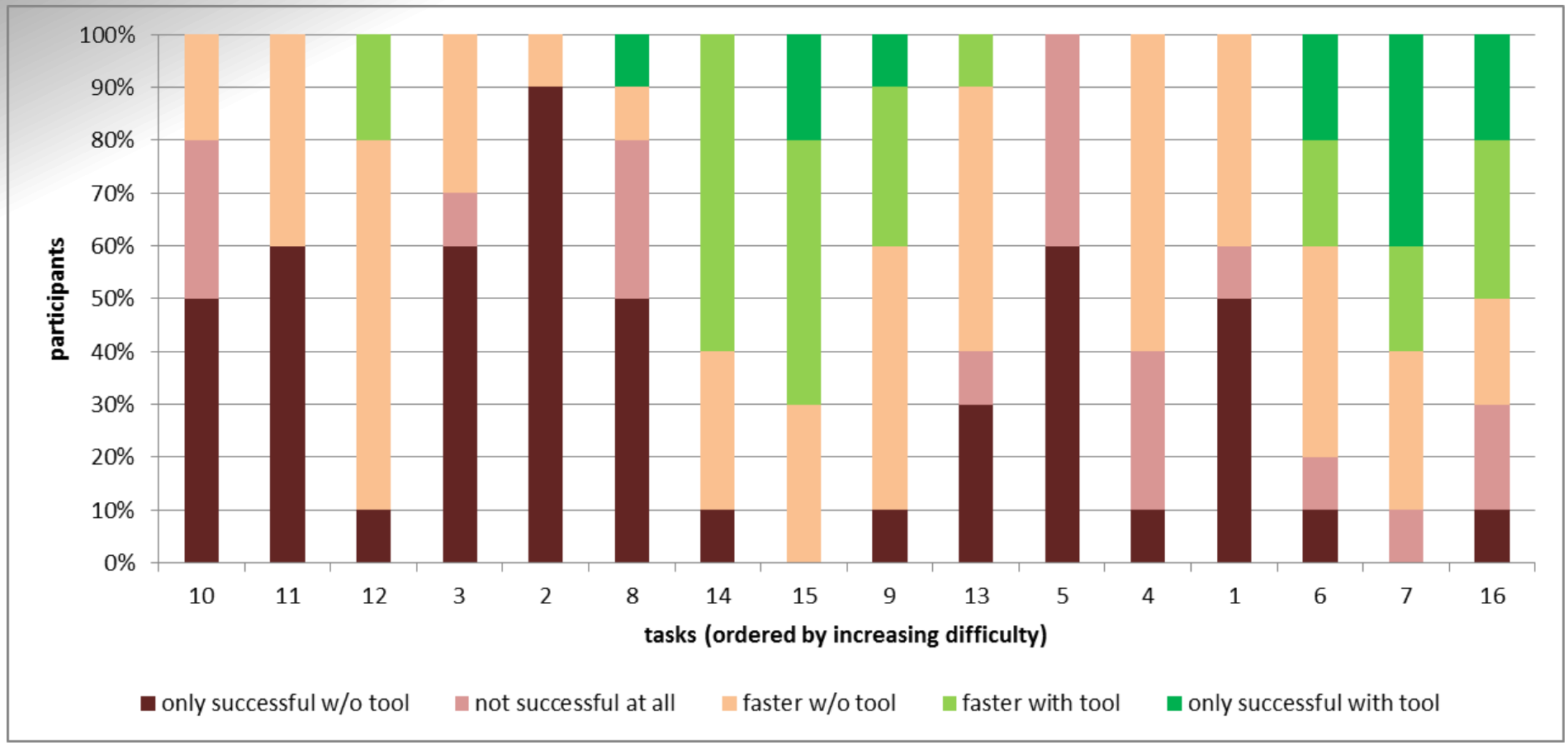
Evaluation

- Users can mainly benefit in two ways
  - GUI search engine may find the UI elements faster
  - GUI search engine may retrieve results when the user has already given up manual search

		User with tool		
		successful	failed	Total
User w/o tool	successful	33,1%	15,0% (tool faster)	31,9%	80%
	failed	7,5%	12,5%	20%
	Total	55,6%	44,4%	100%

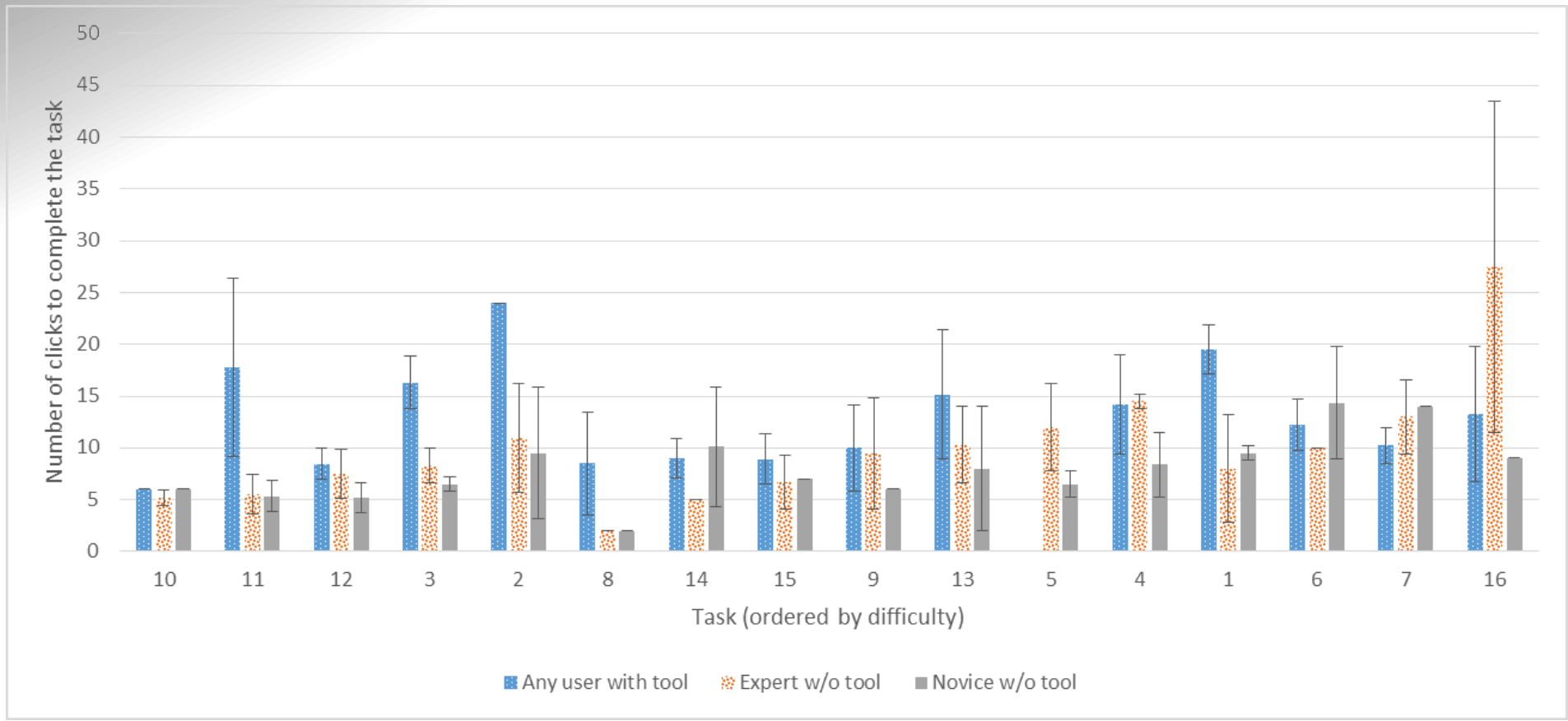
Evaluation

- detailed view (task difficulty = average user time)



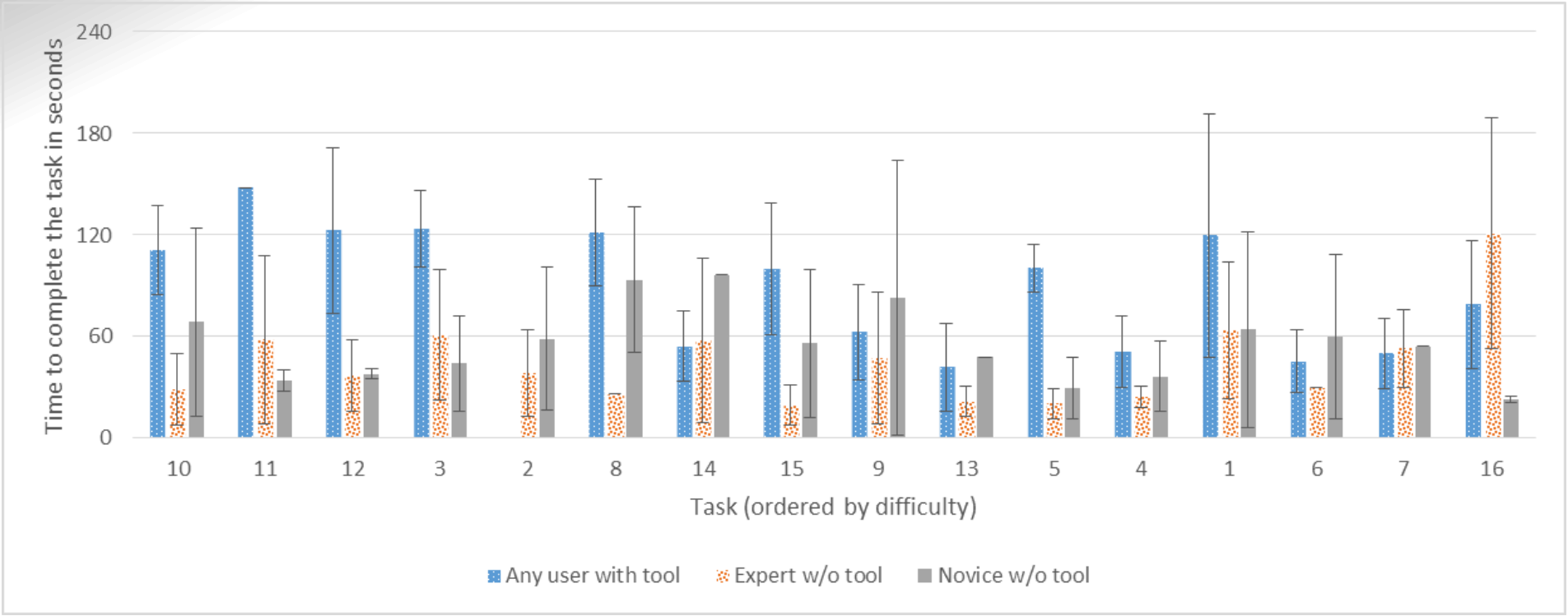
Evaluation

- average number of clicks to complete each task



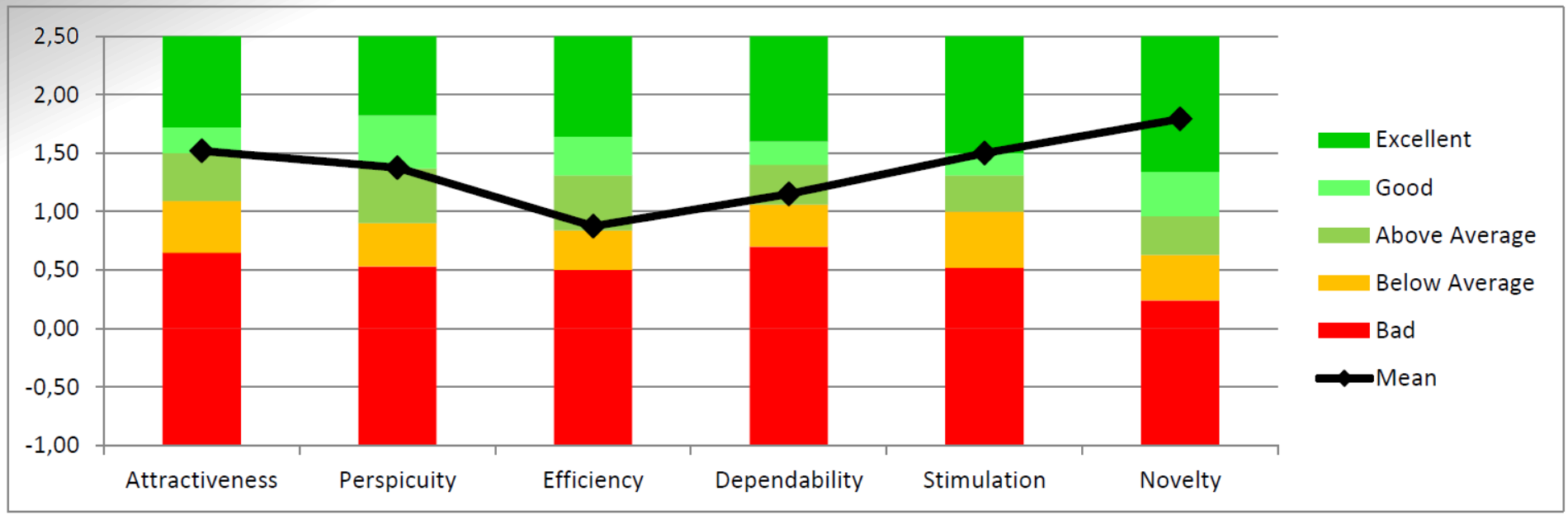
Evaluation

- average time in seconds to complete each task



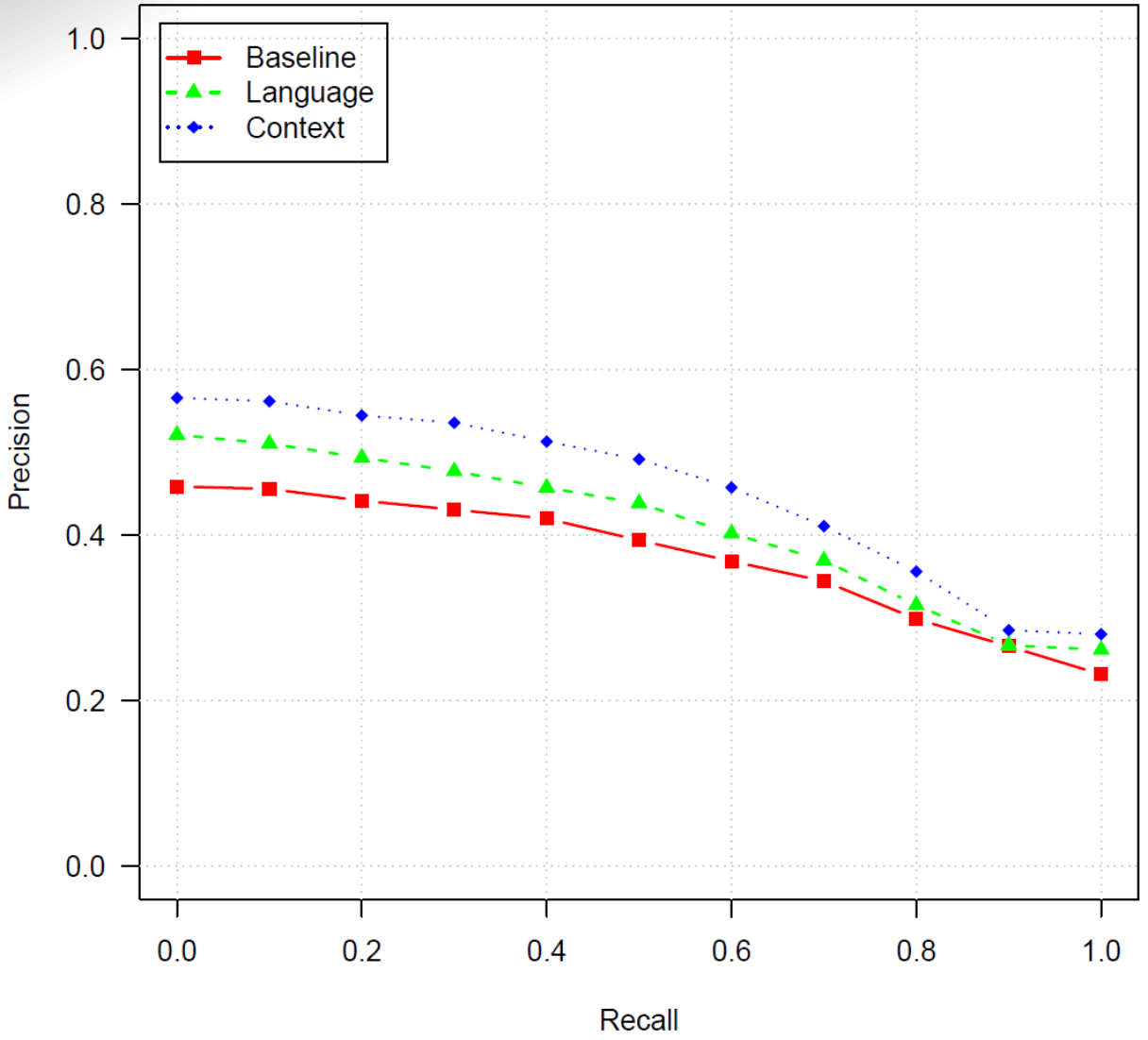
Evaluation

- user experience questionnaire (UEQ)



Evaluation

- recall-precision-curve based on relevance feedback in the user study



Conclusion

- presented the first universal GUI search engine
- works for 16 of 18 very frequently used programs
  - especially features which are not used regularly
- future plans:
  - in-application tutoring
  - further exploit GUI usage behavior (share expert knowledge)

Thank you

- questions, opinions, suggestions, discussions

?