A Qualitative Evaluation of Reverse Engineering Tool Usability

James Mattei
Tufts Security & Privacy Lab

Overview

1. **Background and Motivations**
2. **The Data**
3. **Qualitative Coding**
4. **Results**
5. **Discussion**

1.

Background and Motivations

Vulnerability discovery is important but difficult

- Identifying vulnerabilities in code is predominantly a manual process
- The process is slow and time consuming
  - Reverse engineers (RE) take on average 40 minutes to review ~150 lines of code [Yakdan, 2016]

Example Call Graph -
https://i.stack.imgur.com/Mg9mR.png

How do we improve this problem?

Train more reverse engineers

- Improve the process of educating new RE

Help current engineers work more efficiently

- Identify what tools are usable and how REs interact with them

How do we improve this problem?

Train more reverse engineers

- Improve the process of educating new RE

Help current engineers work more efficiently

- Identify what tools are usable and how REs interact with them

Evaluate what plugins REs are creating

RQ1 What are current interaction modalities for RE tools?

- Determine how REs interact with their tools
- Find out what interaction modalities are associated with expected usability

RQ2 Do they fit the RE's processes and mental models?

- Do RE tools follow usability guidelines?

2.
The Data

Identify what frameworks SEs are using

BINARYNINJA

Sourcing the user created plugins

- Edge of the Art in Vulnerability Research -Two Six Labs
- Google
- Twitter
- Github

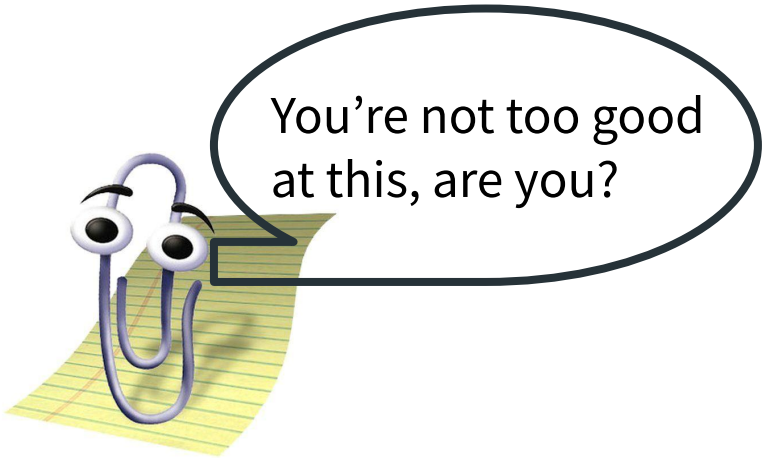
Ghidra	75 plugins
IDA	204 plugins
Binary Ninja	78 plugins
Radare2	50 plugins
Standalone	59 tools
Total	**466 tools**

Exclusion Criteria

- Utility focused tools with functionality outside the RE mental model
- Tools which add support for additional instruction set architectures

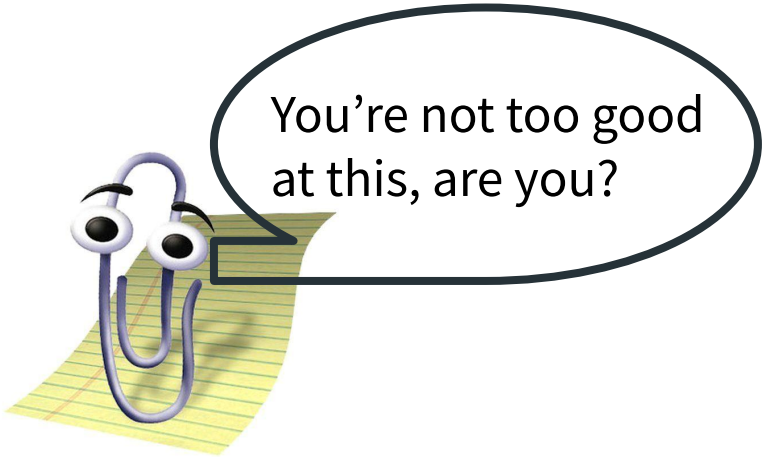
Exclusion Criteria

- Utility focused tools with functionality outside the RE mental model
- Tools which add support for additional instruction set architectures



Exclusion Criteria

- Utility focused tools with functionality outside the RE mental model



- Tools which add support for additional instruction set architectures

Final totals

Ghidra	75 plugins **(56 used)**
IDA	204 plugins **(101 used)***
Binary Ninja	78 plugins **(48 used)**
Radare2	50 plugins **(34 used)**
Standalone	59 tools **(50 used)**
Total	**466 tools (289 used)**

3.
Qualitative Coding

Evaluate what plugins REs are creating

RQ1 What are current interaction modalities for RE tools?

- Determine how REs interact with their tools
- Find out what interaction modalities are associated with expected usability

RQ2 Do they fit the RE's processes and mental models?

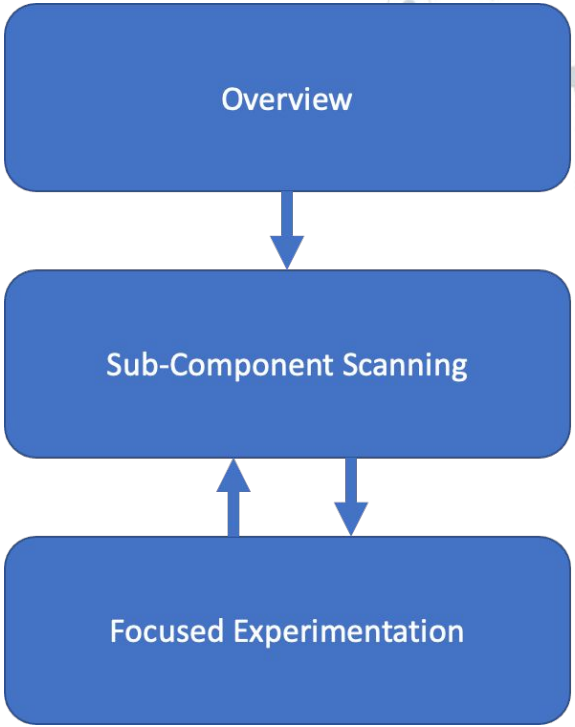
- Do RE tools follow usability guidelines?

Previous work: Reverse Engineering Process

Establish a broad view of the program, develop initial hypotheses and questions for further investigation.

Review relevant sub-components to refine hypotheses and questions

Test hypotheses with in-depth static / dynamic analysis



An Observational Investigation of Reverse Engineers' Processes. [Votipka, 2020]

Develop codebook to classify plugins

Codebook Entry	Potential Values	Significance
Analysis Phase (G1)	Overview / Subcomponent/ Experimentation	Which analysis phase is the intended use of the plugin (RQ1, RQ2)
Input Content (G2)	Binary file/ Selected area/ etc.	What input options does the tool have (RQ1)
Output Content (G2)	Function signatures, Emulated code, etc.	What information is the plugin presenting (RQ1)
Output Method (G2)	Code viewer, Console log, etc.	How is the information presented (RQ1)
Stat. & Dyn. (G3)	Yes / No	Does the tool incorporate different contexts (RQ2)
User Spec (G4)	Yes / No	Does the tool allow more user control (RQ2)
Readability (G5)	Yes / No	Does the tool make code comprehension easier (RQ2)
Functionality Type	Scanning, Fuzzing, etc.	What are the tools used for (RQ1, RQ2)

Develop codebook to classify plugins

Codebook Entry	Potential Values	Significance
Analysis Phase (G1)	Overview / Subcomponent/ Experimentation	Which analysis phase is the intended use of the plugin (RQ1, RQ2)
Input Content (G2)	Binary file/ Selected area/ etc.	What input options does the tool have (RQ1)
Output Content (G2)	Function signatures, Emulated code, etc.	What information is the plugin presenting (RQ1)
Output Method (G2)	Code viewer, Console log, etc.	How is the information presented (RQ1)
Stat. & Dyn. (G3)	Yes / No	Does the tool incorporate different contexts (RQ2)
User Spec (G4)	Yes / No	Does the tool allow more user control (RQ2)
Readability (G5)	Yes / No	Does the tool make code comprehension easier (RQ2)
Functionality Type	Scanning, Fuzzing, etc.	What are the tools used for (RQ1, RQ2)

Develop codebook to classify plugins

Codebook Entry	Potential Values	Significance
Analysis Phase (G1)	Overview / Subcomponent/ Experimentation	Which analysis phase is the intended use of the plugin (RQ1, RQ2)
Input Content (G2)	Binary file/ Selected area/ etc.	What input options does the tool have (RQ1)
Output Content (G2)	Function signatures, Emulated code, etc.	What information is the plugin presenting (RQ1)
Output Method (G2)	Code viewer, Console log, etc.	How is the information presented (RQ1)
Stat. & Dyn. (G3)	Yes / No	Does the tool incorporate different contexts (RQ2)
User Spec (G4)	Yes / No	Does the tool allow more user control (RQ2)
Readability (G5)	Yes / No	Does the tool make code comprehension easier (RQ2)
Functionality Type	Scanning, Fuzzing, etc.	What are the tools used for (RQ1, RQ2)

Develop codebook to classify plugins

Codebook Entry	Potential Values	Significance
Analysis Phase (G1)	Overview / Subcomponent/ Experimentation	Which analysis phase is the intended use of the plugin (RQ1, RQ2)
Input Content (G2)	Binary file/ Selected area/ etc.	What input options does the tool have (RQ1)
Output Content (G2)	Function signatures, Emulated code, etc.	What information is the plugin presenting (RQ1)
Output Method (G2)	Code viewer, Console log, etc.	How is the information presented (RQ1)
Stat. & Dyn. (G3)	Yes / No	Does the tool incorporate different contexts (RQ2)
User Spec (G4)	Yes / No	Does the tool allow more user control (RQ2)
Readability (G5)	Yes / No	Does the tool make code comprehension easier (RQ2)
Functionality Type	Scanning, Fuzzing, etc.	What are the tools used for (RQ1, RQ2)

Develop codebook to classify plugins

Codebook Entry	Potential Values	Significance
Analysis Phase (G1)	Overview / Subcomponent/ Experimentation	Which analysis phase is the intended use of the plugin (RQ1, RQ2)
Input Content (G2)	Binary file/ Selected area/ etc.	What input options does the tool have (RQ1)
Output Content (G2)	Function signatures, Emulated code, etc.	What information is the plugin presenting (RQ1)
Output Method (G2)	Code viewer, Console log, etc.	How is the information presented (RQ1)
Stat. & Dyn. (G3)	Yes / No	Does the tool incorporate different contexts (RQ2)
User Spec (G4)	Yes / No	Does the tool allow more user control (RQ2)
Readability (G5)	Yes / No	Does the tool make code comprehension easier (RQ2)
Functionality Type	Scanning, Fuzzing, etc.	What are the tools used for (RQ1, RQ2)

Develop codebook to classify plugins

Codebook Entry	Potential Values	Significance
Analysis Phase (G1)	Overview / Subcomponent/ Experimentation	Which analysis phase is the intended use of the plugin (RQ1, RQ2)
Input Content (G2)	Binary file/ Selected area/ etc.	What input options does the tool have (RQ1)
Output Content (G2)	Function signatures, Emulated code, etc.	What information is the plugin presenting (RQ1)
Output Method (G2)	Code viewer, Console log, etc.	How is the information presented (RQ1)
Stat. & Dyn. (G3)	Yes / No	Does the tool incorporate different contexts (RQ2)
User Spec (G4)	Yes / No	Does the tool allow more user control (RQ2)
Readability (G5)	Yes / No	Does the tool make code comprehension easier (RQ2)
Functionality Type	Scanning, Fuzzing, etc.	What are the tools used for (RQ1, RQ2)

Limitations

- The scope of this work focuses on what tools exist within our usability metrics, not the **effectiveness** of certain tools
  - Future work will evaluate different interaction modalities with regards to improving RE workflow

- We attempted to use github stargazers to measure tool popularity. Interviews of REs would need to be conducted to gather use data

4.
Results

Evaluate what plugins REs are creating

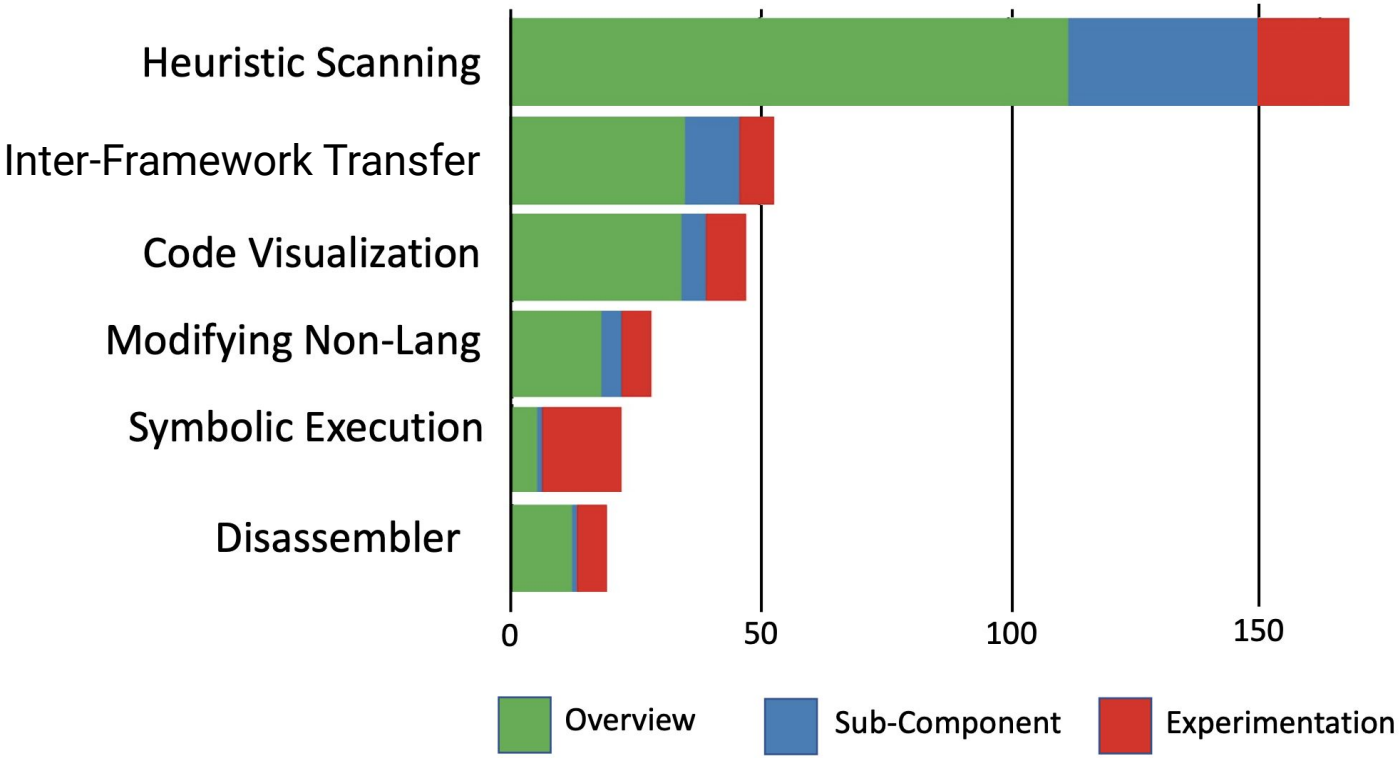
RQ1 What are current interaction modalities for RE tools?

- Determine how REs interact with their tools
- Find out what interaction modalities are associated with expected usability

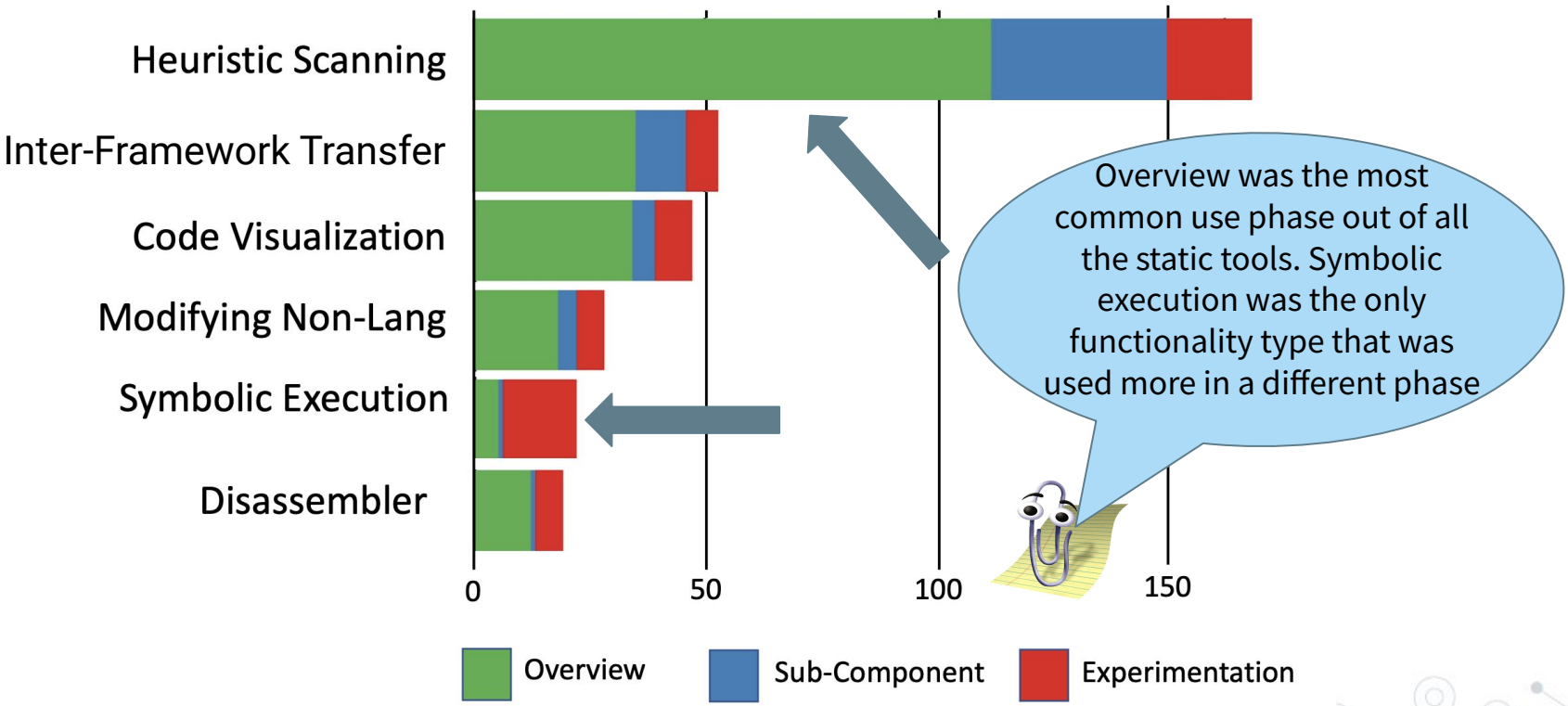
RQ2 Do they fit the RE's processes and mental models?

- Do RE tools follow usability guidelines?

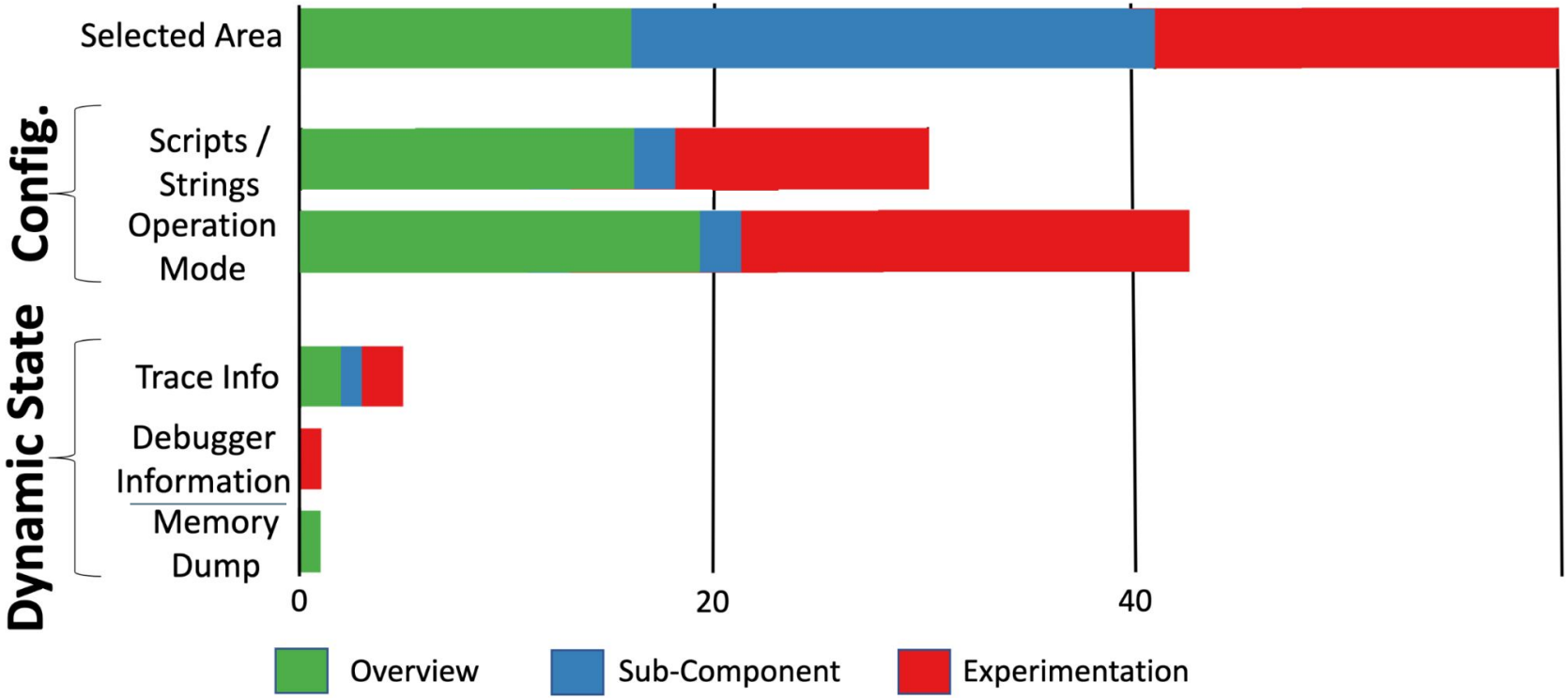
Analysis phase breakdown by static func. type



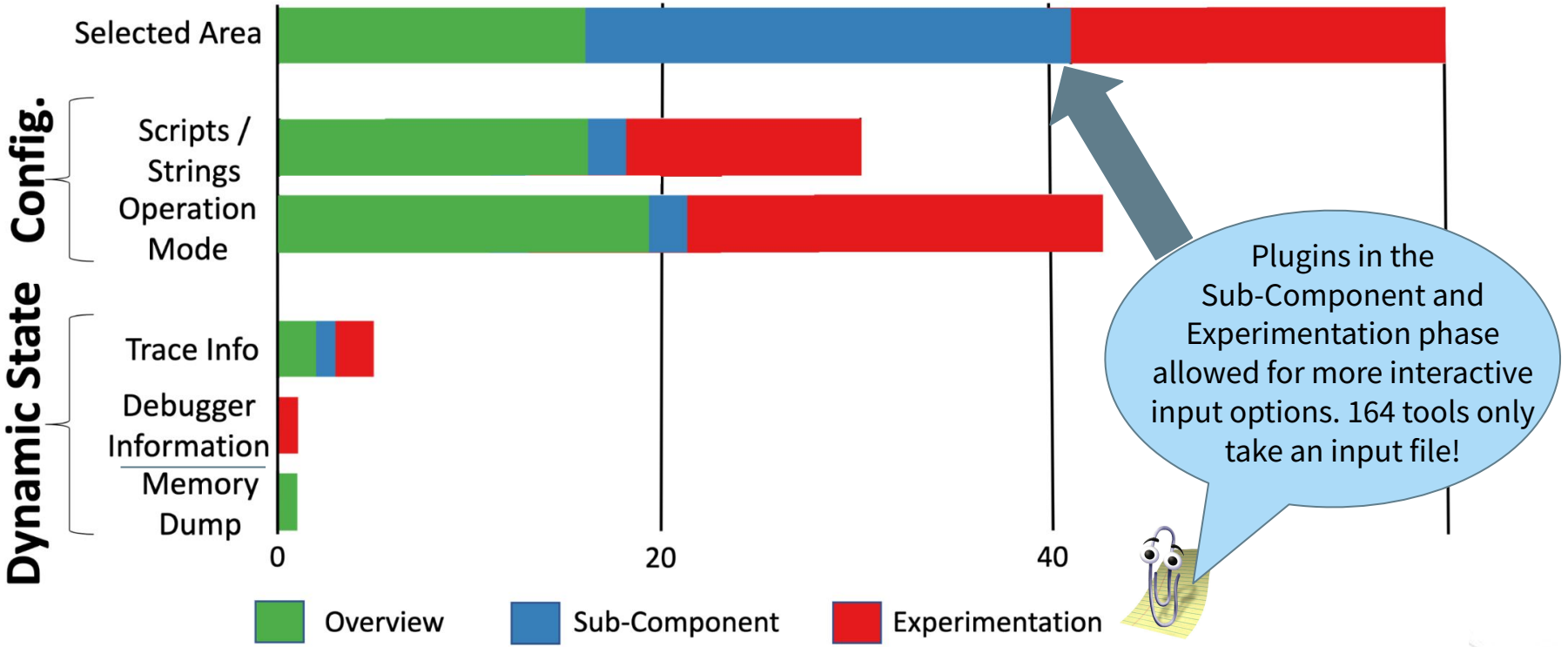
Analysis phase breakdown by static func. type



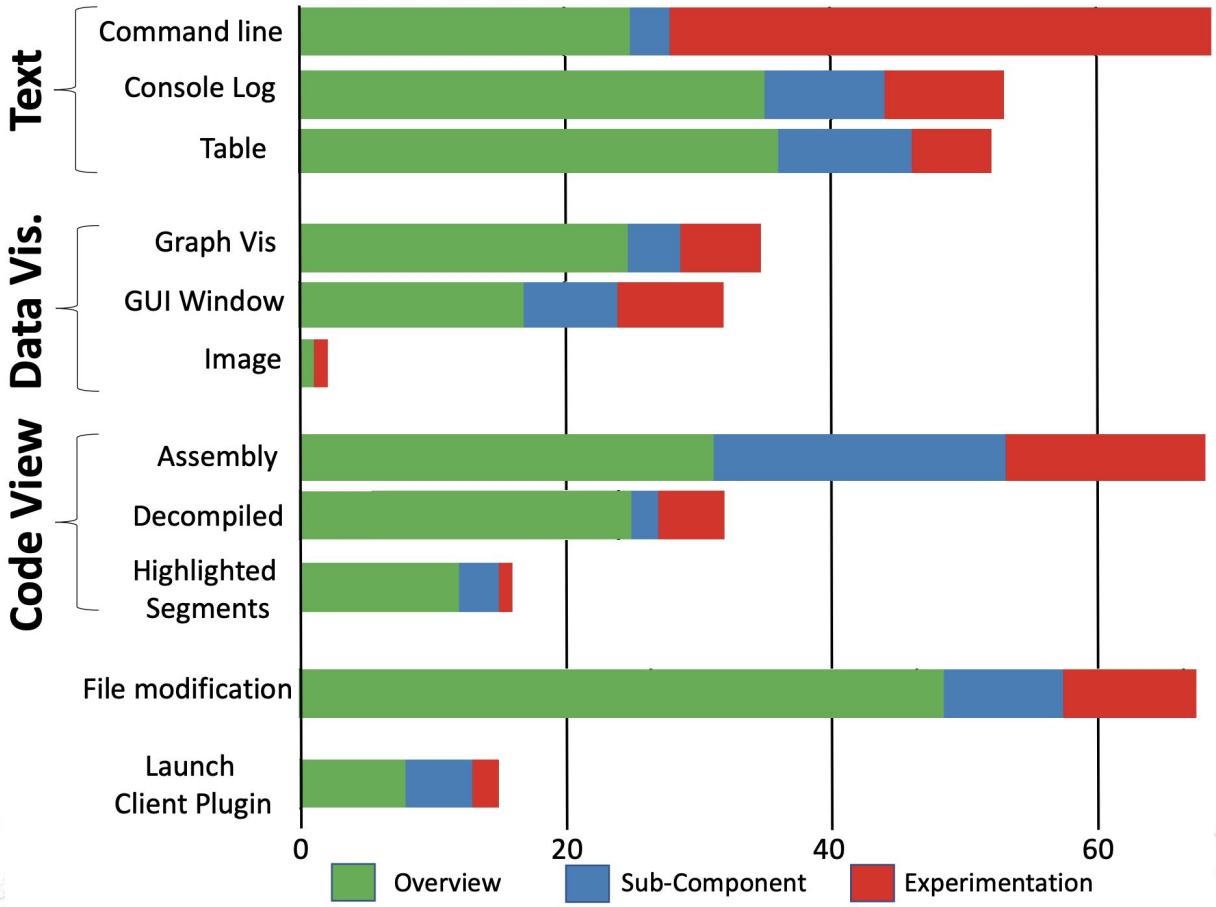
Input types by analysis phase



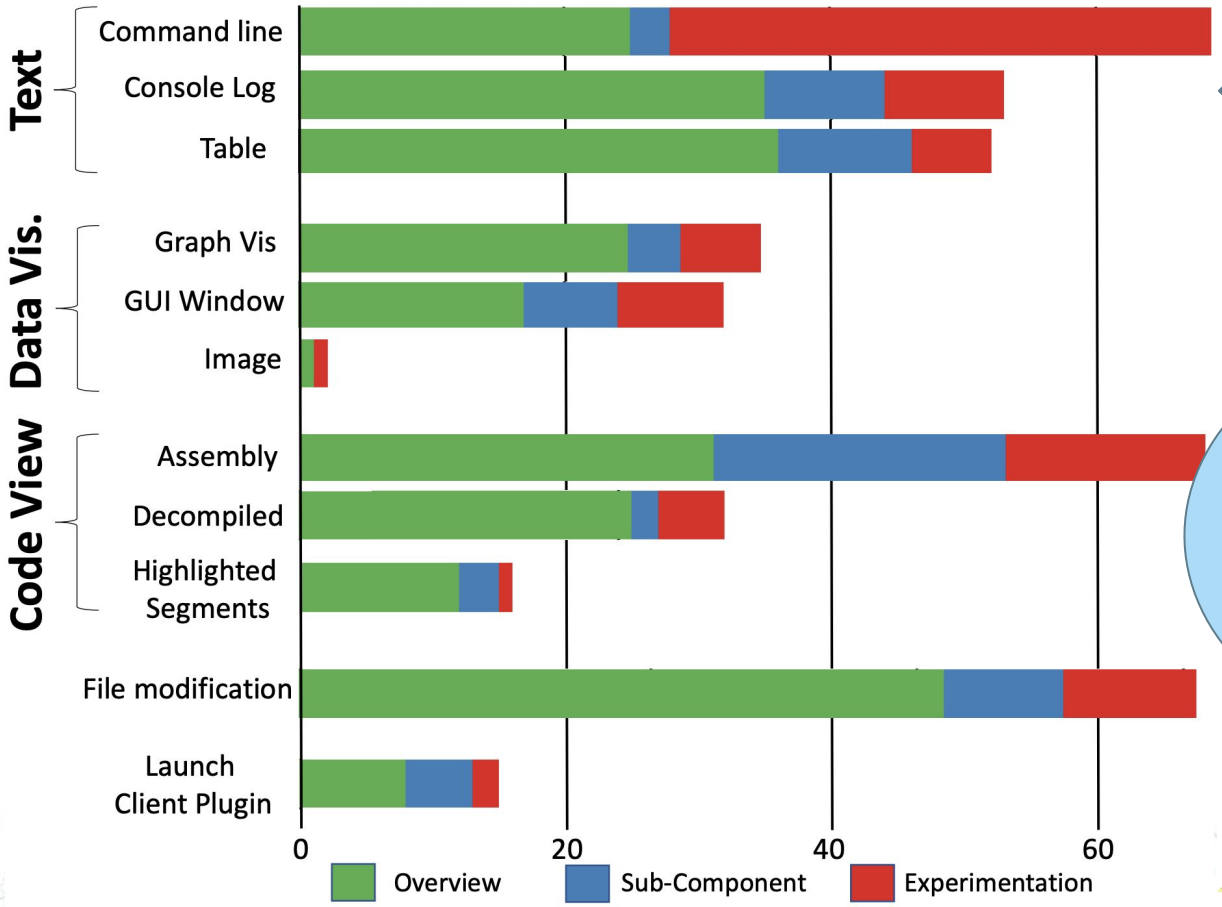
Input types by analysis phase



Output methods by analysis phase



Output methods by analysis phase



Despite many plugins having a GUI they can take advantage of, most present output only as text. However static tools are more likely to use the code viewer!

Evaluate what plugins REs are creating

RQ1 What are current interaction modalities for RE tools?

- Determine how REs interact with their tools
- Find out what interaction modalities are associated with expected usability

RQ2 Do they fit the RE's processes and mental models?

- Do RE tools follow usability guidelines?

Regression results

- Model to predict if a plugin presents interactions in line with code (G2)
  - Tools in the subcomponent phase are almost six times more likely to follow this guideline

- Model to predict if a plugin allows for user configuration (G4)
  - Tools in the experimentation phase are four times more likely to follow this guideline

5.
Discussion

Static is less flexible than dynamic

- Majority of static tools are scanning / visualization tools operating in the Overview phase focused on improving readability
  - Do not promote a transition between phases
  - Do not allow for much analysis tuning

- Most dynamic tools operate in the subcomponent and experimentation phases
  - These tools allow for user selection or integrate with static information (or both!)

Static is less flexible than dynamic

- Majority of static tools are scanning / visualization tools operating in the Overview phase focused on improving readability
  - Do not promote a transition between phases
  - Do not allow for much analysis tuning

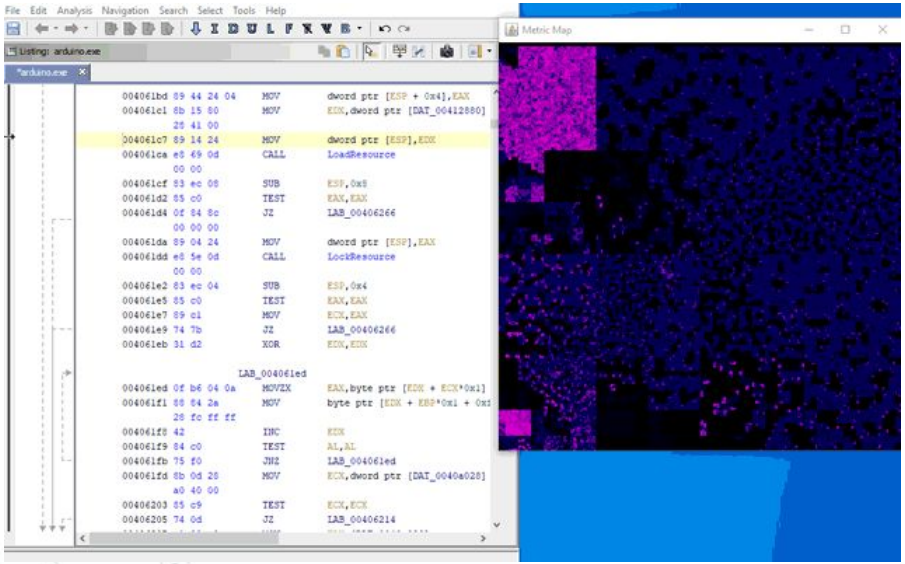
- Most dynamic tools operate in the subcomponent and experimentation phases
  - These tools allow for user selection or integrate with static information (or both!)…but this interaction remains limited

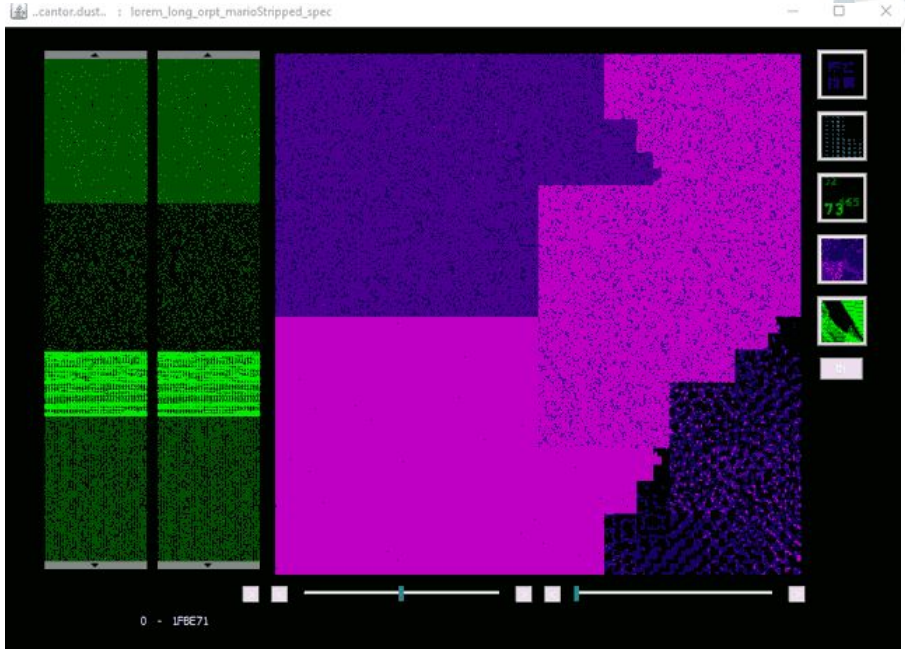
Framework developer takeaways

- The plugins lean towards static functionality type.
  Improvements in API support for dynamic plugins could result in more usable / interactive dynamic plugins being created

Plugin developer takeaways

- Focus on user interaction to validate results





Key Takeaways

- Majority of static tools are scanning / visualization tools operating in the Overview phase focused on improving readability
- Dynamic tools allow for more user interaction and more closely follow usability guidelines, not well integrated into frameworks.
- Framework APIs should provide more emphasis on incorporating input and output interactions with the framework
  - Allow plugin developers to focus on functionality over usability

Questions:
James.mattei@tufts.edu
tsp.cs.tufts.edu