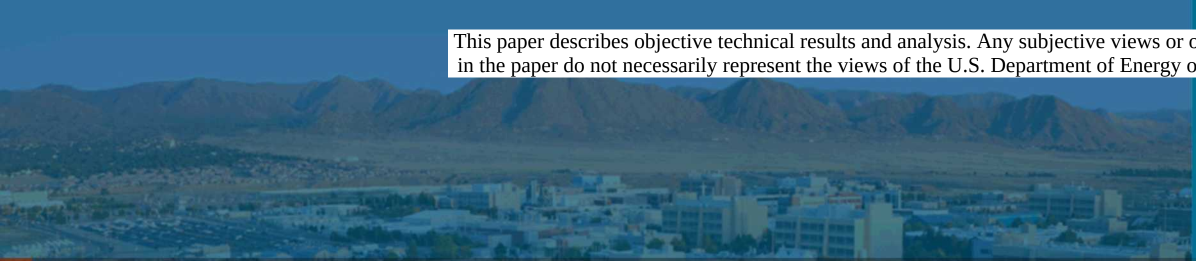
This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

SAND2019-8486C



# Creating a User-centric Data Flow Visualization: A Case Study

Karin Butler, Michelle Leger, Denis Bueno, Christopher Cuellar, Michael J. Haass, Timothy Loffredo, Geoffrey Reedy, and Julian Tuminaro

*Presented at HCI International*

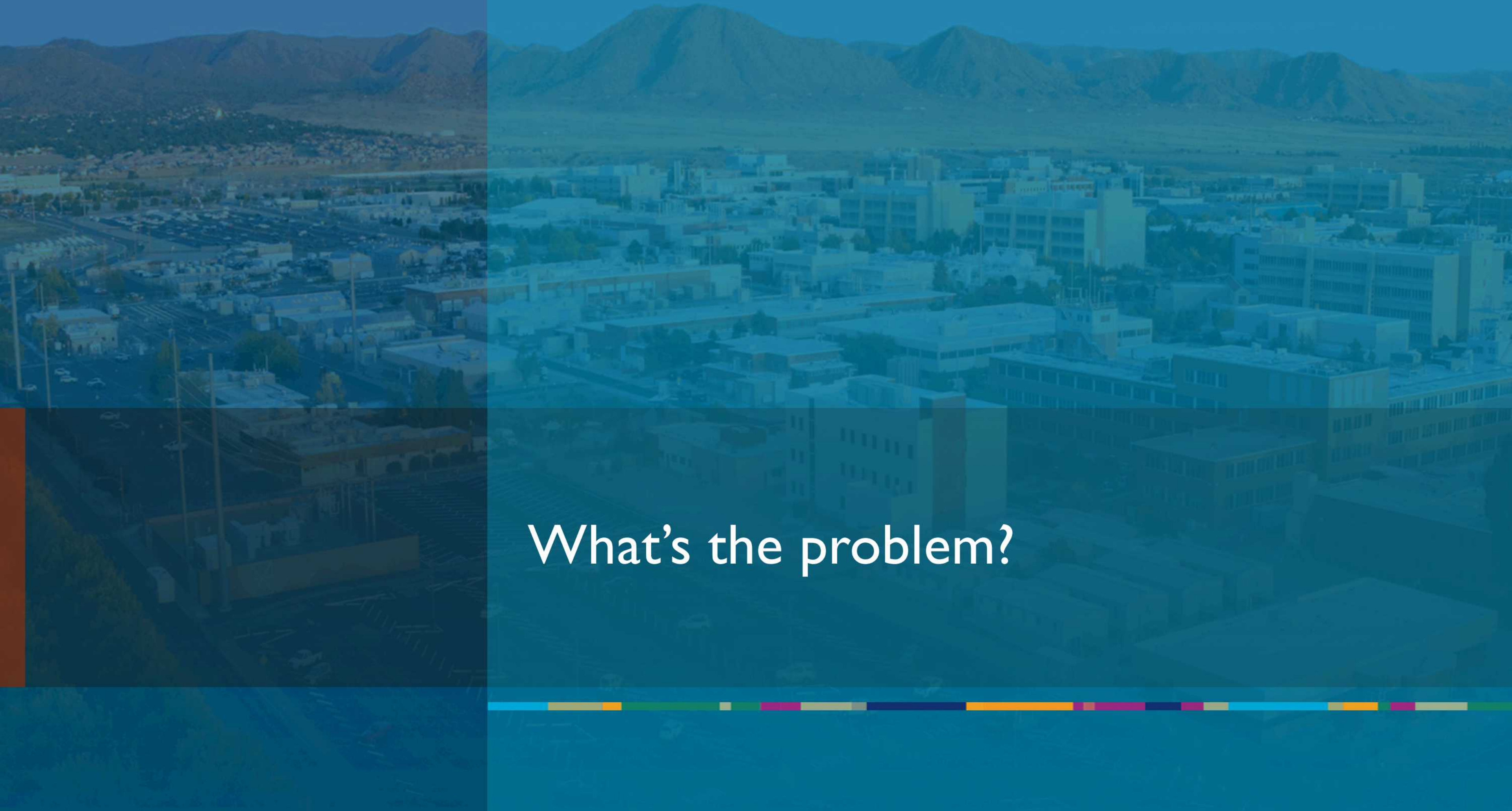
*July 29, 2019*

*Orlando, FL*

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

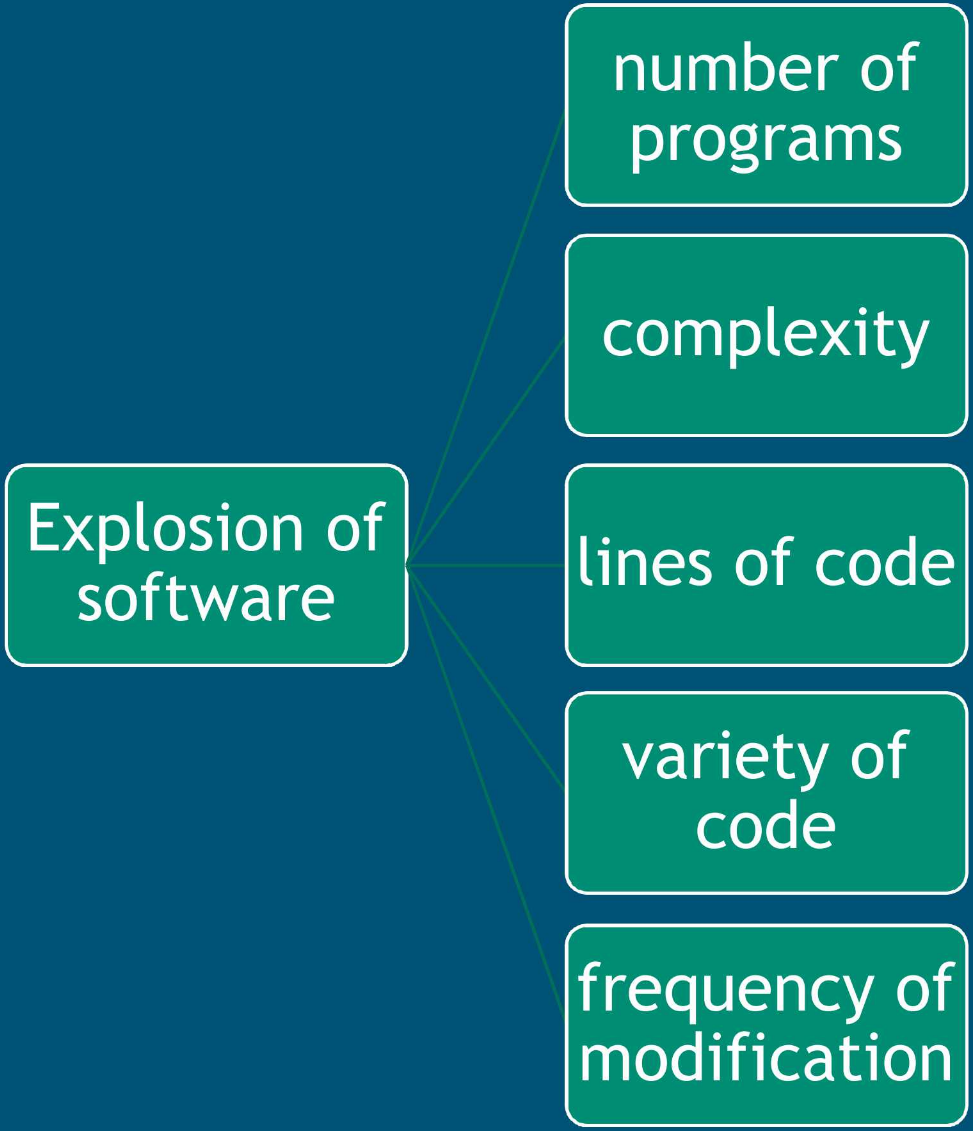
# Outline

1. What's the problem?
2. User-Centered Design Approach
3. Creating a Taxonomy of Interprocedural Data Flow Elements
4. Visualization of Interprocedural Data Flow
5. Evaluation
6. Conclusions and Future Work



## What’s the problem?

# Background



Explosion of software

number of programs

complexity

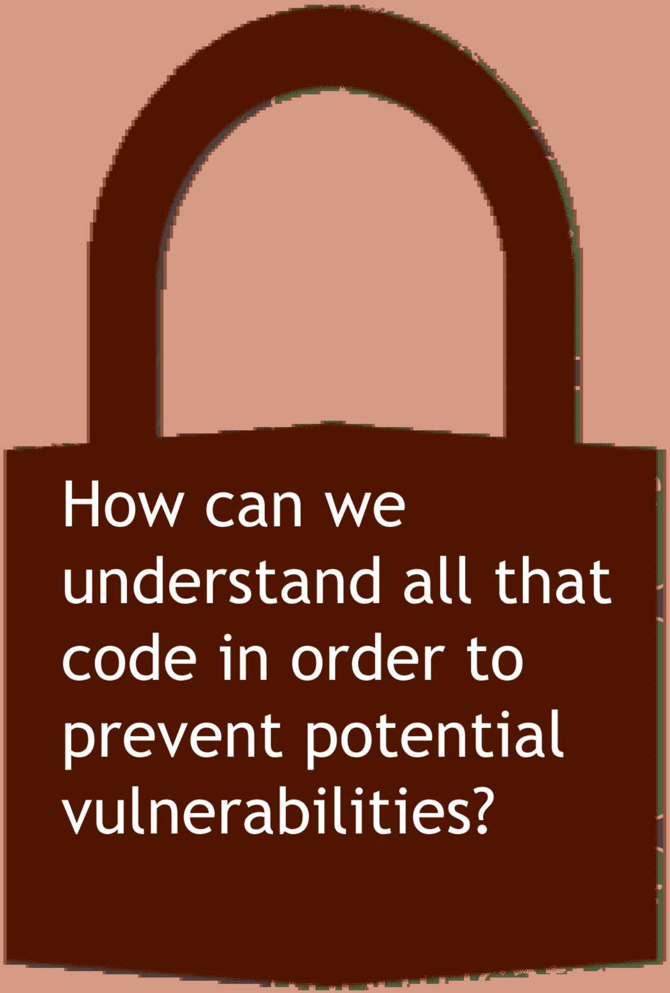
lines of code

variety of code

frequency of modification

# Background

Ex



How can we understand all that code in order to prevent potential vulnerabilities?

# Manual Static Binary Software Analysis

Static → Analyze without execution

- Does not introduce threats from execution
- Does not require access to all support systems

Binary → Detect vulnerabilities introduced during translation from source to binary

Manual → Expert human analysts

- Assisted by automated tools designed to optimize understanding of the execution order of statements (control flow)

# The problem

Current tools are designed to optimize understanding of the execution order of statements (control flow)

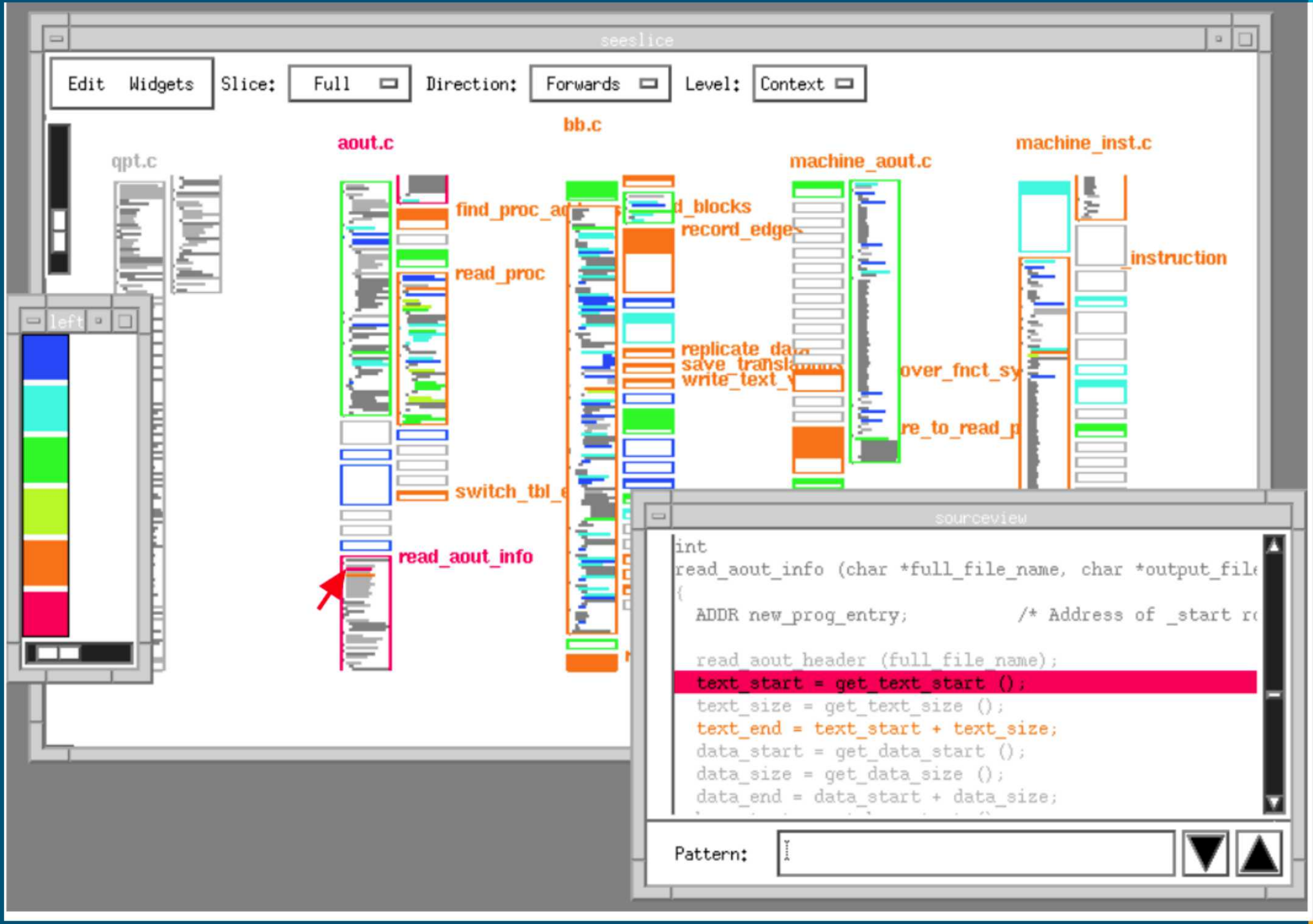
…but attackers increasingly target how data passing through program functions influences other program data and program decisions (data flow).

Tools for human understanding and reasoning about data flow are inadequate.

- Obscure common data flow patterns by overlaying them on control flow abstractions
- Do not allow for analyst updates of content or layout as new information is discovered

# Current Visualizations still look similar

The most advanced data flow visualization tools focus on static visualization for a specific problems.



```
int
read_aout_info (char *full_file_name, char *output_file
{
  ADDR new_prog_entry;            /* Address of _start r
  
  read_aout_header (full_file_name);
  text_start = get_text_start ();
  text_size = get_text_size ();
  text_end = text_start + text_size;
  data_start = get_data_start ();
  data_size = get_data_size ();
  data_end = data_start + data_size;
```

Ball and Eick, “Visualizing Program Slices” in IEEE Symposium on Visual Languages, October 1994.

# User-centered Design Approach

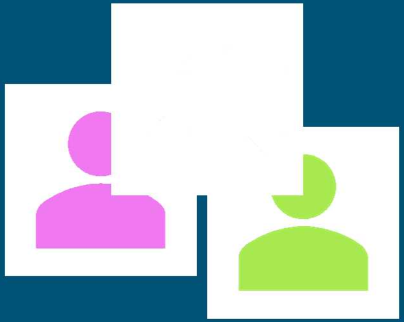
# Creating a taxonomy of data flow elements

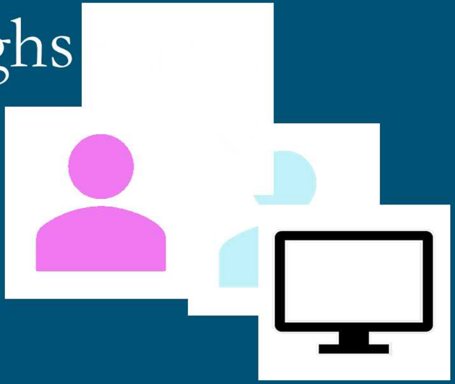
Understanding Essential Data Flow Elements from the perspective of analysts.

# User-Centered Design Approach

GOAL 1: Understand analysis process and extract critical information, knowledge, and relationships for understanding data flow

1. Semi-structured Interviews
2. Applied Cognitive Task Analysis: Knowledge Audits
3. Cognitive Walkthroughs

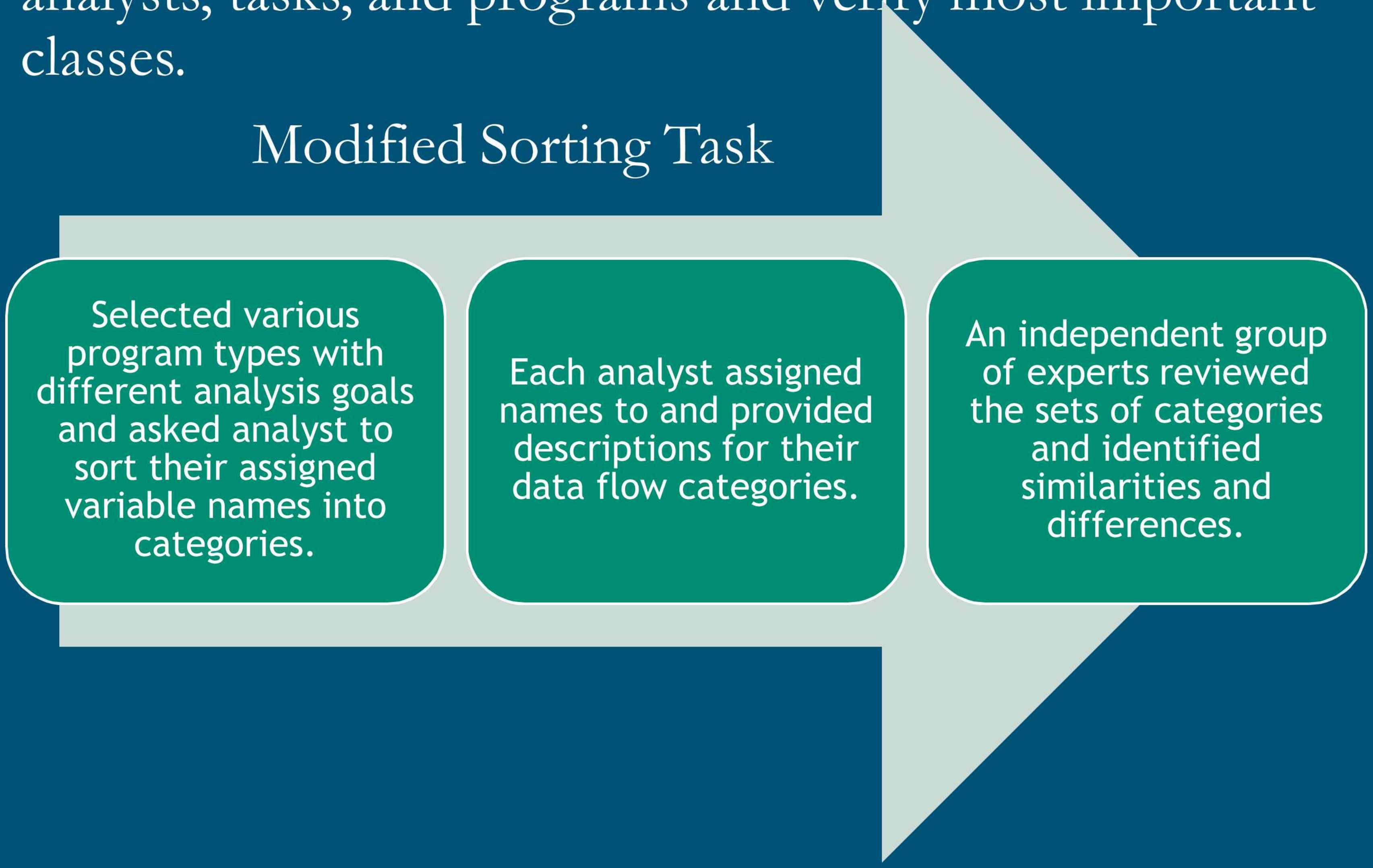




# User-Centered Design Approach

GOAL 2: Generalize data flow elements to other analysts, tasks, and programs and verify most important classes.

## Modified Sorting Task



Selected various program types with different analysis goals and asked analyst to sort their assigned variable names into categories.

Each analyst assigned names to and provided descriptions for their data flow categories.

An independent group of experts reviewed the sets of categories and identified similarities and differences.

# Taxonomy of Data Flow Elements

**Properties of entities**

- Values: constants, computed, constrained, uncertain
- Locations: local, heap, global, shared memory
- Aggregates: arrays, structures, AND, OR
- Code
- Communication: input, output
- Annotations: initial configuration, data type, size

**Properties of relationships**

- Value flow
- Function boundary: parameter, return value
- Points-to
- Comparison
- Control influence: positive, negative
- Length
- Code influence: Allocatable, freeable readable, writeable
- Synchronization: lifetime, sometime
- Colocation: spatial, subset, overlap
- Lifetime

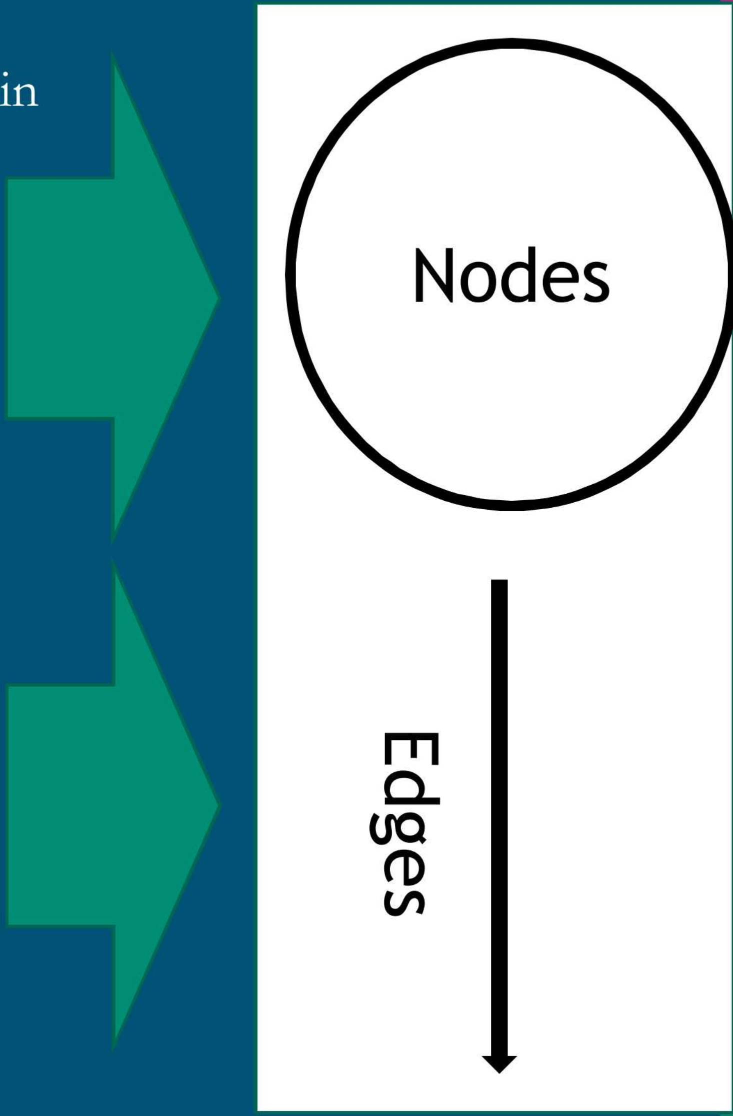
# Taxonomy of Data Flow Elements

**Properties of entities**

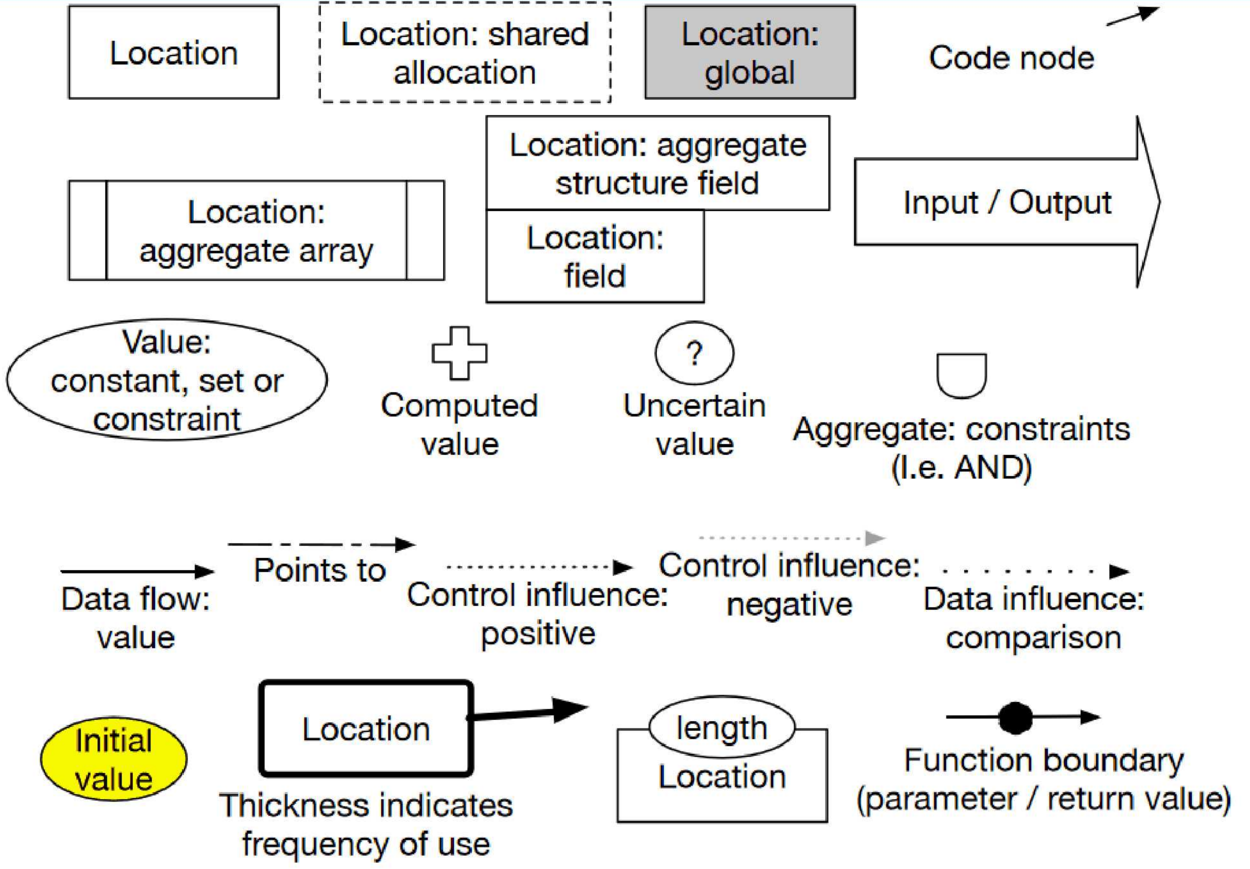
Values: constants, computed, constrained, uncertain
Locations: local, heap, global, shared memory
Aggregates: arrays, structures, AND, OR
Code
Communication: input, output
Annotations: initial configuration, data type, size

**Properties of relationships**

Value flow
Function boundary: parameter, return value
Points-to
Comparison
Control influence: positive, negative
Length
Code influence: allocatable, freeable readable, writeable
Synchronization: lifetime, sometime
Colocation: spatial, subset, overlap
Lifetime



# Assignment of elements to Visual Representation


Location
Location: shared allocation
Location: global
Code node
Location: aggregate array
Location: aggregate structure field
Location: field
Input / Output
Value: constant, set or constraint
Computed value
?
Uncertain value
Aggregate: constraints (I.e. AND)
Data flow: value
Points to
Control influence: positive
Control influence: negative
Data influence: comparison
Initial value
Location
Thickness indicates frequency of use
length
Location
Function boundary (parameter / return value)

Special annotations can indicate priority, uncertainty, safety, boring, and extent of influence

Sequencing information may be implied through left-to-right and top-to-bottom layout

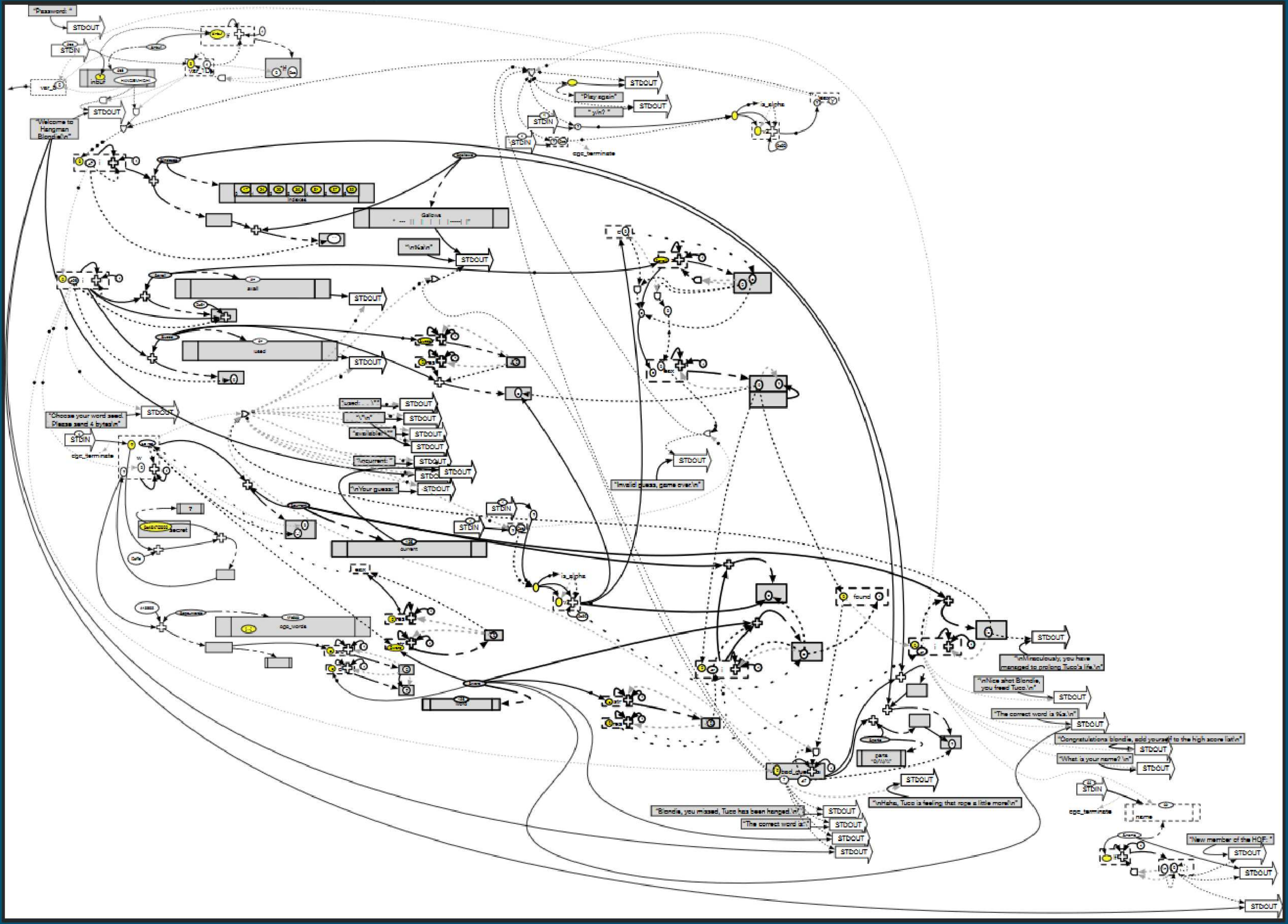
Not yet represented: code influence (allocatable, freeable, visible, updatable), synchronization (lifetime, sometime), colocation (spatial, subset, overlap), or lifetime

# Evaluation of taxonomy

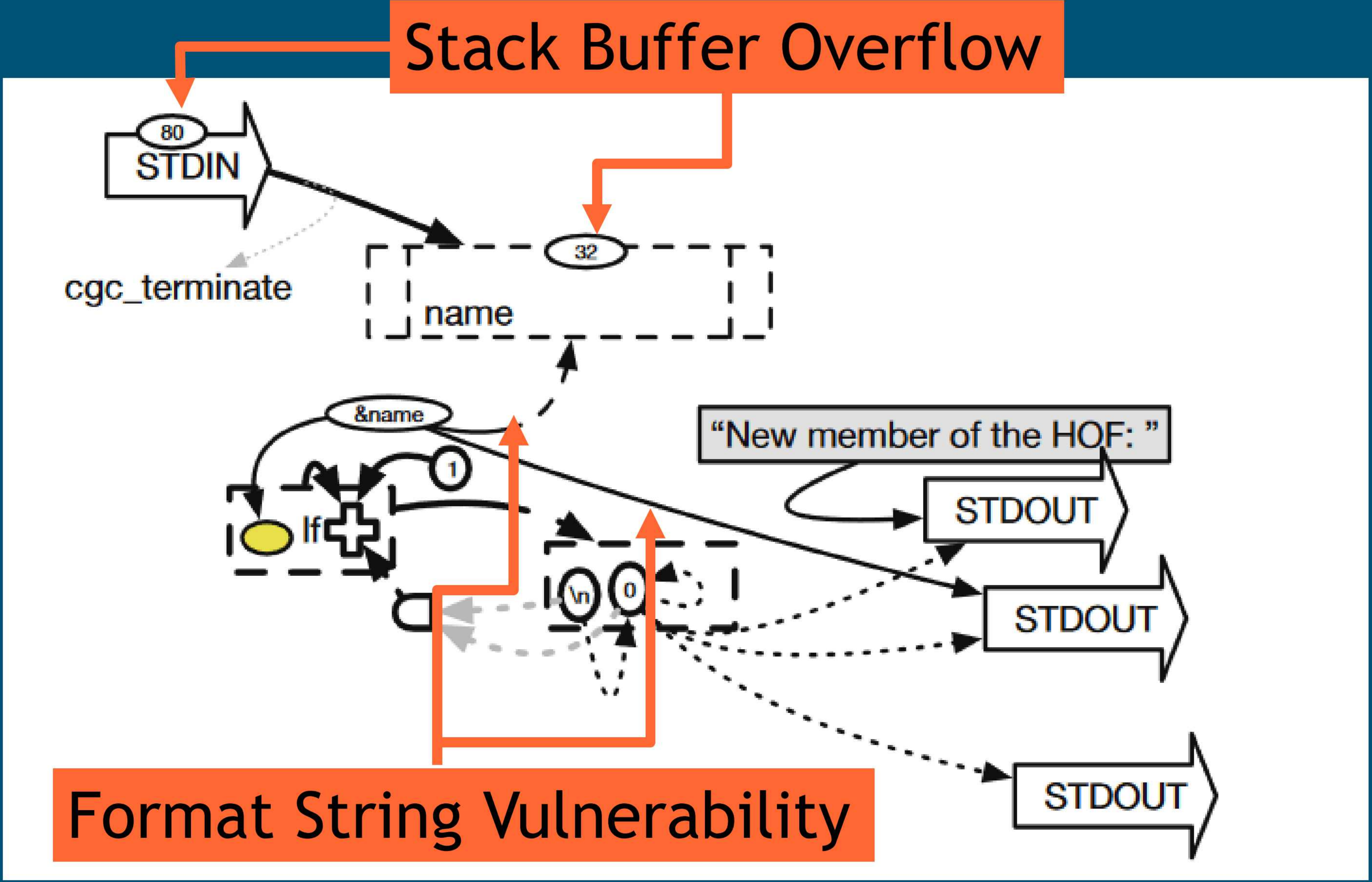
Proof of Principle

- Creation of visualizations from Cyber Grand Challenge (CGC) binaries by team lead and by a project naive analyst with less analysis experience
  - CROMU_00065 (WhackJack)
  - KPRCA_00052 (Pizza Ordering System)
  - EAGLE_0005 (Hangman Game)

# Data flow representation of CGC challenge binary EAGLE_0005



# Data flow representation of vulnerabilities in EAGLE_0005



# Evaluation of taxonomy

Can an analyst inexperienced with the data flow visualization understand data flow from it?

1. 15 minutes of training with the CROMU visualization
2. Given the EAGLE visualization and asked to answer questions about data flow.
   QUESTIONS:
   - Can you find where the global array gallows is written?
   - Looking at the processing of the input buffer inbuf in the function main:
     - What is the initial value of the pointer?
     - When is the pointer incremented?
     - Are values being read or written as the array is walked?
     - What values are being looked for as the array is walked?
     - What values are written?
     - When are those values written?
3. RESULT: Analyst was able to answer questions about 11 of 14 data elements correctly in 40 minutes

# Conclusions and Future Work

- User-centric approach was effective.
  - Taxonomy requirements may be limited to range of programs and data flow represented by programs used in the modified sorting task.
- Visualization still needs to be integrated into analyst workflow.
  - Streamlining the generation of the visual elements during the discovery process through automation and usable interactivity tools.

THANK YOU!

Questions?