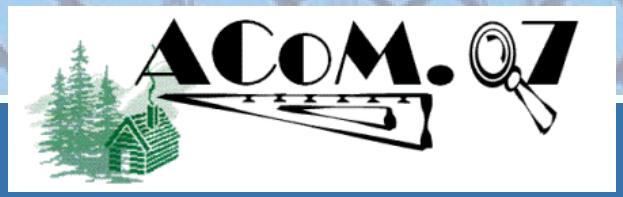
Identifying, Assigning, and Quantifying Crosscutting Concerns

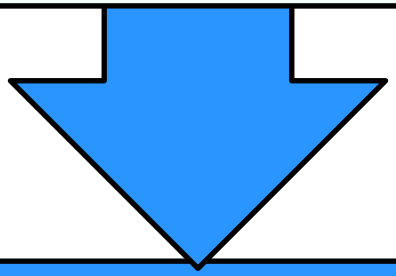
Marc Eaddy and Alfred Aho
Columbia University

Gail C. Murphy
University of British Columbia



Motivation

Concern quantification is important, but current techniques are inadequate



Our Contribution

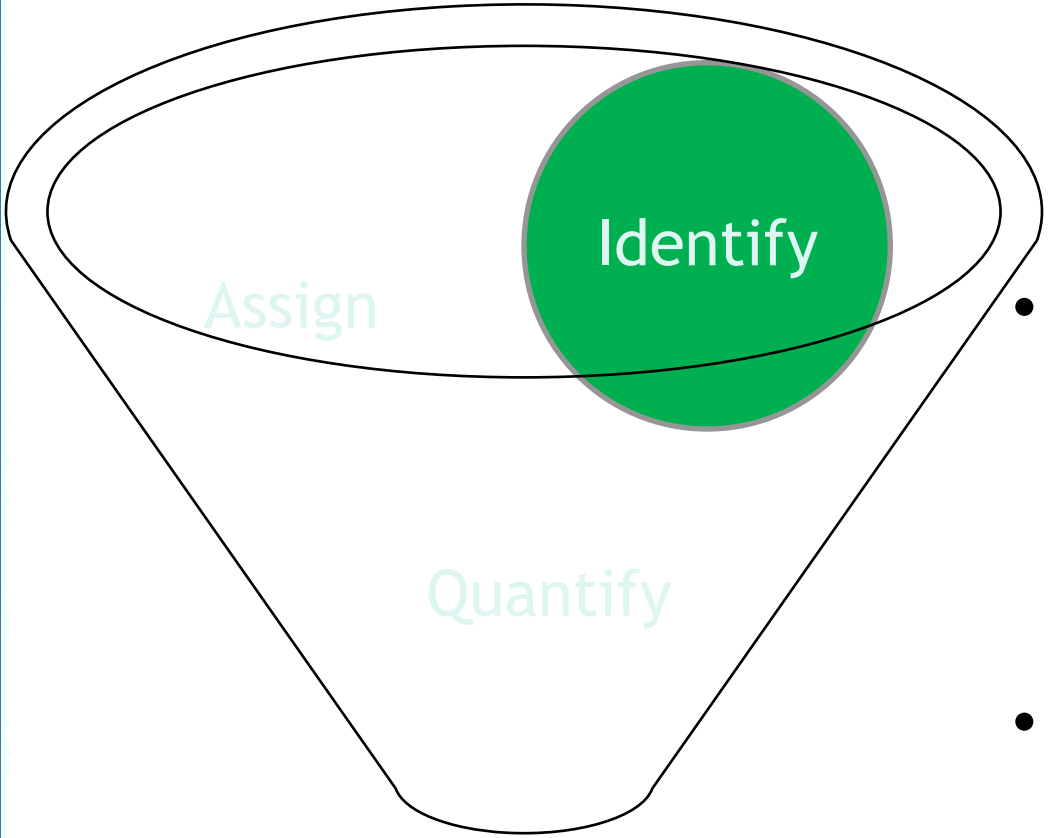
Improved manual technique for locating concerns	New concern metrics (DOS, DOF)

Concerns

Every line of code exists for a reason

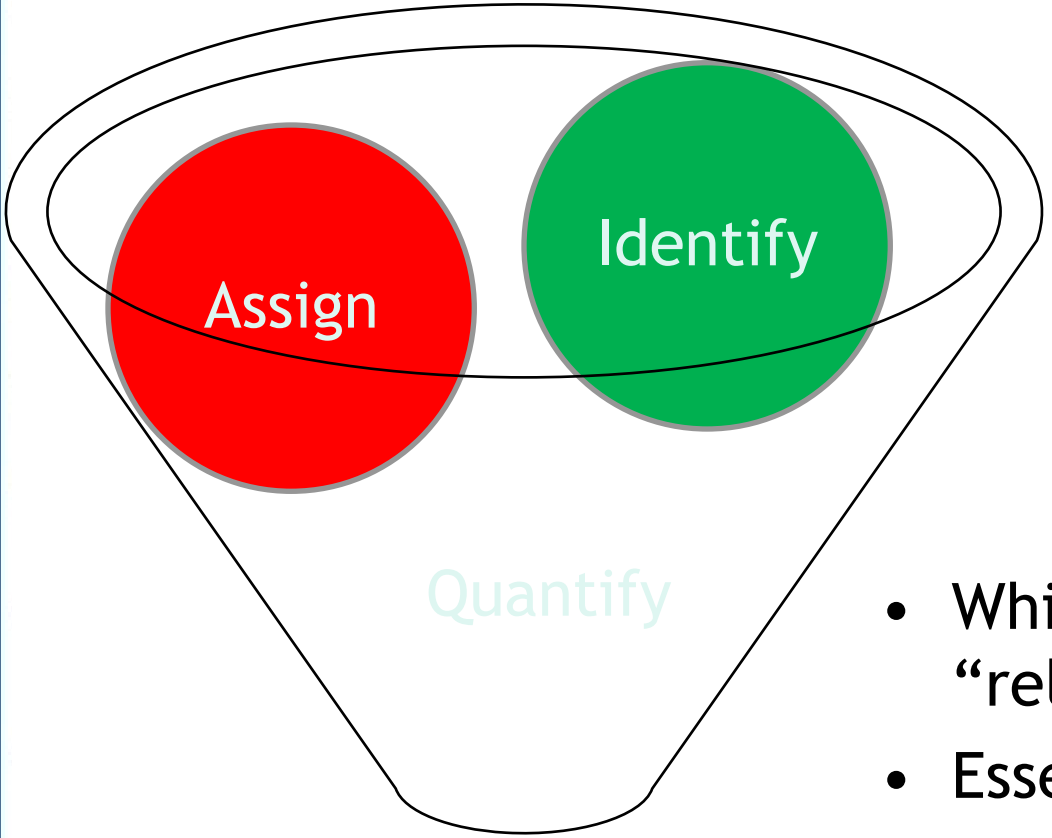
- Anything that impacts the implementation of a program
- Important to understand concerns…
  - "Why is this class/method/line here?"
  - Concerns are likely sources of change
- …and how they are implemented
  - Improves program understanding
  - Improves change impact estimation
  - Impacts software quality (e.g., defects)

Concern Analysis



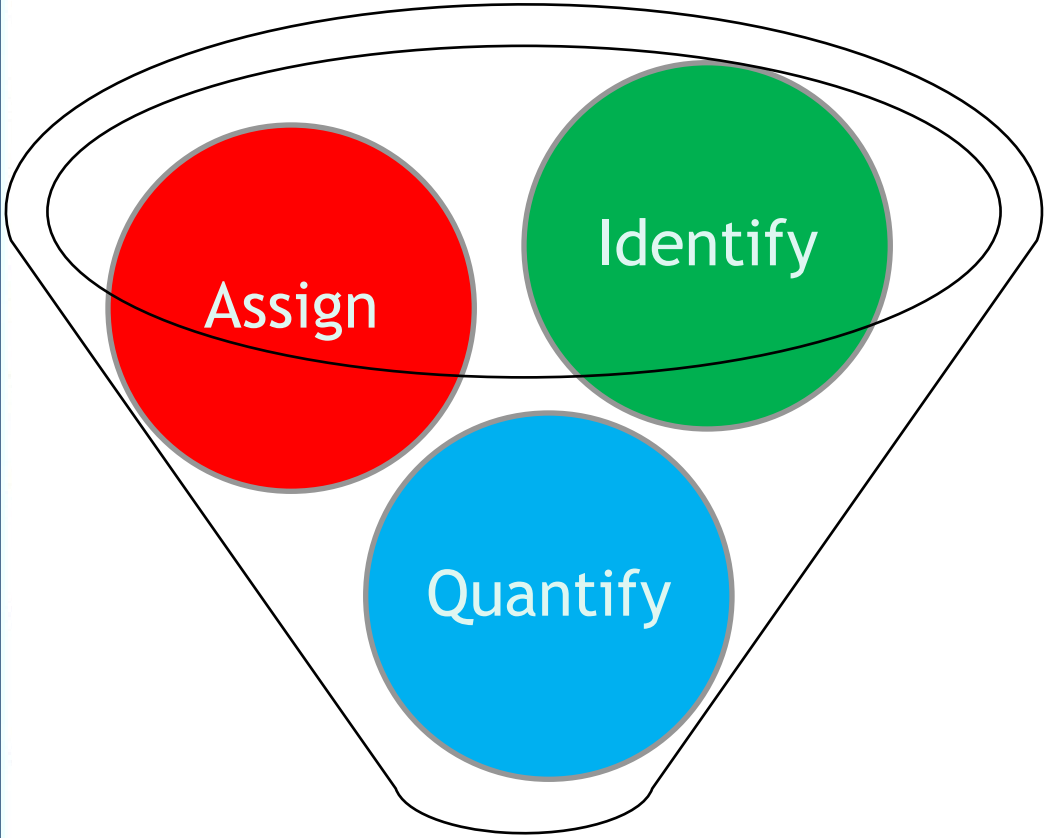
- Reverse engineering
  - Concern may be biased, fake/ unimportant, and inconsistent
- Use an existing spec
  - Concerns more likely relevant, well organized, and consistent

Concern Analysis



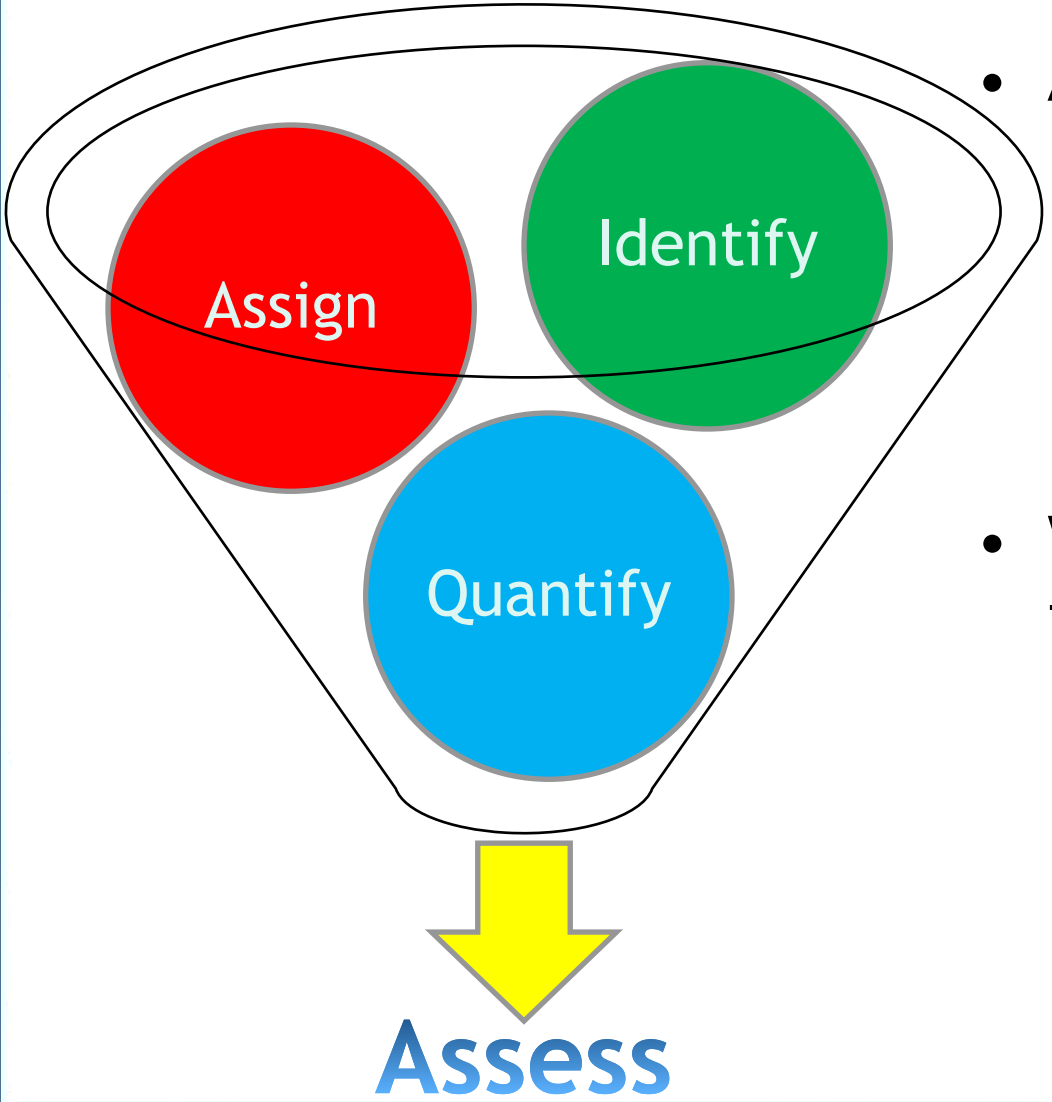
- Which program entities are “relevant”?
- Essentially a reverse engineering exercise
- Hard/impossible to determine accuracy

Concern Analysis



- Need better metrics
- Need empirical validation

Concern Analysis



- Assess impact on
  - Quality (e.g., defects)
  - Understandability
  - Maintenance costs
  - etc.
- Why scattering/tangling impacts quality?
  - Need better theories/models
  - Otherwise, correlations are meaningless

Which entities are relevant?

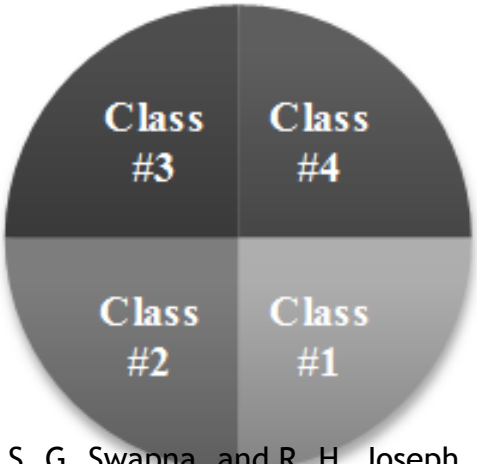
- Entities **dedicated** to one concern
- Entities **shared** by many concerns
  - `Main()` – executed by all concerns
  - `System.String` – reusable/utility class
  - More sharing implies less relevant?
- Our assumption: Relevance based on likely change impact
  - Solves problem for `Main`, but still too vague
  - Cannot consider all possible changes

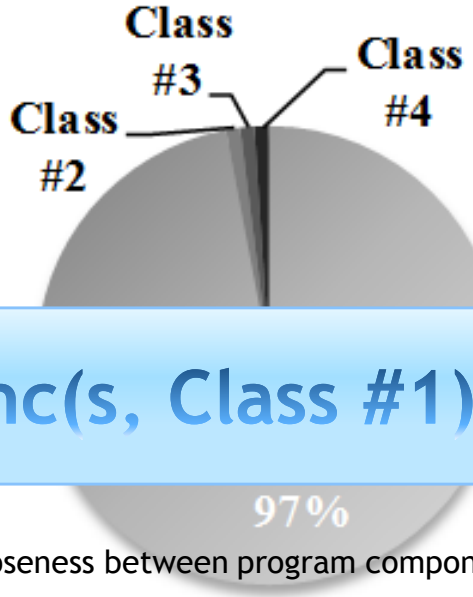
Removal Dependency Assignment

- Our approach: Entities impacted by *eradicating* a concern
  - (Not just disabling the concern)
  - Fairly easy to decide
- Approximates other types of changes
  - Concern modification/addition
  - Empirical evidence needed

Existing Concern Metrics Too Coarse

- Measure *presence* of a concern, not *degree*
- *Diffusion metrics* (Garcia et al.), *spread*





Conc(s, Class #1) = 97%

W. E. Wong, S. G. Swapna, and R. H. Joseph, "Quantifying the closeness between program components and features," *Journal of Systems and Software*, 2000.

Our Metrics: DOS

- *Degree of scattering* – Measures the distribution of a concern's implementation

$$DOS(s) = 1 - \frac{|T|}{|T| - 1} \sum_{t \in T} \left(CONC(s,t) - \frac{1}{|T|} \right)^2$$

- *Average DOS*
  - Overall modularity of concerns
  - Summarizes amount of crosscutting present
- More insightful than traditional product metrics

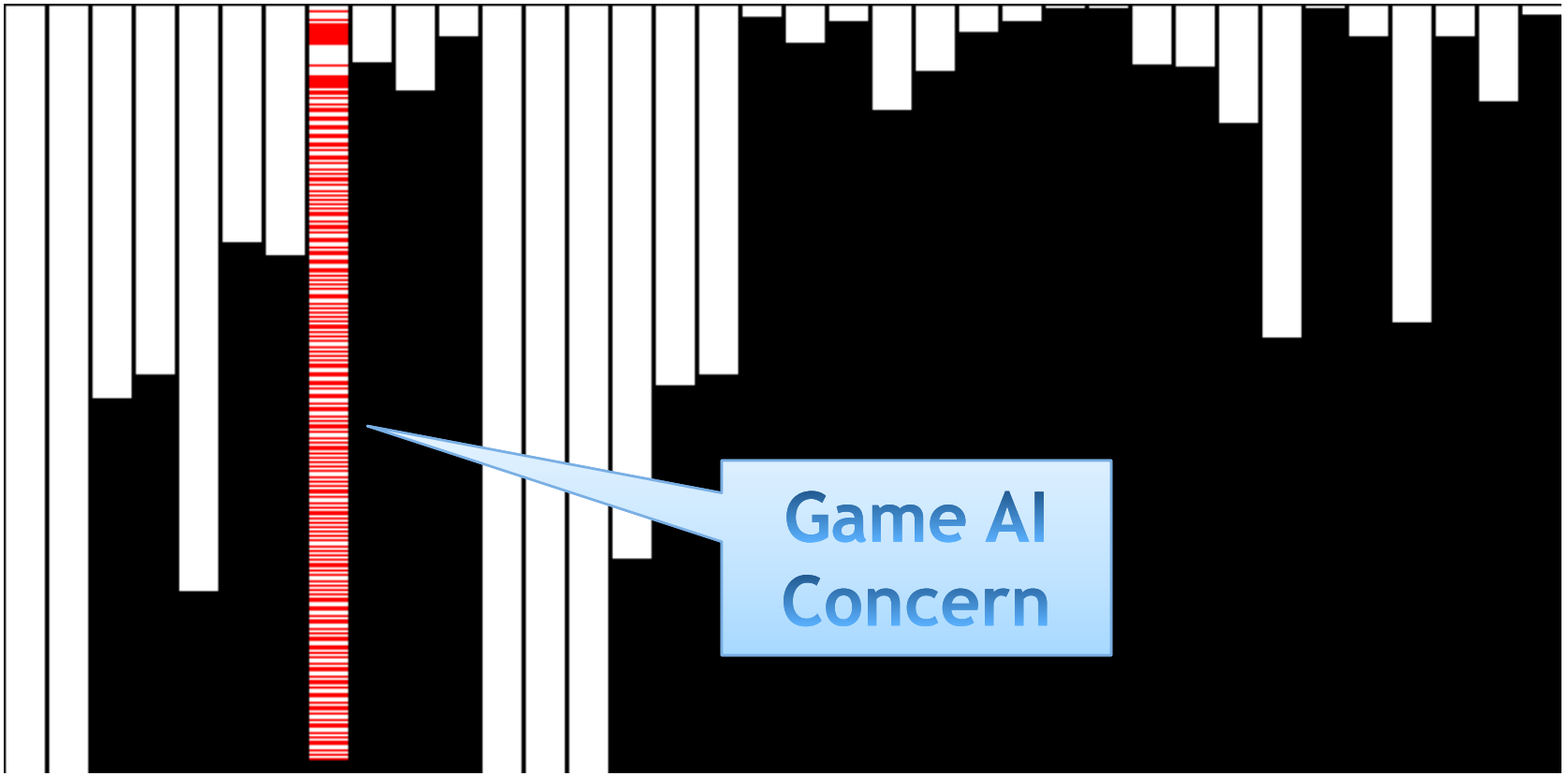
Our Metrics: DOF

- *Degree of focus* – Degree to which component's implementation relates to multiple concerns

$$DOF(t) = \frac{|S|}{|S| - 1} \sum_{s \in S} \left(DEDI(s,t) - \frac{1}{|S|} \right)^2$$

- Reflects amount of tangling
- *Average DOF* – Overall separation of concerns

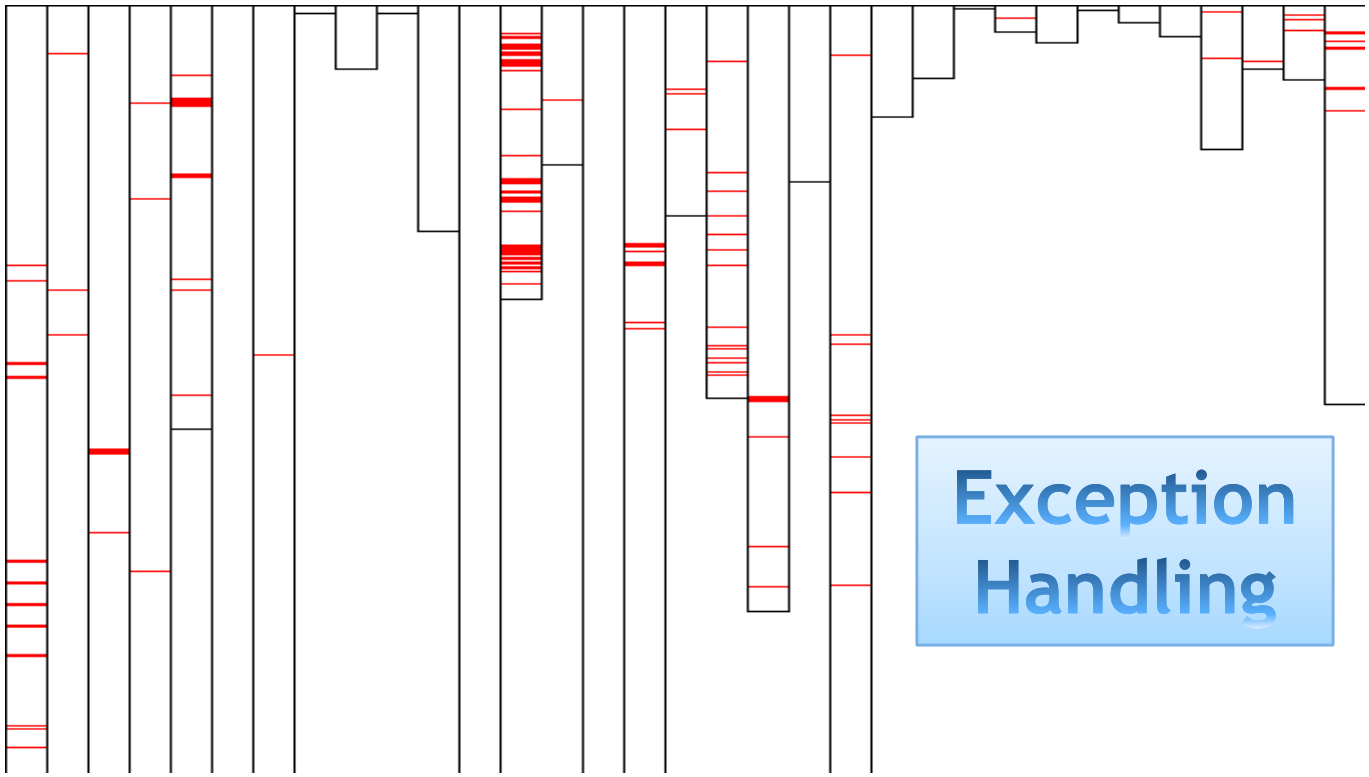
Feasibility Study: Goblin



- Game AI is localized

Goblin: Our Findings

- 95% of Goblin's concerns are scattered
  - Similar findings for two other projects



DOS More Fine-Grained

Concern (Requirement)	*SLOC*	*Concern Diffusion*	*Degree of Scattering*
Graphics API integration	3814	54	0.92
Monocle-display support	2192	10	0.57
Exception/error detection	513	33	0.89
Exception/error handling	455	30	0.80
Collision detection	424	10	0.76
Logging	271	9	0.57
Persistence	190	11	0.83
Clean shutdown	118	13	0.83
Help display	34	1	0.00
Application plug-ins	31	1	0.00

Concluding Remarks

New concern metrics

- Fine-grained measurement of scattering (DOS) and tangling (DOF)
- Recently, found correlation between scattering and defects
- More theoretic and empirical validation needed

Scattering is prevalent

- I.e., a significant problem
- Concerns, bug fixes, changes
- Can we reduce scattering of “real” concerns using OO? AOP?

Thank You!

Contact

Marc Eaddy

me133@columbia.edu

Goblin Dataset

http://www.cs.columbia.edu/~eaddy/goblin