Static analysis

- Widely used.
- Active area.
- Papers: case studies.
  - Google, Facebook

Q: What are specs?

Specs?

- Partial/aspects vs. full correctness.
- Co-design “spec” vs. write full spec.
- Heuristic/“universal” vs. app-specific.
- Internal vs. external behavior:
  - Invariant.
- Locality vs. whole execution.
- Practicality vs. completeness.
Case study: FindBugs @ Google

- Dashboard.
  Developers ignored
  Facebook: 0% fix rate.
  "Master" bug list
  Owned by sec eng team

- File bug reports.
  Not worth fixing
  Facebook: "near silence."
  See engineers file bugs.

- Code reviews. Good plan.
  FindBugs: too much customization

- Compile time.
  Google: review → compile time.
  Facebook: too slow/costly.

Common themes
- "Context switch."
- Who fixes it?
- Shorter feedback loop

Solve problem for dev/eng.
- Right time.
- Right insight.
- Right dev/eng.

How deal w/ bugs?
- Testing
  → integration
  → monitoring
- Runtime checks
  → racey
  → overflows

- Static analysis
  → exec / X inputs.

- Security.

- Mobile devices.
Google: pitfalls, avoidance

- Developer workflow
  - Compile-time
  - Review-time
- Facebook
  - File bugs
  - Batch
- File
  - File bugs

- Actionable warnings
  - Legacy code FP
  - Fix bugs/warnings before deploying tool.

- User trust
- Google: avoid FP $\Delta$ action.
  - FB: pick teams/problems.

- Relevant/fixable
- Understand warnings
  - `printf(\"%s\", x, y, z);`
Focus/metrics.

Google: "effective false positive". EFP.
Fix nuisance warning ⇒ not EFP
Missed real issue. ⇒ EFP.
Flag "not useful".
First lesson: "always bugs".
⇒ Unobjectionable bugs.

FB: "missed bugs".
Q: "How could we have caught it?" / analysis?
Focus areas:
Crashes
Security
Concurrent
Works well:
3 data races missed. ⇒ 50% False positives.
11 see bugs missed. **

Devs to tweak FP!
Interprocedural bugs

$!v.m = \texttt{x509 GMT time adj}(\ldots)$

$!f()$

$!y.g()$

\[ i = \texttt{revm.length + 3}; \]

$!z = \texttt{return NULL}$

![Web browser](render())

```
@id = getID$(id)
out = getForm$(id)
print$(out)$
```

Compositional analysis

"Procedure summary:
intermediate spec
for every func"

who writes intermediate specs?

Infer: deduce specs.
Linux: annotations.