Visualizing the Complexity of Software Module Upgrades

Bram Schoenmakers, Niels van den Broek, Istvan Nagy, Bogdan Vasilescu, Alexander Serebrenik

@bram85 @b_vasilescu @aserebrenik
Is the software easy to upgrade?

Why does upgrading one module require upgrading many other modules?
Goal: Update A to the current version
Goal: Update A to the current version

Solution 1 reduced the amount of code to be shipped to the customer

Solution 2 reduces the amount of shipments
**Goal**: Update A to the current version

**Solution 1** reduced the amount of code to be shipped to the customer.

**Solution 2** reduces the amount of shipments.
Color = #modules needed to upgrade when upgrading row from column to current (8).

AF, AL, AS, AY are easy to upgrade, AC is difficult. Why?

Cliff between 2 and 3???
Upgrade dependency graph
• 1000 software developers
• 40 MLOC
• 327 modules
• 7000 interfaces
• 9 monthly versions of the software

• 327 modules * 8 version updates = **2616 upgrade scenarios**
  – Search space: $8^{327-1} \sim 2.5 \times 10^{294}$ configurations

• Processing time: **16 hours** for all scenarios
  – With limitations on search space.
Inspecting upgrade dependency graphs, we’ve found many red dependencies, caused by symbol removal.
Cliff between 2 and 3

Inspecting upgrade dependency graphs, we’ve found many red dependencies, caused by symbol removal.

Indeed,

Red dependencies included

- 45% ≤10 upgrade dependencies
- 55% >10 upgrade dependencies

Red dependencies excluded

- 91%
- 9%
Inspecting upgrade dependency graphs, we’ve found many red dependencies, caused by symbol removal.

**Suggestion**: symbols are removed only when they are no longer used in any supported release.
Is the software easy to upgrade?

Why does upgrading one module require upgrading many other modules?

- 1000 software developers
- 40 MLOC
- 327 modules
- 7000 interfaces
- 9 monthly versions of the software