Non-Atomic Refactoring & Software Sustainability

Titus Winters (titus@google.com)

Sustainable

Your project is sustainable when you are able to change all of the things that you ought to change, safely, and can do so for the life of your project.

Non-Atomic Refactoring

0. Identify problem API
1. Introduce new variant
2. Migrate old calls to new
3. Remove old

Non-Atomic Refactoring

0. Identify problem API

```
string File::JoinPath(const string&, const string&);
```

Non-Atomic Refactoring

0. Identify problem API

```
string File::JoinPath(const string&, const string&);
```

1. Introduce new variant

```
string file::JoinPath(...);
string file::JoinPathRespectAbsolute(...);
```

Non-Atomic Refactoring

0. Identify problem API

```
string File::JoinPath(const string&, const string&);
```

1. Introduce new variant

```
string file::JoinPath(...);
string file::JoinPathRespectAbsolute(...);
```

2. Migrate old calls to new

Non-Atomic Refactoring

0. Identify problem API
1. Introduce new variant
2. Migrate old calls to new
3. Remove old

```
string file::JoinPath(...);
string file::JoinPathRespectAbsolute(...);
```

Non-Atomic Refactoring Requirements

- Code is well-behaved
- No such thing as provably correct refactoring
- Pervasive unit tests
- Tooling critical at scale

Applications of Non-Atomic Refactoring

Language Change

Go, C++, Python Upgrades to new language version may require tool-assisted refactoring to upgrade.

Monorepo

Change too large to sync/test/commit safely.

Multirepo

Avoid need to synchronize commits across repository boundaries.

Questions + Discussion