

AI-Accelerated Migrations

Charles Covey-Brandt, Developer Platform, Airbnb



Quick story:

3,400 files need migration
Blocking major library update
Last, hardest files to migrate

Est. 1.5 dev years
to complete

For each of the 3.4k files:

1. Understand business context
2. **Understand test intent**
3. Rewrite entire file to new library
4. Maintain coverage and intent

Commit 1.5+ dev years?

Try AI ✨?

Try AI ✨✨!

4 dev weeks to
complete

+ 2 more for code review

Since then, with AI:

12 more migrations

1M lines of code

7+ dev years saved

Just getting started



But how?

- **Four techniques**
- **And a framework**

**#1 Programmatic validation
outside scope of the AI**

Validation outside AI scope

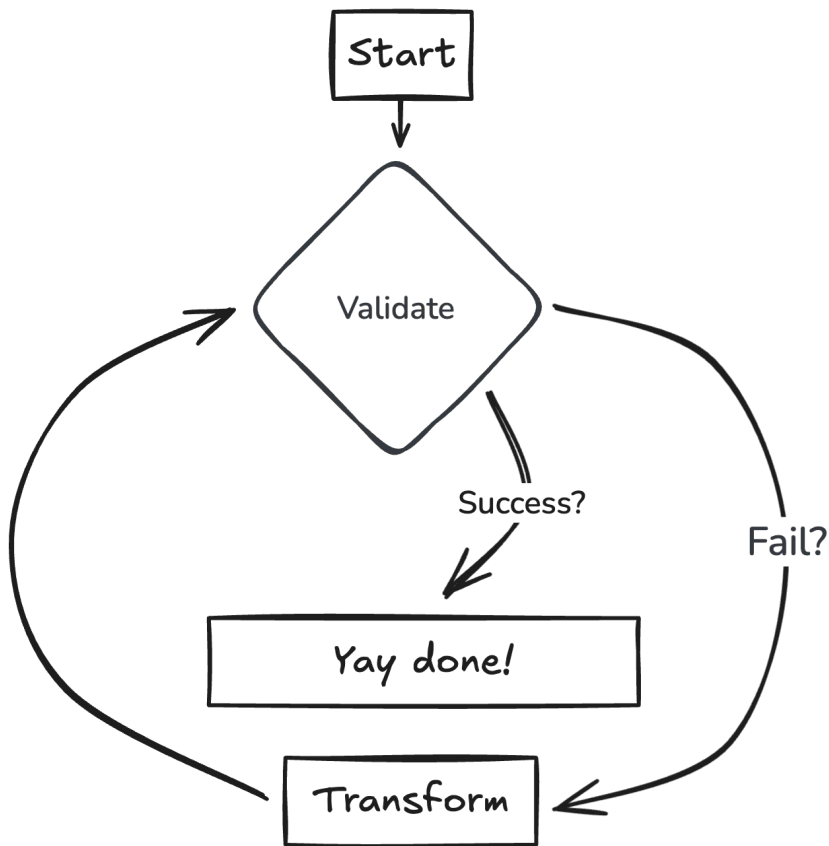
- Programmatic & idempotent
- *Always* runs after the AI transform is complete
- Migration specific

```
# Run the AI agent
ai-agent -p "Follow ${INSTRUCTIONS} for ${TARGET}"

# Run migration-specific validation _after_
./validate-migration $TARGET

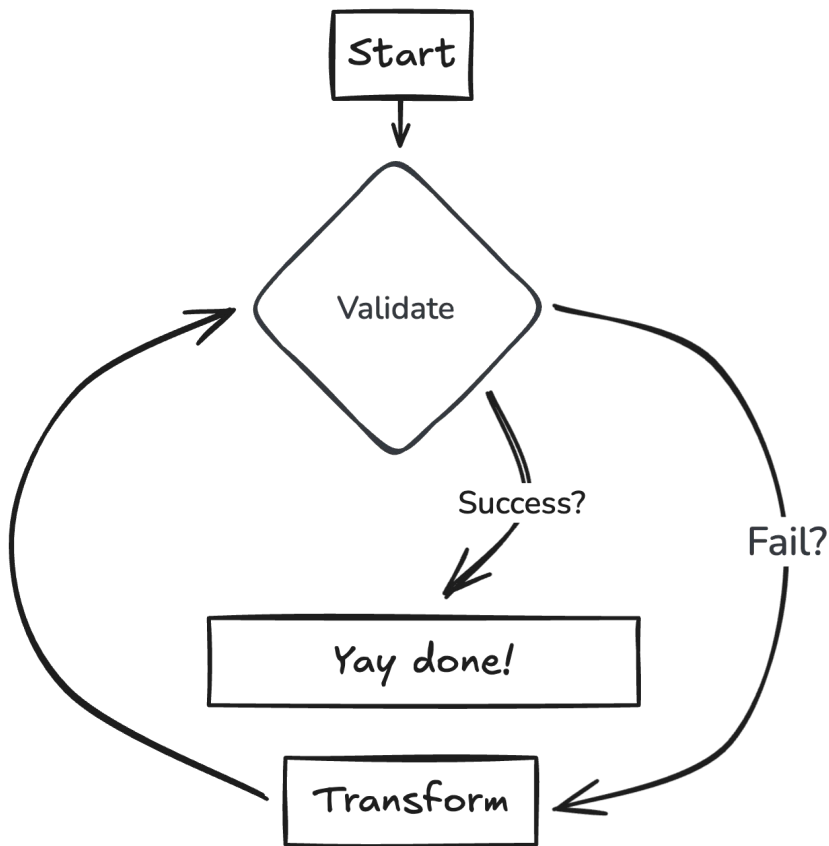
# Then run CI
bazel test ...
```

#2 Script transform & validate as loop



The validate / transform loop

- Starts with validation
- AI transform when validate fails
- Fail on max attempts
- **Enables:**
 - Pause / resume
 - Brute force retries



The validate / transform loop

- Starts with validation
- AI transform when validate fails
- Fail on max attempts
- **Enables:**
 - Pause / resume
 - Brute force retries

\$./run-migration \$TARGET

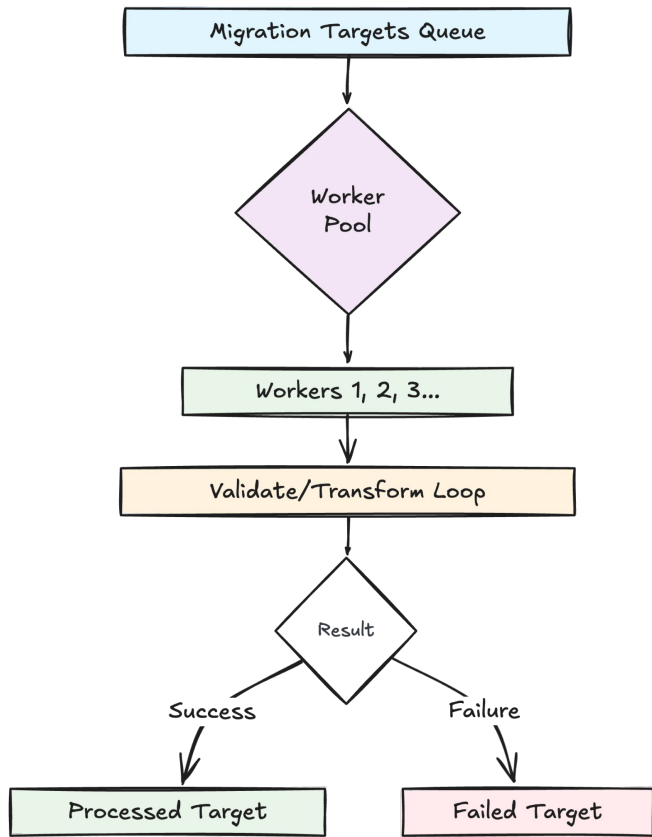
```
let maxAttempts = 5;
while (true) {
  const { error } = await validateTarget(target);

  if (!error) {
    // validation succeeded, we're done!
    return { success: true };
  }

  if (maxAttempts === 0) {
    // max attempts hit, this run failed
    return { success: false };
  }

  const prompt = getPrompt(target, error);
  await spawn(`ai-agent --prompt=${prompt}`);
  maxAttempts = maxAttempts - 1;
}
```

#3 Design for concurrency



Code groups = targets

- Targets could be files, folders, bazel targets
- Loop runs concurrently for *each target*

\$./run-migration-all

```
run(targets, async (target) => {  
  let maxAttempts = 5;  
  while (true) {  
    const { error } = await validateTarget(target);  
  
    if (!error) {  
      // validation succeeded, we're done!  
      return { success: true };  
    }  
  
    if (maxAttempts === 0) {  
      // max attempts hit, this run failed  
      return { success: false };  
    }  
  
    const prompt = getPrompt(target, error);  
    await spawn(`ai-agent --prompt=${prompt}`);  
    maxAttempts = maxAttempts - 1;  
  }  
}, { concurrency: 50 });
```

#4 “Sample, tune, sweep”

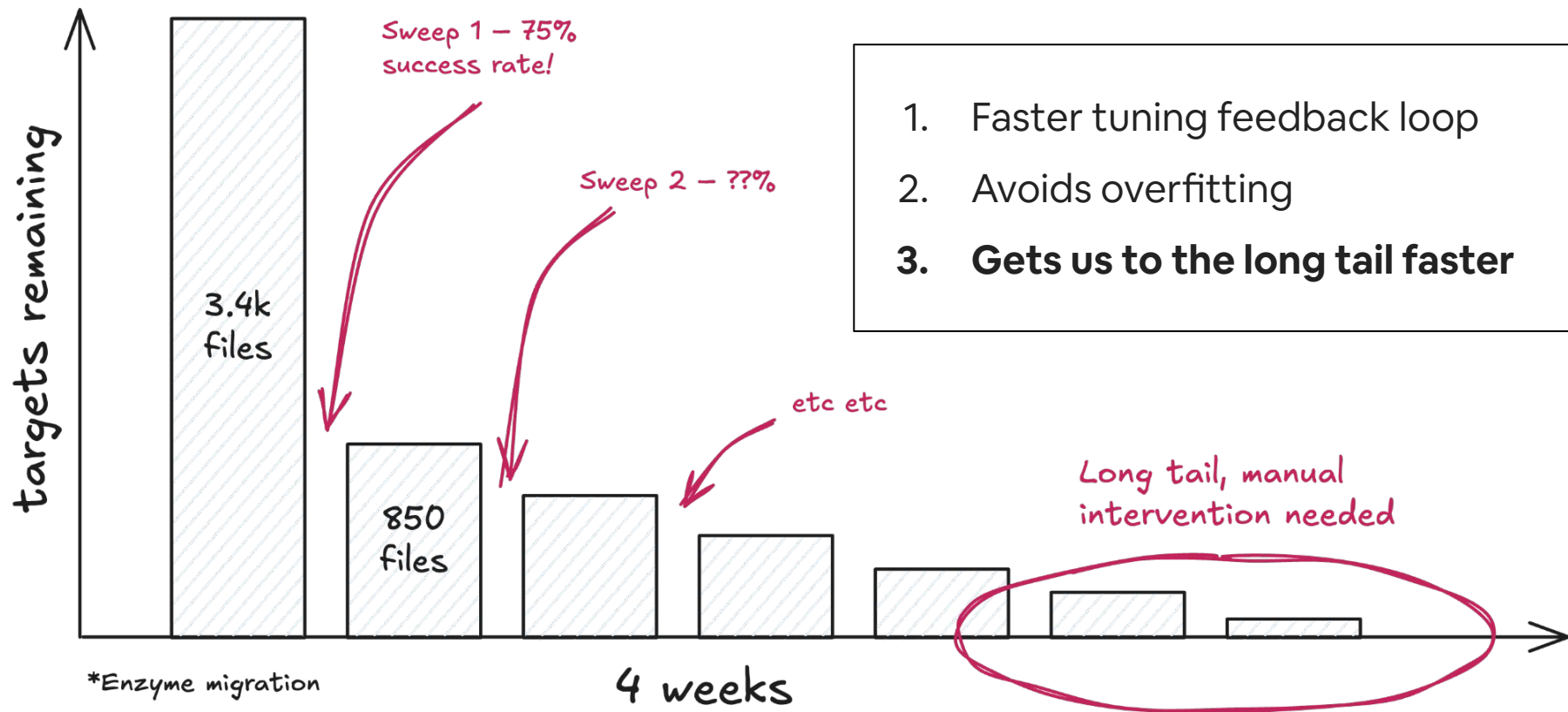
“Sample, tune, sweep”

Fast, *breadth-first* execution:

1. Select (random) sample of targets
2. Tune prompts so selections pass*
3. Sweep across all targets
4. Bank passes, repeat on remaining

*The final version of your migration script (when you're done with the migration) will be *overfitted* for the toughest targets in your migration

“Sample, tune, sweep”



AI Migration Framework

- **Codifies** best practices
- Enables **self-serve**

Defining targets

```
export default createConfig({  
  findTargets: {  
    select: 'frontend/**/*.test.{ts,tsx}',  
    filter: ({ content }) => content.includes('enzyme'),  
  },  
  
  steps: [  
    createStep({  
      name: 'remove-enzyme',  
      validate: async ({ content }) => {  
        if (content.includes("from 'enzyme'")) {  
          return { error: 'Enzyme still imported in file' };  
        }  
      },  
      transform: new CodingAgentTransformer((target) => {  
        return `  
The file ${target} is using enzyme and we need to  
rewrite it to use react testing library.  
  
To do this ...  
Here is more context that will help you ...  
`;  
      })),  
    ],  
  ),  
});
```

Loop definition

```
export default createConfig({

  findTargets: {
    select: 'frontend/**/*.*.test.{ts,tsx}',
    filter: ({ content }) => content.includes('enzyme'),
  },

  steps: [
    createStep({
      name: 'remove-enzyme',
      validate: async ({ content }) => {
        if (content.includes("from 'enzyme'")) {
          return { error: 'Enzyme still imported in file' };
        }
      },
      transform: new CodingAgentTransformer((target) => {
        return `
The file ${target} is using enzyme and we need to
rewrite it to use react testing library.

To do this ...
Here is more context that will help you ...
`;
      }),
    });
  ],
});
```

CLI execution

```
./migrate run enzyme-to-rtl.ts --random=5  
./migrate run enzyme-to-rtl.ts --status=failed  
./migrate run enzyme-to-rtl.ts # run a sweep
```

With or without a framework:

migration specific

1. Idempotent[✓] validation, outside of the AI
2. Script the validate / transform loop
3. Design targets for concurrency
4. Execute w/ “Sample, tune, sweep”

Thank you!

(and ask me about AI accelerated migrations)

