# Rake

Why?

# What is Rake?

- A Ruby library for scheduling and executing tasks
- Includes a Domain-Specific Language that looks like a "configuration" file (but is actually proper Ruby)
- Includes many new utilities specific to

# Rake

Ruby


Rakefile

# Rake

Single Language

One debugger

Ruby

Rake

defines *Domain-Specific Language*

Rakefile

is Ruby code

# We'll leave it there for now...

# Build systems should be:

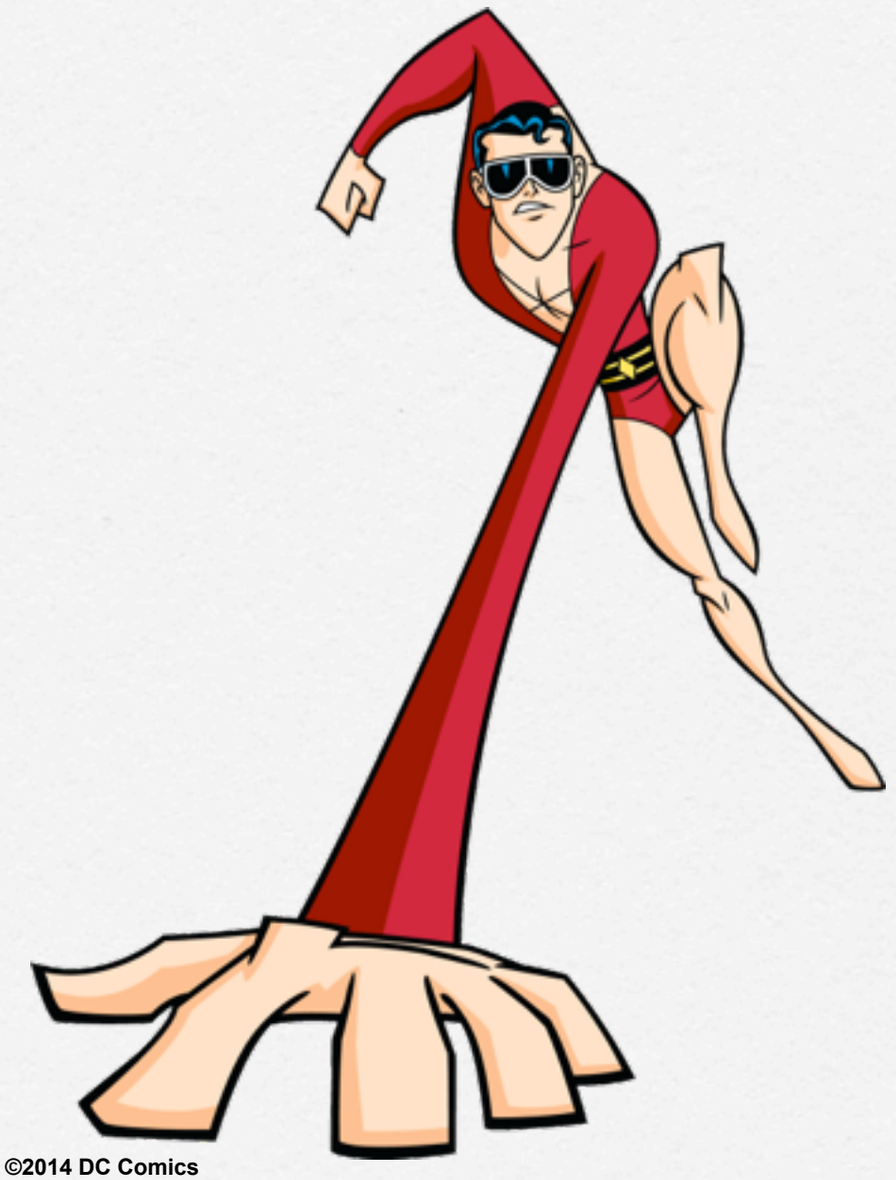
Flexible

Uniform

Maintainable

Speedy

# Flexible



# Make/Ant/Etc...

## Build Interpreter

Config file

*Tasks*

*Dependencies*

Custom Actions

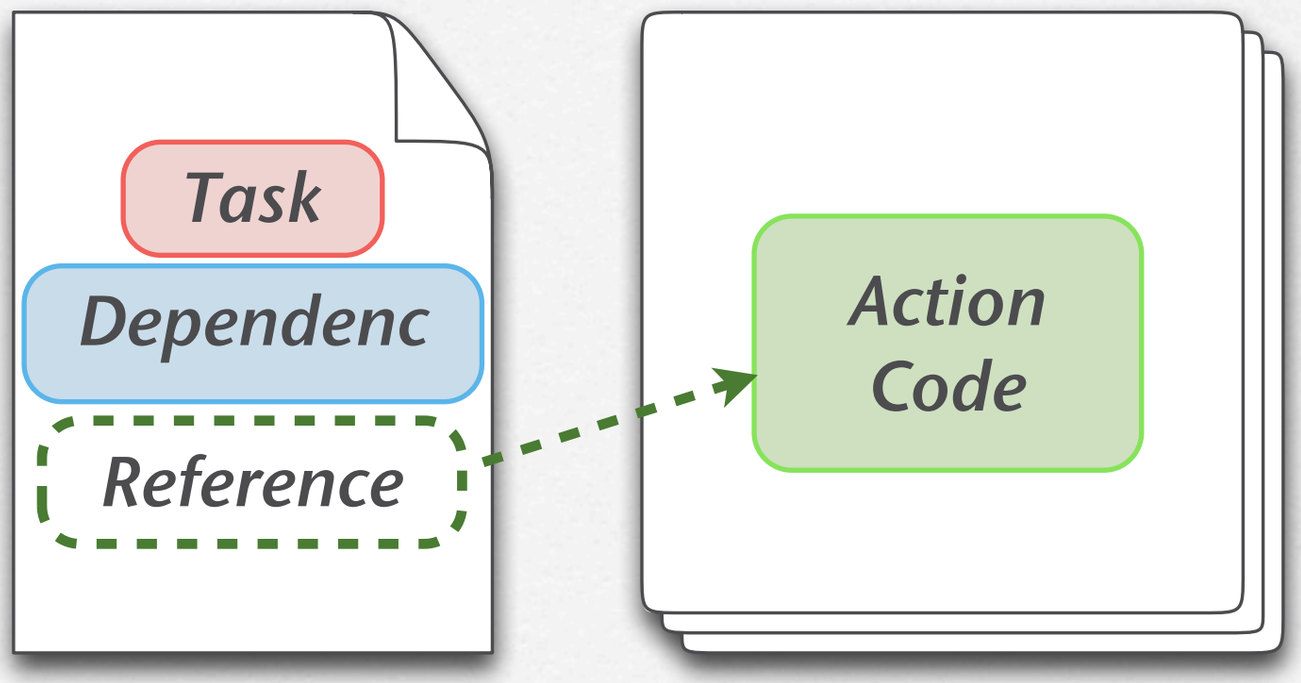
*Actions*

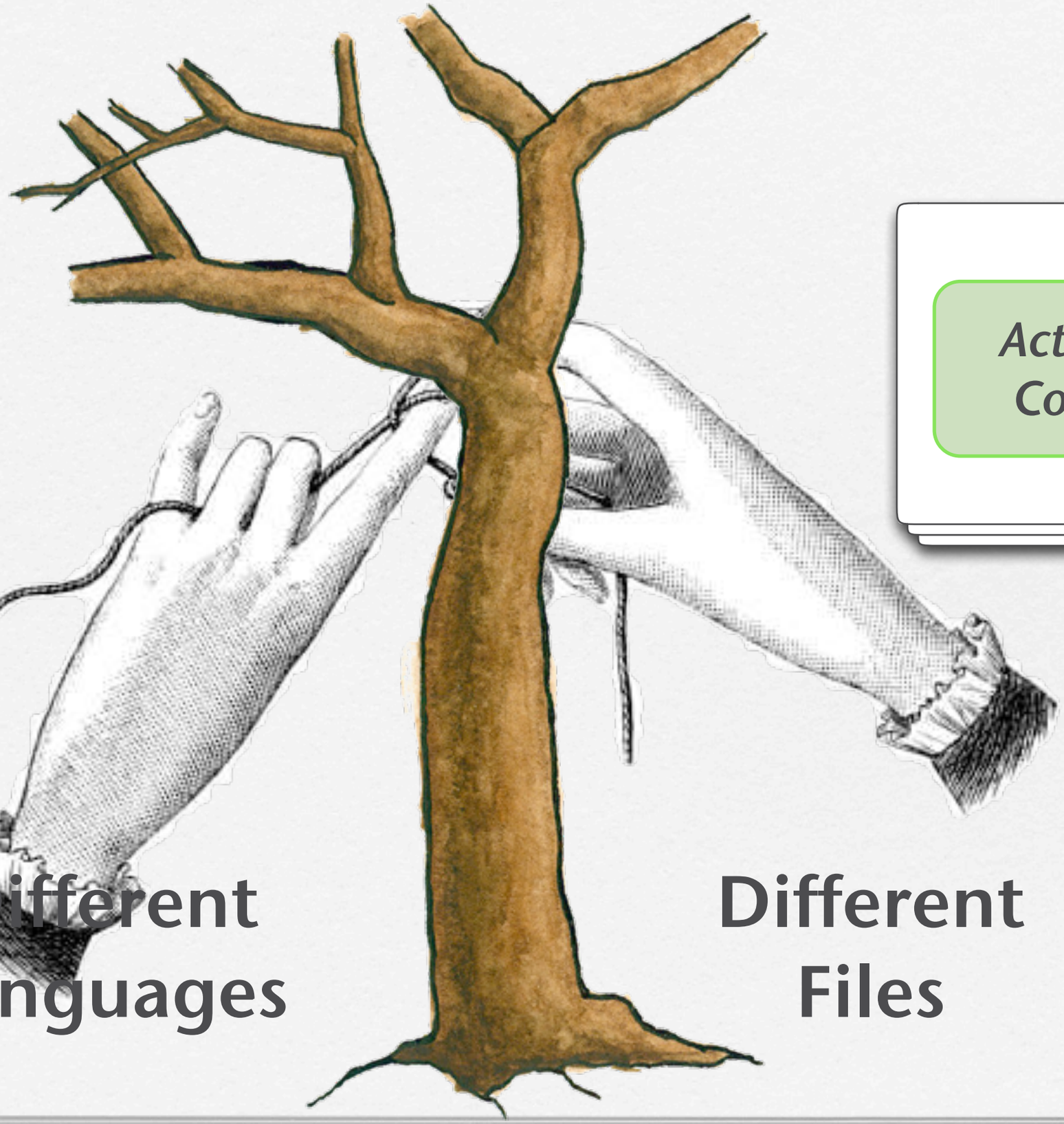
Built-in Actions

*Actions*

# Extending the build

1. Write a custom action
2. Reference it in the configuration file



*Action Code*

*Reference*

Different Languages

Different Files

# Rakefile

```
task :build do |t|
  ...Ruby code (makes the build)...
end


task :zip => :build do |t|
  ...Ruby code (zips the build)...
end
```

# This doesn't look like Ruby. Are you sure it's the same language?

```
task :zip => :build do |t|
    ...Ruby code (zips the build)...
end
```

# It's a Domain-Specific Language defined in Rake's API

```
task :zip => :build do |t|
    ...Ruby code (zips the build)...
end
```

# Rake::Task[] ⇢ task()

```
zip_task = Rake::Task[:zip]
zip_task.enhance(Rake::Task[:build])
zip_task.enhance do |t|
  ...code that zips the build...
end
```

# task()

```
zip_task = task(:zip)
zip_task.enhance(task(:build))
zip_task.enhance do |t|
   ...code that zips the build...
end
```

# task() can take a hash literal

```
zip_task = task(:zip).enhance(task(:buil
zip_task.enhance do |t|
  ...code that zips the build...
end
```

# task( {key => value} )

```
zip_task = task({:zip=>:build})
zip_task.enhance do |t|
  ...code that zips the build...
end
```

# task() can take a block

```
task({:zip=>:build}).enhance do |t|
  ...code that zips the build...
end
```

# task() can take a block

```
task({:zip=>:build}) do |t|
  ...code that zips the build...
end
```

```
task(:zip=>:build) do |t|
  ...code that zips the build...
end
```

```
task :zip => :build do |t|
    ...code that zips the build...
end
```

```
task :zip => :build do |t|
    ...code that zips the build...
end
```

# Highly Dynamic

- Everything the build needs from Ruby is loaded dynamically over the net
- Classes can be built at runtime
- IRB Interpreter to try without-compiling

# Plays well

- You can call Win32 APIs directly from Ruby code
- If you need more than that, you can use IronRuby, which is built using .NET

# Uniform

# Uniform

- The build server uses the same rake commands that you use on your local machine
- All the items needed to make a build are checked in (except for VS)

# Maintainable

# Ruby is a common language

| Position May 2012 | Position May 2011 | Programming Language | Ratings May 2012 | Delta May 2011 | Status |
|---|---|---|---|---|---|
| 1 | 2 | C | 17.346% | +1.18% | A |
| 2 | 1 | Java | 16.599% | -1.56% | A |
| 3 | 3 | C++ | 9.825% | +0.68% | A |
| 4 | 6 | Objective-C | 8.309% | +3.30% | A |
| 5 | 4 | C# | 6.823% | -0.72% | A |
| 6 | 5 | PHP | 5.711% | -0.8% | A |
| 7 | 8 | (Visual) Basic | 5.457% | +0.96% | A |
| 8 | 7 | Python | 3.819% | -0.76% | A |
| 9 | 9 | Perl | 2.805% | +0.57% | A |
| 10 | 11 | JavaScript | 2.135% | +0.74% | A |
| 11 | 10 | Ruby | 1.451% | +0.03% | A |
| 12 | 26 | Visual Basic .NET | 1.274% | +0.79% | A |
| 13 | 21 | PL/SQL | 1.119% | +0.62% | A |
| 14 | 13 | Delphi/Object Pascal | 1.004% | -0.07% | A |
| 15 | 15 | Lisp | 0.941% | -0.01% | A |
| 16 | 24 | Logo | 0.839% | +0.35% | A-- |
| 17 | 17 | Pascal | 0.808% | +0.10% | A |
| 18 | 18 | Transact-SQL | 0.654% | -0.04% | A- |
| 19 | 16 | Ada | 0.649% | -0.1% | B |
| 20 | 12 | Lua | 0.566% | -0.54% | B |

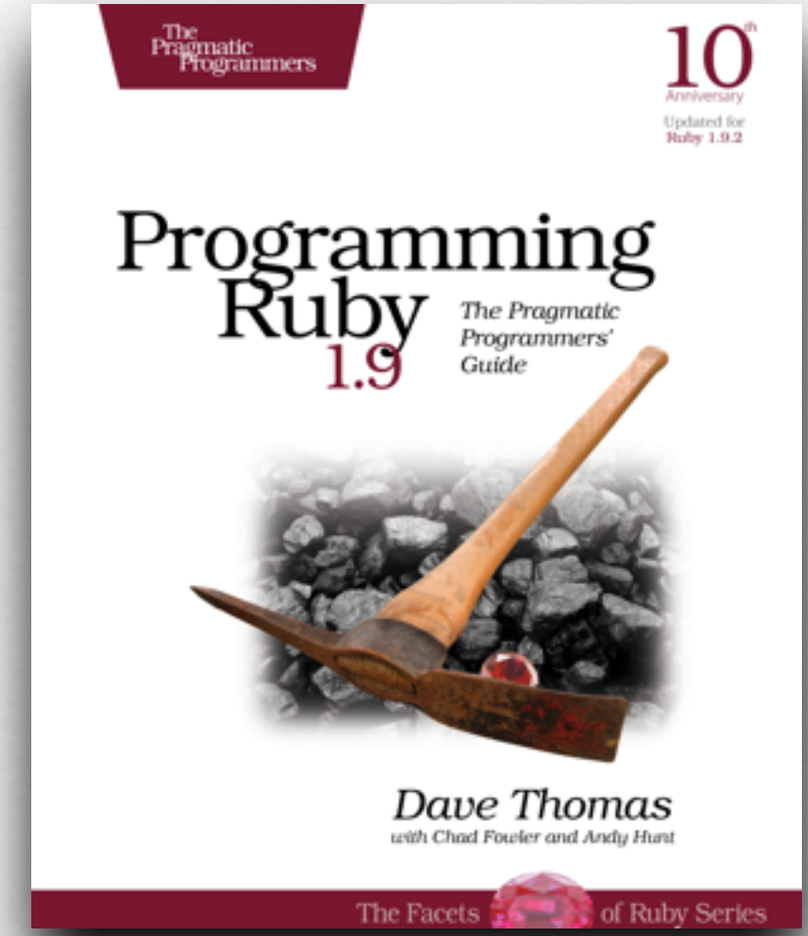
**Ruby**

***Position (around 10)***

**TIOBE Programming Community Index for May 2012**

# Documentation

- http://rake.rubyforge.org/
- "Pickaxe" book (the best)
- Many articles
- Amazon Rubu Books




# Concise

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  printf("Hello, world\n");
  return EXIT_SUCCESS;
}
```

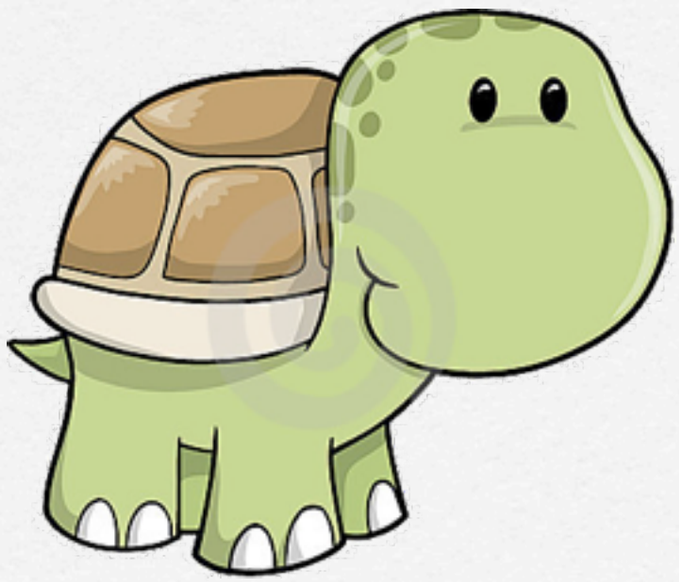
```
class Hello {
 static void Main() {
  System.Console.WriteLine("Hello, World");
 }
}
```

```
document.write("Hello, World");
```

Java C# Ruby Python

# Speedy? Well...

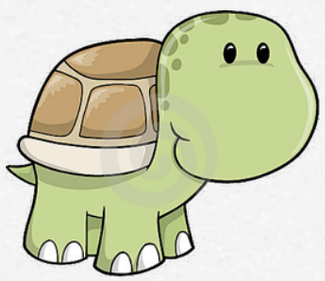
## All that flexibility isn't free



Ruby is not as fast as C, C++, C#, or Java

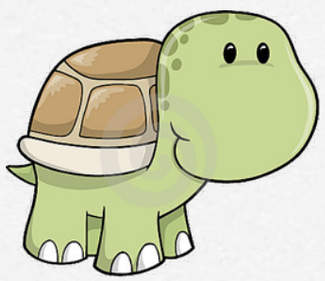
# But...

Think about software. You use a high level of abstraction so you can write software quickly, but when you isolate the slow spots, you write them in assembly or execute a compiled process.

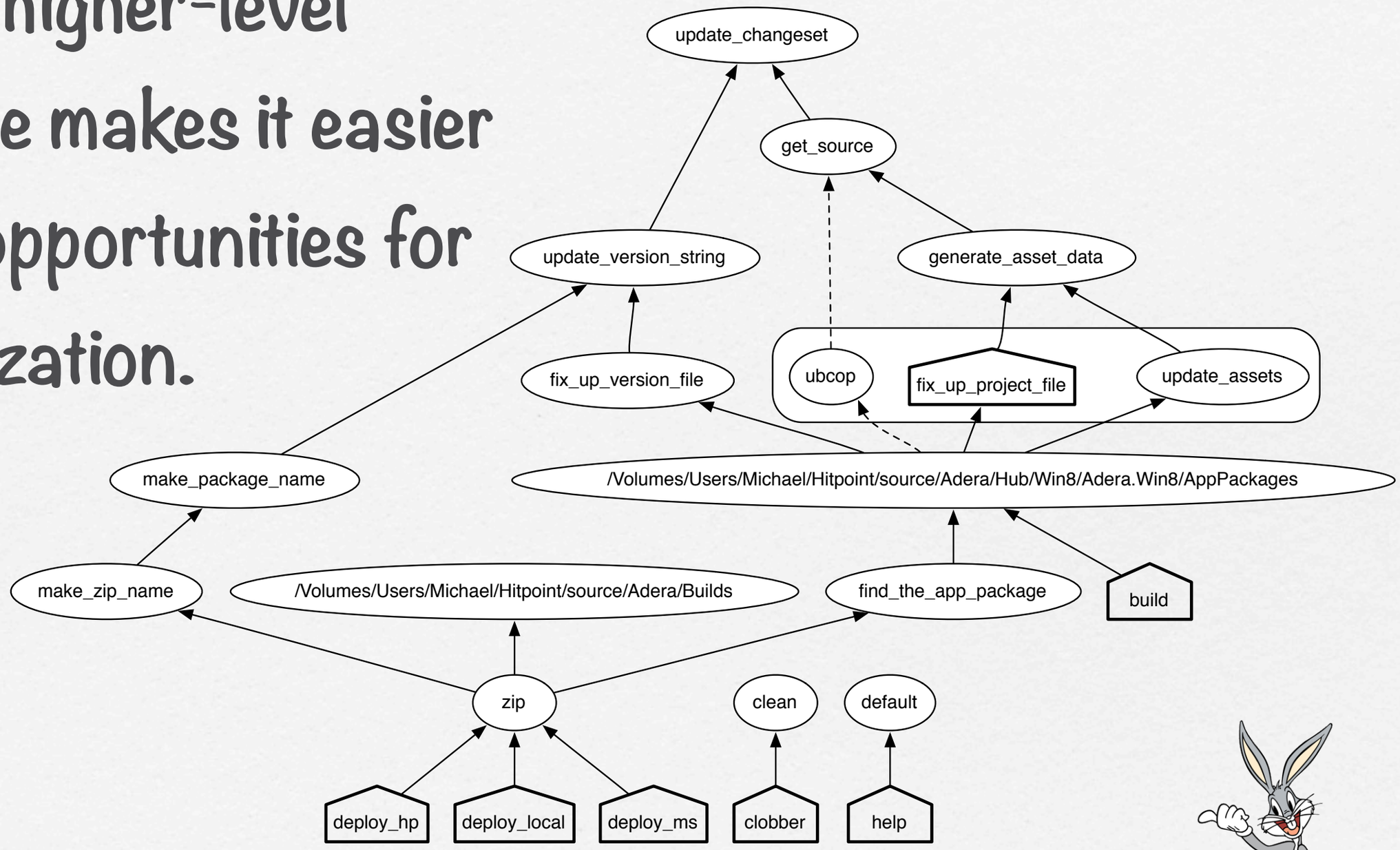


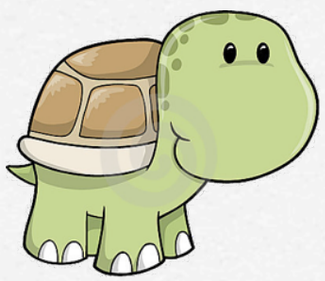
# But...

Build systems are packed with calling external tools to process images, compile files, compress sounds, etc. The demands on the runtime of the build system are very low.



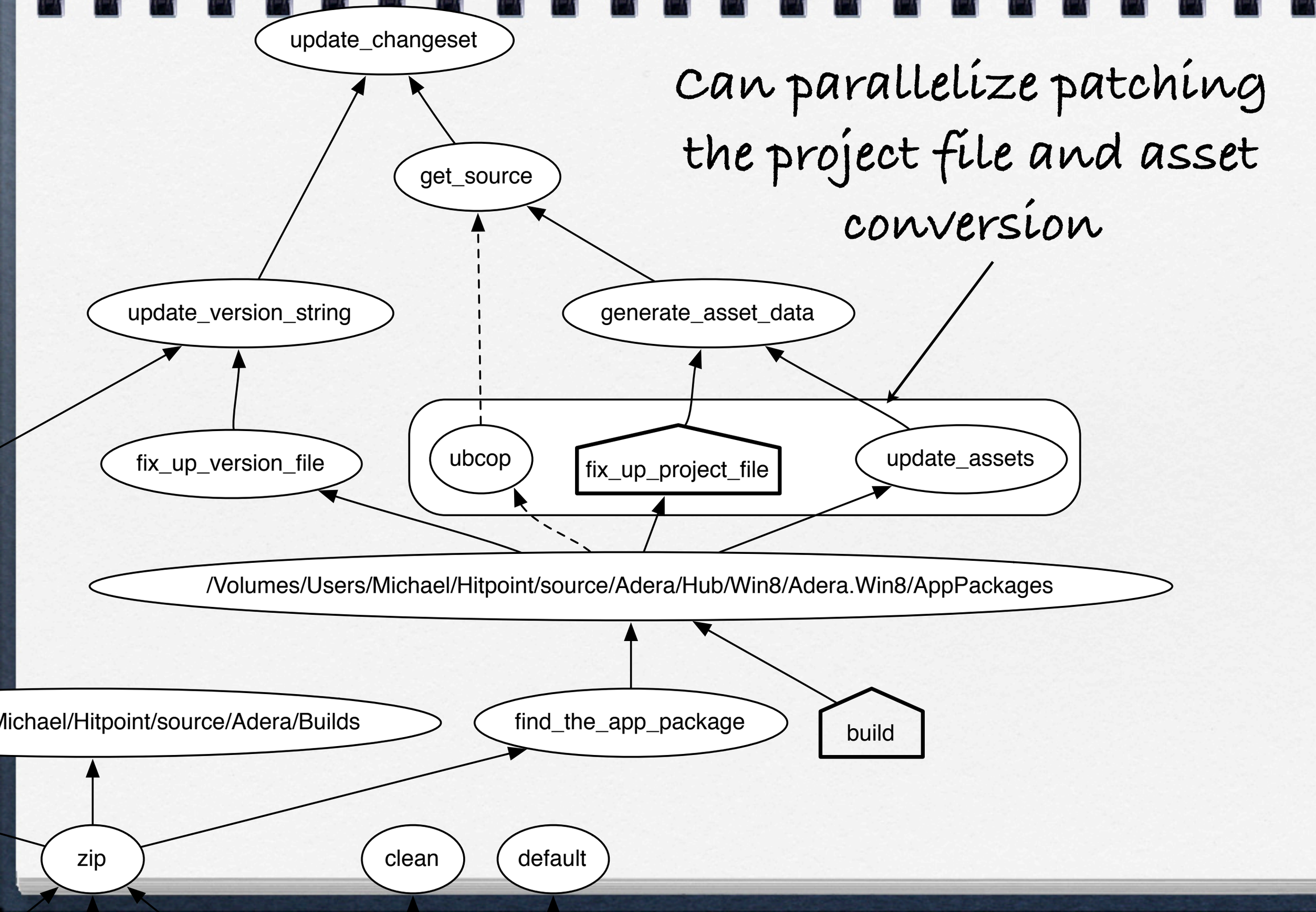
# Using a higher-level language makes it easier to find opportunities for parallelization.



actual rake diagram generated from our build

Can parallelize patching the project file and asset conversion



While your *build* is loading...

# Thanks

## Question Time