Future of OpenEdge build process

a.k.a. Why Ant/PCT should die

Gilles QUERRET ● Riverside Software
Peter JUDGE ● Progress Software
History

- PCT was born in 2003
- Consistent builds across OpenEdge versions, operating systems, DB types, ...
- No Jenkins at this time

- Soon adopted by many companies, and got improvement requests and patches from multiple vendors
What’s wrong

- XML syntax is quite cumbersome
- Conditional behavior is hard to write and maintain (additional libraries required)
- Specific to a directory structure
What happened in the Java world?

- Initial release of Maven in 2004
- Quickly became the de facto build standard for open source Java projects
- And widely used in commercial projects, in-house development...
- Brought two key features:
  - Convention over configuration
  - Dependency management
Convention over configuration

- The directory structure is imposed by the Maven builder
  - Java source code stored in src/main/java
  - Unit tests stored in src/test/java
  - Web application stored in src/main/web
- As long as you follow the conventions, Maven will be able to build your project
- You only have to describe exceptions
  - «Maven turn complex things easy, but easy things can become a nightmare»
Dependency management

- Use public and/or private repositories
- A project only needs to declare its main dependencies
  - Dependency on commons-fileutils 1.0.1 and slf4j 2.6
- Which are downloaded during the build
- Hierarchy generated on the fly
From Eclipse

- Maven projects are configured on the fly
- Classpath is set up automatically
- Source and test entries automatically added
- Initial Maven build is executed
- Dependency management done on the fly
And at the beginning there was

- **make**
- **rake**
- **schmant**
- **rant**
- **ant**
- **maven**
- **gant**
- **ivy**
- **buildr**
- **gradle**

...some loved ruby

- No XML!

- Too much work for some devs!

- The groovy way

- Declarative + dep-mgmt = the standard?

- WTF is maven?

- The finalists?

...with ruby, java, scala

...with groovy, java
With OpenEdge

- Huge monolithic projects are the norm rather than the exception
  - Long build times
- Large build scripts – Reusability is quite limited
- No code convention
- Developer Studio setup is long
- No dependencies
What could we do?