FrOSCon 2009

Gradle

Hans Dockter

Gradle Project Lead

mail@dockter.biz

About Me

- Founder and Project Lead of Gradle
- CEO of Domain Language Germany (www.domainlanguage.de).
    - Consulting and Training for DDD in Germany (with Eric Evans as partner)
- Trainer for Skills Matter (TTD, Patterns, DDD)
- In the old days: Committer to JBoss (Founder of JBoss-IDE)

Agenda

Why a new build system?

Introduction to Gradle

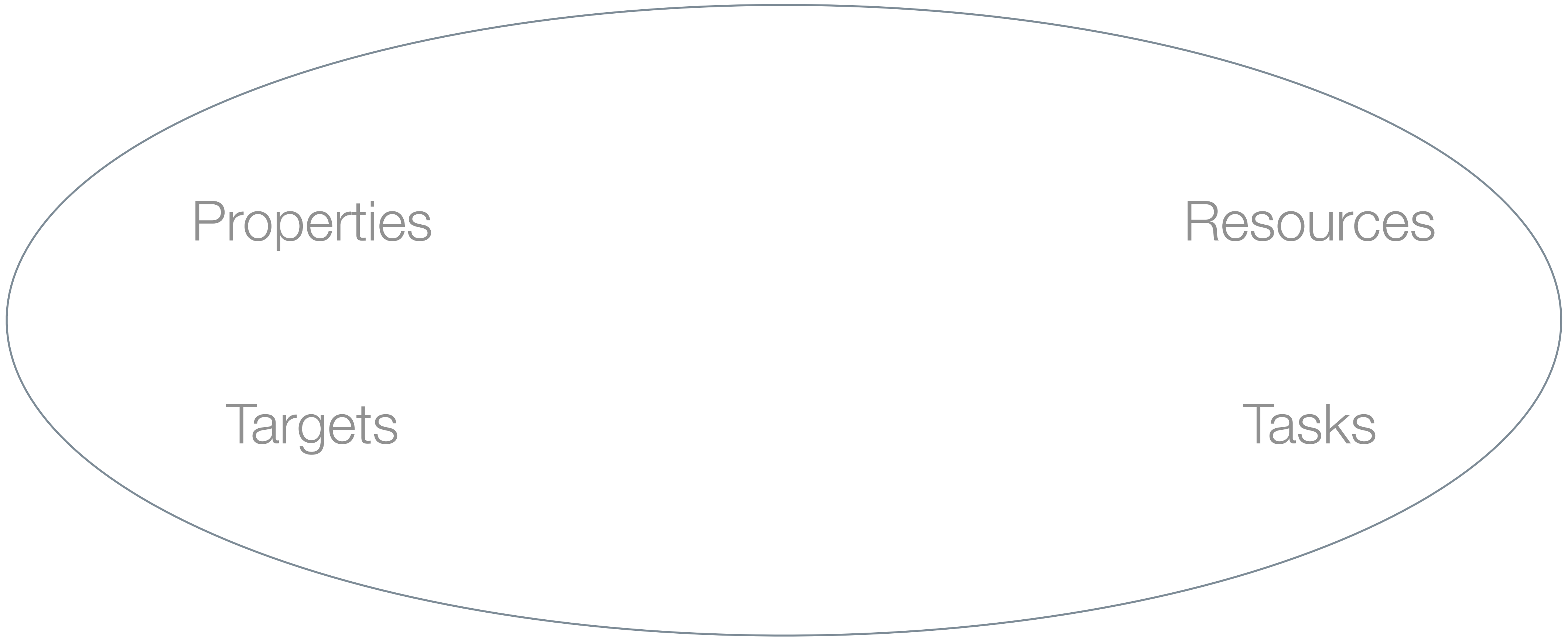
Why a new build system?

Our Vision (Quote from Moshé Feldenkrais)

- Make the impossible possible
- Make the possible easy
- Make the easy elegant

The old bulls …

Ant



Flexible Toolset via dependency based programming

build.**XML**

vs

dynamic language

build.XML

vs

dynamic language

```
4.times { i ->
   task “hello$i” {
       printMeth(i)
   }
}

void printMeth(int i) { println “I am task number $i” }
```

One Way Configuration &
Simple Elements

vs

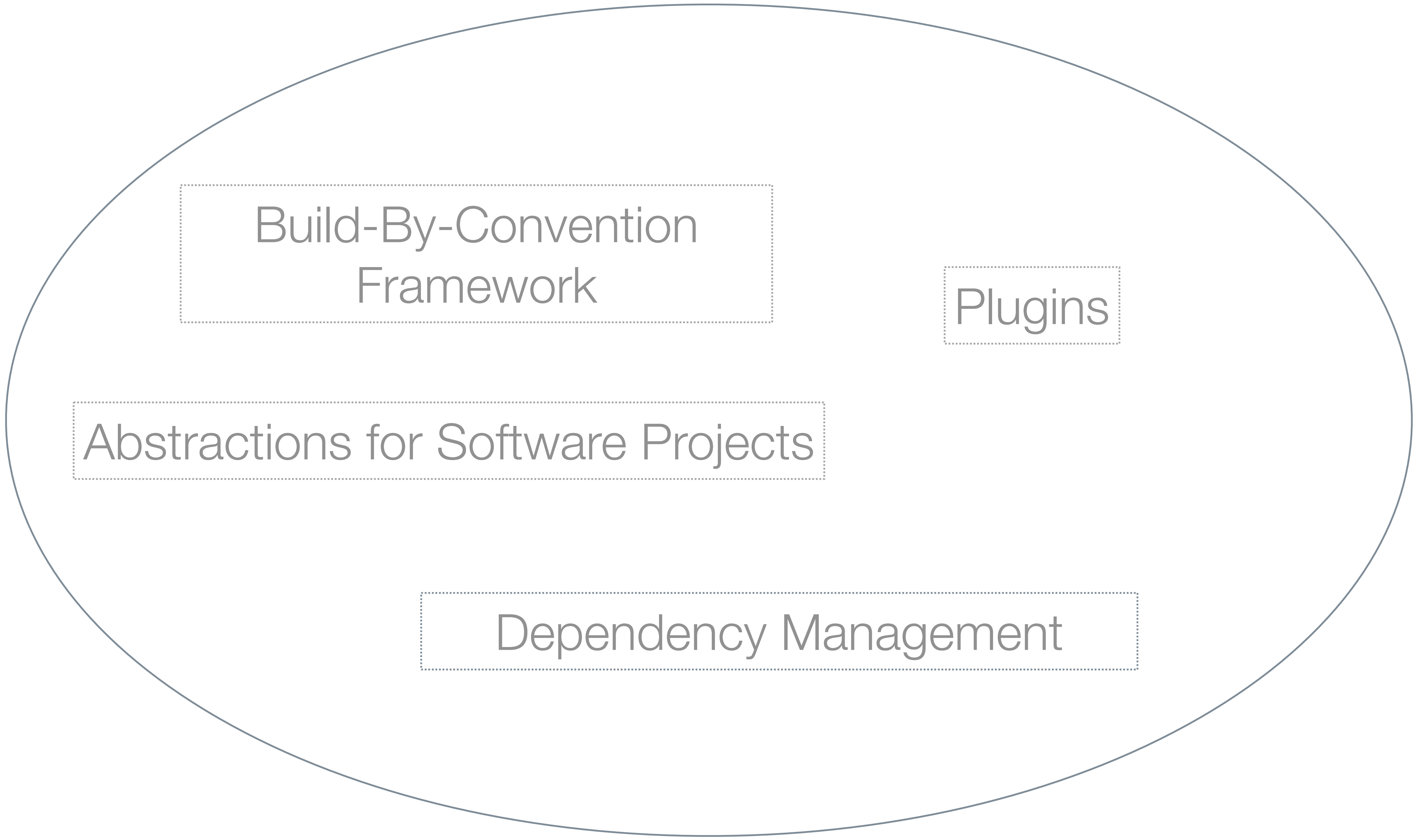
Rich Domain Model

Build By Convention

Multi-Project Builds

Live Demo - Hello World

Maven


Build-By-Convention Framework
Plugins
Abstractions for Software Projects
Dependency Management

pom.**XML**

vs

dynamic language

No dependency based programming

Convention **instead of** Configuration

Framework &
One Way Configuration

vs

Toolset based on
Rich Domain Model

Frameworkitis ...

... is the disease that a framework wants to do too much for you or it does it in a way that you don’t want but you can’t change it. It’s fun to get all this functionality for free, but it hurts when the free functionality gets in the way. But you are now tied into the framework. To get the desired behavior you start to fight against the framework. And at this point you often start to lose, because it’s difficult to bend the framework in a direction it didn’t anticipate. Toolkits do not attempt to take control for you and they therefore do not suffer from frameworkitis.

(Erich Gamma)

Solution

Because the bigger the framework becomes, the greater the chances that it will want to do too much, the bigger the learning curves become, and the more difficult it becomes to maintain it. If you really want to take the risk of doing frameworks, you want to have small and focused frameworks that you can also probably make optional. If you really want to, you can use the framework, but you can also use the toolkit. That's a good position that avoids this frameworkitis problem, where you get really frustrated because you have to use the framework. Ideally I'd like to have a toolbox of smaller frameworks where I can pick and choose, so that I can pay the framework costs as I go.(Erich Gamma)

Live Demo - Java

There
are
no
simple builds

Project Automation

- A build can do far more than just building the jar
- Often repetitive, time consuming, boring stuff is still done manually
  - Many of those tasks are very company specific
  - Maven & Ant are often not well suited for this

The Gradle Build

- Gradle is build with Gradle
- Automatic release management
- Automatic user’s guide generation
- Automatic distribution
- Behavior depends on task execution graph

Release Management

- The version number is automatically calculated
- The distribution is build and uploaded to codehaus
- For trunk releases, a new svn branch is created.
- A tag is created.
- A new version properties file is commited.
- The download links on the website are updated

User’s Guide

- The user’s guide is written in LaTeX and generated by our build
- The source code examples are mostly real tests and are automatically included
- The expected output of those tests is automatically included.
- The tests are run
- The current version is added to the title

Uploading & Execution Graph

- Based on the task graph we set:
  - Upload Destination
  - Version Number

Gradle Overview 1

- A flexible general purpose build tool
  - Offers dependency based programming with a rich API
- Build-by-convention plugins on top
- Powerful multi-project support
- Powerful dependency management based on Apache Ivy
- Ant tasks are first class citizens

Gradle Overview 2

- Build Scripts are written in Groovy
  - We get our general purpose elements from a full blown OO language
  - The perfect base to provide a mix of:
    - Small frameworks, toolsets and dependency based programming
  - Rich interaction with Java
  - Gradle is NOT a framework
- Gradle is mostly written in Java with a Groovy DSL layer on top
- Offers good documentation (70+ Pages user's guide)
- Commiter -> Steven Devijver, Hans Dockter, Tom Eyckmans, Adam Murdoch, Russel Winder

Why **Groovy** scripts?

Why **Groovy** scripts?

```
['Maven', 'Ant', 'Gradle'].findAll { it.indexOf('G') > -1 }
```

Why **Groovy** scripts?

```
['Maven', 'Ant', 'Gradle'].findAll { it.indexOf('G') > -1 }
```

Why not **JRuby** or **Jython** scripts?

Why **Groovy** scripts?

```
['Maven', 'Ant', 'Gradle'].findAll { it.indexOf('G') > -1 }
```

Why not **JRuby** or **Jython** scripts?

Why a **Java** core?

Java Plugin

```
usePlugin(‘java’)
manifest.mainAttributes([
   'Implementation-Title': 'Gradle',
   'Implementation-Version': '0.1'
])
dependencies {
   compile "commons-lang:commons-lang:3.1"
   runtime “mysql:mysql-connector-java:5.1.6”
   testCompile "junit:junit:4.4"
}
sourceCompatibility = 1.5
targetCompatibility = 1.5
test {
   exclude '**/Abstract*'
}

task(type: Zip) {
    zip() {
        files(dependencies.runtime.resolve() // add dependencies to zip
        fileset(dir: "path/distributionFiles")
    }
}
```

Dependencies

- DSL on top of Apache Ivy
- Integrates with Ivy/Maven infrastructure
- Support for client modules.
- Support for storing libs in SVN.
- Extremely powerful and flexible.
- Compile only against first level deps.

Multi-Project Builds

- Configuration Injection
- Partial builds
- Separate Config/Execution Hierarchy
- Arbitrary Multiproject Layout

Multi-Project Build

- ultimateApp
  - api
  - webservice
  - shared

```
// build script ultimate app
subprojects {
   usePlugin(‘java’)
   dependencies {
      compile "commons-lang:commons-lang:3.1"
      testCompile "junit:junit:4.4"
   }
   test {
      exclude '**/Abstract*'
    }
}
```

Organizing Build Logic

- No unnecessary indirections
- If build specific:
    - Within the script
    - Build Sources
- Otherwise: Jar

Gradle Wrapper

- Use Gradle without having Gradle installed
- Useful for CI and open source projects

Production Ready?

- YES! (If you don't need a missing feature).
- There are already large enterprise builds migrating from Ant and Maven to Gradle
- Expect 1.0 in summer
- **Be aware**: No guaranteed API stability until 1.0
- We are transparent:
    - The user's guide list all missing features which might be known from other tools.

Questions