DITA Gradle and Git

DITA-OT day 2018 - Rotterdam

The company - L-Acoustics

French company based near Paris.

Leader in professional audio solutions.

Hollywood bowl

Lorde Melodrama tour

Paris fashion week

The team – the work

We are

-> 5 tech writers (2 apprentices)

-> 1 information architect / toolsmith

-> embedded in R&D

We produce

-> PDFs (user and maintenance manuals, marketing documents)

-> HTML5 (embedded help)

Tools

Docs as Code!

-> Using software tools to build documentation.

-> Leveraging the software team DevOps capability.

-> No CCMS.

Tools

Git

Gradle

Artifactory

Gitlab Runner

Tools – Git

Decentralized version control system.

Widely used today in software development.

Authored files are stored here.

Each system/software has its own repository.

Tools – Artifactory

Package repository manager.

Store resources created from other sources.

Handles dependency declaration.

Compatible with Gradle.

Tools – Gradle

Open-source build automation system.

Groovy-based with a task oriented syntax.

Eero Helenius DITA-OT and Saxon plug-ins.

Used by the DITA-OT team!

Tools – Gitlab runner

GitLab Continuous Integration solution.

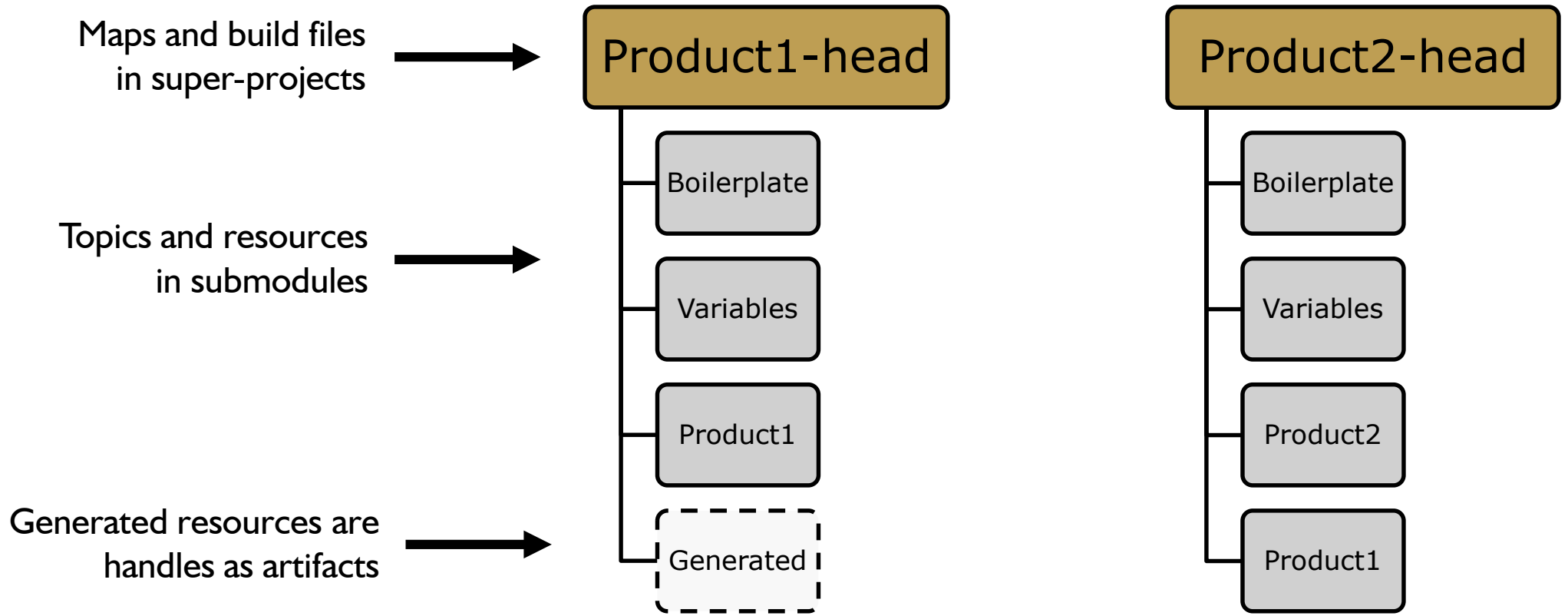
Integrated with Git concepts.

Uses the same Gradle build as the author.

Basic web platform to trigger publishing.

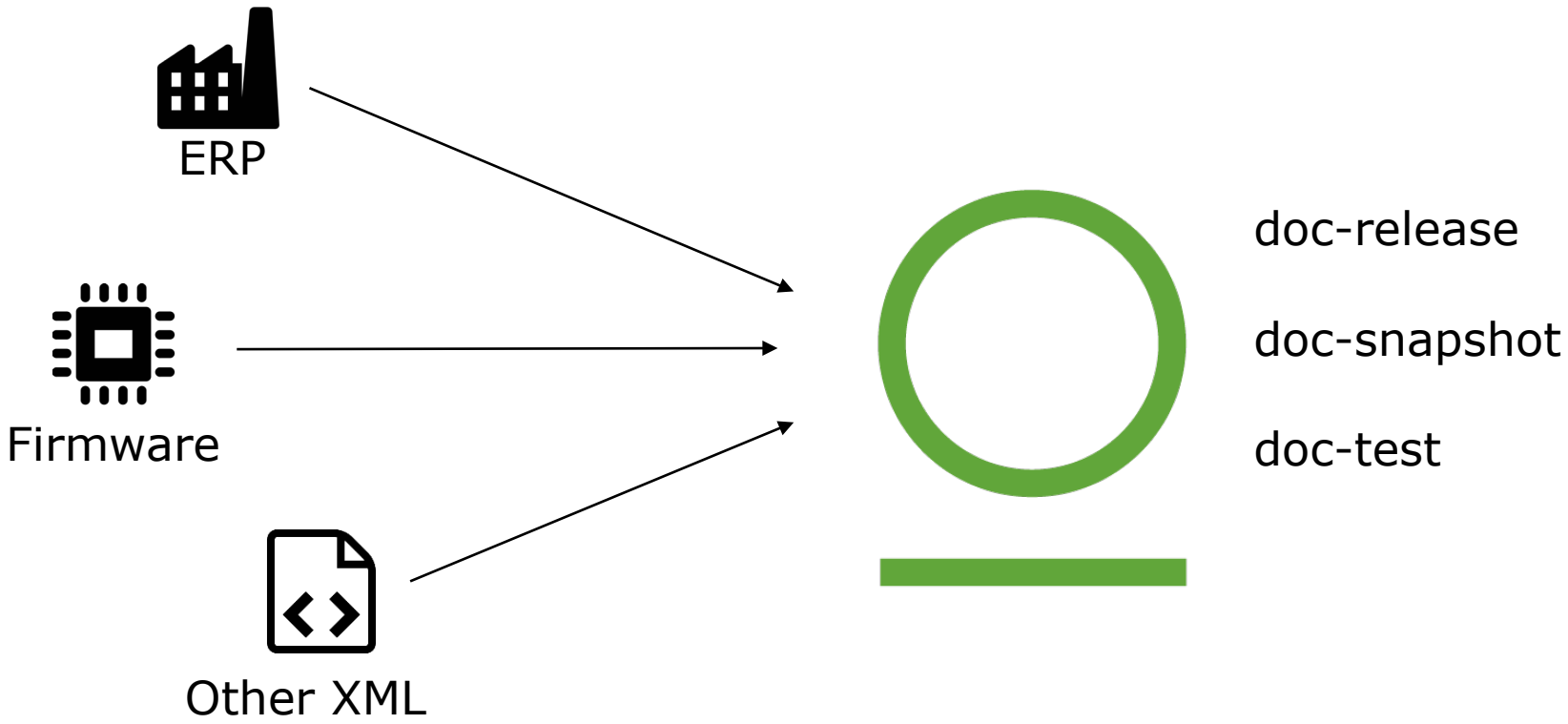
Source management - Git

We make extensive use of **submodules.**



Source management - Artifactory

Generated content is stored in Artifactory.



DITA-OT plug-ins

One document type = one transform

-> 34 plug-ins with transforms

Each feature block has a plug-in

-> 64 plug-ins in total

DITA-OT plug-ins

How do we deal?

Dependency management!

```
+--- com.lgroup:help:2.6
|    +--- com.lgroup:base:1.3
|    +--- com.lgroup:l-html:+ -> 1.2
|    |    +--- com.lgroup:boilerplate:+ -> 1.0
|    |    +--- com.lgroup:strings:+ -> 1.0
|    |    \--- com.oxygenxml:html.embed:+ -> 1.0
|    +--- com.lgroup:help.css:+ -> 1.2
|    +--- com.lgroup:videoJs:+ -> 1.0
|    \--- com.lgroup:bootstrap:3.4
\--- com.oxygenxml:media:+ -> 2.0
```

DITA-OT plug-ins

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="dita-ot/plugin.rnc" type="application/relax-ng-compact-syn
<plugin id="com.lgroup.help.user-guide">
    <feature extension="package.version" value="1.4">
        <!-- 1.2 com.lgroup.help updated RDDOCM-322 -->
        <!-- 1.4 L1ADV-882 features -->
    </feature>
    <require plugin="com.lgroup.help" version="2.6"/>
    <require plugin="com.oxygenxml.media" version="+"/>

    <transtype name="help.user-guide" extends="help" desc="User Guide">
        <param name="cover-style" type="string"/>
    </transtype>

    <feature extension="dita.conductor.target.relative" file="build.xml"/>

</plugin>
```

Uploaded to Artifactory and installed in a local Maven repo.

Dependencies built by a saxon-gradle task from the plug-in `<require>` element.

```
+--- com.lgroup:help:2.6
|    +--- com.lgroup:base:1.3
|    +--- com.lgroup:l-html:+ -> 1.2
|    |    +--- com.lgroup:boilerplate:+ -> 1.0
|    |    +--- com.lgroup:strings:+ -> 1.0
|    |    \--- com.oxygenxml:html.embed:+ -> 1.0
|    +--- com.lgroup:help.css:+ -> 1.2
|    +--- com.lgroup:videoJs:+ -> 1.0
|    \--- com.lgroup:bootstrap:3.4
\--- com.oxygenxml:media:+ -> 2.0
```

DITA-OT plug-ins

Plug-ins dependencies are resolved by the Gradle build.

They are copied in the DITA-OT `plugins` folder.

The `dita --install` command is executed.

DITA-OT plug-ins – development

Developer

Git branch in plug-in repo.

Git branch in build repo.

Plug-ins published to test repo.

Tester

Git branch in build repo

The developer creates a branch in the build repo and changes the version numbers of the updated plug-ins.

Build

Based on the **dita-ot-gradle** plug-in by Eero Helenius.

-> input is a list of files

-> picks up files with the same name as the input (properties, ditaval)

-> great developer!

Build

Gradle handles all dependencies

-> DITA-OT

-> plug-ins

-> generated content

All dependencies are declared in two files.

Build – configuration files

deps.gradle

```
dependencies {
    //ditamaps
    ditamaps fileTree(dir: "../.").include("L1_help_release_notes_publi.ditamap")

    //fluidTopics fileTree(dir: "../.").include("L1_help_release_notes_publi.ditamap")

    //variables
    variables group: 'com.lgroup', name: 'generated', version: "SNAPSHOT", ext: 'zip', changing: true

    //external plugins
    externalPlugins 'dita-community:org.dita-community.jetpack:master@zip'
}

ext{
    category = "software"
    peers = "no"
}
```

Build – configuration files

deps.gradle

input lists

```
dependencies {
    //ditamaps
    ditamaps fileTree(dir: "../.").include("L1_help_release_notes_publi.ditamap")

    //fluidTopics fileTree(dir: "../.").include("L1_help_release_notes_publi.ditamap")

    //variables
    variables group: 'com.lgroup', name: 'generated', version: "SNAPSHOT", ext: 'zip', changing: true

    //external plugins
    externalPlugins 'dita-community:org.dita-community.jetpack:master@zip'
}

ext{
    category = "software"
    peers = "no"
}
```

Build – configuration files

deps.gradle

```
dependencies {
    //ditamaps
    ditamaps fileTree(dir: "../.").include("L1_help_release_notes_publi.ditamap")

    //fluidTopics fileTree(dir: "../.").include("L1_help_release_notes_publi.ditamap")

    //variables
    variables group: 'com.lgroup', name: 'generated', version: "SNAPSHOT", ext: 'zip', changing: true

    //external plugins
    externalPlugins 'dita-community:org.dita-community.jetpack:master@zip'
}

ext{
    category = "software"
    peers = "no"
}
```

input lists

content dependency

Build – configuration files

deps.gradle

```
dependencies {
    //ditamaps
    ditamaps fileTree(dir: "../.").include("L1_help_release_notes_publi.ditamap")

    //fluidTopics fileTree(dir: "../.").include("L1_help_release_notes_publi.ditamap")

    //variables
    variables group: 'com.lgroup', name: 'generated', version: "SNAPSHOT", ext: 'zip', changing: true

    //external plugins
    externalPlugins 'dita-community:org.dita-community.jetpack:master@zip'
}

ext{
    category = "software"
    peers = "no"
}
```

input lists

content dependency

plug-in from Github

Build – configuration files

*_publi.ditamap

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA 1.3 Map//EN" "map.dtd">
<map xml:lang="en" product="  " otherprops="online">

  <title>User guide</title>

  <topicmeta>[11 lines]

  <!-- dita-ot parameters -->
  <data type="document-type">help.user-guide</data>
  <data type="parameter" value="args.help.logo">  </data>
  <data type="parameter" value="args.cover.groupSize">3</data>

```

Build – configuration files

*_publi.ditamap

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA 1.3 Map//EN" "map.dtd">
<map xml:lang="en" product="" otherprops="online">

  <title>User guide</title>

  <topicmeta>[11 lines]

  <!-- dita-ot parameters -->
  <data type="document-type">help.user-guide</data>
  <data type="parameter" value="args.help.logo"></data>
  <data type="parameter" value="args.cover.groupSize">3</data>

```

transtype

Build – configuration files

*_publi.ditamap

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA 1.3 Map//EN" "map.dtd">
<map xml:lang="en" product="" otherprops="online">

  <title>User guide</title>

  <topicmeta>[11 lines]

  <!-- dita-ot parameters -->
  <data type="document-type">help.user-guide</data>
  <data type="parameter" value="args.help.logo"></data>
  <data type="parameter" value="args.cover.groupSize">3</data>

```

transtype

parameters

Build – local vs server

Authors clone the build Git repository.

-> A gradle wrapper is included.

-> Available transtypes are listed in the build.

```
//document types
appli              group:"com.lgroup",    name:"appli",              version:"1.0",  ext:"zip",  transitive:true
help_new           group:"com.lgroup",    name:"help.new",           version:"1.4",  ext:"zip",  transitive:true
help_tutorials     group:"com.lgroup",    name:"help.tutorials",     version:"1.4",  ext:"zip",  transitive:true
help_user_guide    group:"com.lgroup",    name:"help.user-guide",    version:"1.4",  ext:"zip",  transitive:true
```

transtype

Build – authors

Authors set up their maps and update deps.gradle.

local build

```
$ ./gradlew buildAll
```

Only technical writers can build locally.

Build – contributors

Content admins set up the map and deps.gradle.

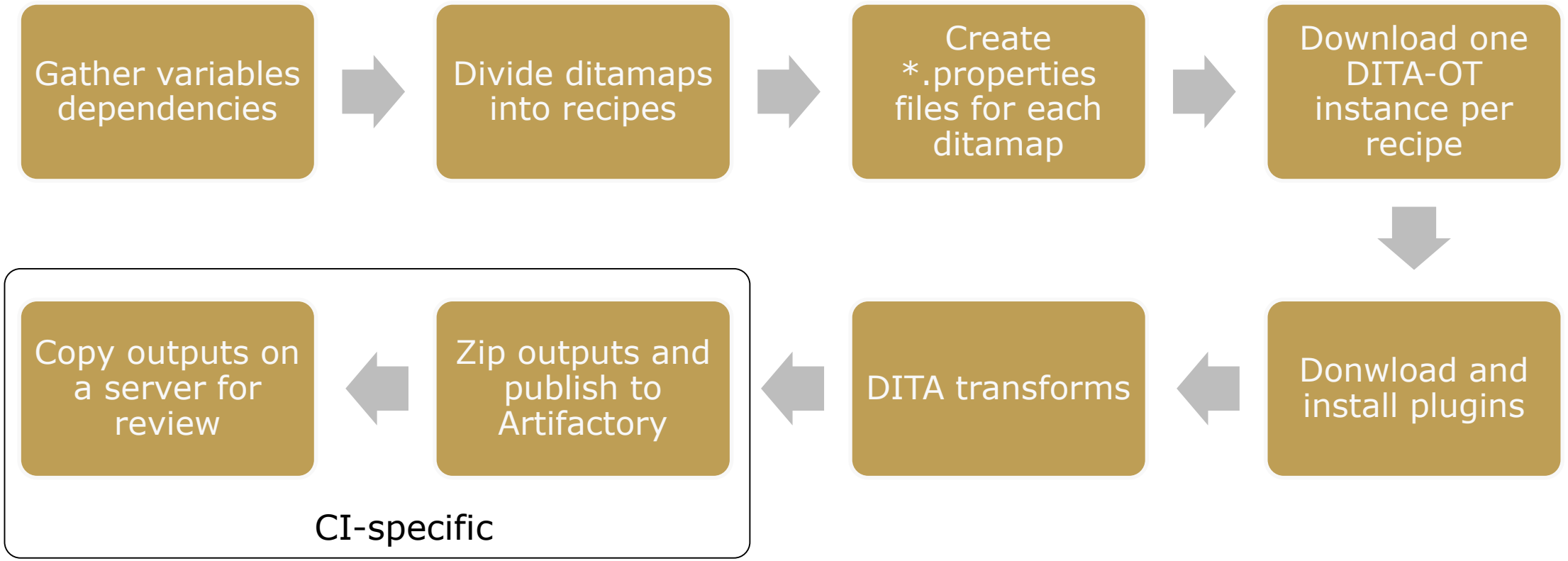
Contributors use XML Web Author

Changes are pushed to the server in a branch.

Branch is picked up by the Gitlab runner

Gradle script is executed and output is published.

Build – script outline



Why it werks

We make time to develop our tools.

We work with the software team.

It is simple enough for our SMEs.

DITA is adopted by our company.

We have full ownership of our tools.

We document our processes.

What is missing?

On our side

-> A more comprehensive dashboard.

-> An xml database to query more efficiently.

-> Unit tests...

On the other side

-> Another indirection level in DITA(-OT).

-> A bare DITA-OT distribution build.

THANKS!

Questions?

Tools

https://github.com/eerohele/

https://gradle.org

https://docs.gitlab.com/ce/ci/quick_start/

http://metadita.org/toolkit/happyhtml-background.html#why__newxform

Contact

lionel.moizeau@gmail.com