Managing dependencies on Document and data projects with Apache Ivy and Arousa.

Introduction

Why?
Avoid non Dry practices, organize code, maintain places where you can test and evolve components, reutilization.
- Where's the last version?
- Another time the same code?

Project Context.

Apache Ivy.
Matches the Ant/Xml/Xslt/XProc environment.
¿ Will it work ? ¿ Is there problems that I'm not aware ?

Arousa
Arousa introduction.

Why are we not using It already?

- Unawareness of the solution?
- Other Options (perceived as better options) ?
- Negative perception.
- Will it be worth the time and effort to adopt it? (learn it, implement it, maintain it?
- Wil it work?
- Will there be adoption/implementation problems that I'm not aware of?

The Plan for today.

We are going to iterate over use cases where we illustrate the Ivy view on its most relevant concepts.
We'll use Arousa as a mean to ease the start and explain how to work with it.

Configuring Arousa.

What do we needed ?

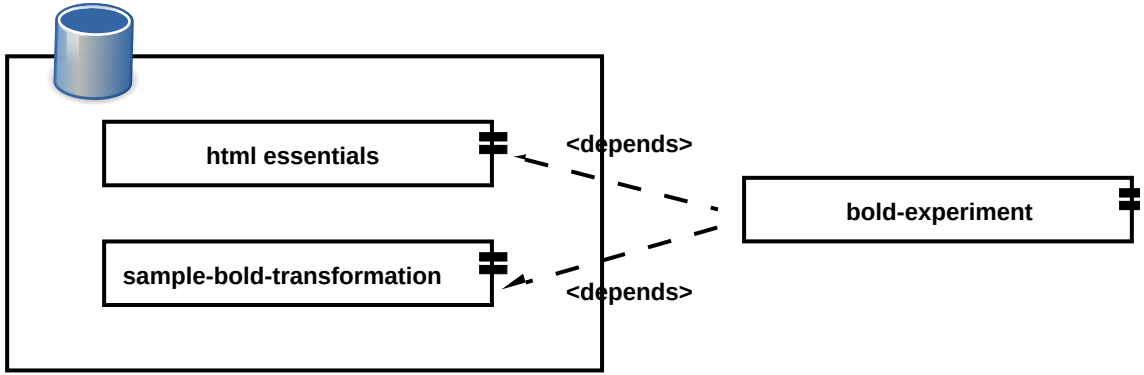
- Bash (cygwin), Java, Saxon, Apache Ant.
- ¿Ivy? !No!
- ¿ Morgana Xproc ? Depending on the project template.

We configure Arousa in a similar way as a JDK or Maven, by setting the Home.

```
export AROUSA_HOME=<Your_Installaton_path>
export PATH=$PATH:$AROUSA_HOME
```

* Some template projects may need configuration too (Xproc, conf/xproc.xml build file needs Morgana path)

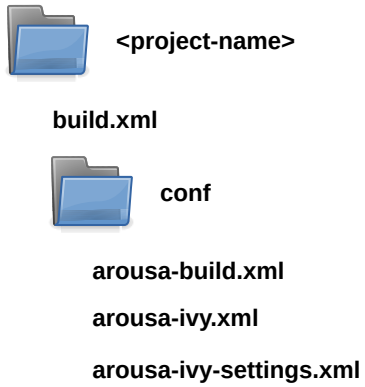
So … ¿ Is it worth it ? [A bold experiment]



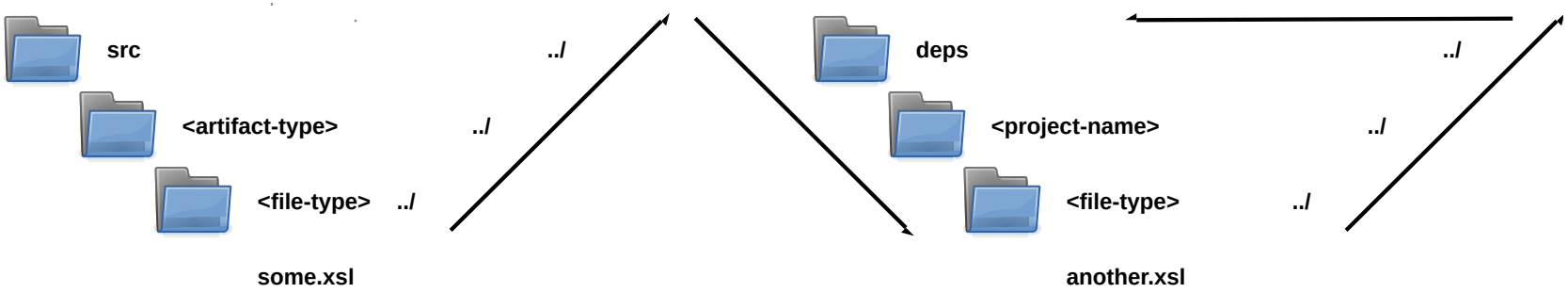
*** Exaxamples are available at https://github.com/vionta/MarkupUK2023**

Arousa File Structure

- Special files.
-- build.xml
-- conf/arousa-build.xml
-- conf/arousa-ivy.xml
-- conf/arousa-ivy-settings.xml



Arousa Folder Name convention



The abstraction problem, Artifact Types.

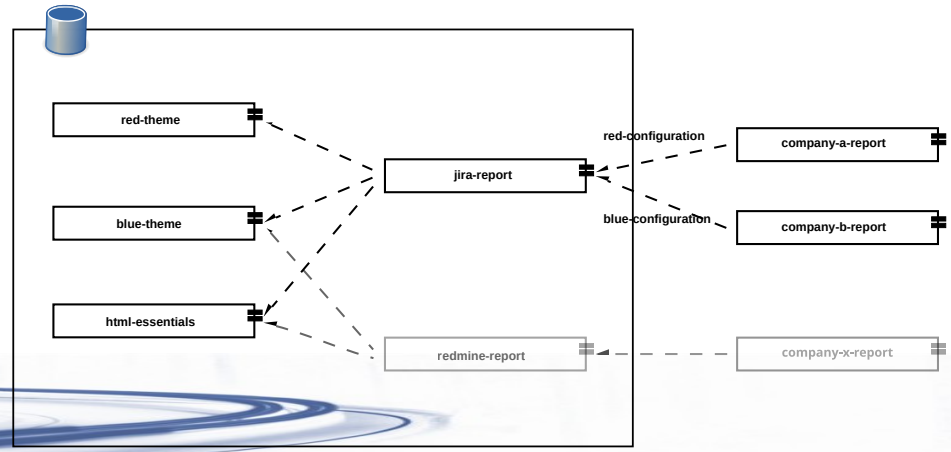
- When you are starting with something small concrete details are easier to understand.
- Ivy is an abstract tool, the lack of guidelines does not help on the first steps.

- What are the artifact types? and why should they be used for ?

Use Case: Manage content elements, from third party teams, with different cycles or steps.

Configuration Chains.

What is an Ivy configuration ? What should it be used for ?
Difference with Maven: Single output artifact.



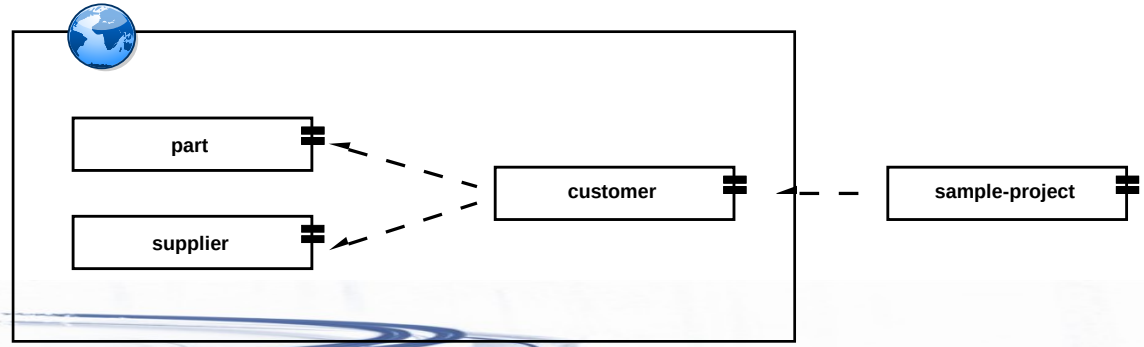
Resolvers.

Big List of resolver types (Local folders, Http, sftp, eclipse, Osgi, ...).
Chains of resolvers (flexibility).

Use case to organize the organization repositories in a decentralized manner and align them with the responsabilities.

Dual resolvers and data.

Dual resolvers allow to integrate third party artifacts and components into your dependency management, even when they don't and they won't use Ivy.



Conclusions

If you work with Ant and Xslt, Xproc, this seems a perfect mix.

It adds some complexity (files, configuration, steps). It may be too much for certain environments.

Ivy has a learning curve. The number of training materials may be small.

With a prebuilt approach like Arousa the point where the benefits overcome the initial difficulties can be reached easily.
fairly easy to add more components.

Conclusions II.

- Dependency management:
-- helps to maintain the code well structured and organized.
-- It helps to maintain the code as you can have specific places with test data and it's own space.

- Consider the adoption design in advance, taking your context into consideration.

- In general, the number of dependencies managed and the benefits from the practice exceeded our initial expectations.

Conclusions.

- Is it worth it?
- Any help ?

… Next Steps …