Alire: a library repository manager for the open source Ada ecosystem

Alejandro R. Mosteo
2018-jun-19
• Motivation
  – Problem
  – Use cases
  – Semantic versioning

• Overview for users
  – Final user
  – Open source developer
  – Software distributor

• Design highlights
  – Staying within pure Ada
  – Index format
MOTIVATION (personal)

• Experience with
  
  – Linux package managers:
    
    ```
    sudo apt install libgtkada16.1.0-dev
    ```
  
  – Java (Android) gradle:
    
    ```
    dependencies {
      compile 'com.example.android:lib-magic:1.3'
    }
    ```
  
  – Python’s pip, javascript’s npm, …
MOTIVATION (general)

CODE REUSE

do not reinvent the wheel

SIMPLICITY

“it just works”

AVAILABILITY / PUBLICITY

reach your audience

REPRODUCIBILITY

tested configurations

PORTABILITY

cross-platform packaging

SAFETY

bug & vulnerability fixes
• **Initial deployment**
  – Find valid combination

• **Subsequent updates**
  – Staying backward compatible
version 1.2.3-prerelease+anything

- major . minor . patch
  - Major changes break compatibility
  - Minor changes add functionality
  - Patch changes fix bugs
- Minor/Patch upgrades “should” be safe.

- Meaningful only when offering an API
- Can assimilate other versioning methods
  - Calendar versioning: 20180501.0.0
ALIRE MOTIVATION

Ada world

Out there

[Images of various software package managers and build tools like dpkg, RPM, npm, PIP, Gradle]
• So, why reinvent the wheel?

Stay within the Ada boundaries
Ada stability makes it an “easy” target
Because why not
THE WORST OF BOTH WORLDS (as of May 2018)

https://xkcd.com/1987/
 EXAMPLES & KINDS

SYSTEM vs SANDBOX

PLATFORM vs LANGUAGE

BINARIES vs SOURCES

OFFICIAL vs COMMUNITY
• Motivation
  – Problem
  – Use cases
  – Semantic versioning

• Overview for users
  – Final user
  – Open source developer
  – Software distributor

• Design highlights
  – Index format
  – Staying within pure Ada
<table>
<thead>
<tr>
<th>ALIRE vs alr</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alire</strong></td>
<td><strong>alr</strong></td>
</tr>
<tr>
<td>- Database project</td>
<td>- Command-line tool</td>
</tr>
<tr>
<td>- “Passive” functionality</td>
<td>- “Active” functionality</td>
</tr>
<tr>
<td></td>
<td>- Dependency solver</td>
</tr>
<tr>
<td></td>
<td>- Project fetching</td>
</tr>
<tr>
<td></td>
<td>- Building process</td>
</tr>
</tbody>
</table>

[https://github.com/alire-project/alire](https://github.com/alire-project/alire)  
[https://github.com/alire-project/alr](https://github.com/alire-project/alr)
$ alr get --compile eagle_lander
$ alr get --compile hangman

$ ls
hangman_1.0.0_a5790492

$ cd hangman_1.0.0_a5790492
$ alr run

***** W E L C O M E T O H A N G M A N *****
  By: Jon Hollan, Mark Hoffman, & Brandon Ball

$ alr run --list
Project hangman builds these executables:
  hangmain (found at /tmp/demo/hangman_1.0.0_a5790492/bin/hangmain)
$ alr list

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ada_lua</td>
<td>An Ada binding for Lua</td>
</tr>
<tr>
<td>adacurses</td>
<td>Wrapper on different packagings of NcursesAda</td>
</tr>
<tr>
<td>adayaml</td>
<td>Experimental YAML 1.3 implementation in Ada</td>
</tr>
<tr>
<td>adayaml.server</td>
<td>Experimental YAML 1.3 server component</td>
</tr>
<tr>
<td>agpl</td>
<td>Ada General Purpose Library with a robotics flavor</td>
</tr>
<tr>
<td>ajunitgen</td>
<td>Generator of JUnit-compatible XML reports</td>
</tr>
<tr>
<td>alire</td>
<td>Alire project catalog and support files</td>
</tr>
<tr>
<td>alr</td>
<td>Command-line tool from the Alire project</td>
</tr>
<tr>
<td>apq</td>
<td>APQ Ada95 Database Library (core)</td>
</tr>
<tr>
<td>aunit</td>
<td>Ada unit test framework</td>
</tr>
</tbody>
</table>

$ alr search x

<table>
<thead>
<tr>
<th>NAME</th>
<th>VERSION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>rxada</td>
<td>0.1.0</td>
<td>RxAda port of the Rx framework</td>
</tr>
<tr>
<td>xml_ez_out</td>
<td>1.6.0</td>
<td>Creation of XML-formatted output from Ada programs</td>
</tr>
<tr>
<td>xstrings</td>
<td>1.0.0</td>
<td>Renaming of gnatcoll.strings w/o other dependencies</td>
</tr>
</tbody>
</table>
“with” a little help from my fellow developers

$ alr init --bin zzz
$ cd zzz
$ alr build ✓
$ alr run # null main
$ alr run # null main

$ vi zzz.gpr
with “xstrings”;
-- alr with xstrings
$ alr with --from zzz.gpr

$ vi src/zzz.adb
$ alr run

Zzz...
$ alr show adayaml
adayaml=0.3.0: Experimental YAML 1.3 implementation in Ada
Origin: commit 2017a7c2523499c03b8d7fe06546a5a8bae6476d
      from https://github.com/yaml/AdaYaml.git
Properties:
  Project_File: yaml.gpr
  Project_File: yaml-annotation_processor.gpr
  Project_File: yaml-utils.gpr
GPR Scenario: Mode := debug | release
Author: Felix Krause
Website: https://ada.yaml.io/
License: MIT
Dependencies (direct):
  aunit is At_Least (2017.0.0)
Dependencies (solution):
  aunit=2017.0.0
commit 2bacba46 from https://.../zlibada.git

zlib_ada=1.3.0

Semantic Version
- Major
- Minor
- Patch

Project
- Name
- Description

Origin
- URL
- Commit

Milestone
- zlib_ada=1.3.0

Release
- Dependencies
- Properties
package Alire.Index.RxAda is

  function Project is new Catalogued_Project ("Rx in Ada");

  Repo : constant URL := "https://bitbucket.org/amosteo/rxada";

  V_0_1_0  : constant Release :=
    Project.Register
    (V ("0.1.0"),
    Hg (Repo, "361d4e2ab..."),
    Properties =>
      Executable ("rx-examples-basic") and
      Author ("amosteo@unizar.es") and
      License (LGPL_3_0));
with Alire.Index.Alire;
with Alire.Index.AJunitGen;
with Alire.Index.XML_EZ_Out;

package Alire.Index.Alr is

   Base : constant Release := Project.Unreleased (...);
   -- All common properties declared here

package V_0_4 is new Project_Release -- Version is “reflected”
   (Base.Replacing (Git (Repo, "721d1112..."))
    .Extending (Dependencies =>
                AJunitGen.Project.Within_Major ("1.0") and
                Alire.V_0_4.Within_Minor and
                XML_EZ_Out.V_1_6.Within_Major));
CONTENTS

• Motivation
  – Problem
  – Use cases
  – Semantic versioning

• Overview for users
  – Final user
  – Open source developer
  – Software distributor

• Design highlights
  – Staying within pure Ada
  – Index format
STAYING WITHIN Ada

• Because we can (we do what we must)
  – Done before:
    • AWS.Resources

• But alr needs to
  – Update the catalog
  – Parse working-project dependencies

• Solution: recompilation
  – With generated session files
alr’s MULTIPLE PERSONALITIES

“stub” in (e.g.) $HOME/bin/alr

- Built at installation time, never recompiled
- Contains minimal index
- Generates full index and compiles:

“rolling” in $XDG_CONFIG_HOME/alire/.../alr

- Built whenever catalog is updated
- Contains full index
- Generates project-specific Ada files and compiles:

“project” in <working project>/alire/.../alr

- Built whenever catalog or metadata changes
- Contains full index + project data
Property’Class
- Encapsulate some environment property
- Unknown until runtime
- Examples:
  - Compiler
  - Operating system
  - Architecture

Requisite’Class
- Parallel hierarchy
- Logical expressions on matching properties
- Evaluated at runtime against available properties

<table>
<thead>
<tr>
<th>GNAT_FSF_7_2</th>
<th>GNAT_FSF_7_3</th>
<th>GNAT_GPL_2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNU_Linux</td>
<td>Windows</td>
<td>OSX</td>
</tr>
<tr>
<td>Unknown</td>
<td>Debian_Buster</td>
<td>Ubuntu_Bionic</td>
</tr>
</tbody>
</table>
| Bits_32      | Bits_64      | }
• All dependencies/properties are conditional
• A requisite tree has to be fulfilled
  – True if omitted

Project_File ("scenarios/catastrophical.gpr") -- unconditional

On_Condition (Operating_System = Windows,
  Project_File ("project_win.gpr")) -- if-then-else

Case_Operating_System_Is
  ((GNU_Linux => Comment ("Long life the penguin"),
    OSX     => Comment ("Oh shiny!"),
    others  => Comment ("Pick your poison"))) -- case is
Three kinds of conditions:

Dependencies =>
    Half_Life.Project >= "3.0" and -- Unconditional

(On_Condition -- Conditional
    (Operating_System = Windows,
        When_True  => Star_Citizen.Project.Current,
        When_False => Nethack.Project /= "1.27")

(GNATCOLL.Strings.Project or -- One of several
    GNATCOLL.Slim.Project or
    GNATCOLL.Project)
package Alire.Index.ZLib is

  function Project is new Catalogued_Project
    ("Library implementing the deflate method from gzip/PKZIP");

package V_1_2 is new Project_Release
  (Base.Replacing
    (Origin =>
      Native ((Debian | Ubuntu => Packaged_As ("zlib1g-dev"),
        others => Unavailable))));

end Alire.Index.ZLib;
CONCLUSION

• Alire + alr exists already
  – Debian testing / Ubuntu 17.10, 18.04 LTS
  – GNAT FSF 7.x / GNAT GPL 2017 / Community 2018

• Userspace-oriented
  – Does not manage a “global view” of installations
  – But can use available system packages
    • Eases initial packaging curve for complex dependencies

• It offers most expected capabilities
  – Flexible dependencies / properties
    • Conditional / Alternatives / Conflicts
  – Relying only on free / open source projects & services
    • Zero-cost at this time
  – Verified through Continuous Integration
## CATALOG STATUS in master branch

<table>
<thead>
<tr>
<th>Status</th>
<th>Project</th>
<th>Version</th>
<th>Build time</th>
</tr>
</thead>
<tbody>
<tr>
<td>pass</td>
<td>aaa</td>
<td>1.0.0</td>
<td>15.13 s</td>
</tr>
<tr>
<td>pass</td>
<td>ada_lua</td>
<td>0.0.0-5.3</td>
<td>23.65 s</td>
</tr>
<tr>
<td>pass</td>
<td>adacurses</td>
<td>6.0.0</td>
<td>33.35 s</td>
</tr>
<tr>
<td>pass</td>
<td>adayaml</td>
<td>0.3.0</td>
<td>119.42 s</td>
</tr>
<tr>
<td>pass</td>
<td>adayaml.server</td>
<td>0.3.0</td>
<td>19.65 s</td>
</tr>
<tr>
<td>pass</td>
<td>agpl</td>
<td>1.0.0</td>
<td>91.46 s</td>
</tr>
<tr>
<td>pass</td>
<td>ajunitgen</td>
<td>1.0.0</td>
<td>17.99 s</td>
</tr>
<tr>
<td>pass</td>
<td>alire</td>
<td>0.6.0</td>
<td>125.50 s</td>
</tr>
</tbody>
</table>

[https://github.com/alire-project/alr/blob/master/status/gnat-fsf-7.3.md](https://github.com/alire-project/alr/blob/master/status/gnat-fsf-7.3.md)
FUTURE STEPS

• Cross-compilation
  – Ada is strong in the *free* embedded world

• Support for more platforms
  – Windows
    • What to do about lack of native manager

• Promotion in the Ada community
  – Ada-Europe / Ada User Journal / comp.lang.ada
  – Await verdict of the masses

• Grow the catalog
THANKS FOR YOUR ATTENTION

🔗 https://github.com/alire-project/
✉ amosteo@unizar.es
🐦 @mosteobotic
My own questions to YOU ;-) 

- **alr compile:**
  - aggregate project
  - builds ev. at once
  - using original GPRs

- **alr install (tbd):**
  - env. var., shared prefix
  - gprbuild + gprinstall
  - Stand-alone, safe order

- **Consistence of the whole:**
  - Libraries decide over:
    - shared / static / etc
  - Requires manual tinkering with most libs (!)
  - Global overriding of library kinds?
    - Or some standardized Externals?