Herodotos

User's manual
release 0.6.0rc1

Nicolas Palix

April 15, 2010

Contents

Foreword

This manual documents the release 0.6.0rc1 of Herodotos.

Copyright

Availability

Herodotos can be freely downloaded from `http://www.diku.dk/~npalix/herodotos/`. This website contains also additional information.

Chapter 1

Installation

1.1 Requirements

To install Herodotos from the source You need the O'Caml compiler. Herodotos parsers are generated with Menhir. However they have been pre-generated in the source package.

To use Herodotos You need the following:

- A org-mode capable matching tool. For instance, Coccinelle (spatch) is perfect.
- Herodotos relies on EPS generation using jgraph and ps2eps. They are thus mandatory.
- To convert to PDF files, Herodotos uses epstopdf which is part of texlive. You may also consider installing pdftk to manipulate PDF files.
- To easy diffusion of your results on the web, Herodotos is able to generate a web site with images in PNG and SVG formats. If you want to use this feature, you need to have imagemagick (for PNG) and inkscape (for SVG). Note that the conversion to SVG needs the PDF version of your graphs. You thus need to have epstopdf available. Finally, this feature requires inscape version to be at least 0.47.
- To improve the execution time of Coccinelle, you may consider using the glimpse indexer.

1.2 Sources

Sources are available on several web sites:

- `http://www.diku.dk/~npalix/herodotos/`
- `https://launchpad.net/~npalix/+archive/herodotos`
- `https://launchpad.net/~npalix/+archive/herodotos-stable`

1.3 Distribution package

If you're under Ubuntu, you may add `ppa:npalix/herodotos` or `ppa:npalix/herodotos-stable` as an other software source in Synaptic. In that case, runtime dependencies are automatically handled.

For more information, you can go to the PPA pages on Launchpad mentioned in previous section.

Chapter 2

How to use Herodotos

At this point, you must have Herodotos installed. You now have to setup your environment and write a Herodotos Configuration file.

1. Setup the environment

   - Copy default make files and define your working directory. If you installed Herodotos from source, the default make files are `herodotos/scripts`. For a package install, copy the files from the Herodotos examples[1].
   - Write a HC file which declares your projects, patterns and graphs.
     If you want to have several .hc files, you should define the CONF environment variable to select a default one. Every following make command will then use that configuration file.

2. Initialize the environment:

   `make init`

3. Compute raw data:

   `make -j2`

4. Correlate bug reports and propose an initial set of correlations:

   `make correl`

   Note: Herodotos will automatically compute the changeset.

5. Manually edit the remaining correlations

   `$EDITOR results/*/*.correl.org`

   Emacs with the Org mode is recommended. Org mode view-link add-on will transform your emacs in a wizard to handle Coccinelle reports.

   See `./tools/org-view-link.el`

6. Use the new complementary correlation information

   `make correl` (recompute the correlation and use the correlation set manually provided)

7. Check the result:

   `make stat`

8. Look for false positive

[1] `/usr/share/doc/herodotos/examples/scripts`

   - `cp results/*/*.edit.org results/*/*.new.org`
   - Edit the new copy and annotate the reported TODO with either BUG or FP.

9. Generate graphs (or website):

   ```
   make [graph|web]
   ```

10. Edit and adjust your HC file

    Iterate on steps 9 and 10.

Chapter 3

HCL grammar

This document presents the grammar of the Herodotos Configuration Language used by the Herodotos tool. The grammar uses some unspecified nonterminals, such as id, etc. These refer to the sets suggested by the name, *i.e.*, id refers to the set of possible identifiers.

A HTML version of this documentation is available online at `http://www.diku.dk/~npalix/herodotos/docs/herodotos.html`.

3.1 Global attributes and top-level definitions

```
config    ::= globattr* project* pattern* graph*
globattr  ::= prefix = string | projects = string
            | patterns = string | results = string
            | findcmd = string | flags = string
project   ::= project id {prjattr*}
pattern   ::= pattern id {patattr*}
graph     ::= graph filepath {grphattr* curve*}
```

3.2 Common attributes

```
styleattr ::= color = float float float
            | linestyle = id | marktype = id
            | legend = string
```

3.3 Project attributes

```
prjattr ::= flags = string | dir = string
          | styleattr
          | versions = {((string,date,int))*}
```

3.4 Pattern attributes

```
patattr ::= flags = string | file = string
          | styleattr
```

3.5 Graphs rules

```
curve       ::= curve project id [{layoutattr*}]
             |  curve pattern id [{layoutattr*}]
             |  curve project id pattern id {layoutattr*}
             |  curve string {layoutattr*}
grphattr    ::= xaxis = xtype | yaxis = ytype
             |  xlegend = string | ylegend = string
             |  pattern = id | project = id
             |  layoutattr
xtype       ::= version | date
ytype       ::= occurrences | count | density | cumulcount
             |  size | sizepct | birth | death | netincrease
             |  birthfile | deathfile | avglifespan
             |  avgage | eldest | eldestdeath
             |  expertise | patchexpert | timeexpert
             |  abstime | reltime | lifeexpect | user
layoutattr  ::= nooccurcolor = float float float
             |  notexistcolor = float float float
             |  filename = boolean
             |  ratio = boolean | factor = int
             |  data = expression
             |  styleattr
```

3.6 Expression rules

The operators * and / have priority over the operators + and -.

```
expression ::= project id
            |  pattern id
            |  expression + expression
            |  expression - expression
            |  expression * expression
            |  expression / expression
            |  (expression)
```