The Quickref Cohort

Didier Verna
EPITA, LRE
Le Kremlin-Bicêtre, France
didier@lrde.epita.fr

ABSTRACT

The internal architecture of Declt, our reference manual generator for Common Lisp libraries, is currently evolving towards a three-stage pipeline in which the information gathered for documentation purposes is first reified into a formalized set of object-oriented data structures. A side-effect of this evolution is the ability to dump that information for purposes other than documentation. We demonstrate this ability applied to the complete Quicklisp ecosystem. The resulting "cohort" includes more than half a million programmatic definitions, and can be used to gain insight into the morphology of Common Lisp software.

CCS CONCEPTS

• **Information systems** → **Information extraction**; **Presentation of retrieval results**; • **Software and its engineering** → *Software libraries and repositories*.

KEYWORDS

Information Extraction, Software Analysis, Morphological Statistics

ACM Reference Format:
Didier Verna. 2024. The Quickref Cohort. In *Proceedings of the 17th European Lisp Symposium (ELS'24).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.5281/zenodo.10947962

1 INTRODUCTION

> *Cohort: a group of individuals having a statistical factor (such as age or class membership) in common in a demographic study.*
> *– The Meriam-Webster Dictionary[a], definition 2.b.*

[a]https://www.merriam-webster.com/dictionary/cohort

1.1 Context

Declt is a reference manual generator for Common Lisp libraries. The project started in 2010, leading to a first stable release in 2013 [2]. Four years later, the Quickref project was born [1, 4–6] (at the time, Declt was at version 2.3 [3]). Quickref runs Declt over the whole Quicklisp[1] repository and offers a website[2], currently aggregating more than two thousand reference manuals for Common Lisp libraries.

[1]https://www.quicklisp.org/
[2]https://quickref.common-lisp.net

ELS'24, May 6–7 2024, Vienna, Austria

ACM ISBN 978-2-9557474-8-3
https://doi.org/10.5281/zenodo.10947962

Declt runs by loading an ASDF system into memory and introspecting its contents. Because it is unrealistic to load the complete set of Quicklisp libraries into a single Lisp environment, Quicklisp runs Declt as a separate process for each library. The unfortunate consequence is that the information gathered by Declt is not directly available to the Quickref instance. Under those conditions, it remains easy to build a library index (by sorting the listing of the generated reference manuals directory), but it is for instance less straightforward to build an author index, as the author information, extracted from each ASDF system, needs to survive each and every Declt run.

Originally, Declt was designed to generate reference manuals in GNU Texinfo[3], an intermediate format suitable for software documentation, which can in turn be converted into a number of user-readable ones such as HTML, PDF, *etc.* Hence its name: Documentation Extractor from Common Lisp to Texinfo…

Over the years, there has been some pressure to extend Declt's rendering capabilities to other output formats (including HTML *without* the Texinfo intermediary). This led to an architecture overhaul, which is ongoing.

1.2 The Declt Pipeline

The goal is to implement Declt as a three-stage pipeline, as depicted in Figure 1. Declt's historical entry point, the `declt` function, triggers the whole pipeline, but for a more advanced usage, each stage of the pipeline is meant to be accessible separately and directly via its own entry point function.

(1) The first stage of the pipeline is called the *assessment* stage. At this stage, Declt loads the library and introspects the Lisp environment in order to extract the pertinent information. This information is stored in a so-called *report*.
(2) The second stage of the pipeline is called the *assembly* stage. At this stage, Declt organizes the information provided by a report in a specific way. The result is called a *script*. A script begins to look like a properly organized reference manual, but is still independent from the final output format.
(3) Finally, the third stage of the pipeline is called the *typesetting* stage. At this stage, Declt renders a script to a file by typesetting its contents in a specific documentation format.

In 2022, we released version 4.0b1 of Declt, marking the achievement of stage 1 of the pipeline [7]. Declt now provides a function called `assess`, which takes an ASDF system name as argument, loads the corresponding library, introspects it, and creates the report. The rest of the pipeline, which is not yet implemented, is wrapped in a temporary function called `declt-1`, going directly from a report to a Texinfo file.

[3]https://www.gnu.org/software/texinfo/

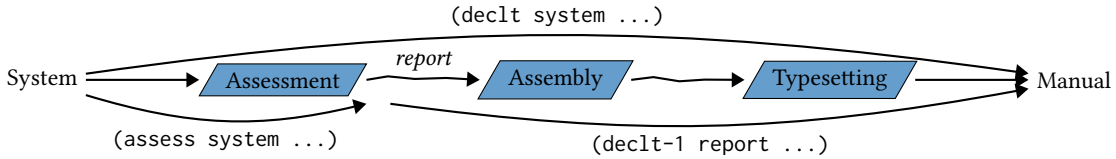

Figure 1: The Declt Pipeline

The first direct benefit of this evolution is the ability for Quickref to build an author index file in a much simpler and robust way. Instead of calling the global `declt` function, Quickref now triggers Declt in two steps. First, it calls the `assess` function to get a handle on the generated report, and then continues with `declt-1`. In the meantime however, the library's contact information is extracted from the report and dumped into a specific file. Once Quickref has finished processing the full set of Quicklisp libraries, it loads back all the contact information for all the libraries to create the index.

The funny thing is that once this was implemented, it quickly occurred to us that Declt reports, now in a stable format, could be fully dumped into files and used for all sorts of purposes other than documentation. In fact, it is relatively easy to "hijack" the Quickref infrastructure in order to dump Declt reports for the whole set of Quicklisp libraries, effectively creating a *cohort* of programmatic definitions.

In the following sections, we describe a preliminary cohort implementation, which currently contains more than half a million entities, and is already publicly accessible. Additionally, we show how such a cohort can be used to gain insight into the current shape of Lisp software.

2 DECLT REPORTS

A Declt report is a data structure containing general information about a library (authors, license, copyright, *etc.*), and a flat list of the discovered ASDF components and programmatic definitions (packages, variables, functions, classes, *etc.*).

2.1 Definitions

Definitions are themselves reified in an object-oriented fashion which is described in the Declt User Manual[4]. An excerpt of the definitions hierarchy is given in Figure 2.

For documentation purposes, the information provided by each kind of definition is as exhaustive as introspection permits. Most of them point back to the original Lisp object, can access the object's docstring if any, *etc.* On top of that, the assessment stage finalizes a library's definitions list by constructing an extensive set of cross-references (definitions pointing to definitions) that will eventually lead to internal hyperlinks in the generated reference manual.

For example, a generic function definition contains a list of method definitions (not raw method objects; pointers to the corresponding method definitions), a reference to it's method combination definition, but also a reference to a setf expander using this function for access, and a list of (short form) setf expanders using this function for update, if applicable.

[4]https://www.lrde.epita.fr/~didier/software/lisp/declt/bibliography/

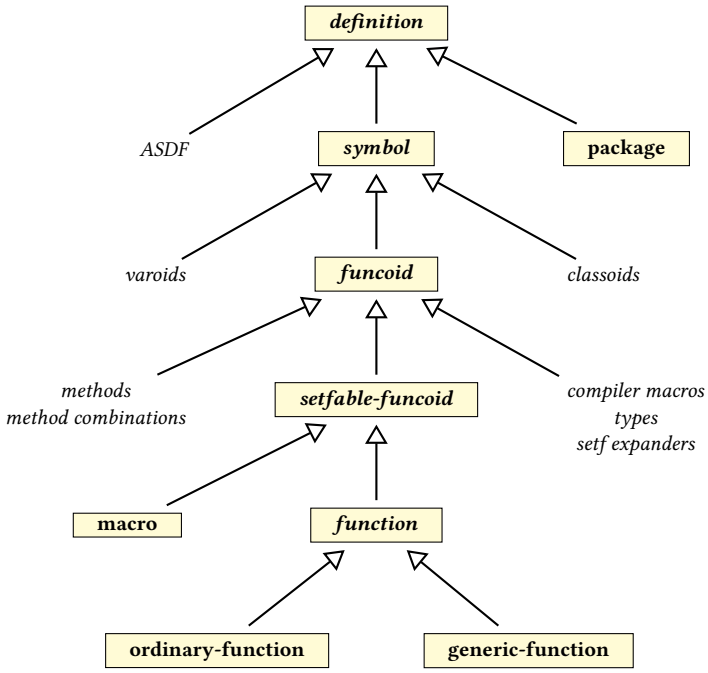

Figure 2: Definitions Hierarchy Excerpt

2.2 Dumping

As mentioned before, Declt reports were originally used by Quickref only to dump library author information, so as to build an author index afterwards. When the idea of a full cohort emerged, we decided to evaluate the potential usefulness of the idea by first creating a quick cohort prototype.

To this aim, the current prototype only dumps an incomplete and simplified version of Declt reports, that is, without performing true serialization. Pointers to the original Lisp objects can of course *not* be preserved in the dump. Cross-references between definitions are not currently preserved either, and only a few interesting attributes of each definition kind are retained, with some amount of pre-processing for subsequent statistical analysis.

Figure 3 provides an excerpt from the dump of Declt's own report. The contents should be mostly self-explanatory. Programmatic definitions start by a keyword denoting the definition kind, and name. Docstrings are replaced by their length, and cross-references by their number.

Such a simple dump already provides enough information to perform all sorts of interesting morphological studies on the 2000+

```
("net.didierverna.declt"
  (:CONTACTS 1)
  ...
  (:SYSTEM "net.didierverna.declt.assess"
   :DOCSTRING 44 :DEPENDENCIES 2 :CHILDREN 2
   :DEFSYSTEM-DEPENDENCIES 0)
  ...
  (:PACKAGE "NET.DIDIERVERNA.DECLT.ASSESS"
   :DOCSTRING 39
   :EXTERNAL-SYMBOLS 169 :INTERNAL-SYMBOLS 119
   :USE-LIST 2 :USED-BY-LIST 1)
  ...
  (:CLASS "GENERIC-FUNCTION-DEFINITION"
   :DOCSTRING 154
   :DIRECT-SUPERCLASSES 1 :DIRECT-SUBCLASSES 1
   :DIRECT-METHODS 11
   :DIRECT-SLOTS 3)
  ...
  (:GENERIC-FUNCTION "DOCUMENT"
   :DOCSTRING 45 :MEHTODS 39)
  ...)
```

Figure 3: Declt Dump Excerpt

libraries available in Quicklisp, as will be exemplified in the next section.

3 QUICKREF COHORT ANALYSIS

The current (beta) version of Quickref dumps Declt reports, as shown in the previous section, for every Quicklisp library. The resulting cohort (containing more than half a million programmatic definitions) is available for download from the website[5]. In order to demonstrate its potential usefulness, Quickref also performs a number of example statistical computations on the cohort, and generates subsequent plots, also visible on the website. Some of them are reproduced below.

3.1 Symbols Morphology

Figure 4 presents the histogram of symbol names lengths in Quicklisp, showing a peak at 11 characters, but also going as far as 135 characters for a single symbol name. Two other plots, not included in this article but visible on the website, show that most composed symbols have a cardinality (the number of `com-po-nents`) of 1, 2, or 3. The longest symbol appears to have 13 components. Most symbol components are 4 characters long, although one symbol (with a cardinality of 2) has a 126 characters long component. In fact, it is the very same symbol that is 135 characters long in total.

3.2 Documentation Shape

Another interesting area of investigation is the current state of Lisp documentation. Figure 5 shows the percentage of documented definitions per kind. For most types of programmatic entities, only 20 to 40% get a docstring. Slightly above this range are method combinations: half of them seem documented. On the other hand,

[5]https://quickref.common-lisp.net/cohort/

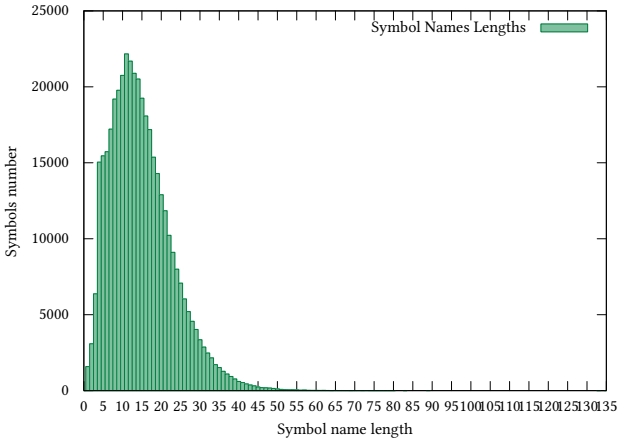

Figure 4: Symbol Names Lengths Histogram

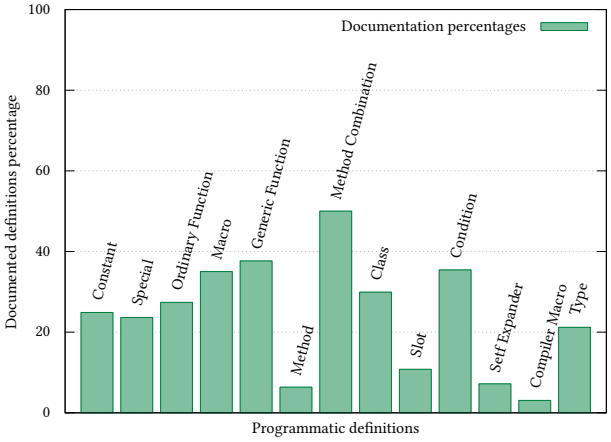

Figure 5: Documentation Percentages

Lisp programmers seem to disregard the documentation capabilities of methods, slots, setf expanders and compiler macros.

3.3 Classoid Profiles

As a final example of cohort analysis, Figure 6 presents the average number of direct slots, methods, parents, and children for structures, classes, and conditions. The most striking element in this plot is the average number of direct methods on classes, a little more than 6, which is much higher than on structures or conditions. It also seems that the multiple inheritance capability of classes and conditions is not used extensively, as the average number of parents remains only slightly above 1 (of course, it is *exactly* 1 for structures). Finally, we can see that the average number of direct slots is significantly higher in structures than in classes (and even more so in conditions), probably because of slot inheritance. Indeed, we can also see, by looking at the average number of children, that subclassing is more frequent than "substructuring".

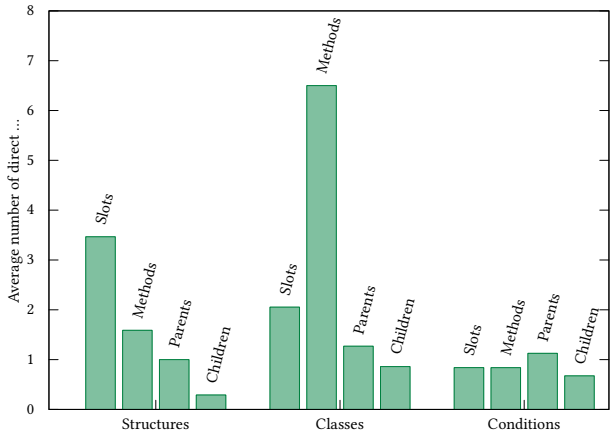

Figure 6: Aggregative Data Structure Averages

4 PERSPECTIVES

The Quickref cohort is currently a proof of concept, but we hope that the existence of a free database of more that half a million programmatic (and ASDF) entities will trigger some interest. Section 3 provided a glimpse at what can be done with it in terms of statistical analysis, but we're eager to hear about other potential use cases.

The cohort is essentially a collection of Declt reports, presented one way or another. Because of that, it makes sense to equip Declt itself with some cohort manipulation ability. For example, it could be interesting for a Lisp programmer to analyze their own (and only their own) library / libraries in a way similar to what was described in Section 3. We definitely are interested in doing so. We plan on extending Declt along these lines in a near future. In such a case, Declt could even manipulate actual reports (Lisp objects) rather than their dumped form.

In order to make the whole Quickref cohort truly usable, the next step is to stabilize the format used for dumping Declt reports. Contrary to the current format illustrated in section 2.2, Declt reports should be preserved as much as possible in order to *not* impose any limit on potential applications. In particular, no precomputation should be performed prior to dumping and cross-references between definitions should be preserved.

On the other hand, some parts of the reports need not (in fact, should not) be preserved in the dump. We want the ability to manipulate reports *without* the corresponding libraries being loaded in memory. This means that the actual Lisp objects corresponding to each definition (whether programmatic or ASDF) should be excluded from the dump.

All in all, it seems that what we are talking about here is some kind of serialization, a topic on which we currently have no experience. Consequently, we're eager to get some advice on that matter.

REFERENCES

[1] Antoine Hacquard and Didier Verna. A corpus processing and analysis pipeline for Quickref. In *14th European Lisp Symposium*, pages 27–35, Online, May 2021. ISBN 9782955747452. doi: 10.5281/zenodo.4714443.

[2] Didier Verna. Declt 1.0 is out. https://www.didierverna.net/blog/index.php?post/2013/08/24/Declt-1.0-is-out, August 2013. Blog entry.

[3] Didier Verna. Declt 2.3 "Robert April" is out. https://www.didierverna.net/blog/index.php?post/2017/10/16/Declt-2.2-Christopher-Pike-is-out, October 2017. Blog entry.

[4] Didier Verna. Announcing Quickref: a global documentation project for Common Lisp. https://www.didierverna.net/blog/index.php?post/2017/12/13/Announcing-Quickref%3A-a-global-documentation-project-for-Common-Lisp, December 2017. Blog entry.

[5] Didier Verna. Parallelizing Quickref. In *12th European Lisp Symposium*, pages 89–96, Genova, Italy, April 2019. ISBN 9782955747438. doi: 10.5281/zenodo.2632534.

[6] Didier Verna. Quickref: Common Lisp reference documentation as a stress test for Texinfo. In Barbara Beeton and Karl Berry, editors, *TUGboat*, volume 40, pages 119–125. TEX Users Group, TEX Users Group, September 2019.

[7] Didier Verna. Declt 4.0 beta 1 "William Riker" is released. https://www.didierverna.net/blog/index.php?post/2022/05/10/Declt-4.0-beta-1-William-Riker-is-released, May 2022. Blog entry.