# Inferred System Descriptions
# ELS 2016

James Anderson <james@dydra.com>

@dydradata @lomoramic

<http://dydra.com/>

# Perspective

- github : nine repositories; (SLOC) = 188,758
- dydra : :depends-on x 22; (SLOC) = 317,746

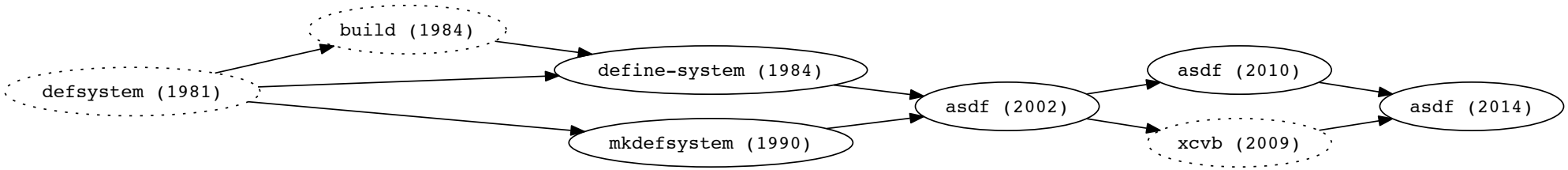
github : <https://github.com/lisp>
dydra : <http://dydra.com>



# Perspective

- github : nine repositories; (SLOC) = 188,758
- dydra : :depends-on x 22; (SLOC) = 317,746

- build systems



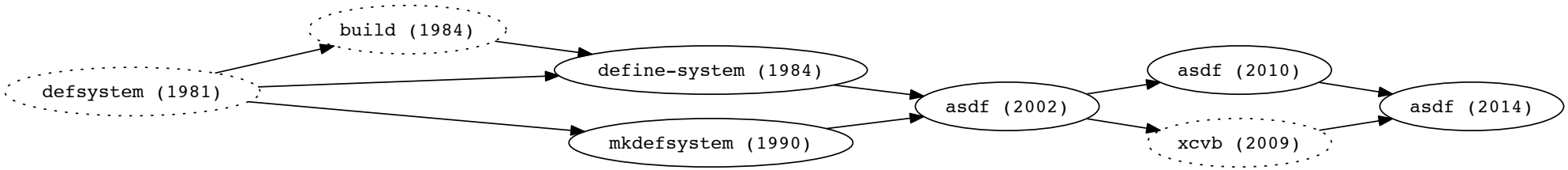

asdf : <http://fare.tunes.org/files/asdf3/asdf3-2014.html>
github : <https://github.com/lisp>
dydra : <http://dydra.com>



# Perspective

- github : nine repositories; (SLOC) = 188,758
- dydra : :depends-on x 22; (SLOC) = 317,746

- build systems




- :depends-on : why?

asdf : <http://fare.tunes.org/files/asdf3/asdf3-2014.html>
github : <https://github.com/lisp>
dydra : <http://dydra.com>



# Perspective: must it be manual?

- :depends-on : why?
- build (robbins 1984)
  - "patching"
  - "more precise change analysis"
- description of large systems (pitman 1984)
  - "transitivity of dependency information"
  - "tight bonding to syntax"

build : <http://hdl.handle.net/1721.1/6909>

dls : <http://www.nhplace.com/kent/Papers/Large-Systems.html>



# Results

- http://dydra.com/data/els2016/index.htm
- 1683 systems



# Perspective: should be possible?

- "patching"
- "more precise change analysis"
- "transitivity of dependency information"
- "loose bonding to syntax"

- LISP
  a programming system called LISP (for LISt Processor)

LISP : <http://www-formal.stanford.edu/jmc/recursive.pdf>



# Perspective: should be possible?

- "patching"
- "more precise change analysis"
- "transitivity of dependency information"
- "loose bonding to syntax"

- LISP
  ~~a programming system called LISP (for LISt Processor)~~
  "Recursive Functions of Symbolic Expressions ..."

LISP : <http://www-formal.stanford.edu/jmc/recursive.pdf>



# Abstraction Alternatives

- ignore any system definition
- macro-expansion just to normalize syntax
- emulate just package symbol identity
- extract relations from source code
- do not compile
- do not build
- do not execute

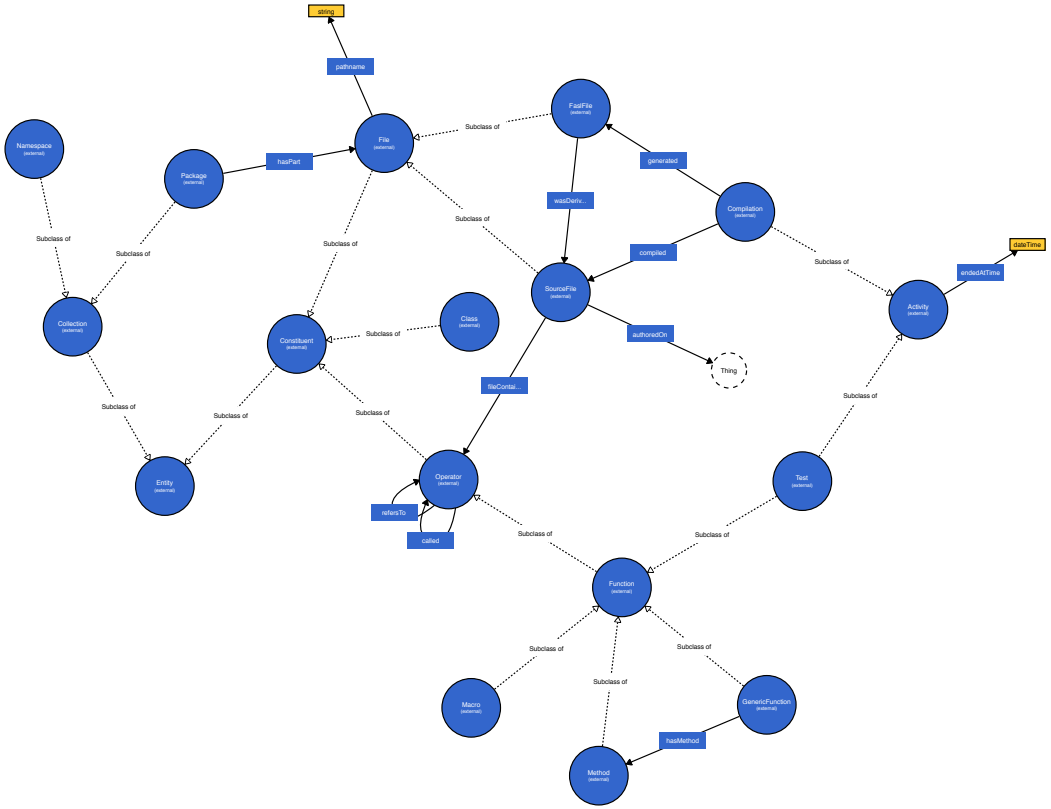


# Abstraction Vocabulary

- doap : http://usefulinc.com/ns/doap :
  packaged product level
- scro : http://www.cs.uwm.edu/~alnusair/ontologies/scro.owl :
  just conceptual structure
- compre : http://www.cs.uwm.edu/~alnusair/ontologies/compre.owl :
  components and depends, but no hierarchy, versions, or generation
- swonto : http://www.cs.uwm.edu/~alnusair/ontologies/swonto.owl :
  data processing in jena
- iao : https://github.com/information-artifact-ontology/IAO/ :
  the entities are too generic
- swo : https://sourceforge.net/p/theswo/code/HEAD/tree/trunk/src/release/swoinowl/inferredswo/swo_inferred.owl :
  overdrawn : 4338 entities
- sioc : http://sioc-project.org :
  process oriented
- oflossc : http://ns.inria.fr/oflossc/ (404) :
  sioc specialized for open source process
- code : http://sioc-project.org :
  provides starting point



# Abstraction Vocabulary



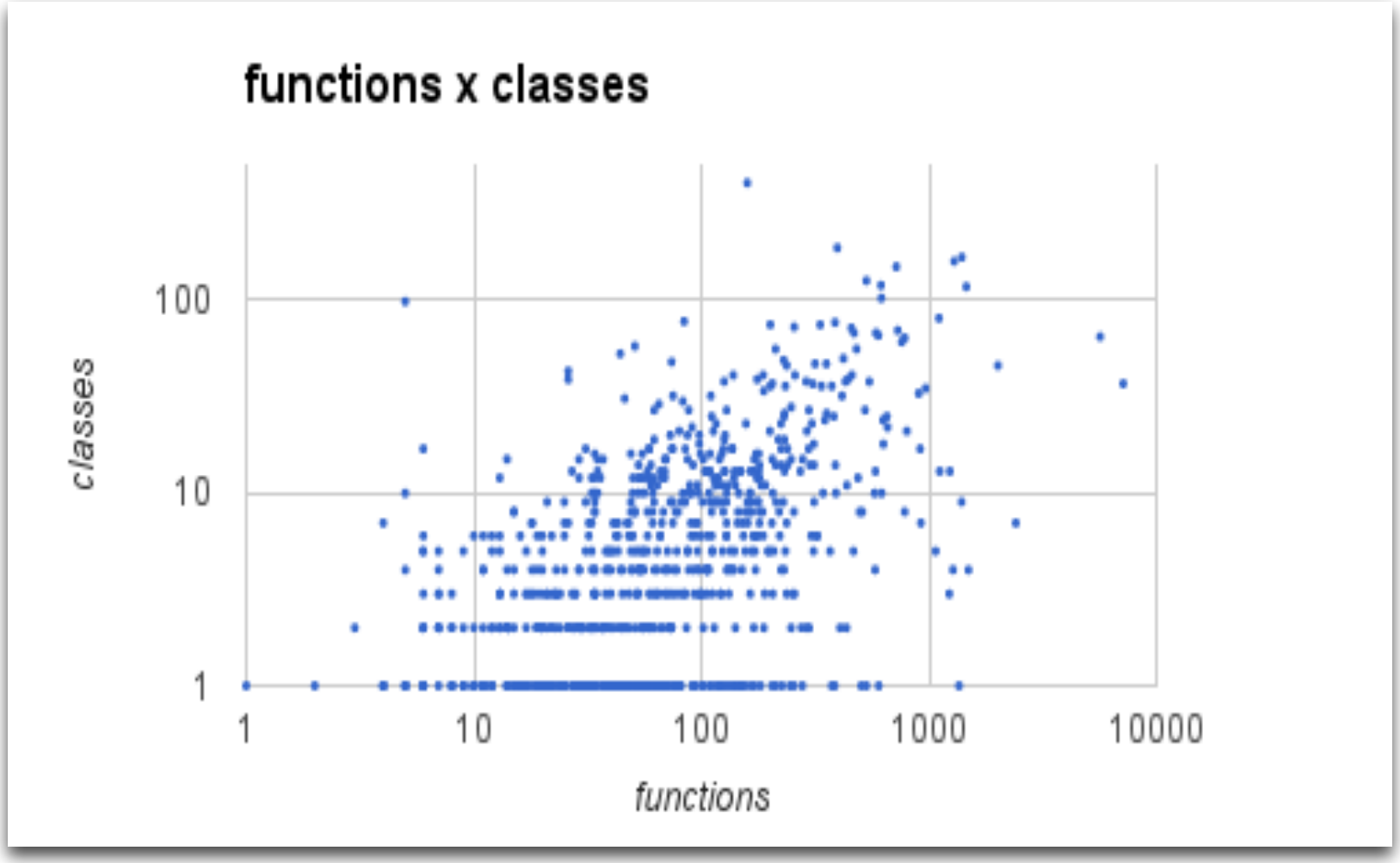



# Goals

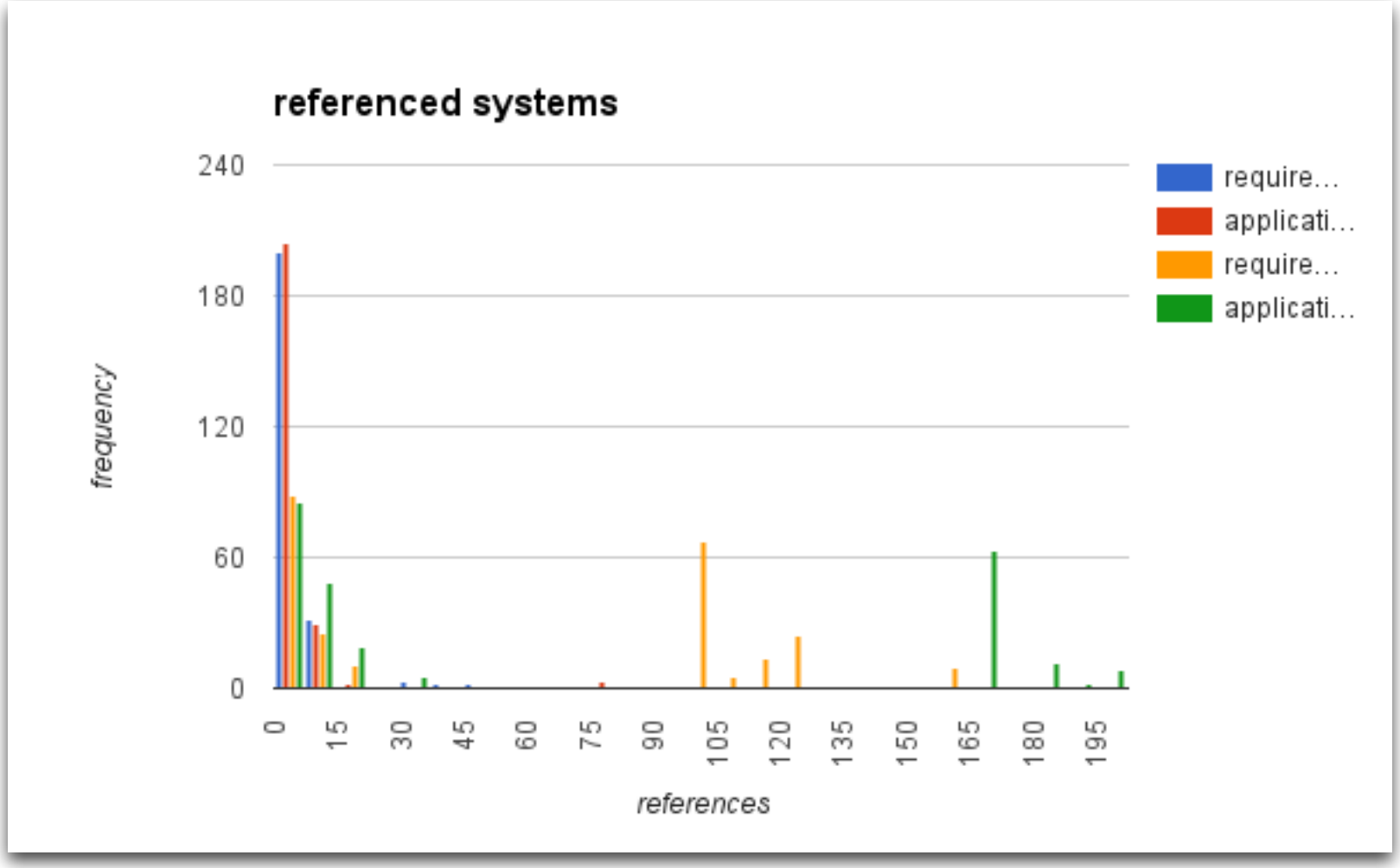
- identify all packages to load to use a given operator
- construct system based on requirements at any of three levels
- derive changes to load between two versions
  per system
  per file
  per function (test)



# Results: Metadata



# Results: Metadata



# Start with Text

test

```
(defun
iex:function (arg)
  (+ (iex:method1
arg) 2))
```

file1

```
(defclass iex:class ()
  ((slot1 :initform 1 :accessor iex:class-slot1)
   (slot2 :initform 2 :accessor iex:class-slot2)))

(defgeneric iex:method1 (object)
  (:method ((object iex:class))
    (+ 1 (iex:method1 (iex:class-slot1 object))))
  (:method ((object t))
    object))
```

file2

```
(defmethod
   iex:method2
((object iex:class))
  (+ (iex:method1
       object)
     (iex:class-slot2
       object)))
```

file2

```
(defun test.function1 ()
  (eql (iex:method1 (make-instance
'iex:class)) 2))

(defun test.function2 ()
  (eql (iex:method2 (make-instance
'iex:class)) 4))
```

```
(system :example
  :components ((:file "file1")
               (:file "file2" :depends-on "file1")
               (:file "file3" :depends-on "file2")
               (:file "test" :depends-on "file2 "file1")
```



# Just the Text

test

```
(defun
iex:function (arg)
  (+ (iex:method1
arg) 2))
```

file1

```
(defclass iex:class ()
  ((slot1 :initform 1 :accessor iex:class-slot1)
   (slot2 :initform 2 :accessor iex:class-slot2)))

(defgeneric iex:method1 (object)
  (:method ((object iex:class))
    (+ 1 (iex:method1 (iex:class-slot1 object))))
  (:method ((object t))
    object))
```

file2

```
(defmethod
   iex:method2
((object iex:class))
  (+ (iex:method1
       object)
     (iex:class-slot2
       object)))
```

file2

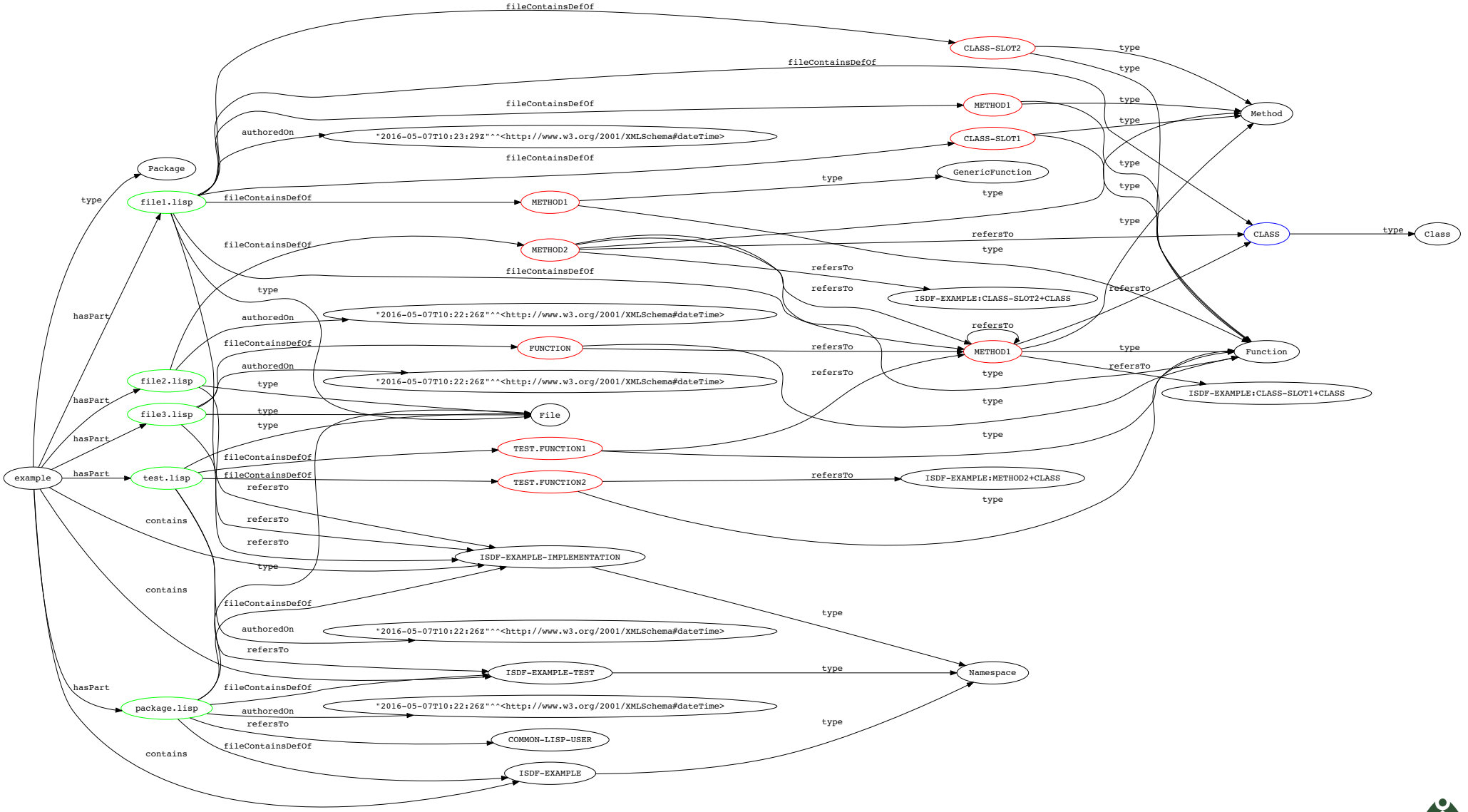
```
(defun test.function1 ()
  (eql (iex:method1 (make-instance
'iex:class)) 2))

(defun test.function2 ()
  (eql (iex:method2 (make-instance
'iex:class)) 4))
```

```
(system :example
  :components ((:file "file1")
               (:file "file2" :depends-on "file1")
               (:file "file3" :depends-on "file2")
               (:file "test" :depends-on "file2 "file1")
```

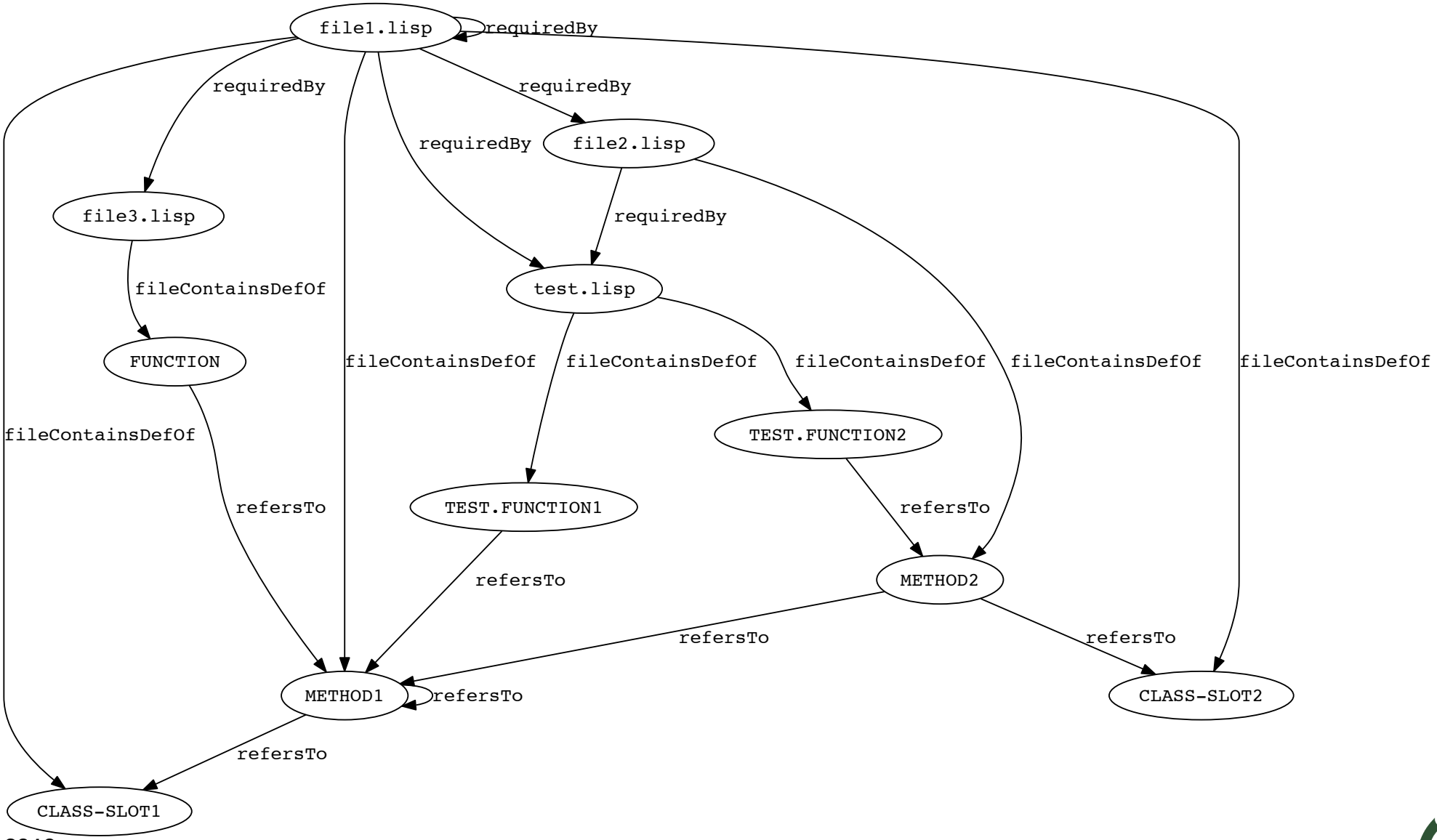


# Abstract Symbols and Relations



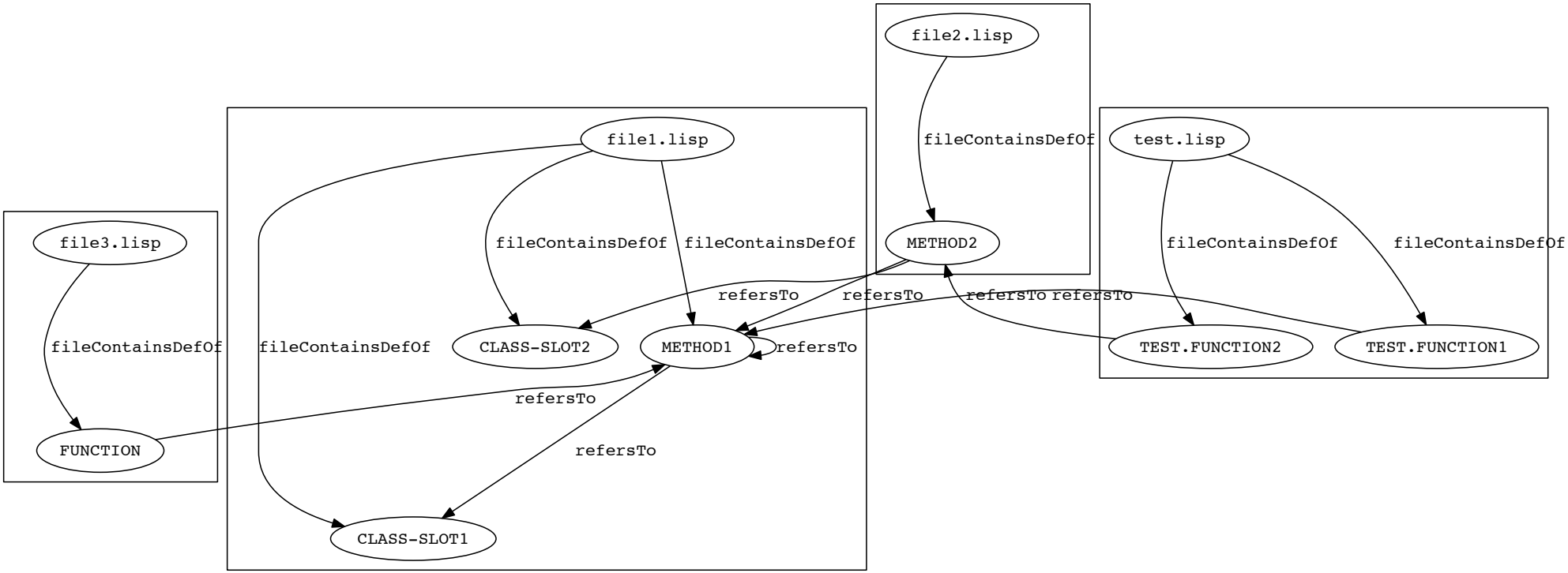



# Identify Types and Relations



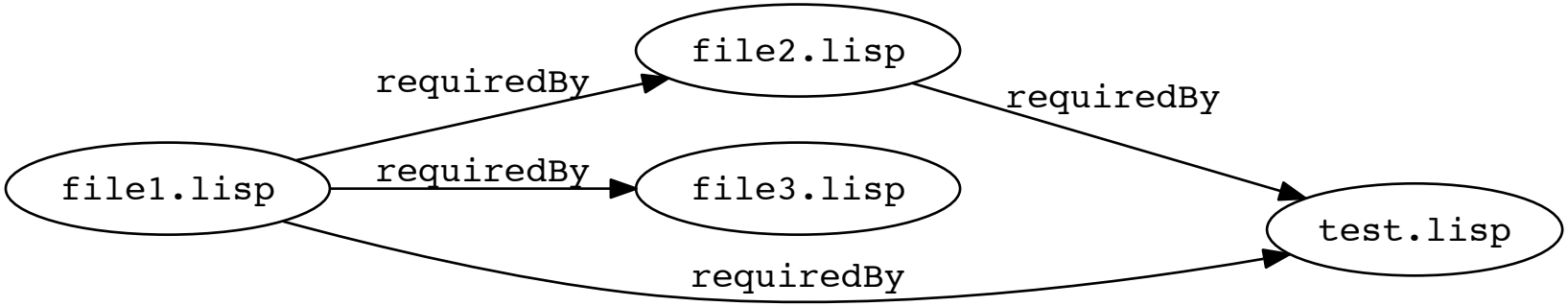



# Establish Source References






# Capabilities: Infer Source Requirements






# Process : Quicklisp as Corpus

```
$ ls /tmp/quicklisp/ | wc -l
1669
$ ls quicklisp/versions/2016-02-08/ | wc -l
1317
$ find quicklisp/versions/2016-02-08/ -name '*.asd' | wc -l
3134
$ find quicklisp/versions/2016-02-08/ -name '*' -type d | wc -l
5839
$ find quicklisp/versions/2016-02-08/ -name '*.asd' \
   | while read dir; do dirname $dir; done \
   | sort | uniq | wc -l
1670
$ find quicklisp/versions/2016-02-08/ -name '*.lisp' | wc -l
20310
$/development/source/library/org/quicklisp/versions/2016-03-18 # sloccount .
...
Total Physical Source Lines of Code (SLOC)                = 4,307,928
```



# Process : Crawl Source Code

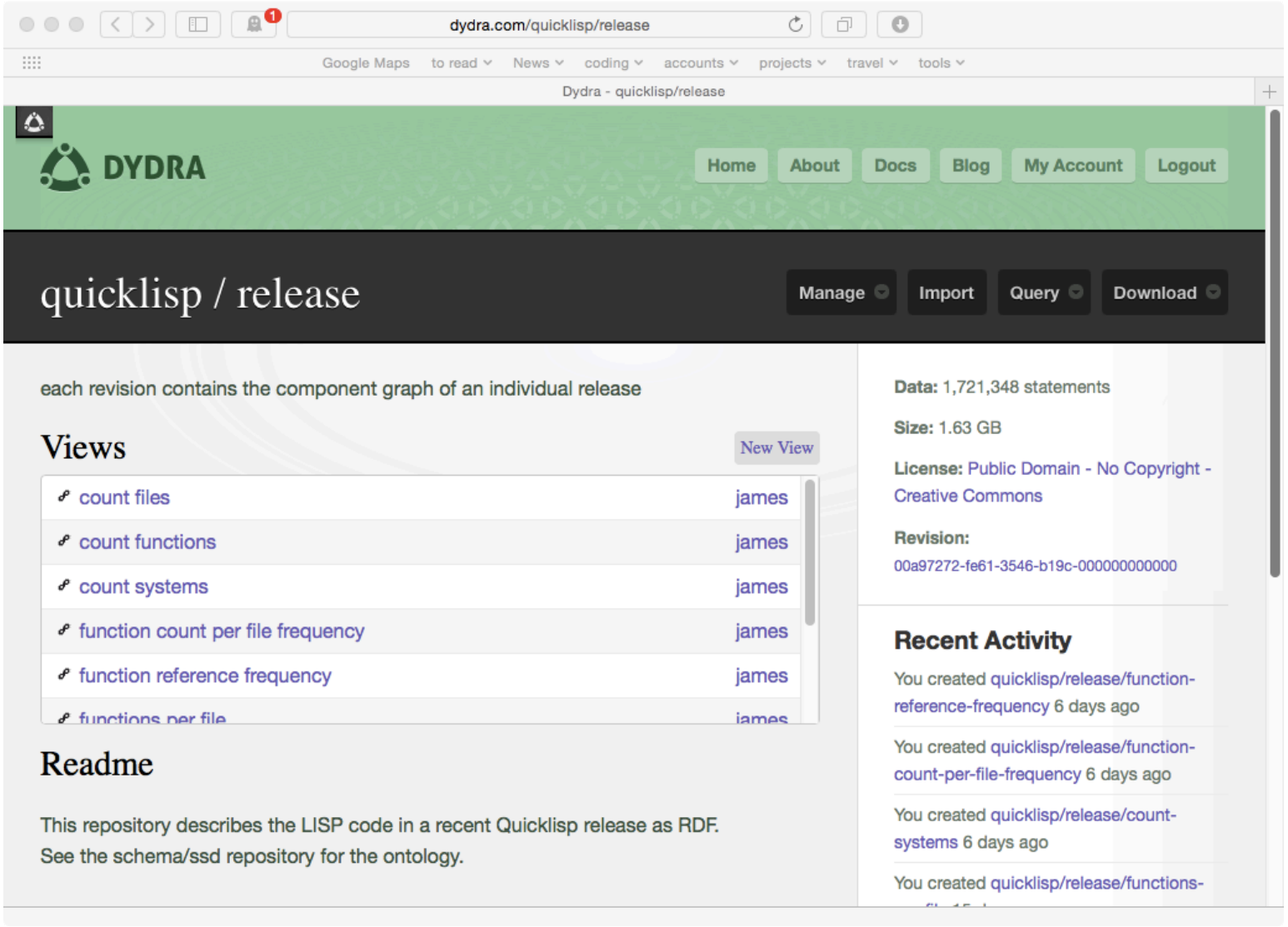
```
* (time
   (scm:abstract-system
    (ensure-system "quicklisp"
                   :location #p"quicklisp/versions/2016-02-08/")))
Evaluation took:
  2178.410 seconds of real time
   323.020348 seconds of total run time (75.350139 user, 247.670209 system)
  [ Run times consist of 0.613 seconds GC time, and 322.408 seconds non-GC time. ]
  14.83% CPU
  3,444,829 forms interpreted
  17 lambdas converted
  7,624,481,182,395 processor cycles
  5,746,702,672 bytes consed

* (/ 2178.4 60)

36.306664
*
```



# Process : Result Repository



# Process : Infer Dependency

```
prefix ssd: <http://dydra.com/schema/ssd#>
prefix code: <http://www.cs.cmu.edu/~anupriya/code#>
clear silent graph <urn:dydra:file-requirements> ;
insert {
 graph <urn:dydra:file-requirements> {
   ?definitionSource ssd:requiredBy ?referenceSource
 }
}
where{
  select ?definitionSource ?referenceSource where {
  graph ?package {
    ?definitionSource code:fileContainsDefOf ?definition .
    # eliminate function/function definitions
    ?definitionSource a <http://www.cs.cmu.edu/~anupriya/code#File> .
    ?referenceSource code:fileContainsDefOf ?reference .
    ?referenceSource a <http://www.cs.cmu.edu/~anupriya/code#File> .
    ?reference code:refersTo ?definition .
    filter( ?definitionSource != ?referenceSource )
  } }
}
```



# Process : Infer Dependency

```
prefix ssd: <http://dydra.com/schema/ssd#>
prefix code: <http://www.cs.cmu.edu/~anupriya/code#>
clear silent graph <urn:dydra:system-requirements> ;
insert {
 graph <urn:dydra:system-requirements> {
   ?definitionPackage ssd:requiredBy ?referencePackage .
   ?referencePackage ssd:requires ?definitionPackage
 }
}
where{
  select ?definitionPackage ?referencePackage where {
  { graph ?definitionPackage {
    ?definitionSource code:fileContainsDefOf ?definition .
    ?definitionSource a <http://www.cs.cmu.edu/~anupriya/code#File> .
    # eliminate function/function definitions
  } }
  { graph ?referencePackage {
    ?referenceSource code:fileContainsDefOf ?reference .
    ?referenceSource a <http://www.cs.cmu.edu/~anupriya/code#File> .
    ?reference code:refersTo ?definition .
  } }
  filter (?definitionPackage != ?referencePackage)
  }
}
```



# Results: Quicklisp Graphs

```
* (quicklisp-sparql "
select ?p (count(?o) as ?count)
FROM <urn:dydra:all>
where {
  ?s ?p ?o
} group by ?p order by desc(?count)"
             :message "predicate count")
---------------
predicate count
---------------
p count
<http://www.cs.cmu.edu/~anupriya/code#refersTo>        743247
type   298610
label  240360
<http://www.cs.cmu.edu/~anupriya/code#fileContainsDefOf>       227370
<http://dydra.com/schema/ssd#requiredBy>       67917
comment        38959
<http://dydra.com/schema/ssd#requires> 37251
<http://www.cs.cmu.edu/~anupriya/code#hasPart> 22017
<http://www.cs.cmu.edu/~anupriya/code#location>        20344
<http://purl.org/pav/authoredOn>       20334
<http://www.cs.cmu.edu/~anupriya/code#contains>        4899
endedAtTime    10
generated      10
used   10
wasDerivedFrom 10
```



# Results: Quicklisp Graphs

```
* (quicklisp-sparql "
select count(*) where {
  graph <urn:dydra:file-requirements> {?s ?p ?o} }"
               :message "intra-system file cross-references")

-----------------------------------
intra-system file cross-references
-----------------------------------
COUNT1
30666
```



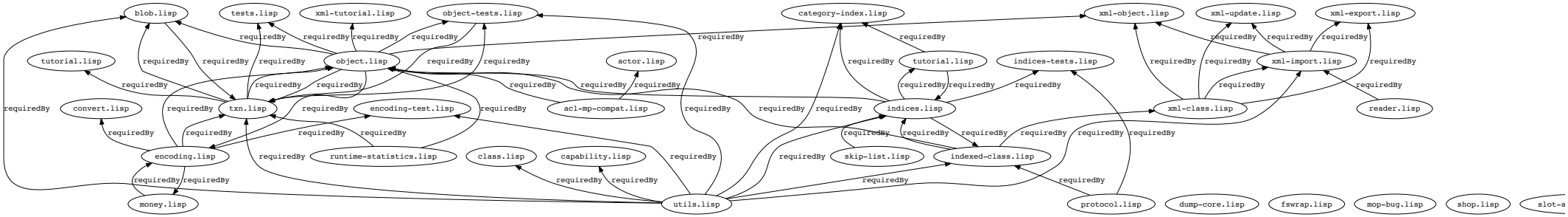
# Results: Quicklisp Graphs

```
* (quicklisp-sparql "
prefix ssd: <http://dydra.com/schema/ssd#>
select count(*) where {
   graph <urn:dydra:system-requirements> {?s ssd:requires ?o} }"
            :message "inter-system cross-references")
-----------------------------
inter-system cross-references
-----------------------------
COUNT1
37251

* (quicklisp-sparql "
prefix ssd: <http://dydra.com/schema/ssd#>
select count(*) where {
graph <urn:dydra:system-requirements> {?s ssd:requiredBy ?o} }"
            :message "inter-system cross-references")
-----------------------------
inter-system cross-references
-----------------------------
COUNT1
37251
```



# BKNR: Infer Source Requirements




LISP : <http://dydra.com/data/els2016>



# BKNR: Load Order

```
* (bknr-sparql "
prefix ssd: <http://dydra.com/schema/ssd#>
prefix prov: <http://www.w3.org/ns/prov#>
prefix pav: <http://purl.org/pav/>
prefix code: <http://www.cs.cmu.edu/~anupriya/code#>
select distinct ?loadFilename ?dependentSource # ?sequenceNr
from  <urn:dydra:all>
where {
  { select * where {
    bind ( '2000-05-08T00:00:00Z'^^xsd:dateTime as ?SYSTEM_LOAD_TIME )
    { select ?source where {
      ?source a <http://www.cs.cmu.edu/~anupriya/code#File> .
      optional { ?otherSource ssd:requiredBy ?source }
      filter ( !bound(?otherSource) )
    } }
    ?source ssd:requiredBy+ ?dependentSource .
    bind (rdf:Seq(xsd:integer) as ?sequenceNr )
  } }
  ?source pav:authoredOn ?timeModified .
  filter ( ?timeModified > ?SYSTEM_LOAD_TIME )
  ?dependentSource code:location ?sourcePathname .
  ?dependentSource pav:authoredOn ?sourceTime .
  ?dependentSource rdfs:label ?sourceFile .
  optional {
    [] prov:used ?dependentSource ;
       prov:generated ?dependentBinary ;
       prov:endedAtTime  ?binaryTime .
    ?dependentBinary code:location ?binaryPathname .
    ?dependentBinary rdfs:label ?binaryFile .
 }
  bind (if( bound(?binaryTime),
            if ((?binaryTime > ?sourceTime),
                ?binaryFile, ?sourceFile ),
            ?sourceFile )
        as ?loadFilename )
} order by desc(?sequenceNr)
  " :message "source/binary load list (but absent binaries)")
```



# BKNR: Load Order

```
-------------------------------------------
source/binary load list (but absent binaries)
-------------------------------------------
loadFilename dependentSource
convert.lisp
encoding-test.lisp
object.lisp
txn.lisp
blob.lisp
object-tests.lisp
tests.lisp
tutorial.lisp
encoding.lisp
xml-object.lisp
xml-tutorial.lisp
xml-export.lisp
xml-import.lisp
xml-update.lisp
indices.lisp
xml-class.lisp
category-index.lisp
indexed-class.lisp
indices-tests.lisp
tutorial.lisp
capability.lisp
class.lisp
actor.lisp
NIL
*
```



# Deficiencies

- macros
- ambiguous packages : non-universal symbols
- non-lisp syntax ".lisp" file content
- colleen-20160208-git/sample.uc.lisp
- bknr-datastore-20150923-git/src/xml-impex/tutorial.lisp
- read-eval
- read-time, compile-time, load-time, run-time
- no tracking for defparameter/-constant/-var
- circularity
- implicit package definitions
- ambiguous methods call sites



# Deficiencies

- ~~macros~~
- ~~ambiguous packages : non-universal symbols~~
- ~~non-lisp syntax ".lisp" file content~~
- ~~colleen-20160208-git/sample.uc.lisp~~
- ~~bknr-datastore-20150923-git/src/xml-impex/tutorial.lisp~~
- ~~read-eval~~
- ~~read-time, compile-time, load-time, run-time~~
- ~~no tracking for defparameter/-constant/-var~~
- circularity
- implicit package definitions
- ambiguous methods call sites



# Deficiencies : ambiguous methods

bknr-datastore-20150923-git/src/indices-tests.lisp

```
(defmethod run-test :around ((test index-test-class)
                             &optional (output *debug-io*))
      ... )
```

unit-test-20120520-git/unit-test.lisp

```
(defun run-all-tests (&key (verbose nil) (output *standard-output*)
                           (unit :all))
  (let (*unit-test-results*)
    (run-test test output)
      ... )
```

would require either type propagation or run-time instrumentation to distinguish specific methods for call sites.



# Thank you

James Anderson <james@dydra.com>

@dydradata @lomoramic

- <http://http://dydra.com/data/els2016/>