Eating Your Own Dog Food

Andreas Abel[1]

[1]Department of Computer Science and Engineering
Chalmers and Gothenburg University, Sweden

25th Agda Implementor's Meeting (AIM XXI)
Teknikparken, Chalmers, Gothenburg, Sweden
9 May 2017

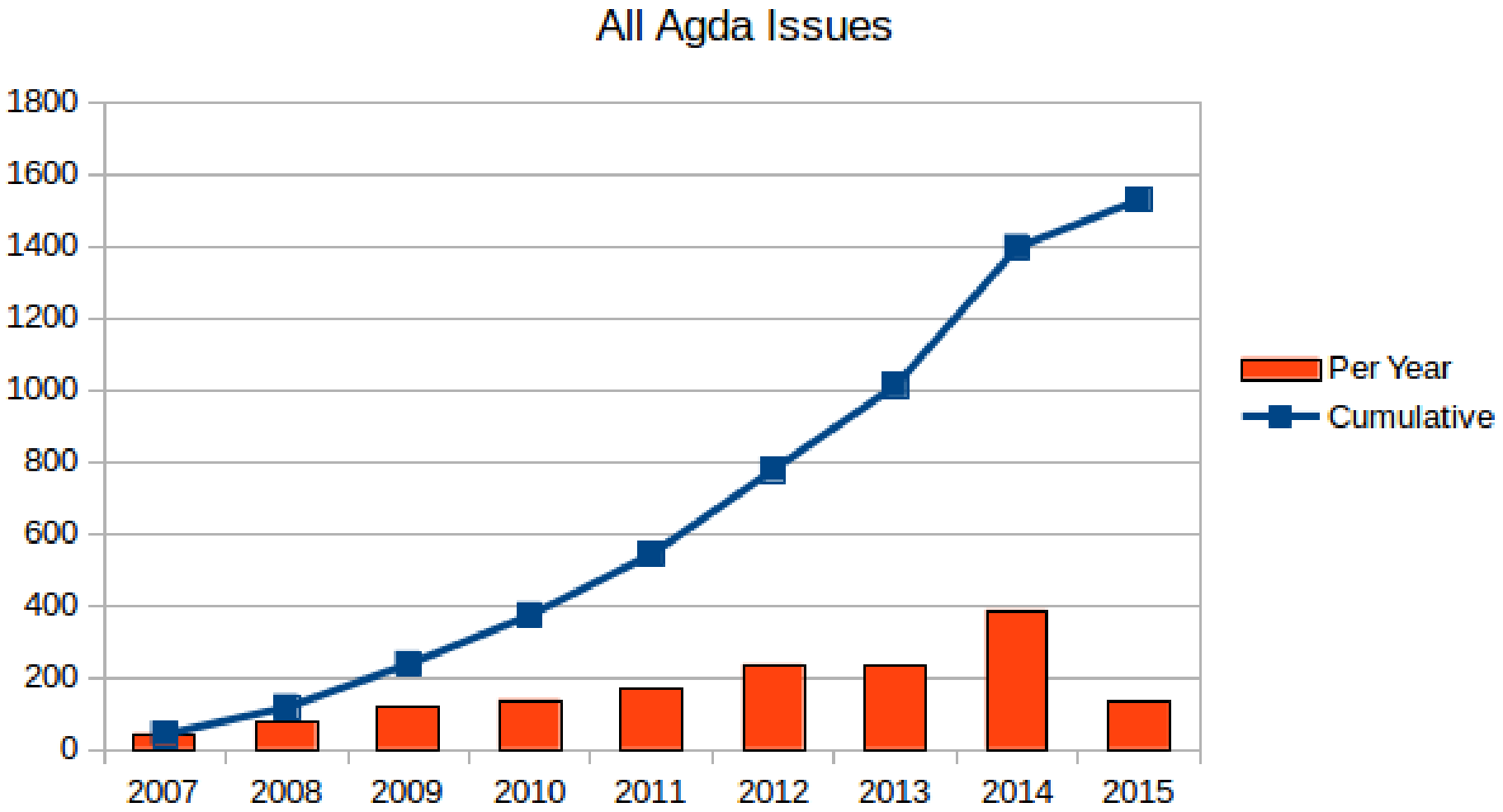
Agda: Inside View

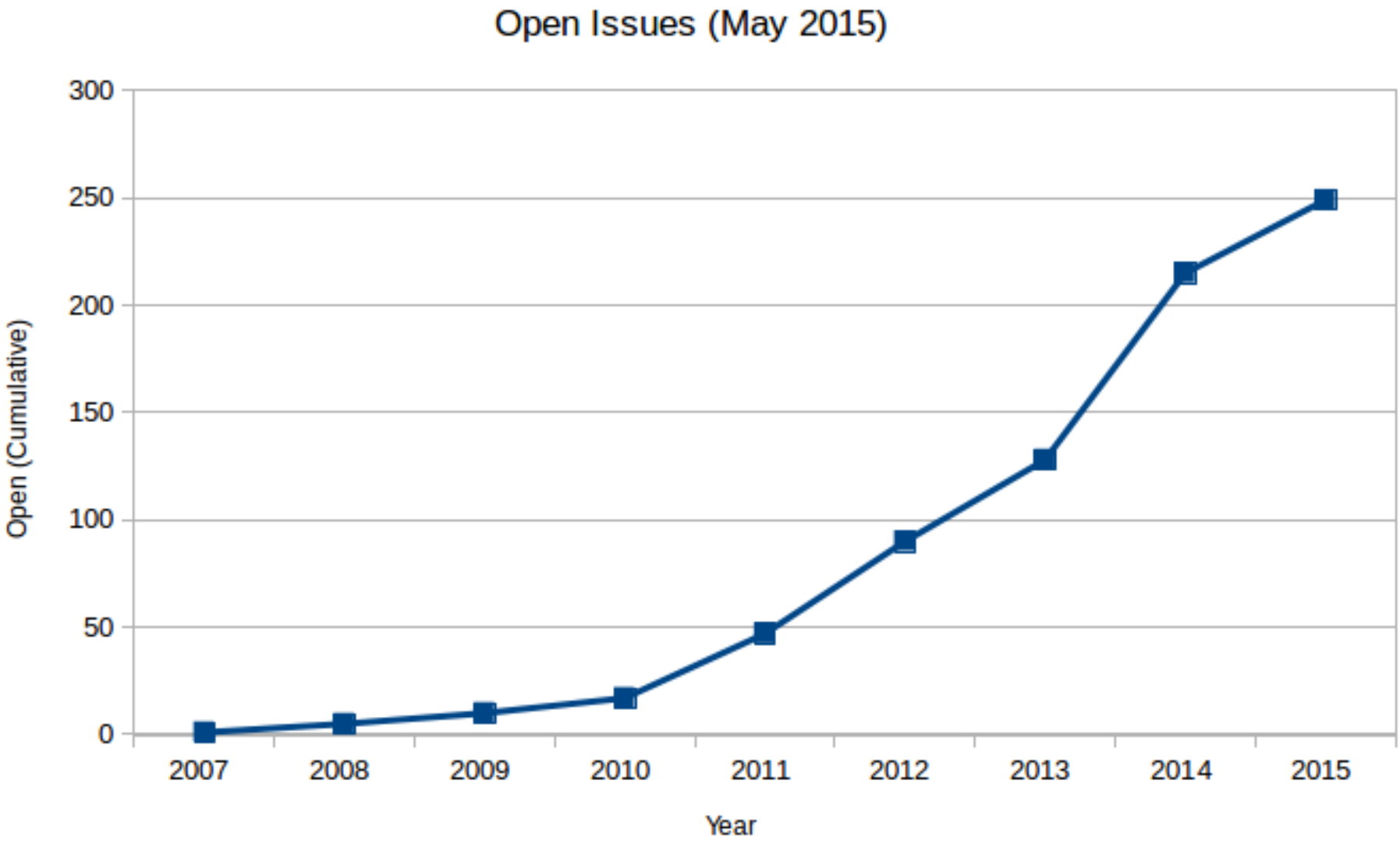
- 3 main developers (part-time)
- 1 packaging, integration, and release engineer (part-time)
- a dozen contributers (sporadically active)
- a long tail of single-patch submitters
- 103.000 loc (3.9MB) [2015: 93.000 loc; 2014: 70.000 loc]
- 2461 issues on the bug tracker (404 open) [2015: 1528 (249); 2014: 1076 (165)]

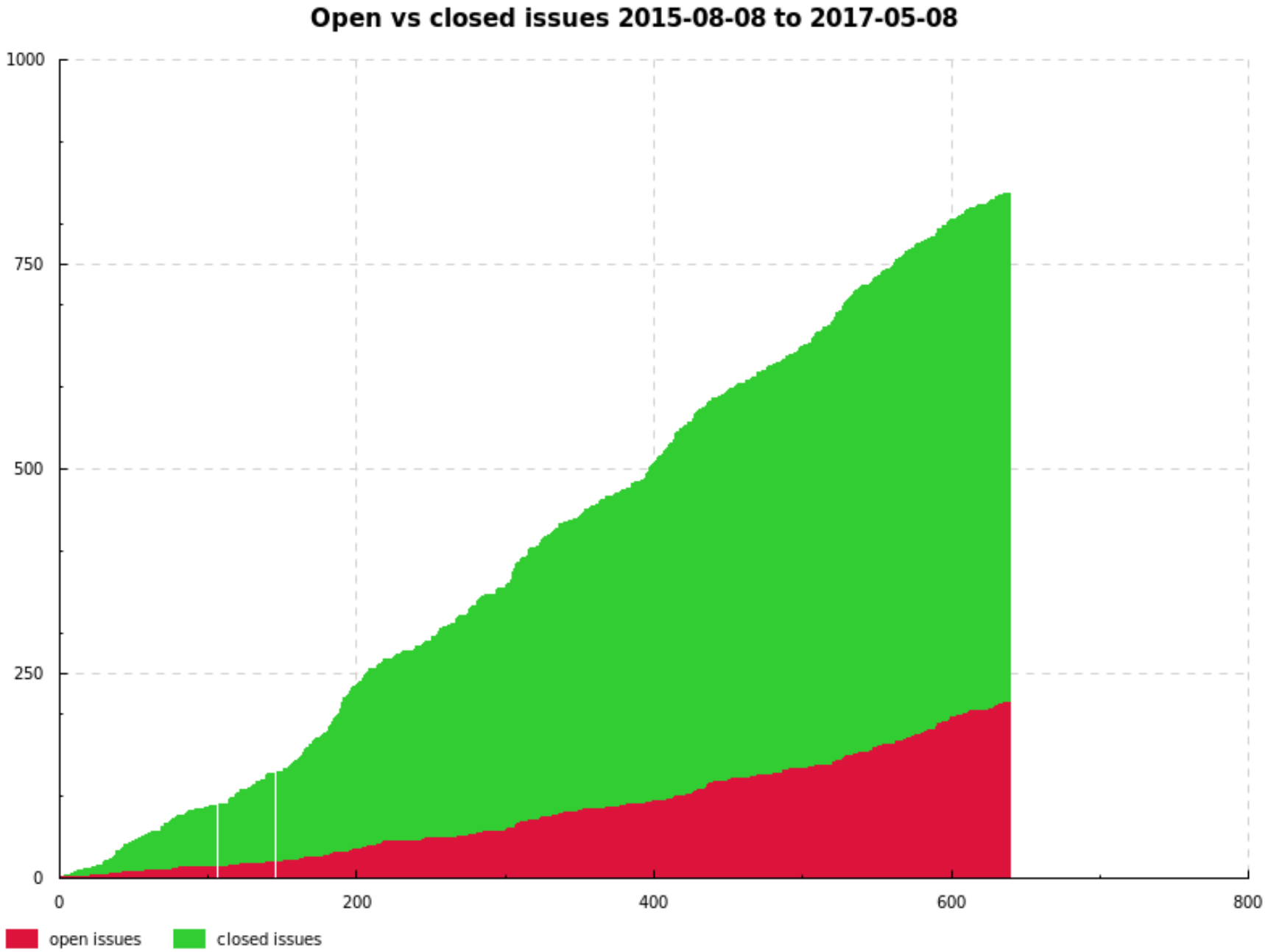
2007-2010	300	100 bugs/year
2010-2013	600	200 bugs/year
2013-2015	600	300 bugs/year
2015-2017	900	$>$400 bugs/year

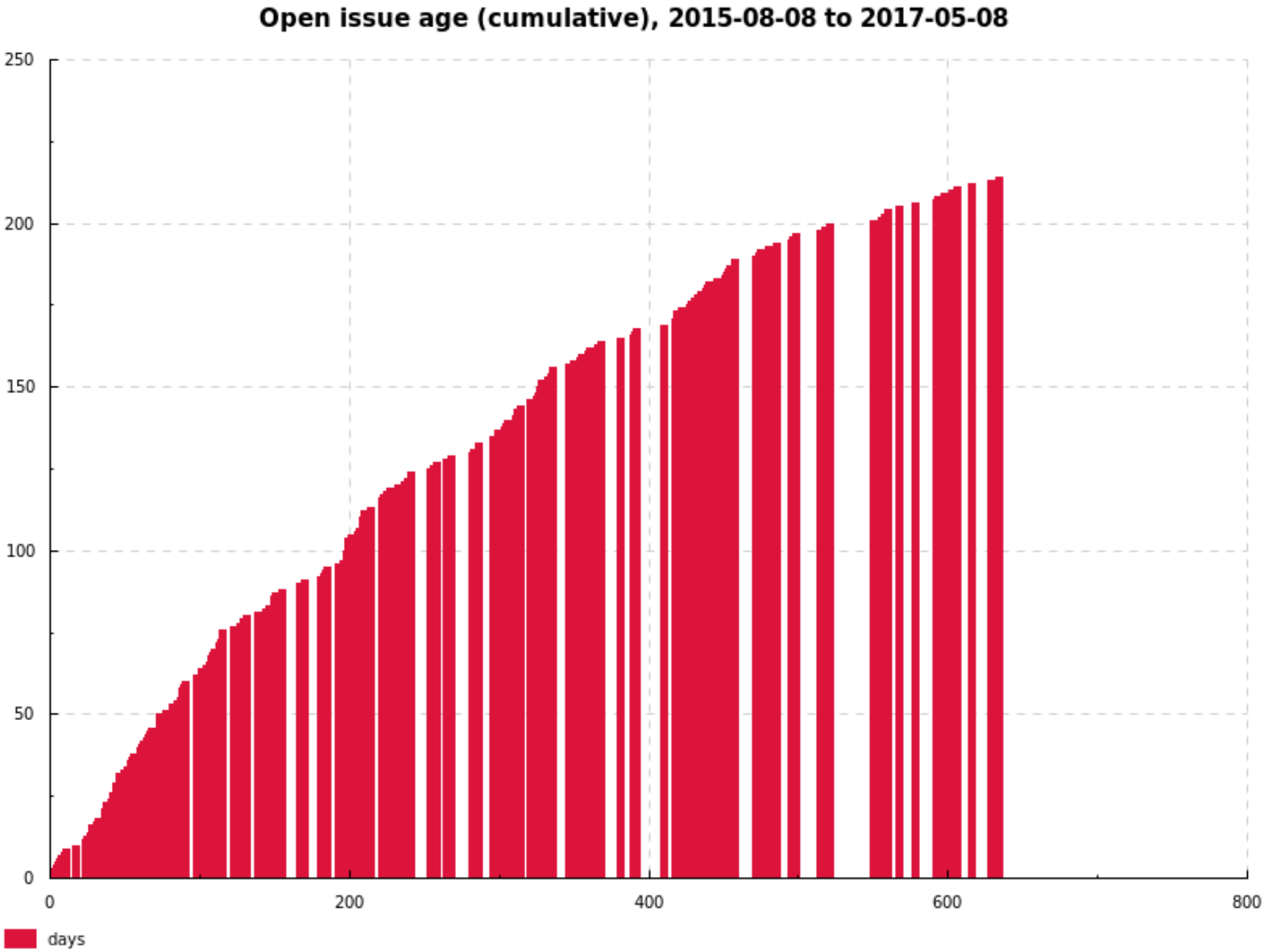
Agda: code by components

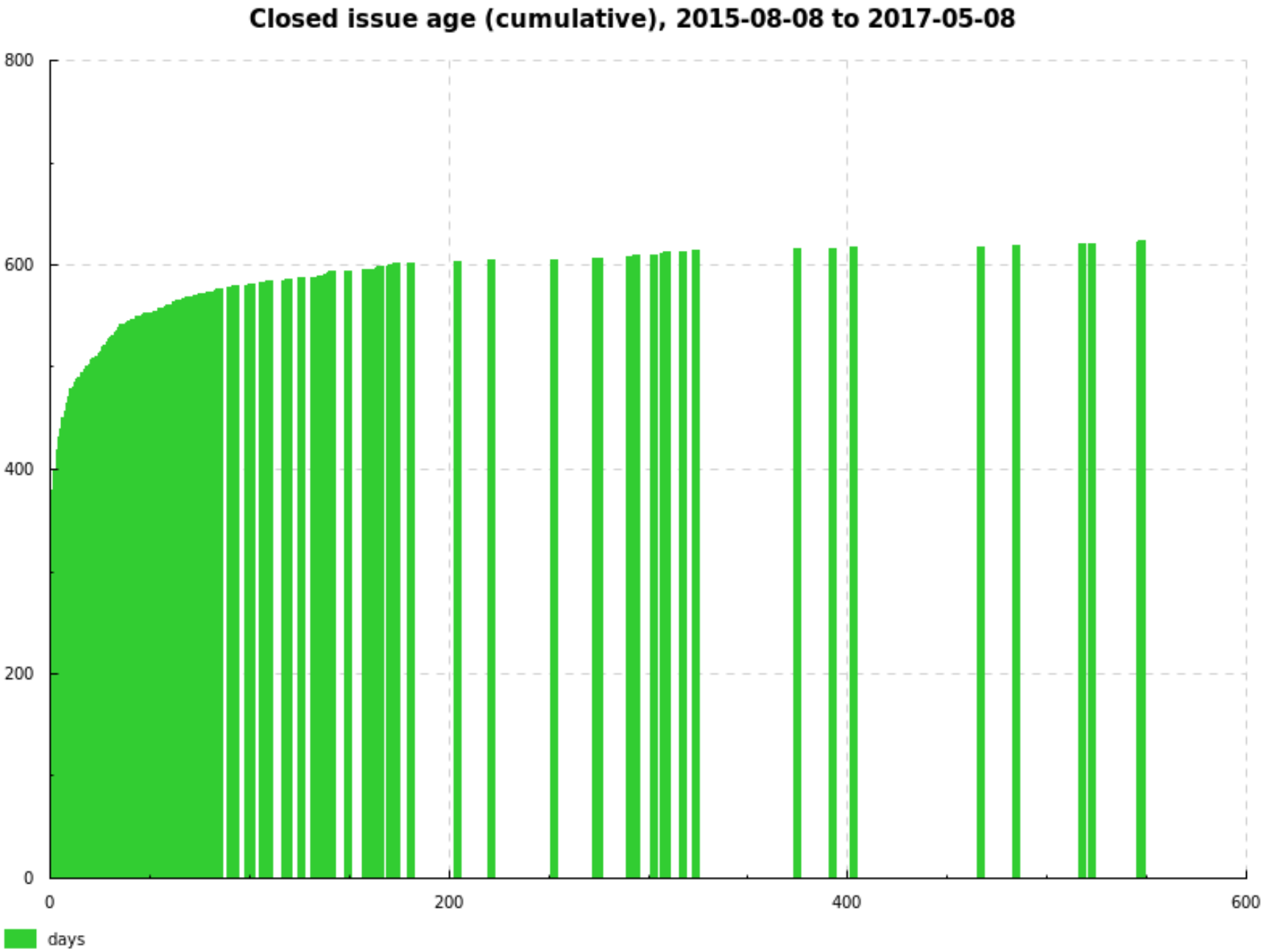
Component	loc ('17)	loc ('15)	loc ('14)
Utility functions	6800	8000	4100
Syntax (parse print scope)	23500	19000	16000
Type checker (eval. cov. pos.)	**49000**	39000	30000
Termination checker	4900	5800	4600
Interaction (imp. highl. LaTeX)	8600	7400	6600
Agsy	4200	4200	4100
Compiler	5500	8000	5200
total	103000	93000	71000

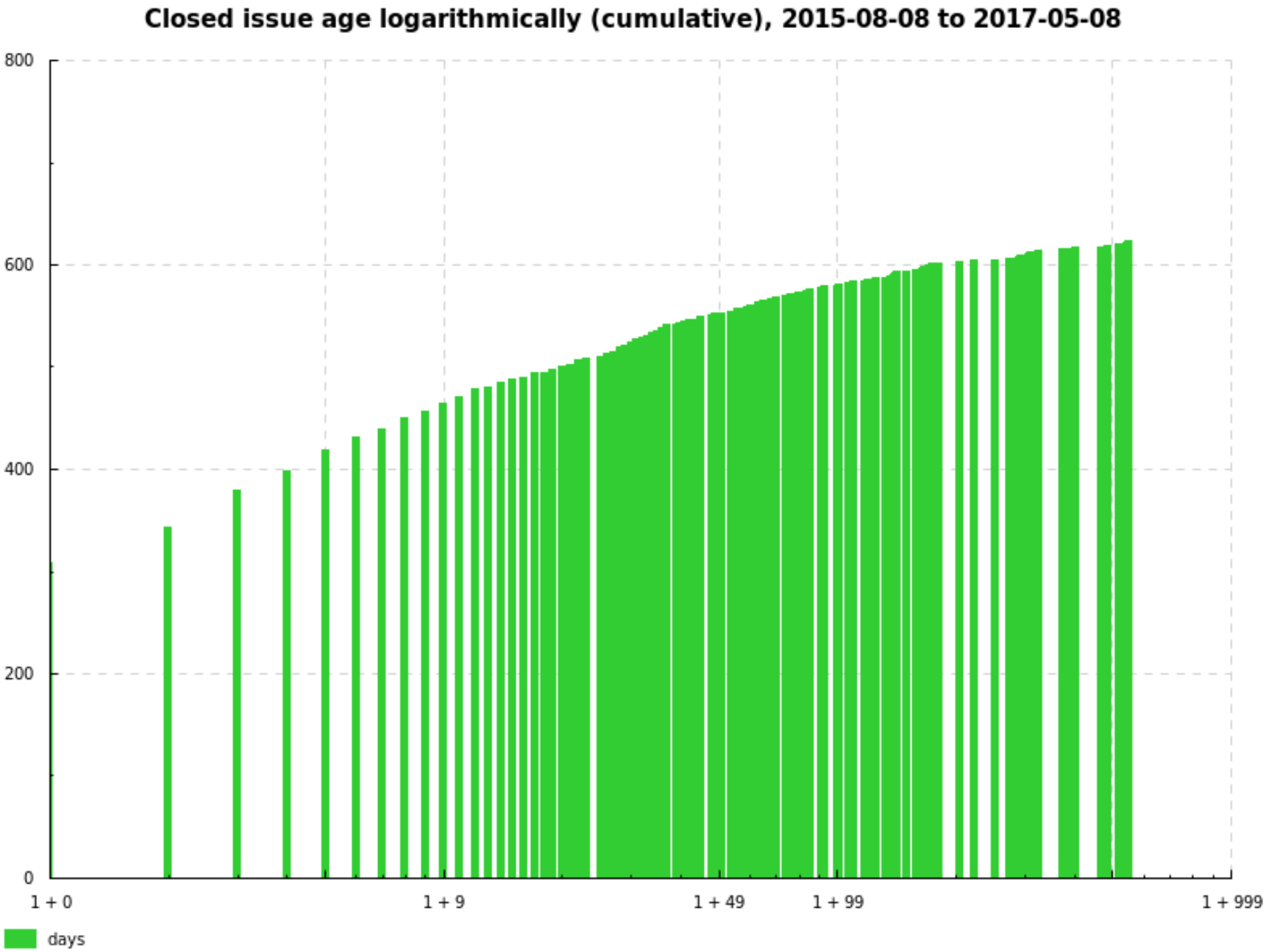

All Agda Issues
1800
1600
1400
1200
1000
800
600
400
200
0
2007
2008
2009
2010
2011
2012
2013
2014
2015
Per Year
Cumulative


Open Issues (May 2015)
Open (Cumulative)
300
250
200
150
100
50
0
2007
2008
2009
2010
2011
2012
2013
2014
2015
Year


Open vs closed issues 2015-08-08 to 2017-05-08
1000
750
500
250
0
0
200
400
600
800
open issues
closed issues


Open issue age (cumulative), 2015-08-08 to 2017-05-08
250
200
150
100
50
0
0
200
400
600
800
days


Closed issue age (cumulative), 2015-08-08 to 2017-05-08
800
600
400
200
0
0
200
400
600
days


Closed issue age logarithmically (cumulative), 2015-08-08 to 2017-05-08
800
600
400
200
0
1 + 0
1 + 9
1 + 49
1 + 99
1 + 999
days

Eating Our Own Dog Food

- We are developing a language for verification.
- Shouldn't we eat our own dog food?
- Develop Agda in Agda?
- Certainly a grand challenge.

Agda in Agda is out of reach, for a while

- We do not even have rigorous pen or paper proofs for our theory.
- We do not even have a theory.
- Most of the theoretical work focuses on the semantics.
- Semantics of small fragments of the language.
- Formalization of decidability $\Pi + \mathsf{Set}_0 + \mathbb{N}$: half a year of dedicated Agda grinding (Joakim hman).
- No advanced technology formalized: unification, pattern matching, termination ...

Santa Claus

Let's get millions of EUR from the EU!

Can we at least apply some dependent types to implement dependent types?

- Well-scoped syntax via Haskell GADTs.
- Could take care of de Bruijn index bugs.
- How does this scale to pattern matching, unification, with-abstraction?
- Type-checking monad needs to be indexed by the context length.
- First try this on a prototype!
- An advanced master thesis?

Prior to Formal Methods

- Agda is a programming language, so it should have a

  SPECIFICATION

- Write an informal specification!
- Flesh it out with test cases.
- We will have plenty of deviations from the specification.
- We can have testcases that should work but do not yet.
- We can gradually approach correctness.

Code quality: documentation

- Module documentation.
- Algorithm explanation by example.
- Documentation of our data structures.
- Pre- and postconditions of functions.
- 100% haddock coverage.
- Can only be reached by *code reviewing* requesting documentation.

Code quality: structure

- Break long procedures into several components!
- Write out the properties of the components.
- We have legacy spaghetti code with impenetrable control flow: `Interaction.Imports`
- Genesis: patches-over-patches
- Counterculture: refactorings, data structure evolution!
- We have monolithic state (`TCState`): pre-OO imperative programming.
- *Gegenentwurf*: modular monadic programming.

Long on the Wish List (2015)

- User manual
- Packaging
- Type classes (WIP)
- Universe cumulativity
- Reflection/tactics (WIP)
- Efficient type-checking
- Usable compiler (WIP)

Core Language / Internal Syntax (2015)

- Sharing
- Independent checking
- Termination certificates
- Shared optimizations/transformations used by compiler backends

Research topics (2015)

- Equality (HoTT, OTT)
- Parametricity/colors
- Sized dependent types
- Proof/instance search and unification
- Foundation for hidden/named arguments
- Telescopes/Σ-types at framework level
- Printing

Action items for us

- Install a coding code of conduct (à la style guide).
- Install mandatory code reviews.
- Framework for test-backed specification.
- Stabilize core features of Agda.