SSBSE Challenge
Optimizing Apache Ant

What can a Big Software Teach us about Optimization?

Márcio Barros
PPGI - UNIRIO

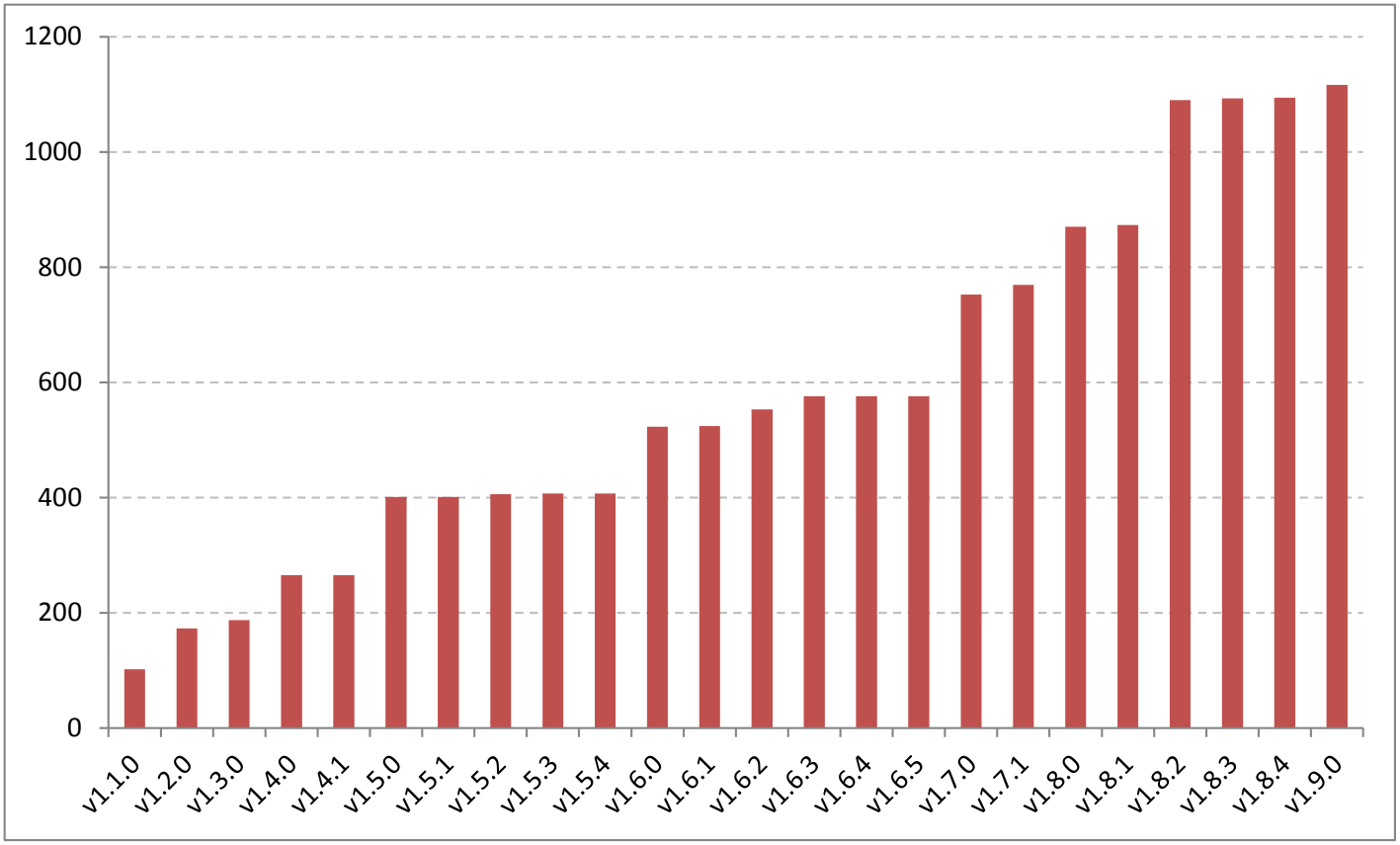
Fábio Farzat
COPPE/UFRJ

Apache Ant

A build automation tool which supports development teams to continuously integrate the results of their software development effort

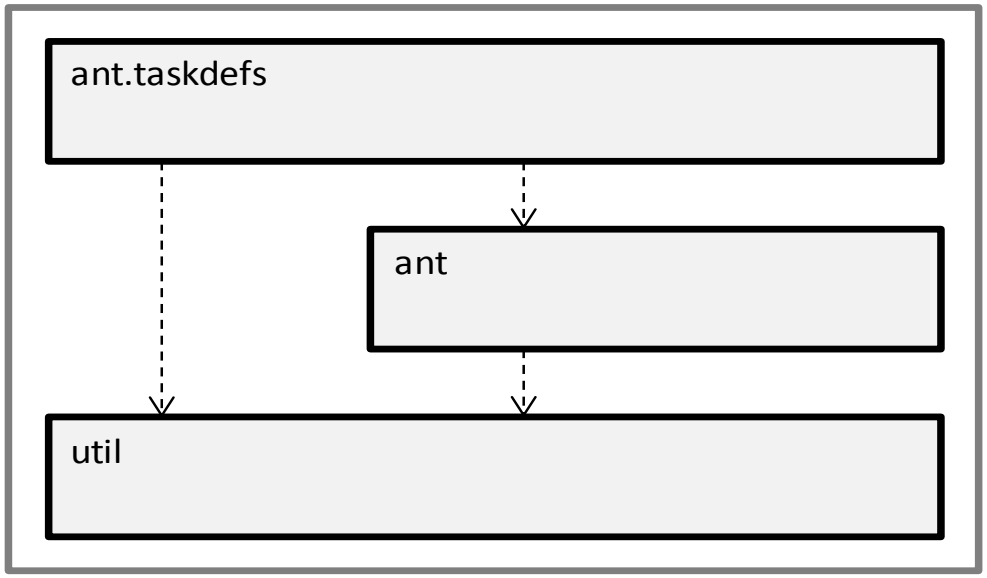
Ant uses a XML files to describe which tasks are required to produce and evaluate the software – and execute these tasks in a proper order.

Is the ANT really big?



Number of classes over different versions

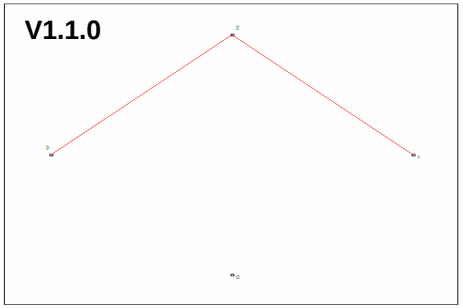
In the beginning, a very simple architecture ...



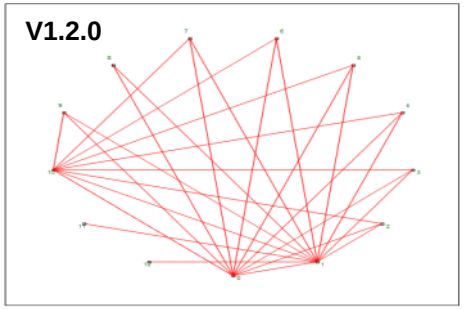
Everything fitted pretty well in just three components

... but evolution has been a quite different story.

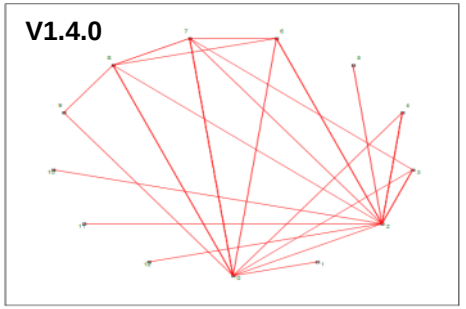
C:102, P: 4 (07/2000)



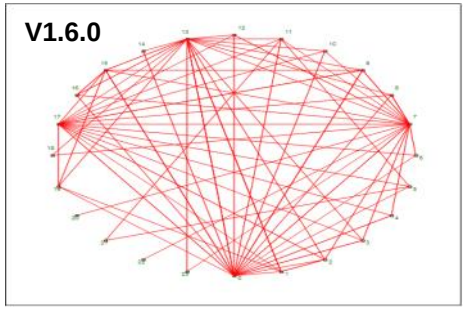
C: 173, P: 13 (10/2000)



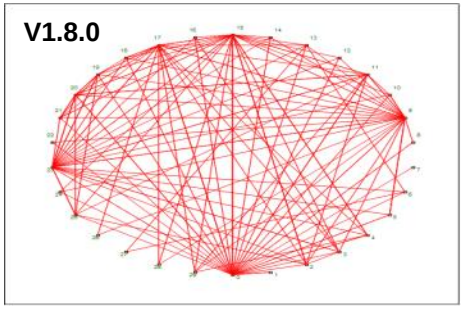
C: 265, P: 13 (09/2001)



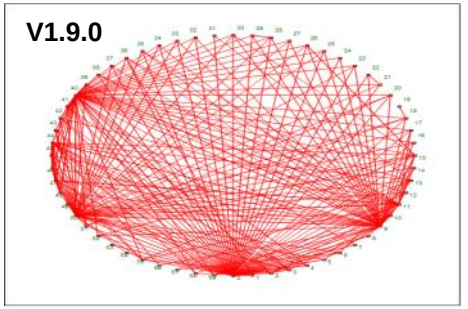
C: 523, P: 24 (12/2003)



C: 870, P: 30 (10/2010)



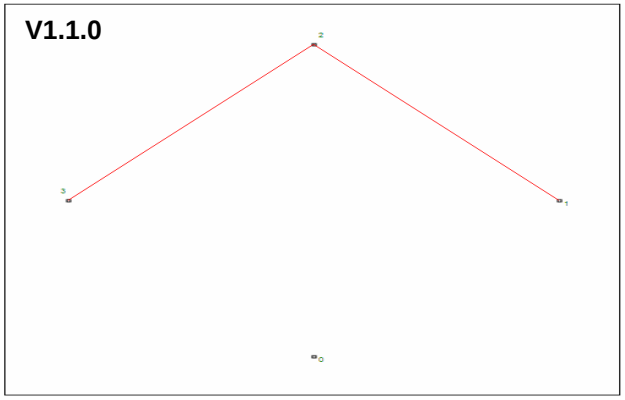
C: 1116, P: 60 (03/2013)



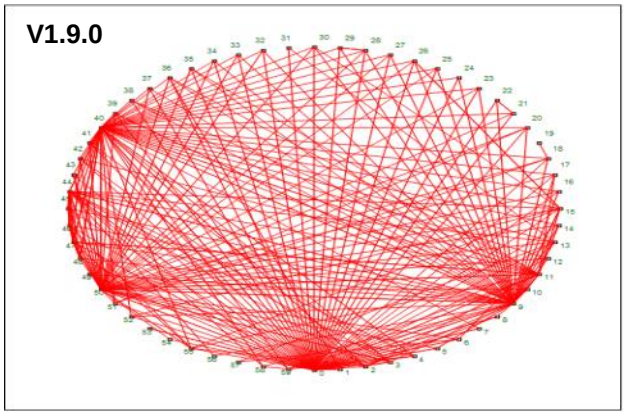
“Architecture erosion is the process by which a system’s architecture gradually degrades as maintainers make changes to the system that violate the original architectural intents.”

Our fundamental question!

Why does something
conceptually as simple as this ...



... become such a complicated
structure as this?



A good distribution of modules creates a set of building blocks in which each type of change is confined to a well-known set of modules and work assignments may be more easily distributed

What to avoid: shotgun surgery

The change looked like a simple drop of water, but in the end ...

What to avoid: shotgun surgery

Average number of classes and packages affected per commit to the version control system (for commits with more than a single class)

Year	Classes	Packages
2008	4.1	2.3
2009	5.1	2.8
2010	5.4	3.2
2011	5.9	3.1
2012	16.8	4.7
2013	6.8	3.0

What to avoid: shotgun surgery

These numbers may not be a sign of the anomaly, but a result of high developer turnover or any other factor which we do not control

Ant's team had on average 7 active developers from 2010 to 2012, but looses 3 developers and gain 2 new developers a year.

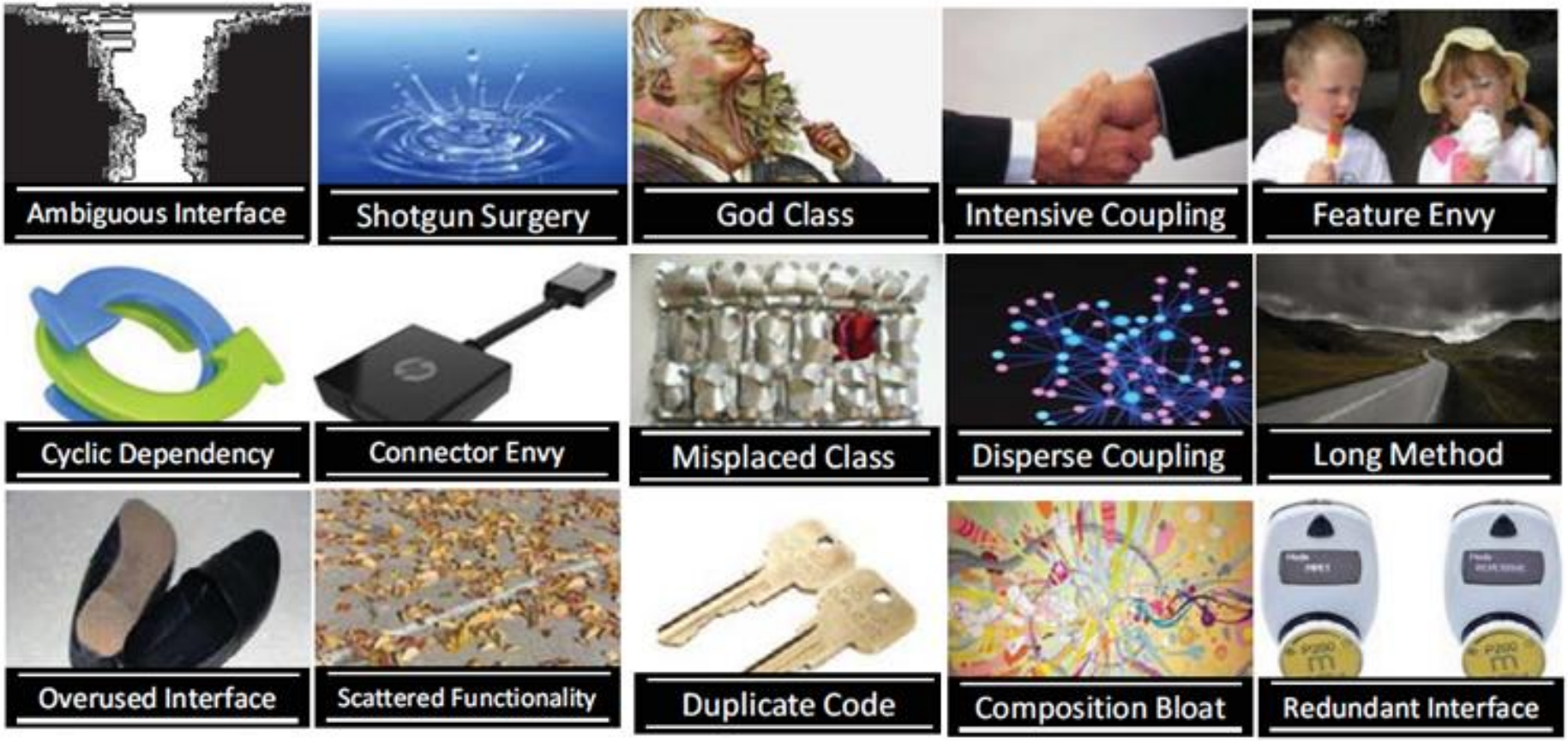
But we have also signs of stability …

The number of classes and packages remained almost constant from Dec/10 to May/12.

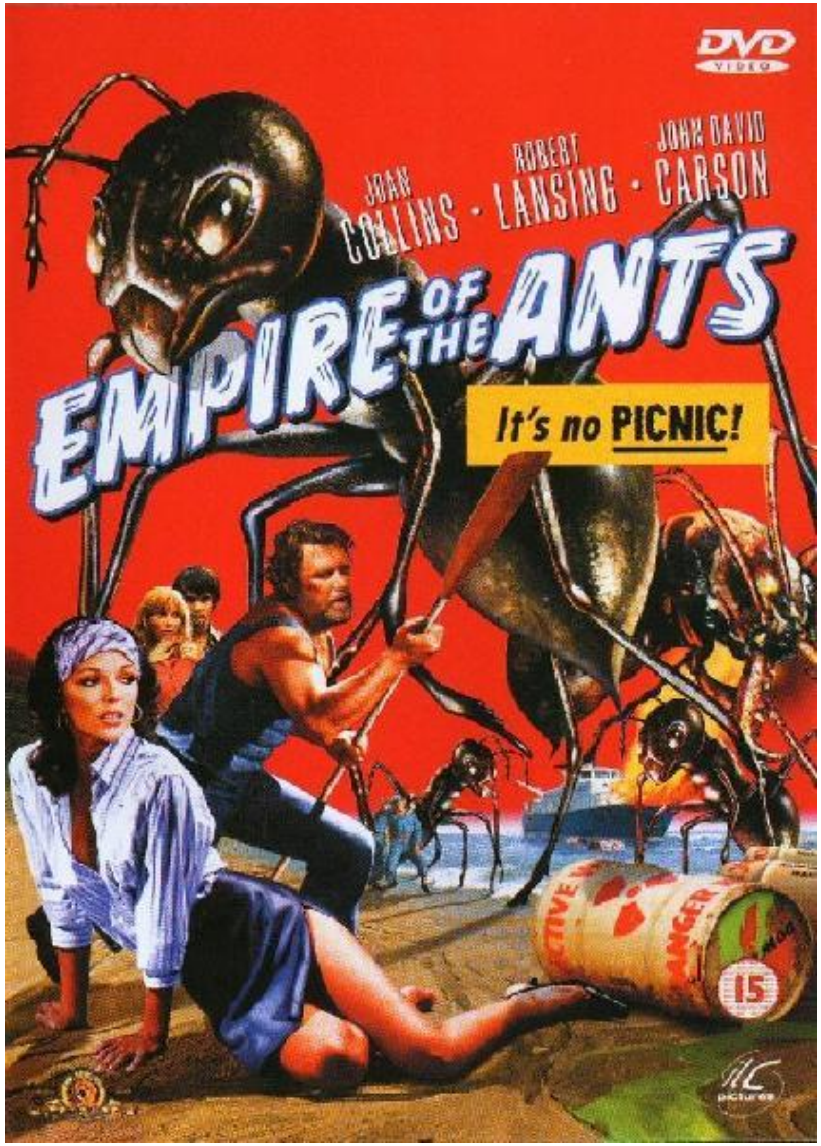
The number of reported bugs is decreasing (200 in 2010, 122 in 2011, 130 in 2012).

So, it seems that every new change is more spread in classes and packages.

This is one guy, but there are many ...



These many coding problems may turn your software into a nightmare



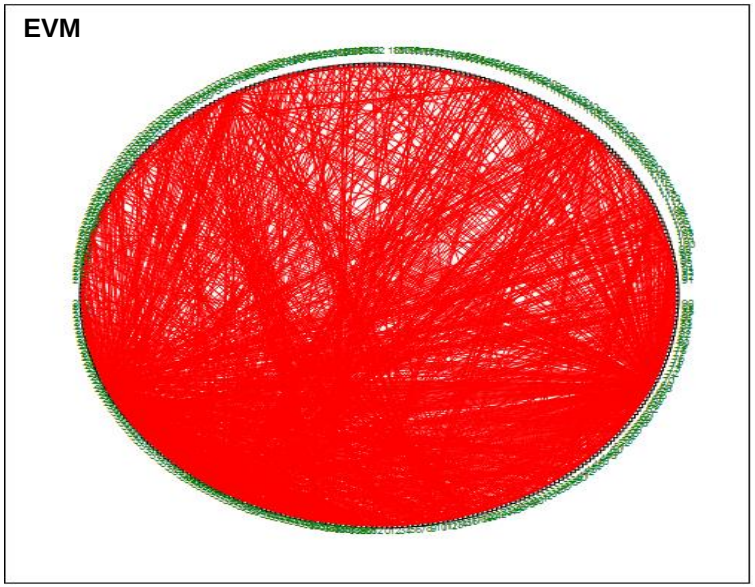
SBSE to the Rescue

Can search-based software module clustering aid in restoring the original architecture?

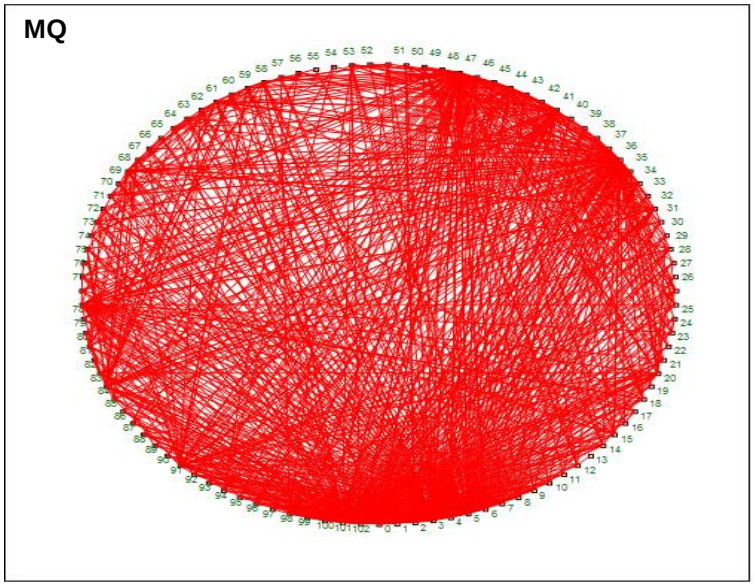
Results were not good with our current models ...

The selected metrics were strongly improved, but the design is quite different from the original architecture.

Original EVM: -49,000 / Optimized EVM: ~750

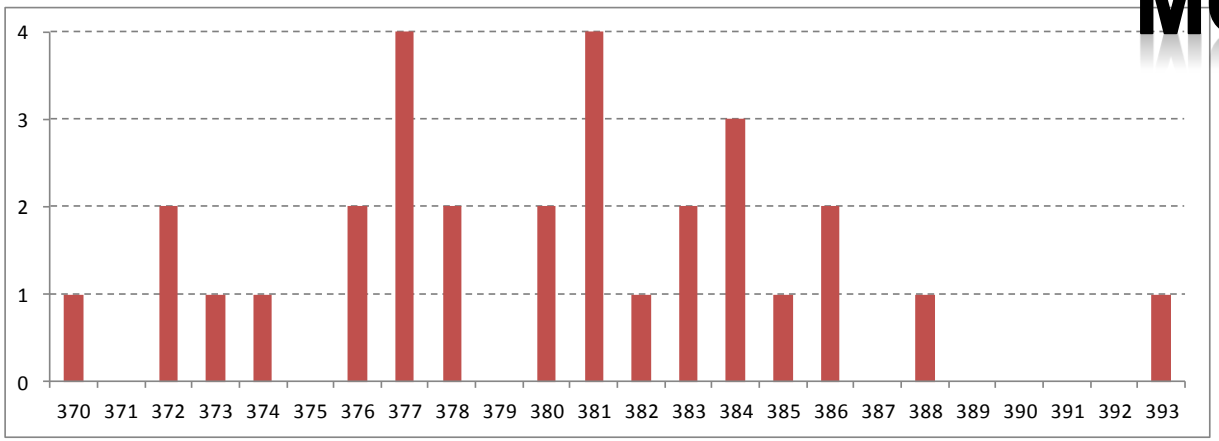


Original MQ: ~21 / Optimized MQ: ~100

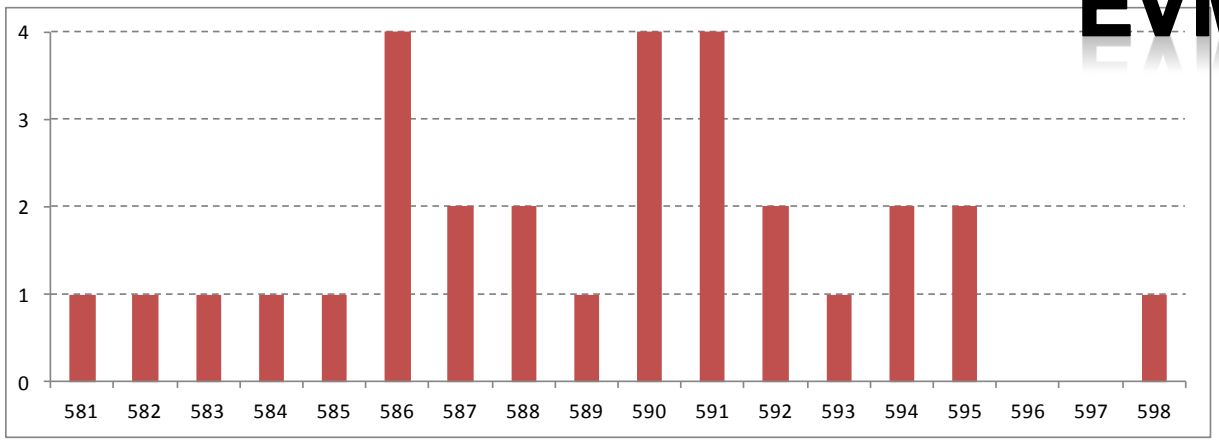


Distribution of the number of suggested packages

MQ



EVM



The scape route ...

- Optimization finds unexpected ways to improve the fitness function

- Like programming, optimization does what the researcher tell it to do, not necessarily what the researcher wants!

So, what can a big software teach us about optimization?

It taught us that SBSE is a learning tool

So, what can a big software teach us about optimization?

- Optimization take our theories and the models that describe them to extremes … and allows us to see their behavior on such scenarios (probably for the first time)

- At least for software clustering, we need better models and metrics that better reflect developer's intention in large software systems

- The dogma of increasing cohesion & reducing coupling is being questioned (e.g. Anquetil & Laval, 2011)

- Optimization based solely on these metrics is under suspicion!

So, what can a big software teach us about optimization?

- We need to find ways to introduce multiple user perceptions into the optimization process (there seems to be a lot of gut feeling here!)

- Some steps toward this goal have been made (Hall, Walkinshaw & McMinn, 2012; Bavota, Carnevale, DeLucia & DiPenta, 2012), but we need to take the problems to a level on which developers can contribute and let optimization fill in the gaps

- Learn from developers using automated learning processes and drive optimization on these uncovered trends

So, what can a big software teach us about optimization?

- As I came to see SBSE as a learning tool, I became convinced that we need the means to transfer knowledge

- A key issue here is visualization – optimization produces large data sets and we need to improve the way we handle this data to really translate the information it carries into knowledge

That is what we have learned.

Thank you!