# Package Dependencies Analysis in Pharo

Baptiste Quidé, Polytech Lille

## Problematics

- Modularity of code in Pharo
- Packages have many dependencies among them
- No tool to visualize package dependencies and detect the cycles in Pharo

## Main goals

- Avoid cycles among the packages
- Analyze automatically your code
- Provide feedback to users on which packages their code depends on
- Detect wrong project description (declared dependencies) or simply visualize the dependencies

## Several kinds

- Reference : explicit reference to a class of another package in a method
- Inheritance : class with a super class hosted in another package
- Trait : use of trait hosted in another package
- Extension : definition of an extension method for a class from another package

# Visualize the dependencies among your packages

- Based on Lukas Renggli project (Pharo 1.4), port to Pharo 4, write tests, add doc.
- Analyze and find all static dependencies
- Showing the results in a UI (using Spec Framework)

## Demo with the packages “Collections” on Pharo

## Detect cycles among your packages

- From a package dependency graph, use of the Tarjan algorithm to find all the strongly connected components
- In a SCC each node can be reach by other node (there is a path)
- Cycles exists only among the nodes (packages) on the same SCC
- For each SCC, run the detect cycle algorithm and find all the cycles

## The algorithm

- Published in JOT 2011 (written by J. Laval, JM. Fellary, P. Vismara, and S. Ducasse)
- Complexity acceptable to be applied at development-time (500 packages as a fair upper-bound)
- Retrieves a set of short cycles that covers all dependencies

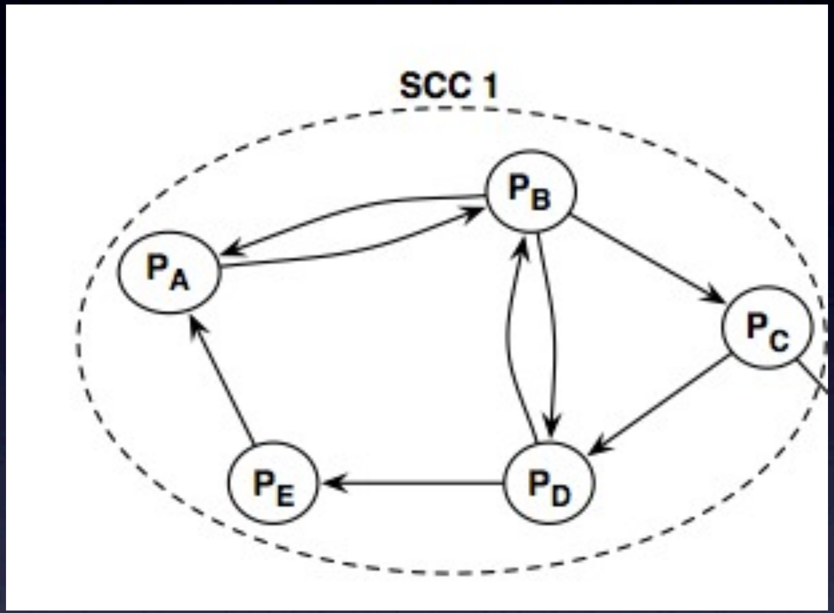
# Intuition

- Retrieve only a polynomial number of the cycles, reducing time and complexity.
- Selecting only a subset of elementary cycles
- Select for each dependency one on the shortest cycles going through the dependency

## Details

- Idea is to gather the parents A of a node x
- Perform a breadth-first search from x until all its parents y are found
- BFS find the shortest path from x to y
- One BFS is performed for each node
- Apply this for each node of the SCC

# Example of application on a SCC



- For the node PB. Parents : PA and PD.
- BFS started from PB will find PA by (PB, PA). PA -> PB is a cycle.
- BFS started from PB will find PD by (PB, PD). PB -> PD is a cycle.
- For the node PC. Parents : PB.
- BFS started from PC will find PB by (PC, PD, PB). PC -> PD -> PB is a cycle.
- repeat this step for each node of the graph...

## Demo with the packages “Collections” on Pharo

# Future improvements

- Visualization with Roassal2 and Telescope
- Integration in Nautilus
- Use packages meta-information to store "normal-cycles"
- Metric for ranking the cycles?
- Live feedback?
- Factorization with Moose?

# Available on SmalltalkHub

- We need feedback !
- SmalltalkHub
- http://smalltalkhub.com/#!/~BaptisteQuide/PackageDependenciesAnalysis

Questions?