But one question we have for discussion at the workshop is: are these techniques also suitable for **detecting ’manually woven’** concerns which are more (and only) characterized by a high degree of scattering? As these consist of single lines of code, techniques for detecting copy/pasted duplication of

larger blocks of code may not be suitable.

Another observation that can be made is that the concerns of logging and updating were implemented using a single central entity, particularly a unique method:

Unique method: “A method without a return value which implements a message implemented by no other method.”

We’ve essentially used fan-in analysis [10] as our approximate heuristic for measuring scattering. A difference with the work of Marin et al. is that we’ve applied the analysis only to unique methods, as our interest is mostly in identifying candidates for applying a method-to-advice refactoring [5], while Marin et al. use it for detecting good ”seeds” from where to start browsing; this can help find larger aspects such as support for undo operations in a graphical editor which however also requires more complex refactorings to extract. **(parece ser que una busca seeds y la otra aspectos directamente. Se podría aplicar primero fan-in, y luego dar una especie de filtro mas con unique methods, chequear esto)**

In this paper we presented a light-weight heuristic for identifying aspect candidates in legacy code. We applied our heuristic to an entire Smalltalk image and were able to detect a number of aspects. The heuristic is based on the observation that in pre-AOP days certain crosscutting concerns were implemented by ’manually weaving’ aspects as method calls to a central entity.