

# Research Statement

Within IFIP WG 2.11

**Cristina Videira Lopes**  
**University of California, Irvine**

My research has always been focused on problems, rather than on solutions. Software engineering offers a myriad of problems, most of them ill-defined. I find it most exciting to look at some of those problems and formulate them in such a way that the solutions emerge naturally. The need for program generation has been an important component of those solutions, and that is the reason why I share common interests with this group. But I see these techniques as a means to an end, and not an end in itself. What follows is a brief description of some of my past work, as well as two of my current projects.

In the past, I helped set the foundations of Aspect-Oriented Programming and its flagship language, aspectj. The emergence of AOP is a great example of my stance on research. The mechanisms supporting AOP existed a long time ago – CLOS had them back in 1988. It is possible to implement AOP with reflection and the right set of macros – a basic, all powerful, mechanism for program generation. But AOP only emerged a decade later. What was missing was an appropriate problem formulation for justifying the use of those mechanisms. The “aspect story” filled that gap. Aspects were defined to be units of program concern that crosscut other units, and that composed by “weaving” – again, related to program generation. Unlike the powerful macros, the composition is not just about anywhere, but only in well-defined points of the programs, called join points. So, the implementation of AOP is about well-behaved program generation, or program generation with semantics, and its purpose is to support modular programming of crosscutting concerns.

I moved on from AOP, but I continue to work on programming languages. I have found two new exciting ill-defined problems. One has to do with the new world of programming Pervasive/Ubiquitous Computing. Take, for example, a sensor network consisting of sensors with very limited resources (e.g. power). Due to these limitations, designers may need to optimize the behavior of each sensor. In other words, the code that runs in each sensor may not be generic at all and, instead, depend on many factors such as the position of the sensor in the network. Obviously, due to the large number of sensors, it is not realistic to produce those specializations by hand, and some form of program generation is essential. I am looking into this problem and trying to devise a good set of high-level specifications that will generate a large number of different sensor programs. It seems that this can take advantage of work in program specialization, where the parameters include, at least, the topology of the network.

Another problem has to do with Web programming, which I recently found out to be a real mess. In spite of all the choices for producing dynamic page content (or because of them!), web programming is a marvel of complexity by means of code tangling. Issues of presentation, content, logic and navigation flow are carefully hand-woven line by line. Things can seriously get out of control when performance is taken into account and the pages that are supposed to be dynamic are generated ahead of time and stored in a database. I still don't know exactly how to formulate the problem, but it seems that there is a strong parallel between web pages and software modules, and that web programming could benefit from ideas coming from programming languages. Program generation emerges here too, as a means for producing the dynamic pages/software modules before run-time.