# Parallel Programming by Transforming “sequential like” Codes

## Context

In the next years every desktop machine will have more than one processing element. To leverage the power of these multi-core systems, software must be updated to include several parallel activities to take advantage of more than one processing element. This is a radical change when compared with previous processor generations and will have a radical impact on the way that software is designed and developed.
In this context, the programming paradigm with the highest probability of success would resemble to traditional sequential programming and should maximise the reuse of existing sequential codes. Unfortunately, current mainstream parallel programming languages require extensive and intrusive source code changes to enable these codes to take advantage of parallel computing resources. In large scale applications this can limit application modularity and consequently the ability to evolve application code in an independent manner. Moreover, many decades of research on parallelising compilers did not introduce a compelling alternative to a programmer based parallelisation.
This proposal aims to explore a new paradigm to develop parallel applications by transforming “sequential like” codes into parallel ones through a set of programmer specified high level transformations.

## Objectives

This thesis aim to develop high-level code transformations that enable ”sequential like” applications to take advantage of multi-core systems. In particular it will explore aspect-oriented techniques (and/or product lines) as a mean to perform this kind of code transformations.
This research will address the migration of software to multi-core systems by attempting to support a traditional “sequential like” style of programming and to minimise the changes performed to the source code to take advantage of parallel processing capabilities. Moreover, it is expected an increased modularity of parallelisation issues, enabling independent development of domain-specific and parallel code. That these are key features to promote a broader usage of parallel processing and easier reuse of legacy sequential code.
Specifically this thesis will help to find answers to the following questions:

What kind of transformations should be supported?
How to specify these transformations?
How to express common parallel patterns in this approach?
What are the main limitations of the approach?

## Orientation:

J. L. Sobral (and possibility to be co-advised by partners at UTexas, Austin)
CCTC – Universidade do Minho

## Contact:

jls@di.uminho.pt

## Bibliographic References:

J. Sobral, Pluggable Grid Services, 8th IEEE/ACM International Conference on Grid Computing (Grid 2007), Austin, Texas, September 2007

J. Sobral. Incrementally Developing Parallel Applications with AspectJ, 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS'06), Greece, Rhodes, April 2006

C. Cunha, J. Sobral, M. Monteiro, Reusable Aspect-Oriented Implementation of Concurrency Patterns and Mechanisms, Fifth ACM International Conference on Aspect Oriented Software Development (AOSD 06), Bonn, Germany, March 2006.

G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, J. Irwin, Aspect-Oriented Programming, European Conference on Object-Oriented Programming (ECOOP), 1997.

G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, W. Griswold, An Overview of AspectJ, European Conference on Object-Oriented Programming (ECOOP), June 2001.

D. Batory: A Tutorial on Feature Oriented Programming and the AHEAD Tool Suite, GTTSE 2006.

N. Mahmood, Y. Feng, James C. Browne: Evolutionary performance-oriented development of parallel programs by composition of components, WOSP 2005.