2018-00395 - [Campagne Doctorant 2018/CRI LILLE] - PhD
Thesis: Programming language migration for numerical modeling applications (M/F)

Type de contrat: CDD de la fonction publique
Niveau de diplôme exigé: Bac + 5 ou équivalent
Fonction: Doctorant
Niveau d’expérience souhaité: Jeune diplômé

A propos du centre ou de la direction fonctionnelle

The Inria Lille - Nord Europe Research Centre was founded in 2008 and employs a staff of 360, including 300 scientists working in sixteen research teams. Recognised for its outstanding contribution the socio-economic development of the Nord - Pas-de-Calais Region, the Inria Lille - Nord Europe Research Centre undertakes research in the field of computer science in collaboration with a range of academic, institutional and industrial partners.

The strategy of the Centre is to develop an internationally renowned centre of excellence with a significant impact on the City of Lille and its surrounding area. It works to achieve this by pursuing a range of ambitious research projects in such fields of computer science as the intelligence of data and adaptive software systems. Building on the synergies between research and industry, Inria is a major contributor to skills and technology transfer in the field of computer science.

Contexte et atouts du poste

Job environnements

Numerical modelling, or Computer simulation, is a primary tool in all fields of engineering and sciences (e.g. physics, astrophysics, climatology, chemistry, biology, automobiles, power plants, economics, . . . ) to study at an arbitrary level of details the phenomena that occur in these domains, to improve the understanding of these phenomena, to propose new solutions, or to test these solutions more rapidly and at a lower cost than physical experimentation.

As the name implies, the domain of numerical modelling, owes much to abstraction and mental conceptualization. The efficiency of these applications relies on the ability of the engineers to correctly and efficiently implement the mathematical abstraction they have in mind. Unfortunately, for historical reasons, these applications are typically implemented in Fortran, a language conceived in the 50’s that lacks high level abstraction concepts such as offered by Object-Oriented programming. The language itself is no longer taught in most universities, which makes it difficult to find skilled programmers. Migrating to an OOP language as C++, should isolate the implemented concepts one from the other; thus allowing to modify any part of the application without impacting the rest of it, which is rarely possible currently. This is a real problem that developers of Code Carmel (our subject application, see below) are experiencing currently.

GOAL: The thesis will provide tools to help developers migrating a legacy procedural FORTRAN application in an modern object oriented C/C++ one.

This thesis project is a collaboration between two research teams of University of Lille: RMod, a Software Engineering research team; and OMN, a Electronics and Electrotechnics research team.

Mission confiée

Research plan

Migrating tens or hundreds of thousands of lines of code from one programming language to another is already a significant challenge, but changing the programming paradigm from procedural to OOP, is even more difficult. It requires defining high level abstractions (i.e. classes) that might already be latent in the developers’ minds, but so tangled with unrelated instructions in the source code that they are difficult to identify.

The thesis will consider code migration at different levels of abstraction and complexity. The plan is to treat each level of abstraction independently from the other in successive phases of the thesis so that one abstract issue will not pollute another simpler one ("Migrate then Redesign" approach [BRPPIQ]).
The PhD student will need to:

- **Syntax**: We need to convert source code from one programming language syntax (Fortran) to another one (C/C++). This is not considered a difficulty for two languages like Fortran and C that are very close one to the other (scientific, statically-typed, procedural languages). Still, some issues may arise from the flexible use of arrays in Fortran [FLQZ97].

- **Idioms**: All programming languages have typical ways of doing things that must be respected for the code to look “natural” ([Fel90, BRPPP10]) (e.g. iterating through a table by using a pointer in C). For the final code to be easily understandable and maintainable, it should follow these standards. This will be achieved by having transformation rules applied on the already converted C code. To follow up on the simple example of manipulating arrays in C, we will identify patterns of FORTRAN code (a loop manipulating a table) that should be transformed and provide a flexible modification of the Abstract Syntax Tree (AST) to transform it in a more natural C/C++ code.

- **Libraries**: Any application depends on some library that needs to be replaced in the migrated code. Such replacement can be one-to-one in the best case, but may often require to develop wrappers, or a new component (when no satisfactory substitution can be found). In our case, we will be helped by the fact that Fortran and C can easily interact. The FORTRAN code always uses some C libraries and the C/C++ code will initially use some FORTRAN libraries. Thus we plan to keep the original libraries in a first time, and only after migration, replace the Fortran libraries by C/C++ ones as the programmers will see fit.

- **Patterns of optimization**: We will look for domain specific patterns of code that would allow for advanced optimizations of the C code or opportunities for parallelization. Again, this will rely on AST pattern matching with modification of the AST to get optimized code. The advantage of this solution is that it can be continue to be used after the end of the project as new optimization patterns are discovered. RMod has an ongoing thesis (Jason Lecerf, in collaboration with CEA) on this subject ([RDDL17]).

- **Paradigm**: Transforming procedural programs into object-oriented ones is an old challenge of software reverse engineering (e.g. [NK95, CCLL01]). Past research focused a lot on full automation and source code analysis. This approach has its limits as the code representing possible classes is often intertwined with other unrelated code. We will offer an interactive environment, where experts of the application can propose domain concepts that are expected to be encountered and look for beacons of these concepts in the source code (e.g. in identifiers or in comments). From the initial set of code entities thus recovered, the experts will be able to add other entities that are highly related to the initial ones through several query mechanisms. The process will be iterative with a gradual refinement of the understanding of the concepts and their implementation. Once the expert is satisfied with the identified set of functions and data, the tool will modify the source code to replace these C entities by C++ classes.

   Note that Grosse-Kunstleve et al. [GKTS12] made an opposite choice and tried to “eliminate the need for manual work”. But this was at the cost of the quality of the OO code, where classes are generated from Fortran programs which are a too coarse level of abstraction.

   On the other hand, we successfully used an iterative and manual approach, with gradual refinement, in the Synectique company, for one of its client, to help split huge classes (thousands of lines of code) into smaller ones.

Bibliography


Principales activités

The PhD student will need to:

- spend some time with the OMN developers group to gain insider understanding of the application domain constraints and get acquainted with the Fortran programming language.
and the Code_Carmel application itself;
- conduct a literature survey on programming language migration, procedural to OO migration, and Fortran to C/C++ migration;
- propose solutions to the Code_Carmel migration problem and implement them as tools within the Moose software analysis platform;
- write scientific article to disseminate the results of the research.

**Compétences**

**Skills:**

- Programming and particularly Object-Oriented programming
- Some mathematical background would help in understanding the constraints of the application domain
- Good English writing skills
- A preliminary knowledge of the Pharo programming language is not required but would be an advantage.

**Avantages sociaux**

**Benefits**

- Subsidised catering service
- Partially-reimbursed public transport
- Social security
- Paid leave
- Sports facilities
- Flexible working hours

More information about Lille:

http://www.lille3000.eu/portail/
http://www.lillemetropole.fr/mel.html

**Rémunération**

**Remunerating**

The gross monthly salary is 1982€ for the 1st and 2nd year, 2085€ for the 3rd year.