# FIT Project Description

# Introduction

Since the 1990’s, many different fuel cycle systems tools have been developed by several institutions (industrial, engineering, academic, etc.). These tools vary in terms complexity, from a simple spreadsheet model to a complex simulation code framework. These tools have evolved and new tools have been developed to leverage more increasingly powerful computational advancements. Many of these tools have the option to be used at different levels of detail and have the flexibility to be adapted for one specific problem or to any problem related to the nuclear fuel cycle.

The fuel cycle systems tools of interest in this activity are dynamic fuel cycle simulators. They are an essential part of the technical evaluation of the future deployment of innovative nuclear systems. Also, they help identify drivers and interactions between parameters in fuel cycle fleet physics. Finally, these tools produce data for further assessments (economy, safety, non-proliferation, etc.) that can be internal or external to the tool. For these reasons, fuel cycle simulators can be viewed as vital tools to inform and support decisions related to the nuclear fuel cycle of a country or multiple countries.

Many of these tools were developed independently by a single institution and oftentimes by a single individual with limited feedback from an established user community or from other developers. There have been multiple international fuel cycle code benchmarks in the past but these were dedicated to comparing detailed time-dependent results for fully-described fuel cycle system evolutions over time with limited comparisons of specific tool functionalities. The goal of this project is to also **improve the confidence in the calculations of these fuel cycle simulators**, but with more of a focus on comparing different tool functionalities in isolation, hence the name Functionality Isolation Tests (FIT). The purpose of the project is to **test the impact of a Fuel Cycle Code (FCC) functionality** and how this **impact is propagated** over time in the fully-described fuel cycle system calculations.

# List of functionalities

In this part, the functionalities to be tested in the framework of the FIT project are listed and described. The list is not exhaustive and all participants can add new functionalities.

1. Updated Fuel Composition vs. Fixed Fraction

This specification aims to compare the impact of the fuel composition update according to fissile isotopic composition. This can be done via a Fuel Loading Model (FLM) or a Recipe approach. This approach is compared to a Fixed Fraction (FF) approach, which means the fissile fraction is constant regardless of the isotopic composition of the fissile material.

2. Fuel Shortage Response

Here, we'll try to assess the impact of the fuel shortage response in the FCC. Usually, the response would be:

An empty cycle

* Reactor is shut down until there is enough material
* Needed material is created
* Etc.

3. Variable Capacity Factor vs. Constant Capacity Factor N G ME

This item is connected to economic evaluation. Variable capacity factor could be due to actual outage modeled, seasonal changes in demand, etc.

4. Accurate physics of mixed cores vs. Assuming constant core

5. Exact startup composition vs. Single composition at beginning of operation G ME

+ same with shutdown

6. Reprocessing As-Needed vs. Constant Reprocessing if decay modeled, it may matter

6’ Fabrication as needed VS constant fabrication

7. Isotopic Decay vs. No Decay First in First Out for all materials

8. Constant XS vs. Updated XS

9. Fuel type switching method

Smooth Transition, brutal transition, etc.

10. Constrained Processing (Fab, Enr, Rep, etc.) vs. Unconstrained Processing

11. Coarse time steps vs. Fine time steps

12. Fleet-based reactors vs. Agent-based reactors

13. Batchwise fuel management vs. Simulated continuous reloading related to MSRs maybe

14. Batchwise fuel management vs. Continuous reloading vs 1 batch reloading

15. Macro reactor versus detailed fleet N M

16. Storage as a “soup” VS kept batch sorting M

# The github repository

You can download the github repository from the following command:

$ git clone https://github.com/FuelCycleFIT/FITProject.git

$ cd FITProject

From that point you can fork the repository from github web site and define upstream and origin remotes.

$ git remote set-url origin https://github.com/[YourAccount]/FITProject.git

$ git remote add upstream https://github.com/FuelCycleFIT/FITProject.git

If your repo is properly defined, you should have following remotes:

$ git remote -v

$ origin https://github.com/[YourAccount]/FITProject.git (fetch)

$ origin https://github.com/[YourAccount]/FITProject.git (fetch)

$ upstream https://github.com/FuelCycleFIT/FITProject.git (fetch)

$ upstream https://github.com/FuelCycleFIT/FITProject.git (push)

From this point, you should be able to push modifications on your fork and to request merge on the upstream repository.

# Participants and Codes

# Participants and Fuel Cycle Codes (FCC)

* CNRS / IN2P3 (Xavier Doligez, Marc Ernoult and Nicolas Thiollière) - CLASS
* CEA (Guillaume Krivtchik) - COSI
* Madison University (Paul Wilson and Baptiste Mouginot) - CYCLUS
* University of Illinois at Urbana-Champaign (Katy Huff) - CYCLUS
* Argonne National Lab (Bo Feng) - DYMOND
* Oak Ridge National Lab (Eva E. Davidson) - ORION
* CIEMAT (Aris Villacorta) - Tr\_Evol / Evol\_code
* TRACTEBEL (Hubert Druenne, Bart Vermeeren) - ANICCA
* Univ. of technology and economics of Budapest (Mate Halasz) - JOSSETTE
* Universidad Católica del Maule (Ivan Merino) - ANICCA

# Codes involved

Each institution describes the FCC that they will use for this project along with the Reactor Physics Code (RPC), if appropriate.

# Fuel Cycle Codes (FCC)

# **CLASS - Developed by CNRS/IN2P3**

The code CLASS (Core Library for Advanced Scenario Simulation) is a dynamic fuel cycle simulation tool developed by CNRS/IN2P3 (Centre National de la Recherche Scientifique / Institut National de Physique Nucléaire et de Physique des Particules) in collaboration with IRSN (Institut de Radioprotection et de Sûreté Nucléaire). The aim of CLASS is to model an evolving electro-nuclear fleet. The main output is the evolution of isotopes everywhere in the fleet.

The CLASS model is a collection of C++ classes that describes facilities in a nuclear fleet. The CLASS model has been built around the reactor class that drives radioactive material flows from reactor front to back end.

The latest shared version is the [official Version 5.1] :

<https://gitlab.in2p3.fr/sens/CLASS/tree/CLASS_V5_Official_Release>

# Reactor Physics Codes (RPC)

# **SMURE - Developed by CNRS/IN2P3**

There is two online sources for the SMURE code, the [official web page of SMURE](http://lpsc.in2p3.fr/MURE/html/SMURE/UserGuide/UserGuide.html) and the [NEA dedicated page](https://www.oecd-nea.org/tools/abstract/detail/nea-1845)

The main aim of the SMURE package is to perform nuclear reactor time-evolution using the widely-used particle transport code MCNP (a Monte Carlo code which is mostly written in FORTRAN) or SERPENT. In SMURE, due to the Object-oriented programming, any user can define his own way to interact with evolution. Moreover, SMURE provides a simple graphical interface to visualize the results. It also provides a way to couple the neutronics (with or without fuel burn-up) and thermohydraulics using either an open source simple code developed in SMURE (BATH, Basic Approach of Thermal Hydraulics) or a sub-channel 3D code, COBRA-EN. But SMURE can also be used just as an interface to MCNP or Serpent to build geometries (e.g. for neutronics experiments simulation).

You can also refer to following publication:

Méplan O., Nuttin A., Laulan O., David S., Michel-Sendis F. et al.: MURE : MCNP Utility for Reactor Evolution. Description of the methods, first applications and results, Proceedings of the ENC 2005 (CD-Rom) ENC 2005. European Nuclear Conference. Nuclear Power for the XXIst Century : From basic research to high-tech industry, France\*