



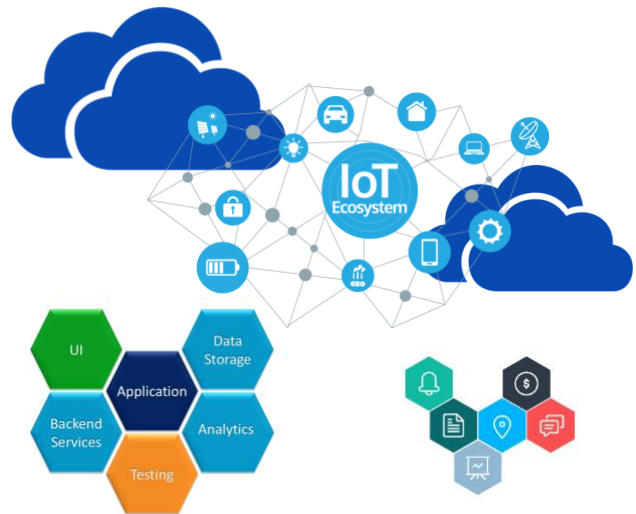
# CS486C – Senior Capstone Design in Computer Science

## Project Description

<b>Project Title:</b> A Graphical Framework for Distributed Software Deployment	
<b>Sponsor Information:</b>	
 	Hélène Coullon, Assistant professor STACK team IMT Atlantique, Inria, France helene.coullon@inria.fr  Frédéric Louergue School of Informatics Computing and Cyber Systems Northern Arizona University frederic.louergue@nau.edu

### Project Overview:

With the increasing popularity of heterogeneous distributed computing infrastructures, the deployment and maintenance of complex computing, storage and network infrastructures continuously increases, leaving developers, infrastructure administrators, scientists, and other end-users to juggle between smart objects, small devices, personal computers, clusters, private, public or hybrid Clouds as well as future generations of infrastructures such as Fog and Edge computing. In parallel, the complexity of distributed software deployed on such infrastructures continues to increase; most distributed software is modular or component-based, with possibly complex communications, interactions and synchronizations between each component.



Due to the complexity of distributed infrastructures, installation (deployment) and regular maintenance (e.g. updates, fault management, and other elements of normal software lifecycles) becomes a complex administrative task, composed of many steps and well-defined procedures. Several higher-level frameworks have been developed to help automate these *distributed software deployment schemes*, including Kubernetes and OpenStack (each specific to a particular type of virtualization), as well as a variety of lower level administrative tools such as Puppet, Chef and Ansible. Among the great number of deployment frameworks and tools currently available, however, none focuses on a critical point: deployment time; specifically, the efficiency and scalability of deployment schemes. Without a stronger focus on minimizing deployment time, future very large-scale infrastructures (e.g. Fog and Edge) and very large-scale applications (e.g. smart-everything apps) cannot be handled.

In the STACK research team at Inria (France), project sponsor Helene Coullon works on Madeus, a novel distributed software deployment model, as well as its Python implementation in a product called MAD (Madeus Application Deployer). At SICCS, project sponsor Frederic Louergue is interested in the formal analysis of component-based software, and in particular, of Madeus assemblies.

The overall approach that Madeus uses to support more efficient deployment is to expose and leverage opportunities for parallelism in the deployment process. Unfortunately, introducing parallelism also introduces more complexity for the end-user/administrator wanting to deploy a distributed software

system: the user has to think about and understand the dependencies between the different tasks of its deployment process, and has to explicitly code these dependencies in MAD. Obviously, this is much more challenging than writing a traditional sequential sequence of deployment actions. To help visualize, create, and maintain the complex parallelized deployment schemes that drive MAD (schemes are directed acyclic graphs in MAD), a Graphical User Interface (GUI) would be highly useful, supporting direct visualization of parallelized deployment tasks (e.g. in a time-based DAG). If successful, a graphical front-end for MAD would dramatically increase MAD's usability and make it much more accessible to the end-user community.

This project aims to prototype a clean, highly-usable GUI front-end to the MAD system outlined above. Specifically, the desired software product would likely consist of (at least) the following three parts:

1. A GUI to allow the graphical design of software deployment according to the Madeus model; the design of the tool should be based on the Model-View-Controller (MVC) architectural pattern,
2. A tree-like data structure to represent Madeus assemblies, and a *documented* API to manipulate it (Visitor pattern),
3. The framework should be extendable by plugins, and the provided plugins should be able to:
  - Read/Write Madeus assemblies from/to files,
  - Simulate the execution of Madeus assemblies,
  - Launch MAD deployments and follow in real time their execution
4. If time permits, other plugins could be provided including:
  - A plugin to generate images of Madeus assemblies (in pdf or/and png formats),
  - A plugin to generate LaTeX (using the tikz packages) representation of Madeus assemblies,
  - More advanced verification plugins such as interactions with the Coq proof assistant.

The GUI code would likely leverage the Python-based MAD implementation under a *GPL* license, but could also be implemented in a different language if the team's preliminary investigations reveal advantages in doing so. In any case, an early challenge in this project will be identifying appropriate existing libraries for doing 2D graph layout, to use as a basis for the DAG-based deployment scheme visualization.

MAD is available for free on an Inria GitLab repository. It is entirely documented under ReadTheDocs. Tutorials and examples could also be found in the repository. It is asked to the student team to use those same examples in the repository and documentation of the MAD-GUI.

### **Knowledge, skills, and expertise required for this project:**

The team will need to have or acquire:

- Skill in development of GUIs, plus coding of such GUIs in the chosen implementation language.
- Basic knowledge of Python, to allow integration of the envisioned front-end with MAD.
- Interest in and basic knowledge of distributed software and Cloud Computing,
- Motivation and drive to develop an innovative contribution to the distributed software community.

**Equipment Requirements:**

- Appropriate development workstations, ideally based on Linux distribution (incl. MacOS)
- All other software tools and modules are anticipated to be open-source, or will be provided by the client.

**Software and other Deliverables:**

Expected deliverables include:

- The software application as described above, deployed and tested successfully on a platform of the client's choosing. Must include a complete and clear User Manual for configuring and operating the software.
- A strong as-built report, implemented as HTML-based online documentation, detailing the design and implementation of the product in a complete, clear and professional manner. This document should provide a strong basis for future development of the product.
- Complete professionally-documented codebase, delivered both as a repository in GitHub, BitBucket, or some other version control repository; and as a physical archive on a USB drive.