Software Design and Architecture

[Introduction](#_ltfxz43lvy0c)

[Architectural Goals and Principles](#_lf33tjen5nk3)

[System Overview](#_l9ten4jnpwl5)

[Architectural Patterns](#_71rcj0j51ki6)

[Component Descriptions](#_lrkv8tz6v3dj)

[Data Management](#_1xlbae4qa4n)

[Interface Design](#_yde8z4fio3au)

[Considerations](#_6vq934e8i85y)

[Security](#_6rwiw2gdazc9)

[Performance](#_dzluejn5gvsr)

[Maintenance and Support](#_8lc935dnx8we)

[Deployment Strategy](#_70pa7uh8tm26)

[Testing Strategy](#_hyz1vkpij8lw)

[Glossary](#_fj5bnerrh6yx)

# Introduction

This document outlines the architecture for a GPU-accelerated simulation of a rate theory radiation damage model. The application must be scalable and usable on an ongoing basis by users without the intervention of a developer, and it must be extensible in the future.

# Architectural Goals and Principles

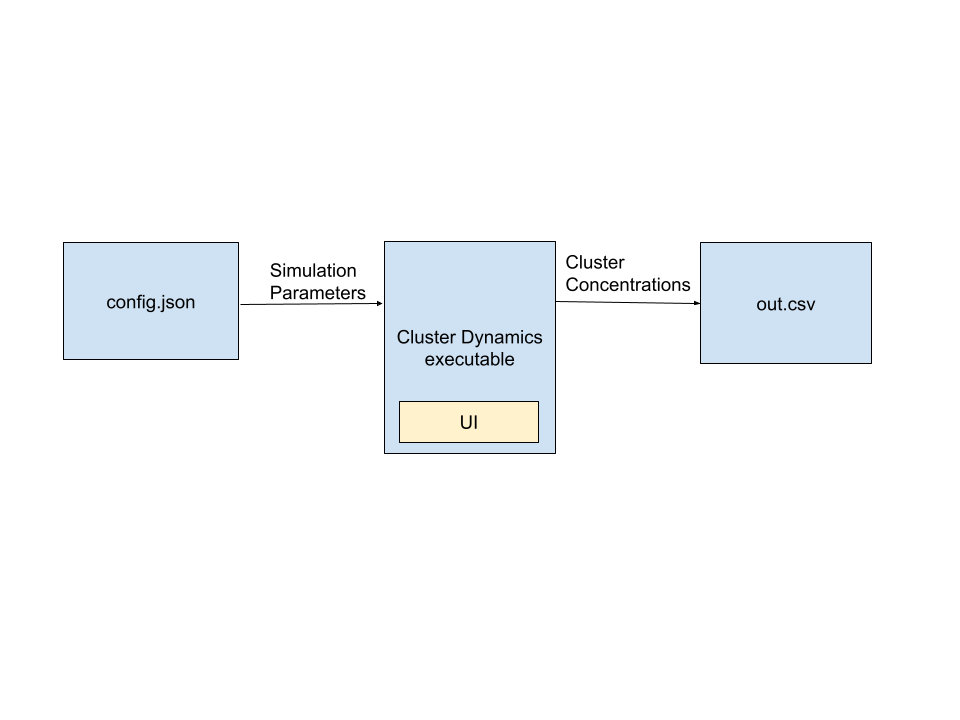
The most important points are scalability, extensibility, and simplicity of maintenance.

Scalability so that we may push the system to get results that have never been done before.

Extensibility so that the simulation may be improved in the future.

Simplicity of maintenance so that the project will remain reasonably usable and stable over time even if the end-user loses access to software developers.

# System Overview



The rough workflow for using the product is:

1. The user configures the simulation using the config.json.
2. The user opens the simulation from the command line.
3. The simulation reads parameters from a config.json file.
4. These parameters are more or less literally used as parameters to a “runModel” function which runs the simulation and outputs the results as a csv file.
5. A window opens which displays the results graphically, if enabled by the user.

The simulation program is implemented in C++ and uses the [ROOT framework](https://root.cern/about/) for visualization.

# Architectural Patterns

We are using the Model View Controller architecture. The model right now is just a config.json and the output csv format within the simulation program. The view will be a UI which is built into the cluster dynamics executable, and the controller is the simulation program itself.

We’ve arrived at this model because it provides a reasonable balance of abstraction and simplicity. Although we aren’t following it strictly yet in our prototypes, we will start splitting our application into these pieces more carefully in the next few weeks.

# Component Descriptions

Current prototypes use a configuration json alongside a command line interface. Input parameters are entered into the config.json, which is read by the simulation program, which runs a simulation based on the parameters and outputs the results as text in csv format.

# Data Management

Data simply consists of simulation input parameters and output concentrations. Simulation input parameters will depend on the material and nuclear reactor being used in a given scenario, and output concentrations will be a list of concentrations for each cluster size over time.

**Simulation Parameters**

Some of these parameters have complicated physical meanings which are not necessary to understanding the overall computation. They are all designed to be under control of the user.

| Attribute Name | Data type | Description |
| --- | --- | --- |
| i\_migration | double | The migration energy of single interstitial defects in eV. Used to compute interaction rates. |
| v\_migration | double | The migration energy of single vacancy defects in eV. Used to compute interaction rates. |
| i\_diffusion\_0 | double | A constant which is used for computing the diffusion coefficient for interstitials. In cm2/s. |
| v\_diffusion\_0 | double | A constant which is used for computing the diffusion coefficient for vacancies in cm2/s. |
| i\_formation | double | Formation energy for interstitials in eV. |
| v\_formation | double | Formation energy for vacancies in eV. |
| i\_binding | double | Binding energy for interstitials in eV. |
| v\_binding | double | Binding energy for vacancies in eV. |
| recombination\_radius | double | The distance at which a vacancy and an interstitial will recombine in cm. |
| i\_loop\_bias | double | Loop bias factor of interstitial clusters. Used to compute how readily interstitials will be emitted or absorbed by larger clusters. |
| v\_loop\_bias | double | Loop bias factor of vacancy clusters. used to compute how readily vacancies will be emitted or absorbed by larger clusters. |
| i\_dislocation\_bias | double | Bias factor used to compute how readily an interstitial cluster will be consumed by existing sinks. |
| v\_dislocation\_bias | double | Bias factor used to compute how readily an vacancy cluster will be consumed by existing sinks. |
| i\_dislocation\_bias\_param | double | A constant which is used in conjunction with i\_dislocation\_bias to compute how readily interstitials will be consumed by the dislocation network. |
| i\_dislocation\_bias\_param | double | A constant which is used in conjunction with v\_dislocation\_bias to compute how readily vacancies will be consumed by the dislocation network. |
| dislocation\_density\_0 | double | The initial density of the dislocation network in 1/cm2 |
| grain\_size | double | The size of the grains that make up the material in cm. |
| lattice\_param | double | A parameter which relates to the size of the unit cell of the material. |
| burgers\_vector | double | An arbitrary constant as far as the developer is concerned. |
| atomic\_volume | double | The average volume of an atom within the material in cm3. |
| flux | double | The intensity of neutrons passing through a material in cm2/s. |
| temperature | double | Temperature in Kelvin. |
| recombination | double | A constant which represents loss of clusters due to recombination. |
| f\_i | double[3] | Numbers which represent the share of the defect generation which goes to interstitial clusters of size 2, 3, and 4. |
| f\_v | double[3] | Numbers which represent the share of the defect generation which goes to vacancy clusters of size 2, 3, and 4. |
| dislocation\_density\_evolution | double | An empirically-determined constant which informs how rapidly the dislocation network will decay. |

**Output Concentration**

| Attribute Name | Data type | Description |
| --- | --- | --- |
| cluster\_concentrations | double[N+1][T] | A 2D N by T array of doubles where each row represents one of T simulation time steps. Each row contains a simulation timestamp followed by concentration of each of the N clusters at the timestep. |

The simulation outputs current time and defect concentrations for each time step throughout the duration of the simulation, written to a csv file.

# Interface Design

A large part of the interface will be command-line. You simply write your input parameters into a json configuration file and then open the program in that directory from the command line, which will save the results as a csv to a filename provided by the user. In addition, optionally, a window can open displaying the results of the simulation graphically.

# Considerations

## Security

The raw application will not communicate over the Internet, so security needs will be minimal. Some special considerations would need to be made if the application were deployed on the cloud, but this is not a consideration for version 0.

## Performance

Performance must be able to improve when the simulation is run using more GPUs.

The simulation must be optimized so that as many timesteps as possible can be run per second. Key factors may be the choice of numerical methods in the simulation, and the choice of GPU framework.

## Maintenance and Support

We anticipate that at a certain point, the project will be fully in the hands of people who are not software developers and who should not be expected to maintain the codebase themselves. The main difficulty this causes is in our ability to use cloud deployments since there won’t be anyone available to maintain the cloud environment if something goes wrong.

In the short term, because our project partner is our end user, we receive weekly feedback from the end user which directly translates into the new issues that are created to be worked on.

# Deployment Strategy

For now, the application is built as an executable that can simply be run end-to-end on the end user’s computer. This places large constraints on memory and computing resources.

# Testing Strategy

CI is used to make sure all of the simulations build successfully. Once we’ve finalized the cluster dynamics model we’re using, we can start thinking about how to test for correctness.

# Glossary

Cluster - A combined group of interstitials or vacancies [see the respective definitions].

Cluster Dynamics - A model of radiation damage which can be used to simulate radiation damage over long timescales. It works by analyzing the rate of change of cluster concentrations over time according to material and environmental parameters.

CSV - Acronym which means Comma Separated Values. It is a simple file type for storing data that can easily be imported into spreadsheet programs.

Dislocation Network - A larger structure of defects in a material which interacts with clusters.

GPU - Graphics Processing Unit - A computer processor chip with a large number of cores that can execute many instructions in parallel. The name comes from the fact that it is mainly used for graphics processing.

Interstitial - An atom in a material which has been knocked out of its normal spot in the crystal structure. Can be generated from radiation exposure.

Sink - A feature of the material which absorbs interstitials and vacancies.

Vacancy - A place in the crystal structure of a material which is missing an atom. Can be generated from radiation exposure.