Software Requirements Specification

for

Runaway Fluid

Documentation version 1.12
Software version 2.0.0

Prepared by Gergő Pokol, Mátyás Aradi, Soma Olasz

BME NTI, Hungary

29. September 2020.

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
GP	2015-03-26	First version: sections 1-2 completed	0.1
GP	2015-03-26	Section 3 completed	0.2
AM	2016-01-29	Update references	0.3
AM	2016-02-02	Update equations from references	0.4
AM	2016-02-08	Update description and figures	0.5
AM	2016-02-10	Correction of errata	0.6
AM	2016-02-11	Temperature dimensions are clarified	0.7
AM	2016-02-15	Mistyping fixed in h(alpha,Z)	0.8
AM	2016-03-17	CPO Input updated& CPO elements edited	0.9
AM	2016-03-18	“Distribution CPO” edited with reference	1.0
AM	2016-03-18	Time step CPO Input	1.1
GP	2016-06-16	Update to present status, scope corrected	1.2
AM	2016-07-01	CPO input and output updated	1.3
AM	2016-07-22	Update to present status, scope corrected, workflow figures added, IOs updated, Dreicer approximations added, runafluid_switch added	1.4
AM	2016-07-27	Erratum fixed in S_D,66	1.5
AM	2016-09-26	Thermal and runaway electron collision time fixed	1.6
AM	2016-10-10	Avalanche formulae added and new runafluid_switch	1.7
AM	2016-10-19	Simulation limitation for plasma edge	1.8
AM	2016-10-25	Toroidicity formulae corrected and toroidicity module added	1.9
OS	2019-10-18	Code parameters implemented, Coulomb logarithm bug fixed, electric field input fixed, Gtest fixed.	1.10
OS	2019-11-27	Fix of Dreicer field formula, hdf5 output parameters modified, figures updated.	1.11
OS	2020-09-29	Physics core moved to separate repository, first IMAS version description added.	1.12

1. Introduction

Purpose

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

This document specifies software requirements of the Runaway Fluid (runafluid) software module. The module is developed and deployed in the European Transport Solver (ETS) framework maintained by the Code Development for Integrated Modelling Project (EU-IM) of the EUROfusion consortium (*https://wpcd-workflows.github.io/*).

Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

Intended Audience and Reading Suggestions

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

This Software Requirements Specification (SRS) document is intended for EU-IM/ETS/H&CD workflow developers, and developers of the Runaway Fluid (Runafluid) module. The document describes the module Runafluid, laying out functional and non-functional requirements. Purpose, overview of the module, interfaces, and CPO/IDS objects used, constraints, assumptions and dependencies, functional requirements are contained in this documentation.

Product Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

The Runaway Fluid (runafluid) project supplies a simulator module assembled as a Kepler workflow actor, which is capable of simulating the generation of runaway electron current with some constrains using analytical formulas that exhibit a perturbative treatment of runaway electrons with respect to the bulk electron population. The output is a 1D radial runaway density and runaway current density. This functionality is able to extend the validity of ITM simulations to a regime with small non-thermal runaway current fraction.

References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

Runaway Fluid is maintained under a Github repository dedicated to runaway electron physics. The project's basic description, documentation and source code is stored in the Github project *https://github.com/osrep*.

Analytical formula used to determine the critical electric field is based on the work of A. Stahl et al [7]. The method of calculating Dreicer runaway generation growth rate stems from the article of J. W. Connor et al [6]. The runaway avalanche growth rate is based on the form in the article by E. Nilsson [12].

The current status of runaway electron modelling in ETS has been published in Nuclear Fusion in 2019 [18].

2. Overall Description

Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

Runaway Fluid module implements an EU-IM Kepler actor.

Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

High level summary of functions:

1. Provide an estimate of the non-thermal current carried by runaway electrons as function of minor radius.
2. Indicate in a message if runaway electrons are present.
3. Indicate if the runaway density exceeds a set fraction of electron density (default is 1%), that might affect the stability of the workflow.
4. Give warning if the plasma regime is not suitable for this type of modeling.

User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

ITM workflow authors, physics module authors will use this product.

Operating Environment

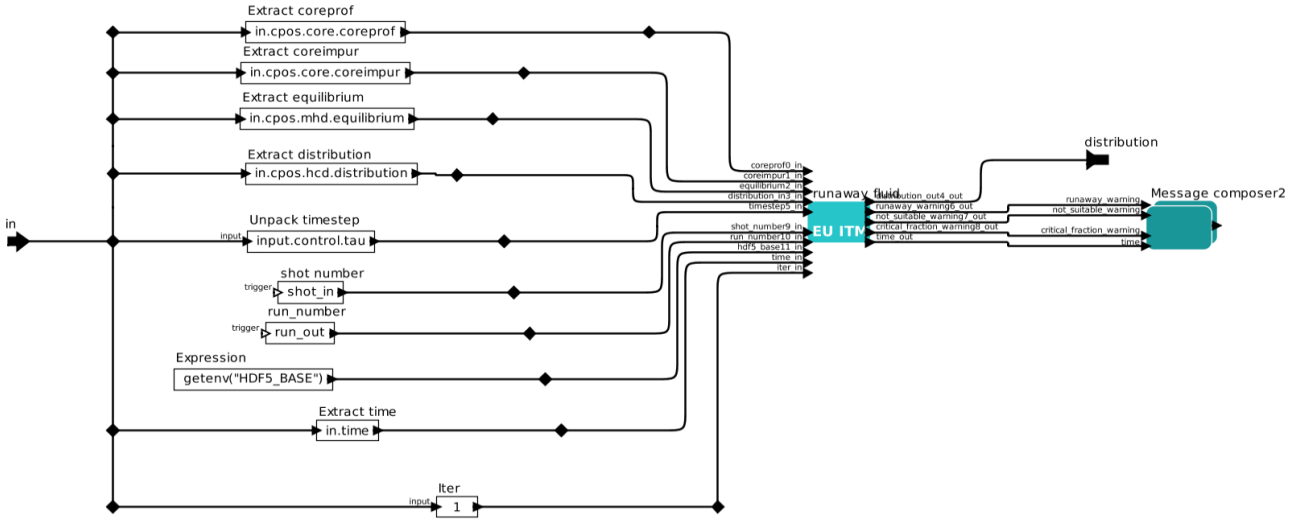
<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

ITM environment

Runaway Fluid module will operate as an embedded Kepler actor in ITM/ETS workflows in the Heating and Current Drive (H&CD) workflow. A dedicated ITM development and execution environment is set up by the H&CD staff.

In H&CD, Runaway Fluid module is embedded into a three layers deep subworkflow structure represented by composite actors. The subworkflows introduced in the composite actors have a specific function and structure. These subworkflows are discussed in the following subsections.

Demultiplexer workflow – `Runaway_Fluid` composite actor



Demultiplexer workflow is contained in composite actor named "Runaway_electrons_one_time_slice". Demultiplexer workflow reads the input CPO-s and time from the input bundle and feeds those values into the "runaway_fluid" actor. The input bundle of demultiplexer workflow (and subsequently input bundle of enabler workflow described in 0) shall contain the following members:

Member name	*Type*
cpos.core.coreprof	coreprof CPO
cpos.core.coreimpur	coreimpur CPO
cpos.mhd.equilibrium	equilibrium CPO
cpos.hcd.distribution	distribution CPO
shot_number	int
run_number	int
hdf5_base	char
dt_in	double
time	double

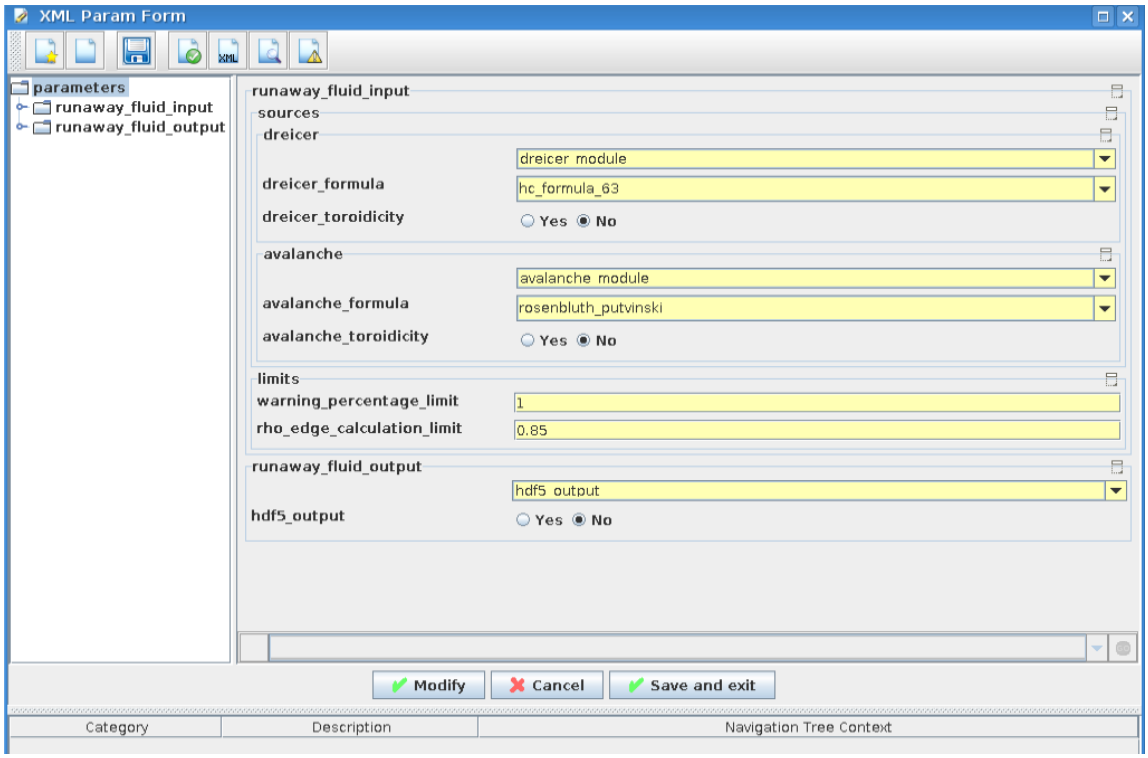
Standardized EU-ITM Plasma Bundle:
https://portal.eufus.eu/documentation/ITM/html/itm_conventions.html#itm_conventions_28

Code Parameters

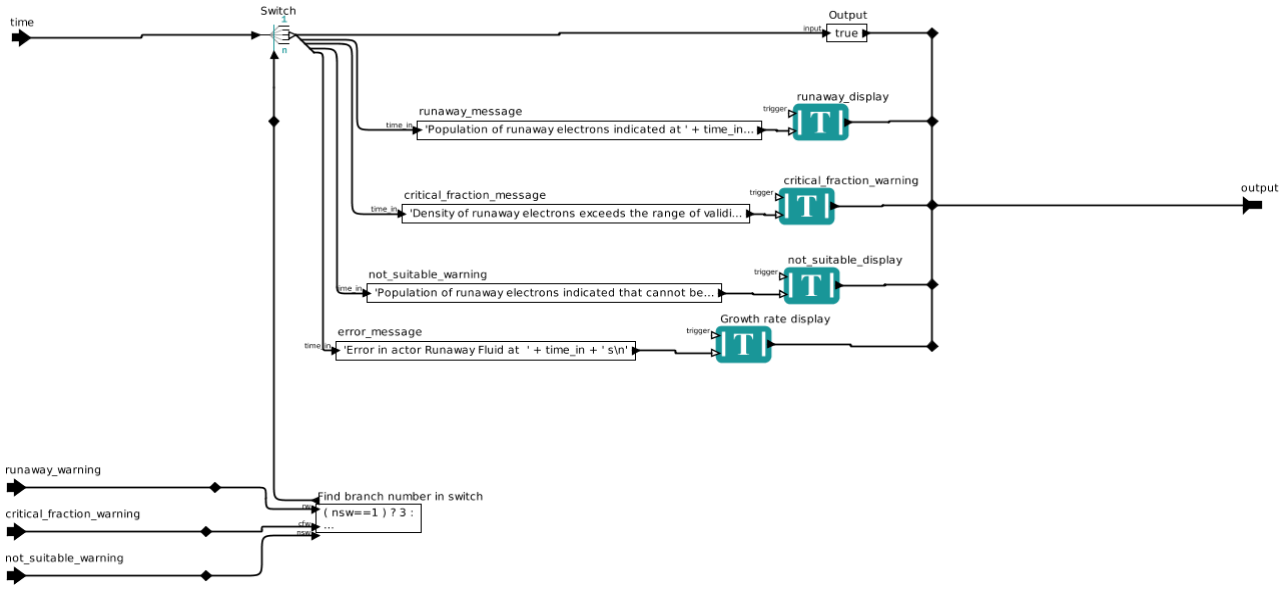
The composite actor has an internal variable named “critical_fraction” with a default value 1. It means the critical fraction is 1% of total electron density. This variable can be set in the code parameters accessible from the configuration of the actor.

The code settings can be modified from the same code parameters page, deciding which features to be used during the simulation. The calculation regime can be set with the rho_edge_calculation_limit option.

The option to write additional data to hdf5 files can be chosen: If turned on, the file is saved to the location given by the $HDF5_BASE environmental variable or give a warning and write to the user’s home folder.



Message composer workflow – `Message Composer` composite actor



Message composer workflow is contained in composite actor named “Message Composer”. This workflow prints a message into a multi tab display defined by the result of the runaway electron modeling. The following messages may be printed:

Case	*Message*
Normal operation – no runaway electrons	-
Normal operation – runaway electrons indicated	Population of runaway electrons indicated: <*time*>
Indicate if the runaway density exceeds a preset fraction of electron density (currently 1%).	Density of runaway electrons exceeds the range of validity: <*time*>
Runaway electrons indicated, but the plasma regime is not suitable for this type of modeling.	Population of runaway electrons indicated that cannot be modeled by Runaway Fluid: <*time*>
Error occurred during calculation	Error in actor Runaway Fluid at time: <*time*>

Message “Runaway electrons indicated, but the plasma regime is not suitable for this type of modeling.” means that the runaway current is higher than the total electron current.

IMAS environment

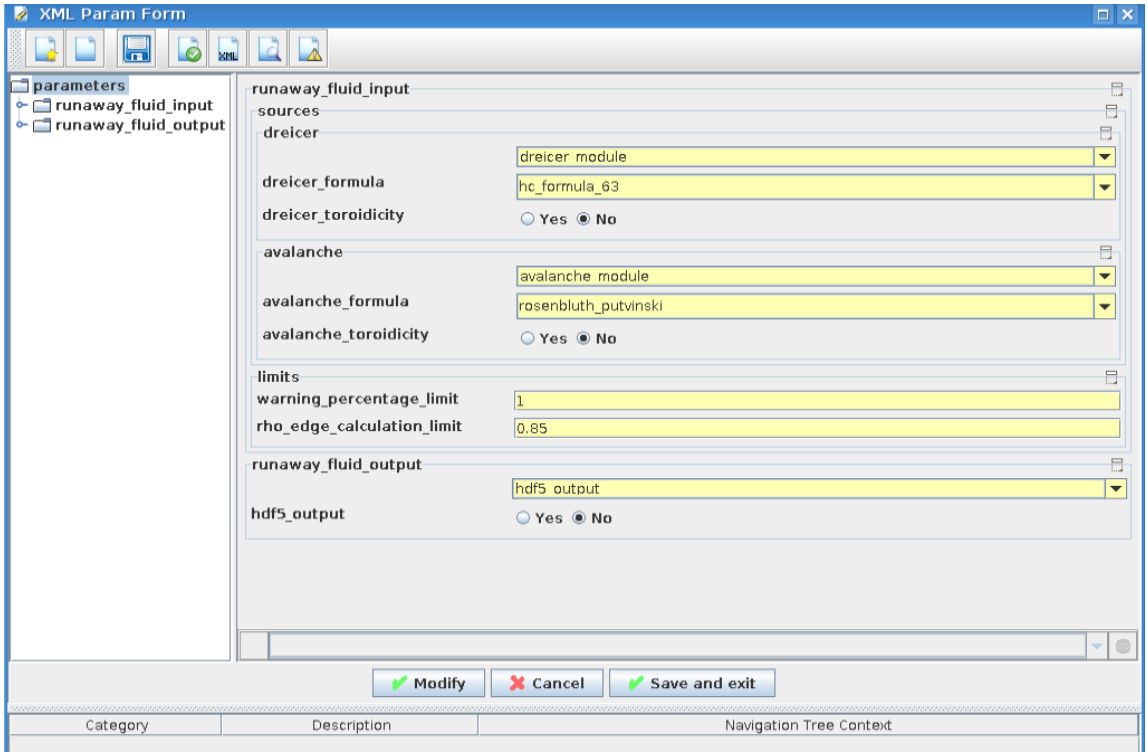
Runaway Fluid module will operate as an embedded Kepler actor in IMAS/ETS workflows, but it is not yet implemented in the IMAS version of the ETS workflow.

Code Parameters

The actor has an internal variable named “critical_fraction” with a default value 1. It means the critical fraction is 1% of total electron density. This variable can be set in the code parameters accessible from the configuration of the actor.

The code settings can be modified from the same code parameters page, deciding which features to be used during the simulation. The calculation regime can be set with the rho_edge_calculation_limit option.

The option to write additional data to hdf5 files can be chosen: If turned on, the file is saved to the location given by the $HDF5_BASE environmental variable or give a warning and write to the user's home folder.



Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

In the framework of the Code Development for Integrated Modelling Project (ITM), the Kepler workflow engine provides the capability of workflow orchestration in simulation. Kepler is a free and open source, scientific workflow application. The Runaway Fluid module implements a Kepler actor.

In the framework of the European Code Development for Integrated Modelling Project (EU-IM), the Universal Access Layer (UAL) provides the capability of storing/retrieving data involved in simulation. The granularity in data access is given by the definition of a set of Consistent Physical Objects (CPOs). The Runaway Fluid module uses the UAL layer for input/output.

Runaway Fluid actor is written in C++ language.

Runaway Fluid actor is implemented using ITM tool “fc2k”.

User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

Basic description and user is provided at *https://github.com/osrep*.

Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

The program is developed using ITM data structure version 4.10b.

3. External Interface Requirements

User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and

describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

CPO Input

Runaway Fluid actor reads input data from ITM CPO-s via UAL. The CPO structures "coreprof", "coreimpur", and "equilibrium" contain the data needed. This input data is typically generated by the actor named "ualinit". The following parameters are used in calculations:

Parameter	*CPO element*	*Unit*
Electron density profile (1d)	coreprof[time]/ne/value[r]	$1/m^3$
Electron temperature profile (1d)	coreprof[time]/te/value[r]	eV
Effective charge profile (1d)	coreprof[time]/profiles1d/zeff[r]	$1/m^3$
Electric field profile (1d) *Runaway density before (1d)*	coreprof[time]/profiles1d/eparallel[r] equilibrium[time]/profiles_1d/b_av[r] distribution[time]/distri_vec[@]/profiles_1d/state/dens[r]	V/m T $1/m^3$
time step	dt_in	s

Here @ is the index where `distsource_identifier` of `distri_vec` refers to the "runaway" distribution (flag 7) as listed in `distsource_types`, as described at Documentation page linked from the Conventions page.

Output

The `runaway_fluid` core actor emits two integer outputs (of possible values 0 and 1), which

1. Indicate in a message if runaway electrons are present.
2. Give warning if the plasma regime is not suitable for this type of modeling.

This output is used in the composite actor to write the output messages.

Properties of the simulated runaway electron distribution are output to the following CPO fields:

Parameter	*CPO element*	*Unit*
Runaway density after (1d)	distribution[time]/distri_vec[0]/profiles_1d/state/dens[r]	$1/m^3$
Runaway current density after (1d)	distribution[time]/distri_vec[0]/profiles_1d/state/current[r]	A/m^3
Geometry information	distribution[time]/distri_vec[0]/profiles_1d/geometry	copy from coreprof

CPO initialization

The output CPO should be initialized in the first call by filling:

Parameter	*CPO element*	*Value*
Initial runaway density (1d)	distribution[time]/distri_vec[0]/profiles_1d/state/dens[r]	0
Initial runaway current density (1d)	distribution[time]/distri_vec[0]/profiles_1d/state/current[r]	0
Geometry information	distribution[time]/distri_vec[0]/profiles_1d/geometry	copy from coreprof
Distribution identifier id	distribution[time]/distri_vec[0]/source_id[0]/type/id	"runaway"
Distribution identifier flag	distribution[time]/distri_vec[0]/source_id[0]/type/flag	7
Distribution identifier description	distribution[time]/distri_vec[0]/source_id[0]/type/description	"Source from runaway processes"
Object instance	distribution[time]/distri_vec[0]/source_id[0]/index	**???**
	distribution[time]/distri_vec[0]/source_id[0]/name	**???**
Species identifier id	distribution[time]/distri_vec[0]/species/type/id	"electron"
Species identifier flag	distribution[time]/distri_vec[0]/species/type/flag	1
Species identifier description	distribution[time]/distri_vec[0]/species/type/description	"Electron"
Gyro-center interpretation	distribution[time]/distri_vec[0]/gyro_type	1

IDS Input

Runaway Fluid actor reads input data from IMAS IDS-s via UAL. The IDS structures “core_profiles” and “equilibrium” contain the data needed. This input data is typically generated by the actor named “ualinit”. The following parameters are used in calculations:

Parameter	*CPO element*	*Unit*
Electron density profile (1d)	core_profiles/profiles_1d[time]/electrons/density[r]	$1/m^3$
Electron temperature profile (1d)	core_profiles/profiles_1d[time]/electrons/temperature[r]	eV
Effective charge profile (1d)	core_profiles/profiles_1d[time]/zeff[r]	$1/m^3$
Electric field profile (1d)	core_profiles/profiles_1d[time]/e_field.parallel[r]	V/m
	equilibrium/time_slice[time]/profiles_1d/b_field_average	T
Runaway density before (1d)	distributions/distribution[@]/profiles_1d[time]/density[r]	$1/m^3$
time step	dt_in	s

Here @ is the index where `distsource identifier` of `distri_vec` refers to the “runaway” distribution (flag 7) as listed in `distsource_types`, as described at Documentation page linked from the Conventions page.

Output

The `runaway_fluid` core actor emits two integer outputs (of possible values 0 and 1), which

1. Indicate in a message if runaway electrons are present.
2. Give warning if the plasma regime is not suitable for this type of modeling.

This output is used in the composite actor to write the output messages.

Properties of the simulated runaway electron distribution are output to the following IDS fields:

Parameter	*CPO element*	*Unit*
Runaway density after (1d)	distribution_out/distribution[0]/profiles_1d[time]/density[r]	$1/m^3$
Runaway current density after (1d)	distribution_out/distribution[0]/profiles_1d[time]/current_tor[r]	A/m^3
Geometry information	distribution_out/distribution[0]/profiles_1d[time]/grid	copy from core_profiles

IDS initialization

The output IDS should be initialized in the first call by filling:

Parameter	*CPO element*	*Value*
Initial runaway density (1d)	distributions/distribution[0]/profiles_1d[0]/density[r]	0
Initial runaway current density (1d)	distributions/distribution[0]/profiles_1d[0]/current_tor[r]	0
Geometry information	distributions/distribution[0]/profiles_1d[0]/grid	copy from coreprof
Distribution identifier id	distributions/distribution[0]/process[0]/type/name	"runaway"
Distribution identifier flag	distributions/distribution[0]/process[0]/type/index	7
Distribution identifier description	distributions/distribution[0]/process[0]/type/description	"Source from runaway processes"
Species identifier id	distributions/distribution[0]/species/type/name	"electron"
Species identifier flag	distributions/distribution[0]/species/type/index	1
Species identifier description	distributions/distribution[0]/species/type/description	"Electron"
Gyro-center interpretation	distributions/distribution[0]/gyro_type	1

Physics core

The physics calculations are included in a separate repository named Runaway Physics. This is included in Runaway Indicator a git submodule. The Runaway Physics is maintained at https://github.com/osrep/Runaphys.

Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by

use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

Plasma edge cutdown threshold

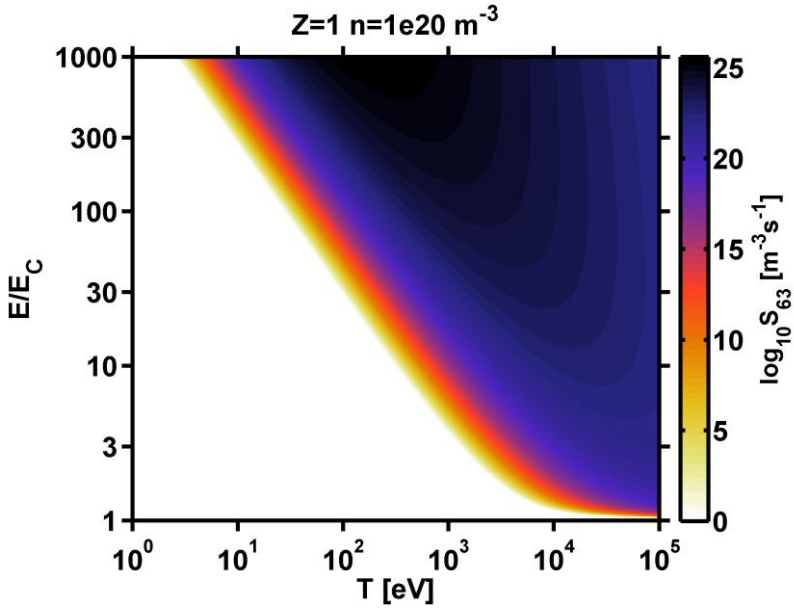
Because of in edge region of the plasma we do not expect runaway electrons a cutdown is needed to be implemented because of scattered electric field and low electron density to avoid false expectation of runaways.

A cutdown threshold is set at $\rho = 0.85$. For higher normalized minor radius runaway density is set to zero. This variable can be modified in the code parameters.

Dreicer generation rate

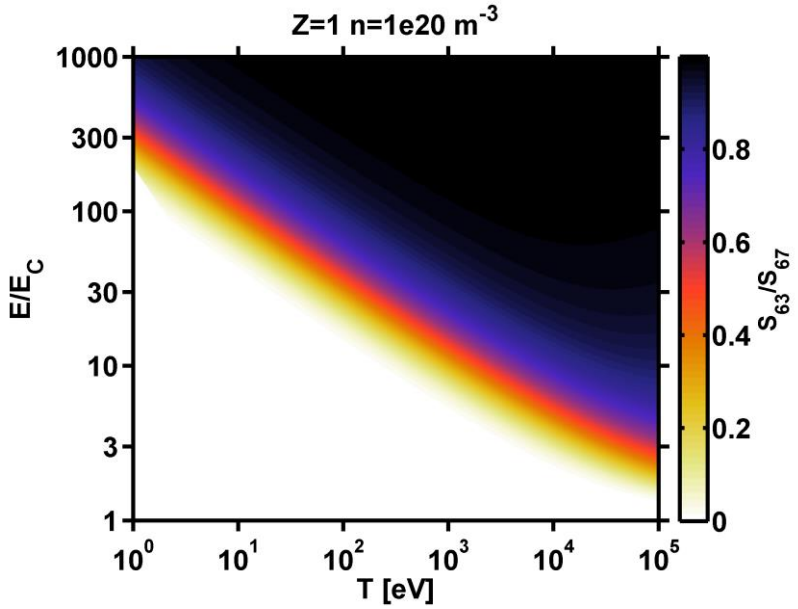
Dreicer generation rate is calculated using the formula [6] (63):

$$S_{\mathrm{D}} = \frac{\Delta n_R}{\Delta t} = \frac{C_{\mathrm{R}} n_{\mathrm{e}}}{\tau} \cdot \left(\frac{E}{E_{\mathrm{D}}}\right)^{-h(\alpha,Z)} \cdot \exp\left(-\frac{\lambda(\alpha)}{4} \cdot \frac{E_{\mathrm{D}}}{E} - \sqrt{\frac{2E_{\mathrm{D}}}{E}} \cdot \gamma(\alpha, Z)\right)$$



Dreicer generation rate approximation in general case [6] (67):

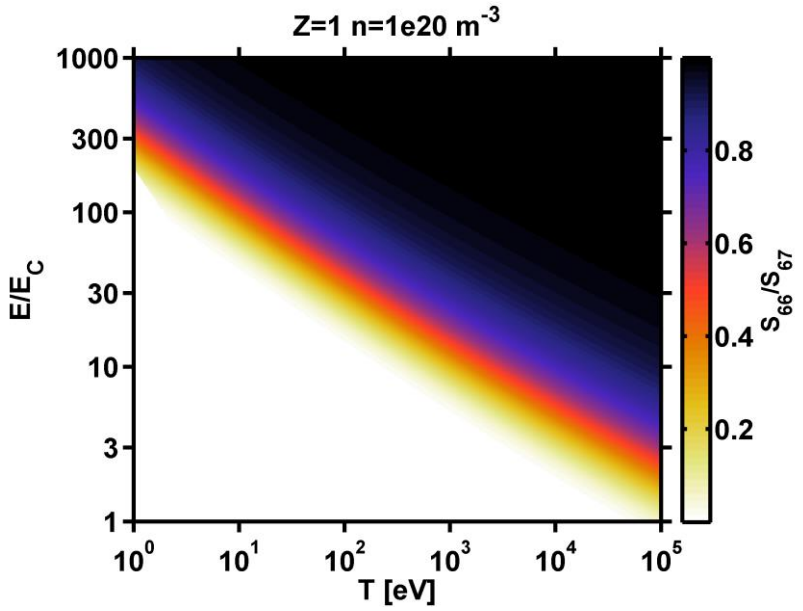
$$\boldsymbol{S_{\mathrm{D},67} = \frac{C n_{\mathrm{e}}}{\tau} \cdot \left(\frac{E}{E_{\mathrm{D}}}\right)^{-3/16(Z+1)} \cdot \exp\left(-\frac{1}{4} \cdot \frac{E_{\mathrm{D}}}{E} - \sqrt{(1+Z) \cdot \frac{E_{\mathrm{D}}}{E}}\right)}$$


Z=1 n=1e20 m-3
1000
300
100
30
10
3
1
E/EC
10^0
10^1
10^2
10^3
10^4
10^5
T [eV]
0.8
0.6
0.4
0.2
0
S63/S67

Dreicer generation rate approximation with high temperature correction [6] (66):

$$S_{\mathrm{D},66} = S_{\mathrm{D},67} \cdot \exp\left(-\frac{T_\mathrm{e}}{mc^2} \cdot \left(\frac{1}{8}\left(\frac{E_\mathrm{D}}{E}\right)^2 + \frac{2}{3}\left(\frac{E_\mathrm{D}}{E}\right)^{3/2}\sqrt{1+Z}\right)\right)$$

T_e electron temperature in joules ($k_\mathrm{B}T$)



Description and Priority

Implemented

The module outputs a double array what contains Dreicer generation rate.

Stimulus/Response Sequences
Dreicer generation rate is calculated in every value in row of E and n

Functional Requirements

REQ-1: Dreicer field [16] (63):

$$E_\mathrm{D} = \frac{n_e e^3 \ln\Lambda}{4\pi\epsilon_0^2 T_\mathrm{e}}$$

$$E_\mathrm{D} = E_c \frac{m_e c^2}{T_\mathrm{e}}$$

T_e electron temperature in joules ($k_\mathrm{B}T$)

REQ-2: thermal electron collision time

$$\tau_{\mathrm{th}} = 4\pi\varepsilon_0^2 \cdot \frac{m_{\mathrm{e}}^2 {v_{\mathrm{th}}}^3}{e^4} \cdot \frac{1}{n_{\mathrm{e}} \ln\Lambda}$$

where

$$v_{\mathrm{th}} = \sqrt{2\frac{T_{\mathrm{e}}}{m_{\mathrm{e}}}}$$

REQ-3: Coulomb logarithm [17]

$$\ln\Lambda = 14.9 - 0.5\log(n_{\mathrm{e}} \cdot 10^{-20}) + \log(t_{\mathrm{e}} \cdot 10^{-3})$$

t_{e}electron temperature in electronvolts

REQ-4: h factor [6] (62):

$$h(\alpha, Z) = \frac{1}{16(\alpha - 1)} \cdot \left(\alpha(Z+1) - Z + 7 + 2 \cdot \sqrt{\frac{\alpha}{\alpha - 1}} \cdot (1+Z)(\alpha - 2)\right)$$

REQ-5: lambda factor [6] (64):

$$\lambda(\alpha) = 8\alpha\left(\alpha - \frac{1}{2} - \sqrt{\alpha(\alpha - 1)}\right)$$

REQ-6: multiplication factor [6] (64):

$$\gamma(\alpha, Z) = \sqrt{\frac{(1+Z)\alpha^2}{8(\alpha - 1)}} \cdot \left(\frac{\pi}{2} - \sin^{-1}\left(1 - \frac{2}{\alpha}\right)\right)$$

REQ-7: relative electric field

$$\alpha = \frac{E}{E_{\mathrm{D}}} \cdot \frac{m_{\mathrm{e}} c^2}{T_{\mathrm{e}}}$$

T_{e} electron temperature in joules ($k_{\mathrm{B}}T$)

REQ-8: When some input data is missing in CPO input, return error code.
REQ-9: Error code is an integer value of -999999999.
REQ-10: When some input data is missing in CPO input, prints error message.

Additional Requirements

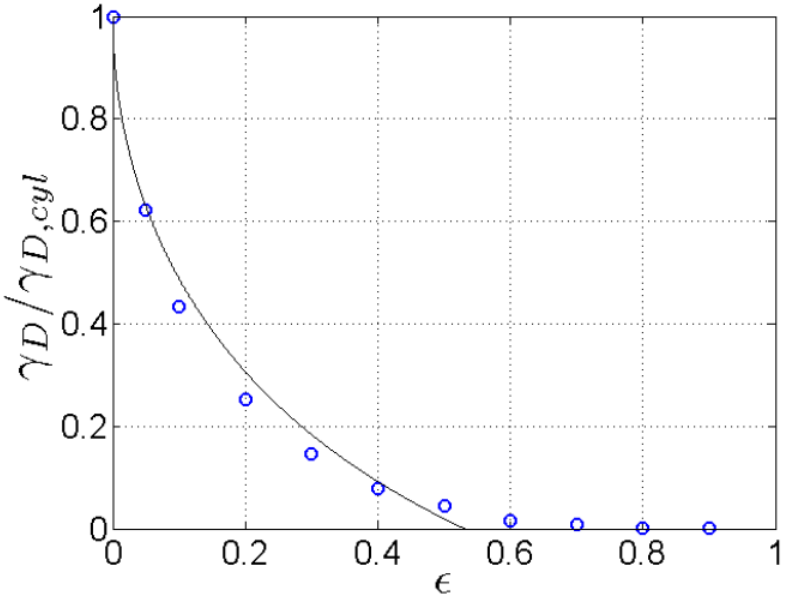
Toroidicity
Implemented

REQ-11:toroidicity [11, 12]

$$\frac{S_{\mathrm{D}}}{S_{\mathrm{D,\,cyl}}} = 1 - 1.2\sqrt{\frac{2\epsilon}{1+\epsilon}}$$

REQ-12: inverse aspect ratio

$$\epsilon = \frac{a}{R}$$



Reduction of Dreicer generation due to toroidicity as calculated by LUKE, Figure 7 of [11] and Figure 2 of [12]

Avalanche generation rate

Description and Priority

Implemented

The module outputs a double array what contains avalanche generation rate.

Avalanche generation rate [8,13]:

$$\Gamma \equiv \frac{1}{n_\mathrm{R}}\frac{\partial n_\mathrm{R}}{\partial t} = \frac{1}{2\tau_\mathrm{R}\ln\Lambda}\left(\frac{E}{E_\mathrm{c}} - 1\right)$$

Large electric field ($E/E_\mathrm{a} \gg 1$) [9]:

$$\frac{\partial n_\mathrm{R}}{\partial t} \approx \frac{n_\mathrm{R}}{2\tau_\mathrm{R}\ln\Lambda}\left(\frac{E}{E_\mathrm{c}} - 1\right)$$

it is implemented as

$$\Delta n_\mathrm{R} \approx \frac{n_\mathrm{R}}{2\tau_\mathrm{R}\ln\Lambda}\left(\frac{E}{E_\mathrm{c}} - 1\right)\Delta t$$

Stimulus/Response Sequences

Avalanche generation rate is calculated in every value in row of *E* and *n*

Functional Requirements

REQ-1: Coulomb logarithm [17]

$$\ln\Lambda = 14.9 - 0.5\log(n_\mathrm{e}\cdot 10^{-20}) + \log(t_\mathrm{e}\cdot 10^{-3})$$

t_e electron temperature in electronvolts

REQ-2: critical electric field

$$E_\mathrm{c} = \frac{n_\mathrm{e} e^3 \ln\Lambda}{4\pi\varepsilon_0^2 m_\mathrm{e} c^2}$$

REQ-3: runaway electron collision time

$$\tau_\mathrm{R} = 4\pi\varepsilon_0^2 \cdot \frac{m_\mathrm{e}^2 c^3}{e^4} \cdot \frac{1}{n_\mathrm{e}\ln\Lambda}$$

REQ-4: avalanche generation rate (large field, $E/E_\mathrm{c} \gg 1$)

$$\Delta n_\mathrm{R} \approx \frac{n_\mathrm{R}}{2\tau_\mathrm{R}\ln\Lambda}\left(\frac{E}{E_\mathrm{c}} - 1\right)\Delta t$$

REQ-5: avalanche generation rate (small field, $E/E_\mathrm{c} \ll 1$)

$$\Delta n_\mathrm{R} = 0$$

REQ-6: when some input data is missing in CPO input, return error code.
REQ-7: error code is an integer value of -999999999.
REQ-8: when some input data is missing in CPO input, prints error message.

Additional Requirements

onset threshold
Implemented

REQ-9: Avalanche sustainment threshold electric field

$$E_\mathrm{a} \approx E_0 \approx 1 + \frac{\frac{Z+1}{\sqrt{\bar{\tau}_\mathrm{rad}}}}{\sqrt[6]{\frac{1}{8} + \frac{(Z+1)^2}{\bar{\tau}_\mathrm{rad}}}}$$

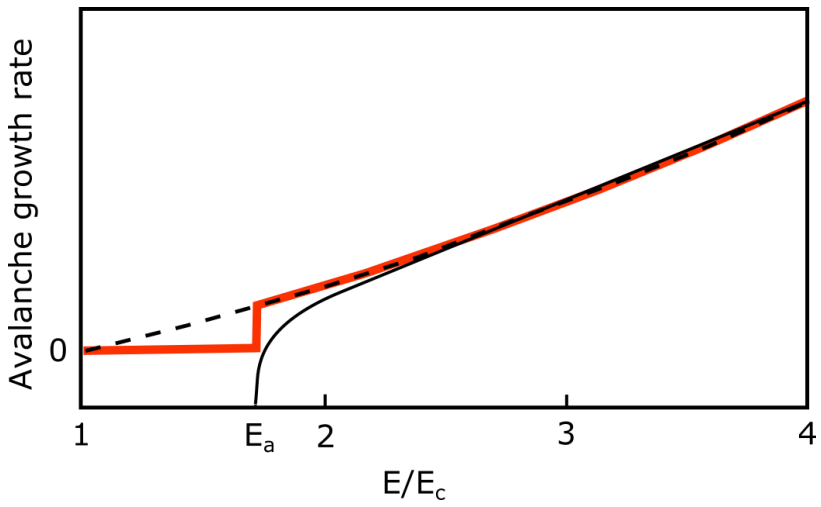
REQ-10: normalized time of synchrotron losses

$$\bar{\tau}_\mathrm{rad} \equiv \frac{\tau_\mathrm{rad}}{\tau_\mathrm{R}} = \frac{1}{\tau_\mathrm{R}} \cdot \frac{6\pi\varepsilon_0 m_0^3 c^3}{e^4 B^2}$$

where τ_R in REQ-3

REQ-11:
Small electric field ($E/E_a \ll 1$) [8, 10]:

$$\frac{\partial n_R}{\partial t} = 0$$



E_a: avalanche onset threshold electric field

Additional Requirements (implemented)
Toroidicity
Implemented

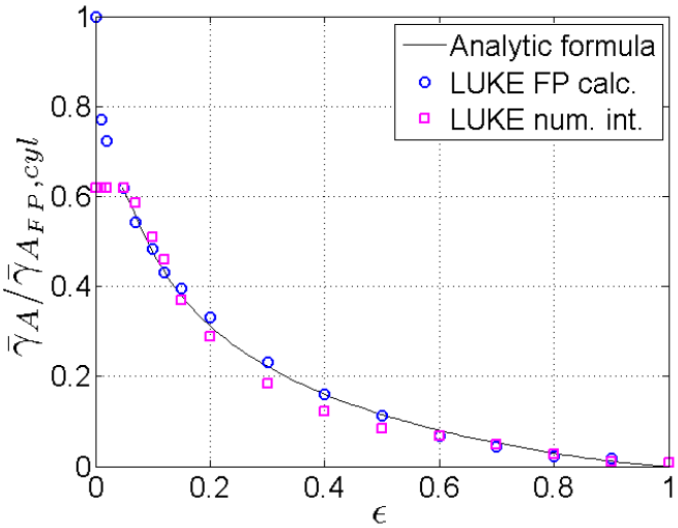
REQ-12:toroidicity [12 A.4]

$$\frac{S_A}{S_{A,\,cyl}} = \frac{(1-\epsilon)^2}{\pi\sqrt{\epsilon E/E_c}}$$

where E_c in REQ-2

REQ-13: inverse aspect ratio

$$\epsilon = \frac{a}{R}$$



Reduction of avalanche generation due to toroidicity as calculated by LUKE, Figure 8 of [11] and Figure 2 of [12]

Additional Requirements (not yet implemented)

near-threshold theory
Priority level: 2

REQ-14:

$$\Gamma = \frac{1}{4\ln\Lambda\,\gamma_0\sqrt{\gamma_0^2-1}} \cdot\left(-\frac{(2\gamma_0-1)}{\gamma_0-1}\,2\ln\left(1+\frac{\gamma_0+1-2\gamma_{\min}}{\gamma_{\min}-1}\right) + (\gamma_0+1-2\gamma_{\min})\left(1+\frac{2\gamma_0^2}{(\gamma_{\min}-1)(\gamma_0-\gamma_{\min})}\right)\right)$$

Aleynikov et al., PRL 114,155001(2015) Eq(11)

REQ-15: flow velocity

$$U(p) = E\cos\theta_{\text{av}} - 1 - \frac{1}{p^2} - \frac{Z+1}{E\bar{\tau}_{\text{rad}}}\cdot\frac{p^2+1}{p}\cdot\cos\theta_{\text{av}}$$

REQ-16:

$$\cos\theta_{\text{av}} = \left(\frac{1}{\tanh(A(p))} - \frac{1}{A(p)}\right)$$

REQ-17:

$$A(p) \equiv \frac{2E}{Z+1} \frac{p^2}{\sqrt{p^2+1}}$$

REQ-18:

$$U(p) = -B(p)E - 1 - \frac{1}{p^2} + \frac{Z+1}{E\bar{\tau}_{\mathrm{rad}}} \cdot \frac{p^2+1}{p} \cdot B(p)$$

p_0 and $p_{\max}$:

$$0 = -B(p)E - \frac{p^2+1}{p^2} + \frac{Z+1}{E\bar{\tau}_{\mathrm{rad}}} \cdot \frac{p^2+1}{p} \cdot B(p)$$

Loss mechanism (not yet implemented)

Description and Priority

Priority level: 3

Diffusive radial transport [11]

diffusion coefficient

$$\chi_{\mathrm{r}} = \frac{\langle (\Delta r)^2 \rangle}{2\tau} = D_{\mathrm{st}} v$$

Non-diffusive transport [12]

Rechester-Rosenbluth diffusion coefficient

$$D_{\mathrm{RR}} = \pi q v_{\parallel} R \left(\langle \frac{\delta B}{B} \rangle \right)^2$$

where
q safety factor
$v_{\parallel} \cong c$parallel velocity
R major radius

$$\langle \frac{\delta B}{B} \rangle = \sqrt{\left(\frac{\delta B}{B} \right)_{\psi}^2}$$

flux surface averaged normalized magnetic perturbation amplitude as a function of radius

Stimulus/Response Sequences

one dimensional diffusion population by major radius

Functional Requirements

REQ-1: Flux surface averaged normalized magnetic perturbation amplitude

$$\langle \frac{\delta B}{B} \rangle = \sqrt{\left(\frac{\delta B}{B} \right)_{\psi}^2}$$

REQ-2: Numerical diffusion solver

5. Other Nonfunctional Requirements

Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>

7. References

[1] G.I. Pokol, R. Lohner, G. Papp, B. Erdos and EU-IM Team. Application limits of runaway electron modeling based on analytical formulas of generation and loss rates. *42nd EPS Conference on Plasma physics*, P5.169 (2015)

[2] G. Papp, T. Fülöp, T. Fehér, P.C. de Vries, V. Riccardo, C. Reux, M. Lehnen, V. Kiptily, V.V. Plyusnin, B. Alper, and JET EFDA contributors. The effect of ITER-like wall on runaway electron generation in JET. *Nuclear Fusion* **53**(12), 123017 (2014)

[3] G.L. Falchetto, D. Coster, R. Coelho, B.D. Scott, L. Figini, D. Kalupin, E. Nardon, S. Nowak, L.L. Alves, J.F. Artaud, et. al. The European Integrated Tokamak Modelling (ITM) effort: achievements and first physics results. *Nuclear Fusion***54**(4), 043018 (2014)

[4] Y. Peysson and J. Decker. Numerical Simulations of the Radio-Frequency–driven Toroidal Current In Tokamaks. *Fusion Science and Technology* **65**, 22 (2014)

[5] D. Kalupin, I. Ivanova-Stanik, I. Voitsekhovitch, J. Ferreira, D. Coster, L.L. Alves, Th. Aniel, J.F Artaud, V. Basiuk, João P.S. Bizarro et al. Numerical analysis of JET discharges with the European Transport Simulator. *Nuclear Fusion***53**(12), 123007 (2013)

[6] J.W. Connor and R.J. Hastie. Relativistic limitations on runaway electrons. *Nuclear Fusion***15**, 415 (1975)

[7] A. Stahl, E. Hirvijoki, J. Decker, O. Embréus, and T. Fülöp. Effective Critical Electric Field for Runaway-Electron Generation. *Physical Review Letters***114**(11), 115002 (2015)

[8] P. Aleynikov and B.N. Breizman. Theory of Two Threshold Fields for Relativistic Runaway Electrons. *Physical Review Letters***114**(15), 155001 (2015)

[9] R.M. Kulsrud, Y.C. Sun, N.K. Winsor and H.A. Fallon. Runaway electrons in a plasma. *Physical Review Letters***31**(11), 690 (1973).

[10] P. Aleynikov, K. Aleynikova, B.N. Breizman, G. Huijsmans, S. Konovalov, S.V. Putvinski, V. Zhogolev. Kinetic Modelling of Runaway Electrons and their Mitigation in ITER. *Proceedings of IAEA FEC 2014*, TH/P3-38 (2014)

[11] E. Nilsson, J. Decker, Y. Peysson, E. Hollmann and F. Saint-Laurent. Kinetic modelling of runaway electrons in tokamak plasmas. *41st EPS Conference on Plasma physics*, O2.303 (2014)

[12] E. Nilsson, J. Decker, N.J.Fisch, Y. Peysson. Trapped-Electron Runaway Effect.*Journal of Plasma Physics* **81**(4), 475810403 (2015)

[13] M.N. Rosenbluth and S.V. Putvinski. Theory for avalanche of runaway electrons in tokamaks.*Nuclear Fusion***37**(10), 1355 (1997)

[14] A.B. Rechester and M.N. Rosenbluth. Electron heat transport in a Tokamak with destroyed magnetic surfaces.*Physical Review Letters***40**(1), 38 (1978)

[15] K.Särkimäki, E.Hirvijoki, J. Decker, J.Varje, T.Kurki-Suonio. An advection-diffusion model for cross-field runaway electron transport in perturbed magnetic fields. *arXiv*, 1606.04409 [physics.plasm-ph] (2016)

[16] P. Helander, F. Andersson, L.-G. Eriksson, T. Fülöp, H. Smith, D. Anderson, M. Lisak. Runaway electron generation in tokamak disruptions. *Proceedings of the 20th IAEA Fusion Energy Conference* .TH/P4-39 (2004)

[17] J. D. Huba. *NRL: Plasma formulary*. Naval Research Laboratory Washington DCBeam Physics Branch (2004)

[18] G.I. Pokol, et. al, Runaway electron modelling in the self-consistent core European Transport Simulator, ETS, Nuclear Fusion 59, 076024 (2019)