Software Requirements Specification

for

Runaway Indicator

Version 2.2

Prepared by Roland Lohner and Gergő Pokol and Mátyás Aradi

BME NTI, Hungary

26. September 2016.

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
RL	2014-12-15	Integer output, no exception in case of error	1.1
RL	2014-12-16	Introduced embedding subworkflows	1.2
GP	2015-03-24	More consistent naming of actors in 2.4 Operating environment, reference to user documentation added to 1.5 References	1.3
GP	2015-03-26	Small formal corrections	1.4
RL	2015-04-15	Use SWITCH composite workflow in enabler	1.5
RL	2015-04-20	Define growth rate limit in Demultiplexer WF	1.6
RL	2015-04-30	Rename document to runaway-indicator-specification.doc	1.7
RL	2015-04-30	Update workflow images	1.8
AM	2016-01-29	Update figures, references	1.9
AM	2016-03-31	Update Inputs	2.0
PG	2016-06-20	Calculation for absolute value of electric field, description of higher level workflows removed	2.1
AM	2016-09-26	Thermal electron collision time fixed	2.2

Introduction

1.1 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 Indicator (runin) 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 (*http://portal.efda-itm.eu*).

1.2 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.>

1.3 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 workflow developers, and developers of runin module. The document describes the module runin, laying out functional and non-functional requirements. Purpose, overview of the module, interfaces, and CPO objects used, constraints, assumptions and dependencies, functional requirements are contained in this documentation.

1.4 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 Indicator (runin) project supplies a simulator module assembled as a Kepler workflow actor, which is capable of indicating whether runaway electron generation is to be expected during tokamak operation. This functionality is highly valuable in ITM simulations, since present equilibrium and transport calculations neglect the generation of runaway electrons. The runin module can determine whether runaways are generated thus validate the results of equilibrium and transport modules in this manner.

1.5 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.>

Basic description is provided at *http://portal.efda-itm.eu/twiki/bin/view/Main/HCD-ElectronRunawayPhysics?sso_from=bin/view/Main/HCD-ElectronRun-awayPhysics*.

User manual is maintained at *http://portal.efda-itm.eu/twiki/bin/view/Main/HCD-codes-runin-usermanual*.

Runaway Indicator is maintained under the ITM-TF Collaborative Software Development Environment using Gforge. The project documentation is accessible via *http://gforge.efda-itm.eu/gf/project/runin/*. Source code is stored in the SVN repository *https://gforge.efda-itm.eu/svn/runin*.

Analytical formula used to determine the critical electric field is based on the work of J.W. Connor and R.J. Hastie [1]. The method of calculating Dreicer runaway generation growth rate stems from the article of H. Dreicer [2].

2. Overall Description

2.1 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 Indicator module implements an ITM Kepler actor.

2.2 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. Indicate, whether the absolute value of the electric field exceeds the critical level, thus runaway generation is possible.
2. Indicate, whether runaway electron growth rate exceeds a given limit. This calculation takes only the Dreicer runaway generation method in account and assumes a velocity distribution close to Maxwellian, therefore this result should be considered with caution.

2.3 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.

2.4 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.>

Runaway Indicator module operates as an embedded Kepler actor in EU-IM ETS workflows in the INSTANTANEOUS EVENTS & ACTUATORS module. A dedicated EU-IM development and execution environment is set up by the EU-IM staff.

In ETS, Runaway Indicator 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 part of ETS, and thus are not described here in detail, only the function is described.

2.4.1 Enabler workflow – `Runaway indicator` composite actor

Enabler workflow is contained in composite actor named “Runaway indicator”. Its aim is to control the execution. Execution can be enabled or disabled using the workflow parameter “actuator_runaways”. Setting this parameter to “ON” enables the “runaway_indicator” actor.

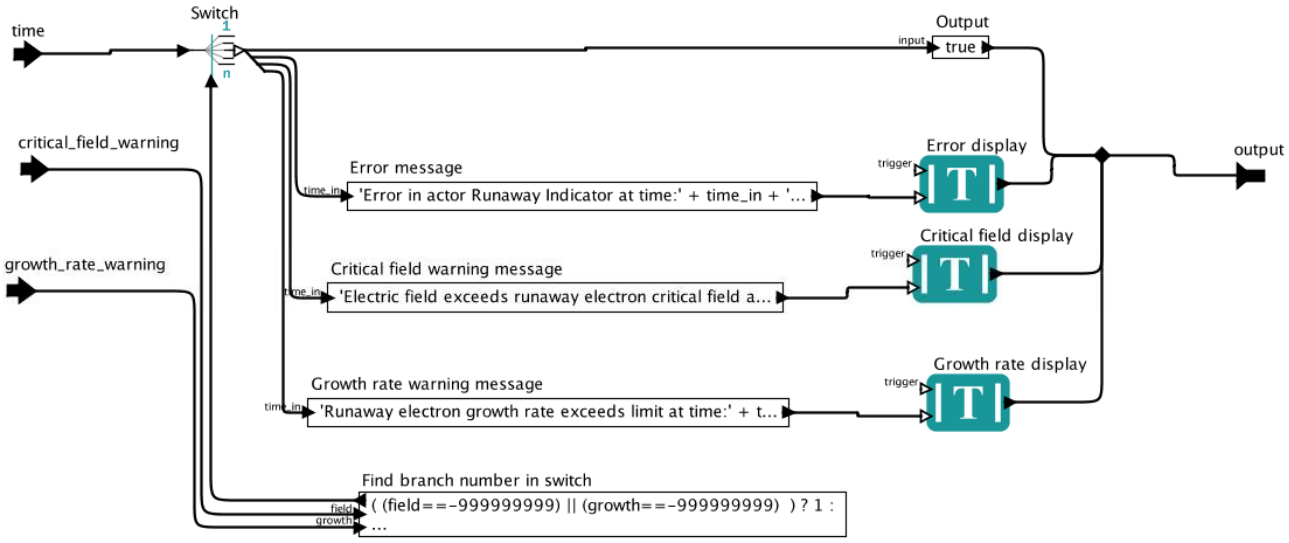
2.4.2 Demultiplexer workflow – `RUNIN` composite actor

AT the next level there is a demultiplexer workflow that reads the input CPO-s and time from the input bundle and feeds those values into the “runaway_indicator” actor. The input bundle of demultiplexer workflow (and subsequently input bundle of enabler workflow) shall contain the following CPOs in the input bundle:

Member name	*Type*
cpos.core.coreprof	coreprof CPO
cpos.core.coreimpur	coreimpur CPO
cpos.mhd.equilibrium	equilibrium CPO
time	Double

The “runaway_indicator” actor has an input variable named as “growth_rate_limit” what is defined and set at this level having 1e12 as default value that is to be understood in the units $s^{-1}m^{-3}$.

2.4.3 Message composer workflow – `Message Composer` composite actor



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

Case	*Message*

Case	*Message*
Electric field is below critical field	-
Electric field exceeds critical field	Electric field exceeds runaway electron critical field at time: <time>
Runaway electron growth rate exceeds the limit given in Demultiplexer workflow.	Runaway electron growth rate exceeds limit at time: <time>
Error occurred during calculation	Error in actor Runaway Indicator at time: <time>

2.5 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 (EU-IM), the Kepler workflow engine provides the capability of workflow orchestration in simulation. Kepler is a free and open source, scientific workflow application. Runaway Indicator module implements a Kepler actor.

In the framework of the 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). Runaway Indicator module uses the UAL layer for input/output.

Runaway Indicator actor is implemented in C++ language.

Runaway Indicator actor is implemented using EU-IM tool "fc2k".

2.6 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 is provided at *http://portal.efda-itm.eu/twiki/bin/view/Main/HCD-ElectronRunawayPhysics?sso_from=bin/view/Main/HCD-ElectronRun-awayPhysics*.

User manual is maintained at *http://portal.efda-itm.eu/twiki/bin/view/Main/HCD-codes-runin-usermanual*.

2.7 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

3.1 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.>

3.2 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.>

3.3 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.>

3.3.1 CPO Input

Runaway Indicator 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)	coreimpur[time]/impurity[species]/z[r] coreimpur[time]/impurity[species]/nz[r]	1 $1/m^3$

Parameter	*CPO element*	*Unit*
	coreprof[time]/ni/value[r, species]	$1/m^3$
	coreprof[time]/compositions/ions[species]/zion	1
Electric field profile (1d)	coreprof[time]/profiles1d/eparallel[r]	V/m
	coreprof[time]/toroid_field/b0	T
	equilibrium[time]/profiles_1d/b_av[r]	T

3.3.2 Growth rate limit input

The actor reads the limit value of the runaway growth rate using this input. A warning is raised if calculated runaway growth rate exceeds this limit. Dimension of input value is $s^{-1}m^{-3}$. Recommended limit value is 10^{12} particle per second. (This growth rate generates a runaway current of approximately 1kA considering a 10 seconds long discharge.)

3.3.3 Output

The `runaway_indicator` core actor emits two integer outputs (of possible values 0 and 1 and -999999999 in case of internal error), which

1. Indicate, whether electric field is below the critical level, thus runaway generation is impossible.
2. Indicate, whether runaway electron growth rate exceeds the limit given in Demultiplexer workflow. This calculation takes only the Dreicer runaway generation method in account and assumes a velocity distribution close to Maxwellian, therefore this result should be considered with caution.

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

3.4 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.>

4.1 Critical field warning

<Don't really say "System Feature 1." State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

The module outputs an integer value (0 or 1) which indicates, whether electric field is above the critical level, thus runaway generation is possible. When the electric field exceeds the critical level this warning raises, a value of 1 is outputted. However it does not mean that runaway electrons are present, the warning only signs the possibility.

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

The warning output is calculated every time the actor is called.

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1: If a radius exists where electric field is above critical, returns 1.
REQ-2: If electric field is below critical across the whole profile, returns 0.
REQ-3: Critical electric field is determined based the formula:

$$E_c = n_e e^3 \ln \Lambda / 4\pi\epsilon_0^2 m_e c^2$$

REQ-4: Coulomb logarithm is calculated as:

$$\ln \Lambda = 14.9 - 0.5 \cdot \log (n_e \cdot 10^{-20}) + \log (t_e \cdot 10^{-3})$$

REQ-5: When some input data is missing in CPO input, return error code.
REQ-6: Error code is an integer value of -999999999.
REQ-7: When some input data is missing in CPO input, prints error message.

4.2 Growth rate warning

4.2.1 Description and Priority

The module outputs an integer value (0 or 1) which indicates, whether the growth rate of runaway electron generation is above the limit value given in Demultiplexer workflow, thus runaway generation is to be expected. If the growth rate exceeds the limit value this warning raises.

4.2.2 Stimulus/Response Sequences

The warning output is calculated every time the actor is called.

4.2.3 Functional Requirements

REQ-1: If a radius exists where growth rate exceeds the limit, returns 1.
REQ-2: If growth rate is below limit across the whole profile, returns 0.
REQ-3: Growth rate is calculated using the formula (67) of [2]:

$$\dot{n}_r^{\mathrm{I}} \simeq \frac{n_e}{\tau}\left(\frac{m_e c^2}{2T_e}\right)^{3/2}\left(\frac{E_{\mathrm{D}}}{E_\parallel}\right)^{3(1+Z_{\mathrm{eff}})/16} \times \exp\left(-\frac{E_{\mathrm{D}}}{4E_\parallel}-\sqrt{\frac{(1+Z_{\mathrm{eff}})E_{\mathrm{D}}}{E_\parallel}}\right)$$

$$\tau = 4\pi\varepsilon_0^2 m_e^2 v_{th}^3 / n_e e^4 \ln\Lambda$$

$$v_{th} = (2T_e / m_e)^{1/2}$$

$$E_{\mathrm{D}} = m_e^2 c^3 / (e\tau T_e)$$

REQ-4: Coulomb logarithm is calculated as:

$$\ln \Lambda = 14.9 - 0.5 \cdot \log (n_e \cdot 10^{-20}) + \log (t_e \cdot 10^{-3})$$

REQ-5: When some input data is missing in CPO input, return error code.
REQ-6: Error code is an integer value of -999999999.
REQ-7: When some input data is missing in CPO input, prints error message.

5. Other Nonfunctional Requirements

5.1 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.>

5.2 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.>

5.3 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.>

5.4 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.>

5.5 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] J.W. Connor, R.J. Hastie, Relativistic limitations on runaway electrons, *Nuclear Fusion* **15**, 415 (1975)
[2] H. Dreicer, Electron and Ion Runaway in a Fully Ionized Gas, *Physical Review* **115**, 238 (1959)