Spacecraft Plasma Interaction Analysis and Simulation Toolkit

User Requirements Document (URD)

ESTEC Contract No. 16806/02/NL/JA

ONERA

ARTENUM

UNIVERSITY PARIS 7

User Requirements Document (URD)

Contributors

Jean-François Roussel **François Rogier** **Michel Lemoine**	**ONERA/DESP** 2 Av. Edouard Belin 31055 Toulouse cedex France
Gérard Sookahet	**Artenum** 25 Rue des Bas 92600 Asnières sur Seine France
Guillaume Rousseau	**Université Paris 7 Denis Diderot / BVRI** 2 place Jussieu 75251 Paris cedex 5 France

Document Status Sheet			
Document Title:			
Issue	**Date**	**Author(s) of document/change**	**Reason of change**
0.0	20/01/2003	JFR	
0.1	14/02/2003	JFR	Reorganisation
0.2	19/02/2003	JFR, GR, GS	Content update
0.3	21/02/2003	JFR, GR, GS, ML, FR	Minor corrections
1.0	06/03/2003	JFR	Changes requested during 4th SPINE workshop
1.1	20/03/2003	JFR, GS	Changes requested after 4th SPINE workshop
1.2	02/04/2003	AH, JFR, GR	Changes after contractor-ESA TO discussion
1.3	11/04/2003	GR	Changes after SDAB meeting discussion

Contents

1. Introduction

1.1. Purpose and scope

The "Spacecraft Plasma Interaction Analysis and Simulation Toolkit" project aims at developing a toolkit for modelling spacecraft-plasma interactions. A first definition of this software aims and structure was given in ESA invitation to tender [ITT] and the ONERA-Artenum-UP7 proposition [PROP] in response to this ITT. This toolkit shall include a general infrastructure, allowing to conveniently embed an increasing number of modules, or routines. This toolkit is usually called SPIS for Spacecraft-Plasma Interaction System (http://www.spis.org/spis). The contractor is responsible for the design and implementation of the framework, implementation and integration of some of the numerical routines. The SPINE community shall provide guidelines or requirements so that this software answers its needs, develop some of the routines and test the software on test cases defined in the framework of three working groups.

It is the purpose of this User Requirement Document to gather, organise and rationalise the requirements of SPINE community. This URD is considered as in a quasi final version.

The scope of these requirements extends from the general or high level requirements of the end-users, to the requirements of developer-users, who are able to express their needs at the routine level. It does not extend yet as far as the Software Requirement Document, which will go into more technical details.

1.2. Overview

The context of this software development and the situations that the code should be able to model are described in section 2. They naturally lead high level requirements, presented in section 3. The consequences are given in the next sections (3 to 8) in term of detailed requirements, which are numbered and will be tracked throughout the project. Annex A reports a state of the art of spacecraft-plasma interaction modelling through a listing of the major existing codes and their description. Eventually, the annex B contains the URD of IPICSS project (written by Julien Forest within SPINE community), which can be considered as a basis to the present work (different file for online download). It is annexed to this document mostly to avoid repeating here general considerations about the context, which were fully developed in IPICSS URD.

1.3. Definitions

Users: a first class of users will use the software as a finite product to model their problems. They will only use a standard packaged version of the software behind a user friendly front end. They should express their requirements in terms of functionalities (ex: "predict the ion distribution into my detector in an ionospheric plasma"), which are called **high level requirements.**

Developers: a second category of users will combine routines in the software framework to perform the desired simulations. They may also develop their own routines. They will thus be able to express their requirements at a lower level, typically at routine level (ex: “need for a PIC solver with such boundary conditions on such a mesh”), called **low level requirements**.

Contractor: the contractor consortium, ONERA, Artenum, and University Paris 7 is in charge of the development of the software architecture and a first set of routines.

1.4. Nomenclature

User Requirements referencing
User requirements were numbered to allow a proper tracking throughout the project.
UR0.zzz are high level requirements
URx.y.zzz for x > 0 are detailed requirements
When several versions of a requirement correspond to different versions of a routine, the different versions are designated with consecutive numbers (ex: *UR2.1.1.101, UR2.1.1.102,* etc. for various versions of Poisson equation solver requirements).
Some requirements are general rules or constraints (ex: “proper documentation of the source code”), others are routines or capabilities to be developed. This distinction is at the base of the UR classification in [IPICSS/URD]. Whereas the general rules simply apply throughout the project, the routines or capabilities can have variable status:

- *URx.y.zzz* (no *): they shall be provided by the contractor
- ***URx.y.zzz*: they will simply be taken into account in the software architecture but their development is left to the community (the ** sign is thus not equivalent to suppressing the requirement, since the integration of such a routine will be made possible)
- **URx.y.zzz*: intermediately, the routines/capabilities designated with one * are desirable, but their implementation is not yet decided in the current status of the project and URD. Their future implementation, or not, depends on future findings (availability of resources) and discussions.

1.5. Acronyms

API	Application Programming Interface
CAD	Computer Aided Design
DM	Developer Manual
DXF	Drawing eXchange Format
EP	Electric Propulsion
ESD	ElectroStatic Discharge
FAQ	Frequently Asked Questions
FEEP	Field Effect Electric Propulsion
GEO	Geostationary Earth Orbit
GTS	GNU Triangulated Surface Library
GUI	Graphics User Interface
IGES	Initial Graphics Exchange Specification
LEO	Low Earth Orbit
MEO	Medium Earth Orbit
MHD	Magneto-Hydrodynamics

ONERA	Office National d'Etudes et de Recherches Aérospatiales
OO	Object Oriented
OOA	Object Oriented Approach
PEO	Polar (low) Earth Orbit
PIC	Particle-In-Cell
SDAB	Software Development Advisory Board
SPINE	Spacecraft Plasma Interaction Network in Europe
SPIS	Spacecraft Plasma Interaction System
TRD	Theoretical Reference Document
UM	User Manual
UP7	Université Denis Diderot - Paris 7
UR	User Requirement
URD	User Requirements Document
VRML	Virtual Reality Modelling Language
VTK	Visualization ToolKit
WG	Working Group
XML	eXtensible Markup Language

1.6. Reference Documents

[ITT]	Invitation to Tender AO/1-4147/02/NL/JA – Spacecraft Plasma Interaction Analysis and Simulation Toolkit
[PROP]	Proposal for a Spacecraft Plasma Interaction Analysis and Simulation Toolkit, in response to AO/1-4147/02/NL/JA, ONERA, Artenum, UP7
[IPICSS/URD]	“Investigation of Plasma Induced Charging of Satellite Systems” (IPICSS) URD, Julien Forest, v1.4, October 200
[ECSS-E-40]	European Cooperation for Space Standards – Engineering – Software

2. The context: space systems and the physics to model

The context in which the present software shall be developed and operated must first be described. It has been discussed in details in [IPICSS/URD] annexed to this document, so this section tries to be more concise and synthetic. With respect to previous URDs (IPCSS in annex B, and SpaceGRID) the state of the art of plasma simulation codes yet required an update, which is presented in Annex A.

The developed software shall address all, or most **problems** related to spacecraft-plasma interactions. Depending on the missions and environments the problems can be very diverse. Generically, plasma may involve potentials, charges and currents. When short transients or electromagnetic waves are involved electromagnetic compatibility issues may also arise. More specifically the problems to be modelled are thus: spacecraft charging whatever the sign and amplitude of the potentials (kilovolts negative in GEO or PEO, a few Volts positive in quiet low density plasma…), current collection (on solar arrays, on active devices…), phenomena in the surrounding plasma that can affect the spacecraft by propagation (plasma-wave interactions…). Only surface charging issues shall yet be addressed and deep dielectric charging shall be excluded from the field of the modelled problems, for technical reasons detailed below.

The type of **mission** to be modelled is very variable, and none should be excluded from the application field of the code, even though not all the necessary routines may be immediately developed. The mission may be located in earth orbit (LEO, PEO, MEO, GEO or higher), planetary, interplanetary, or comet environment, etc. The mission can be scientific, technological or commercial. The spacecraft attitude can be three-axis stabilised, spinned, or other. The equipment and payloads on board can be very diverse: chemical or electrical propulsion (Hall or gridded thruster, FEEP…), other plasma sources (contactors, ion or electron guns…), telecom equipment, optics, scientific detectors in particular plasma analysers, etc. Depending on the mission, very variable perturbations due to plasma interactions are tracked: for instance charging beyond a fraction of Volt can be damageable to scientific plasma measurements, but ESD risk only shows up above thousands of Volts on a commercial telecom spacecraft.

An important consequence of the mission is the definition of the plasma **environment** to model. Considering all types of natural or artificial plasma, typical ranges of parameters are the following:

- density: 10^{-2} cm^{-3} (external magnetosphere) to 10^{10} cm^{-3} (EP thruster plume)
- temperature of thermal populations: 0.01 eV (cometary plasma) to 100 eV (“cold” plasma in magnetosphere)
- drifting velocity in spacecraft frame: ~ 0 km/s for plasma from a contactor, 0.1 to a few km/s for comet plasma, 2 to tens of km/s in earth orbit (plasma convection), several hundreds of km/s in solar wind.
- energy of high energy populations: up to a few 100 keV (higher energy fluxes are too small to affect surface charging, they only affect deep dielectric charging, which should be excluded from this software, cf. below)
- magnetic field: from a few 10^{-9} T in the solar wind, a few 10^{-5} T in LEO, and much more close to some device (near field of plume Hall thruster, etc.)

The **physics** originating from these variable space systems and environments is also very broad. The plasma dynamics is most of the time electrostatic (with background magnetic field), sometimes electromagnetic (mostly for waves), and may also sometimes involve collisions. The plasma interaction with surfaces mostly involves the collection of charges and the emission of electrons (secondary emission under electron or ion impact, photo-emission). The distribution and re-distribution of charges within the spacecraft is also complex. It involves the transport of high energy particles in matter, conductivity of dielectrics, with possible radiation enhancement. A general modelling of these phenomena is indeed almost impossible. Two classes of effects are thus distinguished. On the one hand, particles of relatively low energy (below a few tens of keV for electrons) have a penetration depth small enough to be considered as remaining at the dielectric surface. It simplifies the physical model to a deposition of charges on dielectric coatings (capacitors) and a distribution of the image charges on the underlying conductors. The leakage resistances have then to be added (volume and surface conductivity of dielectrics, with possible radiation or field enhancement). The inductance of spacecraft circuit may also be important for specific very quick phenomena. On the other hand, the particles of higher energy result in a deep charging of dielectrics, possibly well inside the spacecraft. This deep charging can indeed largely be disconnected from the surface charging considered above (little effect of surface charging on so high energy particles, or on deep electric fields). Since moreover the tools and physics for deep charging are very different than for surface charging (need to mesh the volume below spacecraft external surface, model the transport in matter…), the deep dielectric charging is indeed likely to be excluded from the field of application of SPIS software.

Some more descriptions of the missions, environment and physics can be found in the annexed [IPICSS/URD] and in conference proceedings (7th Charging Conference ESTEC 2001…).

The software development shall be conducted for a **community**, and (partially) by a community. This SPINE community (Spacecraft Plasma Interaction Network in Europe) is thus a key element of the project. It comprises researchers and engineers from academic, institutional, and commercial companies. Some members of SPINE community are specialists of spacecraft plasma interaction (high level charging, or interactions with cold plasma…). But most SPINE members do not indeed properly work in the spacecraft-plasma interaction field. The finality of their work may be either scientific (planetary environment study, detector deign…) or technological (equipment development, system integration…), but they need to considered spacecraft-plasma interactions to insure proper operations of their spacecraft or equipment. As a consequence, they necessarily have at least a basic knowledge of spacecraft plasma interactions physics. Some other members are also working in domains useful to this development (plasma physicists, computer scientists…).

The skills and the **roles** of all of these community members is thus likely to be also very different. Some of them will be simple ***users***. They will first express requirements at a rather high level, i.e. in terms of global capabilities, as e.g. the capability of modelling their detector on their spacecraft with such and such outputs. They want to have a user friendly software, and control it through a few parameters. Since the degree of automation of the software is expected to be low, these users are assumed to use their scientific skills to define these simulation parameters. Some other members will be more involved in the modelling and will enter deeper into the code and their structure. Their requirements may be expressed at routine level, closely controlling the modelling architecture and algorithms. They may then build their own application by combining the routines provided in SPIS toolkit. They may also implement some routines. They are called ***developers***, although the development may be limited for some of them to the combination of objects, in a fashion similar to GEANT4. Another role can be distinguished, the one of ***expert***, mostly in relevant physics or computer domains. Some of the users are of course also experts, but all experts may not really be users (external invited experts, scientists…). The last and central role is the one of ***integrator*** or co-ordinator, which is mostly played by the contractor consortium and ESA. It is central to the community since it organises the community work around SPIS software development, and it is central to the software development since the integrator will design and implement the software framework and co-ordinate the routine development.

From the **software** point of view, several types of architecture and design can exist. Some software can be designed as a closed system, not expected to evolve, at least frequently. Others are designed in a modular fashion, allowing the interfacing with external tools and the integration of new routines or modules. SPIS software should be in the second category, to allow a co-development in SPINE community and the efficient integration of existing tools. In this framework, the most efficient developing method is through open source, which allows an easy integration and modification of modules.

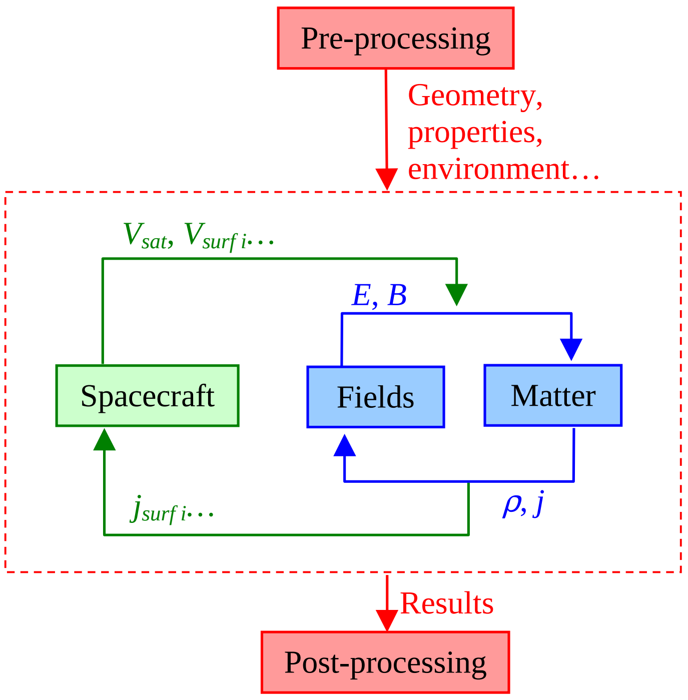


Figure 1: the general simulation process

Although the requirements and the design of the software are still to be defined, the general structure of the **modelling process** is known, and summarising it here may allow a better understanding of the forthcoming requirements (see diagram of Figure 1 above). A simulation is based on data which must be prepared and provided to the code through a pre-processing task. It mostly amounts to defining the spacecraft geometry and properties, and the environment it interacts with (*UR1* section below, called modelling). It may also involve simulation control commands. The simulation is the next step. In the case of spacecraft plasma interactions, the general coupling of the solvers naturally follows the physical couplings. The plasma dynamics (*UR2.1*) involves the coupling of the field equations (Poisson in its simplest electrostatic version, cf. *UR2.1.1*) and the matter transport equations (Vlasov in a kinetic approach, cf. *UR2.1.2*). The numerical way to mimic their physical coupling is to iterate the solver runs for fields and matter (*UR2.1.3*). Of course, this is only the general scheme, and in some simplified approaches the field equations can for example be eliminated through quasi-neutral approximations, or the full plasma dynamics can also be reduced to an approximate analytical law. The plasma is then coupled to the spacecraft, since the potentials on the spacecraft influence the plasma, whereas the currents collected from the plasma modify spacecraft potentials. The interactions at plasma-spacecraft boundary are ruled by particle collection and emission (*UR2.2*). As discussed above, in the superficial charging approximation the spacecraft is modelled by an equivalent circuit (*UR2.3*). The coupling of the spacecraft circuit with the plasma dynamics must eventually be performed (*UR2.4*) as an other iteration loop, external to the plasma dynamics loop. It is indeed also far from trivial in particular because of the very different time scales for plasma dynamics, relative and absolute charging of spacecraft. The faster dynamics of plasma is also the reason why the plasma loop is usually treated as an inner loop of the spacecraft loop, whereas everything is in reality simultaneously coupled The post-processing of simulation results (*UR3*) can then be subdivided in two step, the post-treatment of data (extra computation on results, *UR3.1*) and their representation, most of the time graphical (*UR3.2*).

3. UR0: High level requirements

The high level requirements are presented here. They address the needs of users, without entering into the details of the modelling method. The lower level needs, at routine level, are detailed in the next sections.

- **UR0.100:** Flexible spacecraft model
  - Geometry:
    - volumes, thin surfaces, wires
    - no limitation on size and element number
    - no limitation on the type of surface mesh (unstructured!)
  - Surface properties:
    - basic set of properties (NASCAP or extended/modified NASCAP properties list)
    - extension possible

- **UR0.200:** Multi-scale geometry should simultaneously allow the modelling of:
  - large scales: large computation box to allow large sheaths modelling
  - small scales: small scale detectors or devices

- **UR0.300:** User-friendly interface (graphical and command line) for pre/post-processing and simulation control

- **UR0.400:** Modelling of general physical phenomena involved in space
  - Plasma dynamics in volume:
    - electrostatic or electromagnetic dynamics
    - involving kinetic or fluid behaviour
    - virtually unlimited size to Debye length ratio
    - collisions (elastic, inelastic including charge exchange)
    - arbitrary sources of plasma at computation box boundaries
    - library of pre-defined sources at boundaries (environment)
  - At plasma-spacecraft interface, all possible phenomena:
    - charge collection form volume dynamics
    - secondary emission (from electrons or ions), including back-scattered electrons
    - photo-emission
    - arbitrary plasma source, either fixed or depending on the incoming flux
    - library of pre-defined source on spacecraft (artificial plasma)
  - Within the spacecraft:
    - Equivalent circuit of spacecraft, involving:
      - conductors
      - capacitors (coatings, possibly absolute capacitance)
      - conductivity through dielectrics (surface conductivity, bulk conductivity, induced conductivity)
      - inductance (for short transients)
    - Coupling of this circuit with volume dynamics through surface phenomena (collection, emission)

4. UR1: Modelling

4.1. UR1.1: Spacecraft

UR1.1.1 Spacecraft geometrical description
UR1.1.1.100 The software should be able to describe and model the spacecraft shape with a realistic 3D geometry.
UR1.1.1.200 The software will be able to locally refine/coarsen the mesh (structured or unstructured).
UR1.1.1.300 The spacecraft must be described with subsets (groups of surface nodes).

UR1.1.2 Model size limitation
UR1.1.2.100 No limitation other than machine memory should apply on the number of nodes and elements

UR1.1.3 Spacecraft electrical model
UR1.1.3.100 The user must be able to arbitrarily describe the spacecraft in a set of electrical nodes, with at least the possibility to have one per surface element. This may be done via commande line or GUI.
UR1.1.3.200 The user must be able to activate/deactivate each node with respect to equivalent circuit of spacecraft, or merge some nodes.
UR1.1.3.300 For each electrical node, the user must be able to define all electrical properties (stored in a database) with at least, and without limitation to the extension of properties:

- Secondary emission parameters (under electron and ion impacts)
- Photoemission parameters
- Surface and volume conductivity for dielectrics (and induced conductivity)

UR1.1.3.400 The software should model the complete electrical circuit of the spacecraft resulting from these properties (network of capacitors, resistors)
UR1.1.3.500 The equivalent electrical circuit of the spacecraft can be automatically derived from the properties of the electric nodes (conductor/insulator…) or edited by the user via command line or GUI, allowing the modelling of extra properties: active bias, inductances…

UR1.1.4 Spacecraft electrical state
UR1.1.4.100 The software should model the complete electrical state of the spacecraft with at least:

- the spacecraft ground potential ϕ_{sat} and the electrical potential ϕ_i at all surface points of the vehicle (differential charging)
- the global and local electrostatic charges Q_{sat} and Q_i
- the local currents j_i

4.2. UR1.2: Volume geometry

UR1.2.1 Mesh edition

UR1.2.1.100 The software must be able to create/delete mesh entity.

UR1.2.1.200 The software should provide automatic meshing facilities.
UR1.2.1.300 The software shall determine the surface or the volume of an element.
UR1.2.1.400 The user must be able to define node groups or element groups.

UR1.2.2 Mesh control

UR1.2.2.100 The software shall determine the edge's length.
UR1.2.2.200 The software shall determine the aspect ratio of the elements.
UR1.2.2.300 The software shall determine the surface or the volume of an element.
UR1.2.2.400 The software will be able to locally refine/coarsen the mesh (structured or unstructured).

UR1.2.3 Mesh constraint

UR1.2.3.100 The software must at least provide one type of meshing algorithm, with the possibility to include adaptive meshing capability.
*UR1.2.3.200 The software must be able to control the distribution of the mesh's elements with respect to numerical (CFL conditions) and physical constraints.

4.3. UR1.3: Environment

Several predefined environments (with some adjustable parameters) can be defined, with the extra possibility to instead use a completely new environment definition (through new routine writing).

- **UR1.3.100:** model of ionospheric plasma, with at least:
  - Ions: multispecies Maxwellian distributions of given densities, mean velocities, temperatures
  - Electrons: Maxwellian distribution of given density, temperature
  - Homogeneous ***B*** field

- ****UR1.3.200:** model of auroral electrons

- **UR1.3.300:** model of magnetospheric plasma (typically GEO)
  - High energy electrons: two maxwellian distributions
  - Cold plasma: Maxwellian distribution of given density, temperature
  - Predefined worst case values for these parameters (or user-defined value)
  - Homogeneous ***B*** field

- ****UR1.3.400:** model of solar wind

- ****UR1.3.500**: open environment model:
  - Generic model including arbitrary particle populations and maybe non-uniform ***B*** field (a magnetosphere or a Hall thruster discharge channel for instance)
  - Interfacing with existing models with pre-defined environment conditions (e.g. SPENVIS/LEOPOLD)

- **UR1.3.600:** allow the modelling of transient environments, including the effects on spacecraft (e.g. solar array polarisation when coming into sunlight)

- **UR1.3.700:** model of sun exposition (global solar flux depending on distance to sun, shadowing, and angle between surface normal and sun direction)

4.4. UR1.4: particle and plasma sources

Similarly to the environment models, several predefined particle or plasma sources (with a few adjustable parameters) can be defined, with the extra possibility to define a completely new source (through new routine writing).

- **UR1.4.000**: interfacing with open plasma source model

- **UR1.4.100:** Maxwellian source
  - Maxwellian distribution of given density, temperature, mean velocity
  - Source extension: a point or a surface

  NB: it should allow to models beams (small temperature) and simplified thruster plumes

- ****UR1.4.200:** Hall thruster sources library, for each thruster:
  - Ions: full velocity distribution function, possibly variation of the distribution depending on the exit point
  - Electrons: temperature at exit, or better energy distribution
  - Possibly time variation of these parameters (oscillations)
  - Possibly magnetic field due to the thruster (close to exit)
  - Possibly emitted electromagnetic noise

- ****UR1.4.300:** Gridded ion thruster sources library
  - …

- ****UR1.4.400:** FEEP thrusters sources library:
  - …

5. UR2: The physics and the solvers

5.1. UR2.1: Plasma dynamics

As explained in section 2 above, the equations to solve are:

1- field equations (Poisson, Maxwell…)

2- matter dynamics equations, either kinetic (Vlasov, possibly with collisions) or fluid (variants of Euler equation, possibly with higher momentum equations)

3- the coupling of field and matter equation, with stability issues (for instance “PIC” is usually used for a specific Monte Carlo solver of Vlasov equation, but also its coupling with Poisson equation)

The plasma dynamics solvers shall basically be 3D solvers. However the possibility to integrate 2D (planar or cylindrical) and 1D (spherical) solvers is also to be envisaged.

5.1.1. UR2.1.1: Field equations solvers

The field equations shall be limited to Poisson equation in most cases (*UR2.1.1.100*), but the possible integration of a Maxwell equation solver should be allowed by the software framework (*UR2.1.1.200*).

- **UR2.1.1.100:** Poisson equation solver, with the possible following instances:
- **UR2.1.1.101:** basic Poisson equation solver
  - Mesh: rectangular
  - Spacecraft model: unstructured
  - Boundary conditions: mixed Dirichlet-Neuman on mesh external boundaries, Dirichlet on spacecraft
  - Method: finite differences, over-relaxed Gauss-Seidel or conjugate gradient
- **UR2.1.1.102:** improved Poisson equation solver
  - Mesh: multiscale (nested rectangular, or unstructured)
  - Spacecraft model: unstructured
  - Boundary conditions: mixed Dirichlet-Neuman on mesh external boundaries, Dirichlet on spacecraft
  - Method: finite differences, finite elements, or finite volume over-relaxed Gauss-Seidel if possible or preconditioned conjugate gradient or TBD method
- ***UR2.1.1.103:** non linear Poisson equation solver
  - Specific handling of electron distribution: implicit method for solving Poisson equation including Boltzmann distribution for electron distribution
  - Mesh: multiscale (nested rectangular, or unstructured)
  - Spacecraft model: unstructured
  - Boundary conditions: mixed Dirichlet-Neuman on mesh external boundaries, Dirichlet on spacecraft
  - Method: finite differences, over-relaxed Gauss-Seidel if possible, or preconditioned conjugate gradient, or TBD method

- ****UR2.1.1.104:** more general Poisson equation solver, with generalised mesh type or boundary conditions, other methods…

- ***UR2.1.1.200:** Maxwell equations solver:
  - Mesh: TBD
  - Boundary conditions: TBD
  - Method: TBD

5.1.2. UR2.1.2: Matter dynamics solvers

- **UR2.1.2.101:** basic PIC model, matter only (kinetic, Monte Carlo)
  - Electric field: local, time dependant
  - Magnetic field: global, static
- ****UR2.1.2.10x:** extended PIC model (electromagnetic field, etc.)

- ***UR2.1.2.201:** Simplest analytic local fluid model:
  - Boltzmann distribution: $n \sim \exp(q\,\phi)$
- ***UR2.1.2.202:** General analytic local fluid model:
  - Any law: *current* or *density* = *f*(*local_potential*, *local_electric_field*)

- ****UR2.1.2.300:** Euler equation solver for one fluid

- ****UR2.1.2.400:** Collision model, in particular charge exchange (CEX) with a given background density of neutrals

5.1.3. UR2.1.3: Coupling of matter and field dynamics

- **UR2.1.3.100:** coupling of PIC matter dynamics and Poisson solver (sometimes also called PIC)
  - User control over time steps and iteration parameters, in particular allowing:
    - different electron and ion time steps (“sub-cycling”)
    - different integration duration for ions and electrons (“numerical times” for convergence towards steady state)
    - under-relaxation control…
  - Possibility of implementing customised particle injection methods (e. g. injection flux modulated by ***E*** field at boundary)

- ****UR2.1.3.200:** various global fluid models (quasi-neutral, with collisions…)

- ****UR2.1.3.300:** MHD:

5.2. UR2.2: Plasma-surface interactions

- **UR2.2.100:** Secondary emission under electron impact:
  - Yield function: either classical function (NASCAP) based on energy and yield at maximum, or a new function based on new data (experimental ITT)
  - Distribution function of secondary electrons (reflected and true secondary electrons)

- **UR2.2.200:** Secondary emission under ion impact

- **UR2.2.300:** Photo-emission:
  - Total current for one sun illumination
  - Distribution function of photo electrons: energy distribution, Lambertian law, or better

- **UR2.2.400:** possibility of interfacing other routines (sputtering…)

5.3. UR2.3: Spacecraft circuit solvers and coupling with plasma

Very different time scales coexist in this problem: very fast plasma dynamics (fraction of second), fast absolute charging (second scale or below), slow relative charging (can be minutes in GEO). Since plasma dynamics cannot be modelled over seconds or minutes, in case of time dependent computations the potentials on the spacecraft need to be integrated on the basis of $I(V)$ laws provided by the plasma dynamics module. Moreover, the very different time scales easily generate instabilities, which can be handled by implicit schemes.

- **UR2.3.100:** Spacecraft circuit model:
  - Network of capacitors (coatings)
  - Spacecraft absolute capacitance: explicit or implicit
  - Leakage through coatings (surface and volume conductivity, environment dependent for induced conductivity)
  - Inductances (used for fast transients only)
  - Active voltage or current sources

- **UR2.3.201:** spacecraft circuit solver, with:
  - Use of collection law $I(V)$ obtained from plasma dynamics solving
  - User controlled or automated parameters for time integration and refreshing of the $I(V)$ law
  - *Possibility to use customised analytical emission/collection laws $I(V)$ for part or all of the surfaces
  - *Possibility to use customised emission/collection laws $I(V)$ for part or all of the surfaces depending on the plasma characteristics (e.g. for re-collection of zero temperature emitted particles in case of attractive $\boldsymbol{E}$ field)
- ***UR2.3.202:** implicit spacecraft circuit solver (requires more information from plasma model, such as dI/dV)

6. UR3: Post-processing

6.1. UR3.1: Post-treatment of results

The main type results to be used for the post-processing are:

UR3.1.1 Potential

UR3.1.1.100 The software should give as output the electrostatic potential of the spacecraft with respect to the unperturbed plasma included the equilibrium potential.

UR3.1.1.200 The software should give as output a detailed 3D map of the electrical potential in all points of the computational space around the spacecraft. In particular, the code should give as output the detailed 3D structure of the electrostatic sheath.

UR3.1.2 Density

UR3.1.2.100 The software should give as output a complete 3D map of the charge density, (i.e. net electrical charge by unit of volume) around the vehicle.

UR3.1.2.200 The software should give as output a complete 3D map of the density of particles, number of particles by unit volume, for each species at all points of the computational space around the spacecraft.

UR3.1.3 Flux

UR3.1.3.100 The software should be able to give as output the particle flux, or its equivalent in fluids description, for each species on each node or element of surface of the spacecraft.

UR3.1.4 Electric fields

UR3.1.4.100 The software shall give a complete 3D map of the electrical field, or at least the strength of the electrical field, in all points of the computational space around the vehicle.

UR3.1.4.200 The software shall give as results the value and the direction of the electric field at the surface of the spacecraft.

UR3.1.5 Electric state of the spacecraft

UR3.1.5.100 The software shall give as results the complete electrostatic state of the spacecraft or at least for each electrical node the potential, the net charge, the net current for each species, etc ….

UR3.1.6 Distribution functions

UR3.1.6.100 The software shall give as results the distribution function of a specified population, either on a surface element or at a volume point

UR3.1.7 Particle tracking

UR3.1.7.100 The software shall give as results the trajectories of defined test particles

UR3.1.8 Virtual instruments

*UR3.1.8.100 The user may place virtual instruments at various places in the simulation box or on the spacecraft. They provide local parameters (potential, density, flux, etc.) in a separate file.
*UR3.1.8.200 More complex virtual instruments can also be integrated by the users. Typically, they mimic the functioning of real instruments, e.g. by only integrating fluxes above an energy threshold like and RPA (Retarding Potential Analyser).

UR3.1.9 Time evolution

UR3.1.9.100 The time evolution of maps (of potentials, densities, etc.) is reported through sequential map files (they can be used as basis to produce animations).

UR3.1.9.200 The time evolution of local values defined by virtual instruments is reported in a single file (possibly one file per instrument, with the time dependence presented in column).

UR3.1.10 Output files

UR3.1.10.100 Each output files formats should be clearly defined and specified, in order to let the user to develop *a posteriori* its own post-processing subroutines.
UR3.1.10.200 The output file formats should respect the complete accuracy of the data (double or float).
UR3.1.10.300 The software should be able to create a partial output files specified by the user (from ranges, every step, etc ….).
UR3.1.10.400 In priority, the output formats should be in human-readable ASCII format. However, depending on technical constraints (size for example) the output format may be also in binary format.

***UR3.1.11 Post-processing API**

UR3.1.11.100 The software should provide an API (Application Programming Interface) to enhance visualization capabilities (such as virtual instrument) via plug-ins developed by users.

6.2. UR 3.2: Representation of results

The results should be represented by different methods

UR3.2.1 Files

UR3.2.1.100 Files that contain numbers shall be represented in a table (spreadsheet-like).

UR3.2.1.200 Matrices shall be represented in a table.

UR3.2.2 2D representations

UR3.2.2.100 Files that contain columns of numbers shall be represented by 2D graphs.
*UR3.2.2.200 2D graphs can be saved in standards formats (.ps, .gif, .vtk, HDF, NetCDF).
*UR3.2.2.300 Several 2D representations must be available (points, lines, curves, histograms, iso-curves, vector fields, streamlines).
UR3.2.2.400 The user must be able to zoom in/zoom out and translate the 2D graph.

URD3.2.3 3D representations

UR3.2.3.100 Files that contain columns of numbers shall be represented by 3D graphs.
*UR3.2.3.200 3D graphs can be saved in standards formats (.ps, .gif, .vtk, VRML, HDF, NetCDF).
UR3.2.3.300 Several 3D representations must be available (points, lines, curves, volumes, vector fields, streamlines).
UR3.2.3.400 The user must be able to zoom in/zoom out the 3D graph.
UR3.2.3.500 The user must be able to rotate the 3D graph.
UR3.2.3.600 The user must be able to translate the 3D graph.
UR3.2.3.700 The user must be able to visualize the spacecraft with mapped representation of all relevant physical data (potential, charge, current, surface current, electric fields).

URD3.2.4 Cutting plane and projections

UR3.2.4.100 A 3D volume shall be projected on a plane (XY, XZ, YZ).
UR3.2.4.200 A 3D volume shall be sliced by an arbitrary plane defined by its equation.
UR3.2.3.300 A 3D volume shall be represented by iso-curves.

7. UR4: Interface and control

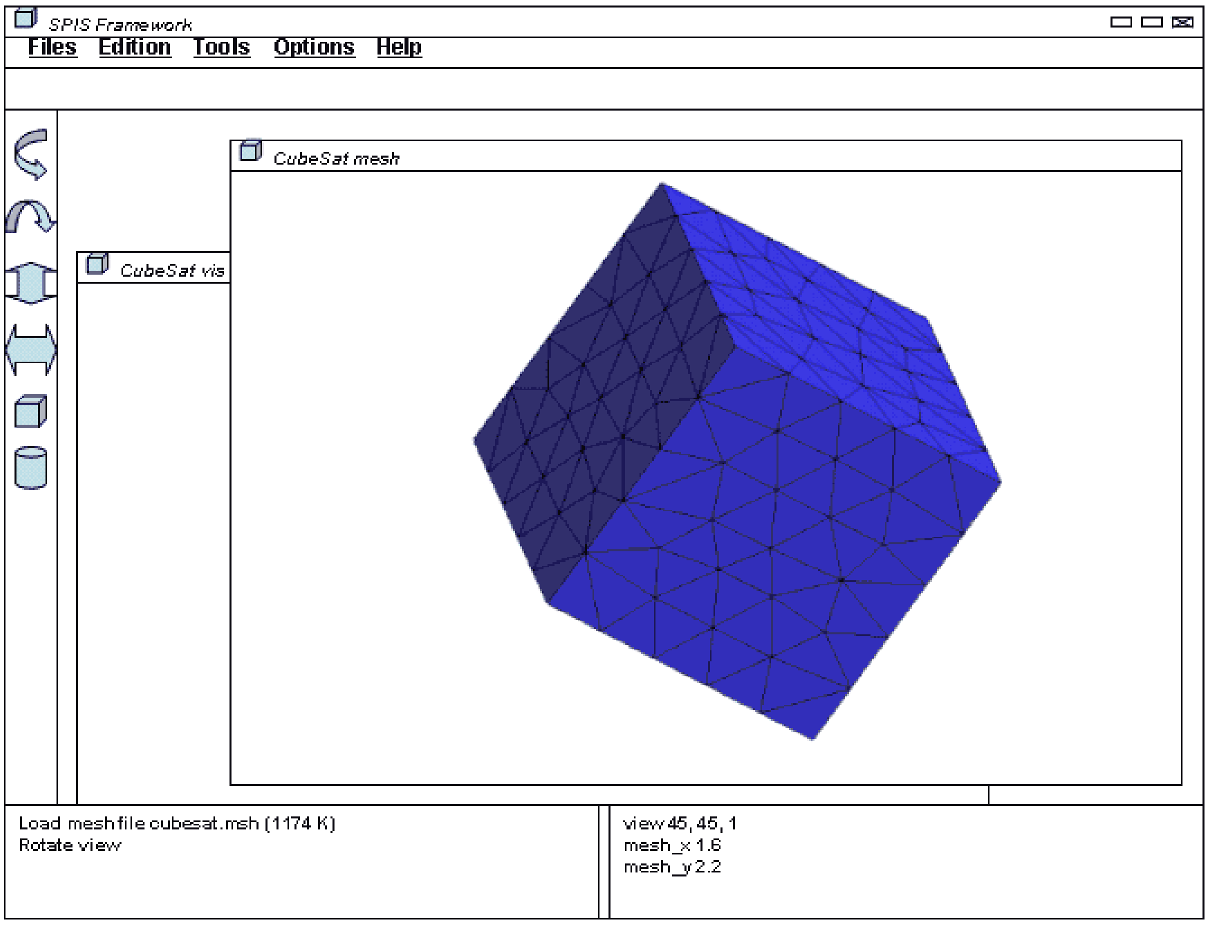
7.1. UR4.1 Graphical interface for spacecraft model building

Remark : The Graphics User Interface (GUI) should be designed with an open architecture in order to ease its improvement with the feedback of the community along the software development cycle.
(Ref. Extreme Programming and Waterfall design).

SPIS must provide a user-friendly GUI defined as a workspace and allowing the interactive creation of projects.
Components of this workspace are:

- A main frame
- One or several visualisation windows
- A log file window
- A command line window
- A supervision window

Only as a simple and illustrative example see the picture below.



```
Load meshfile cubesat.msh (1174 K)
Rotate view
```

```
view 45, 45, 1
mesh_x 1.6
mesh_y 2.2
```

UR4.1.1 The main frame

The main frame is the delimitation of the SPIS framework.
It is composed by:

- A menu bar
- Tool bar
- Buttons

UR4.1.2 Visualization windows

There are several visualization windows contained in the workspace as a MDI (Multiple Document Interface). They allow us to work with multiple views at the same time making it easier to review or compare multiple items.

Visualisation windows are:

- Geometry window
- Mesh window
- Physical properties window
- Post-processing window

. UR4.1.2.100 The geometry window

- UR4.1.2.110 The *geometry window* shall display the whole geometry or a part of the geometry.
- UR4.1.2.120The *geometry window* must allow the user to enter/delete entities (points, lines, surfaces, volumes) interactively.
- UR4.1.2.130 In case of the geometry is composed by several part, they must be displayed in different colours.
- *UR4.1.2.140 If the development of geometry is shared, it shall be designed in collaborative manner.
- *UR4.1.2.150 Several standards primitive entities must be available (point, line, arc, rectangle, box, cylinder, sphere) in a toolbox.
- *UR4.1.2.160 Boolean operations must be available on primitive entities.
- UR4.1.2.170 The geometry shall be imported via file selection box.

. UR4.1.2.200 The mesh window

- UR4.1.2.210 The *mesh window* shall display the whole mesh or a part of the mesh.
- *UR4.1.2.220 The *mesh window* must allow the user to enter additional nodes interactively.
- UR4.1.2.230 Elements (triangle or tetrahedral) should be identified by number.
- UR4.1.2.240 In case of several meshes, they must be displayed in different colours.
- UR4.1.2.250 The *mesh window* must allow the user to locally refine or coarsen a mesh.

. UR4.1.2.300 The material property window

- UR4.1.2.310 The *material property window* must allow the user to enter data related to materials.

- UR4.1.2.320 The materials properties shall be entered interactively via form fields.
- UR4.1.2.330 The materials properties shall be imported via file selection box.

. UR4.1.2.400 The Post-processing window

- UR4.1.2.410 The *post-processing window* must allow the user to see results of his simulation.
- UR4.1.2.420 Those results shall be displayed after the simulation
- *UR4.1.2.430 Those results must be displayed during the simulation.
- UR4.1.2.440 The user must be able to choose the results to be displayed via form fields.
- UR4.1.2.450 The geometry must be visualized in any direction (X,Y,Z).

. UR4.1.2.500 The log file window

. The *log file window* must provide status messages, including error identification, warning and status. Such diagnostics must be both understandable and useful. Where appropriate, such messages should be logged (in a file). It displays information about the occurred session of a project.

. UR4.1.2.600 The command line window

The *command line window* must allow the user to enter commands in order to modify or to provide information or new data. It is based on a scripting language easy to learn by the user.

**UR4.1.2.700 The supervision window (supervisor)
The *supervision window* must allow the user to start/stop and monitor his simulation.

- UR4.1.2.710 The supervisor must be able to display information during the simulation

- UR4.1.2.710 The supervisor must allow the user to choose interactively what parameters should be written in output files.

- UR4.1.2.710 The supervisor must allow the user to choose interactively what output files should be written.

- UR4.1.2.710 The supervisor shall store user preferences about the configuration of the user simulation.

- UR4.1.2.710 The user must be able to quit the supervisor without stopping the simulation.

- *UR4.1.2.710 The user must be able to monitor an ongoing simulation by starting the supervisor.

- **UR4.1.2.710 The user must be able to define the start of his simulation by entering the time parameters (DD:HH:MM:SS) in form fields.

- **UR4.1.2.710 The user must be able to monitor multiple tasks

4.1.3 GUI toolkit

UR4.1.3.100 The toolkit used to develop the SPIS GUI must be portable for Linux and Windows 2000 platforms, and *Unix Mac OS.

7.2. UR4.2 Geometry definition

UR4.2.1 The software must allow several means to insert geometry data.

- UR4.2.1.100 The user must be able to define geometry interactively by using mouse or form fields.
- */**UR4.2.1.200 The user must be able to import geometry in SPIS in standard formats commonly used in free or proprietary software (IGES, DXF, OBJ and VRML 97).
- UR4.2.1.300 The user must be able to edit and modify geometry.
- UR4.2.1.400 The imported geometry file must be backed up before any modification.
- UR4.2.1.500 The geometry must be visualized in any direction (X,Y,Z).

7.3. UR4.3 Mesh definition

UR4.3.1 The software must allow several means to insert mesh data.

- UR4.3.1.100 The user must be able to define mesh interactively by using mouse.
- */**UR4.3.1.200 The user must be able to import mesh in SPIS in standard formats (MED, GTS, VTK, UNV, PLOT3D).
- UR4.3.1.300 The imported mesh file must be backed up before any modification.
- UR4.3.1.400 The mesh must be visualized in any direction (X,Y,Z)
- UR4.3.1.500 Mesh refinement

7.4. UR4.4 Material properties definition

UR4.4.1 The software must allow several means to define material properties.

- UR4.4.1.100 The user must be able to define material properties interactively by entering data in form fields.
- UR4.4.1.200 The user must be able to define material properties by importing a material property file defined as a structured file (a database).
- *UR4.4.1.300The software must contain a minimal database of common materials used in spacecraft charging as defined in the current version of NASCAP for instance.
- *UR4.4.1.400 The user must be able to edit and add/delete materials in the database.
- *UR4.4.1.500 The user must be able to reload the database.
- UR4.4.1.600 Materials must be defined by their properties (density, temperature, etc ….).
- UR4.4.1.700 The user must be able to assign materials to each entities defined by the meshed geometry.
- UR4.4.1.800 Each material must be displayed with different colours.
- **UR4.4.1.900 Possibility of graphical display of some properties, as e.g. secondary emission yield ($Y = f(E)$)

7.5. UR4.5 Interface for Spacecraft Circuit handling

UR4.5.100 The software must allow the user to edit the spacecraft equivalent circuit:
- Text display of electric nodes and components (capacitors, resistors, etc.)
- Capability to modify them (circuit and values of *R*, *C*, etc.)

**UR4.5.200 A graphical interface for editing the spacecraft equivalent circuit may be interfaced with the software:
- Graphical display of electric nodes and components (there is usually a lot of nodes, so some advanced capability of grouping them, or equivalent method, would be welcome)
- Capability to modify them through the GUI
- Enter coupling matrices

7.6. UR4.6 Command line interface for simulation control

UR4.6.100: A proper simulation control must be offered
- The control of the simulation in command line must be eased by a scripting language.
- The user must be able to monitor the simulation in command line.
- The user must be able to display the status of the simulation in command line.
- The user must be able to start/stop the simulation in command line.
- The user must be able to provide parameters to the simulation in command line.
- The user must be able to define the start of his simulation by entering the time parameters (DD:HH:MM:SS) in command line.
- The user must be able to restart the simulation from a stopping point.
- Simple test procedure must be available prior to launch long run to be sure that there is no simple mistakes with initial set-up of launch and simulation parameters

8. UR5: Requirements originating from software development process

8.1. UR5.1 Computing Structure and Method of Development

UR5.1.1 Object Oriented rules

UR5.1.1.100 It is recommended to use the Object Oriented Approach and rules of development (number of line for a subroutine, number of object in a class, etc ….)
UR5.1.1.200 A naming convention should be defined and respected for the definition of each objects and method.

UR5.1.2 Open Source and community considerations

UR5.1.2.100 It is explicitly required that the final product will be released under an Open Source licence. All source code will be accessible by all potential users.
UR5.1.2.200 The language of development must be at least available as Open Source (compiler, scripting language, etc ….).
UR5.1.2.300 The software must be released in two forms: source code and executable.
UR5.1.2.400 The software should be designed and provided to be developed in the context of a network of developers and users or in a community development based approach.

8.2. UR5.2 Platform Operational Requirements

UR5.2.1 The SPIS framework must be portable to all platforms that run a Linux based operating system and Windows NT/2000 operating system.
UR5.2.2 New versions of the framework must be compatible with the older versions so that a continuous use of the system is guaranteed.

8.3. UR5.3 Documentation requirements

Following [ECSS-E-40], [ITT] and [PROP] specifications, the documentation shall include the following documents.

UR5.3.1 Technical documentation

UR5.3.1.100 A complete technical documentation shall be given with the product.
*UR5.3.1.200 The technical documentation must be provided on-line or in standard file formats (.doc, .pdf, .ps, etc ….)
UR5.3.1.300 All the technical documentation shall be written in English.

UR5.3.2 User Manual (UM)

UR5.3.2.100 This document shall give a complete explanation of utilisation of the software in the standard configuration.
UR5.3.2.200 The UM shall include a general description of the code structure and possibilities.
UR5.3.2.300 The UM shall include a brief tutorial, with a complete example of demonstration, including real model data, input files, step-by-step simulation, output results and corresponding analysis.
UR5.3.2.400 The UM shall include a complete description of input and output file formats, if they are not standard.

UR5.3.3 Developer Manual (DM)

UR5.3.3.100 The DM shall give the detailed structure, complete description and the limitations of the code. This shall be done for the general structure of the library and for each module or sub-routine.
UR5.3.3.200 The DM should include a complete description of methods and subroutines, including a brief summary of its function and a description of input and output parameters. A link with theoretical documentation shall be done if the subroutine or method uses a mathematical model or a specific method. The DM should provide a complete description of all input and output files.
UR5.3.3.300 The DM should contain many basic working examples. In particular, it should be provided an example able to perform a complete simulation in its simplest form.
UR5.3.3.400 A structured map of compatibility between each module, subroutine or method should be provided. In particular, this map should contain the computing compatibility, but also the numerical compatibility between various used numerical methods. In case of object oriented approach, a tree of heritage is strongly recommended.
UR5.3.3.500 The DM should give the complete numerical and physical limits, the criterion of stability and level of accuracy (i.e. order of the used scheme) for each numerical subroutine and method. If it is not possible to give this information for a specific method, such as Monte-Carlo method, examples and results of validations tests should be given.
UR5.3.3.600 The DM should include documentation about the API for the development of plug-ins (cf. UR3.1.7).
UR5.3.3.700 It is strongly recommended to generate automatically the API documentation from comments in source codes using a similar approach to JavaDoc or DOxygen software.

UR5.3.4 Other useful documentation

UR5.3.4.100 A detailed Theoretical Reference Documentation (TRD) regarding the used models and numerical methods, if there are not standard, shall be provided.
UR5.3.4.200 An *How to* document, presenting classic difficulties of installation or use, and the corresponding solutions should be provided.
UR5.3.4.300 A *Frequently Asked Questions* (FAQ) list should be given with the documentation.

Annex A: State of the art of Spacecraft-plasma models

The major codes devoted to plasma dynamics modelling are briefly described here. This can be considered as an update of the previous states of the art performed in the framework of IPICSS and above all SpaceGRID studies (Spacecraft Plasma Interactions Software Analysis Technical Note SGD-SPI-QIN-TN-002-1.2, 2/05/2002). With respect to this previous study, the present description of the codes is a little more detailed, and a few other codes were found, although not of prime interest for our application.

Name	Geometry	Physics/methods	Comment	Availability
XPDP1, XPDC1, XPDS1, XPDP2	- Space: 1D bounded, planar (XPDP1), cylindrical (XPDC1) or spherical (XPDS1), or 2D (XPDP2) - Velocity: 3D	- Electrostatic (PIC) - Collision with neutrals (MCC) - External RLC circuit	- Devoted to plasma device modelling (bounded + circuit) - GUI	Free Berkeley http://ptsg.eecs.berkeley.edu/ (older not well maintained codes from Berkeley were discarded) Same
XOOPIC	- Space: 2D bounded - Velocity: 3D	- Electrostatic or electromagnetic - Object Oriented (C++) - Parallelised (MPI)		
ES3D	3D periodic boundaries	- Electrostatic (PIC, FFT for Poisson) - Vectorised and parallel versions	- limited graphical interface - available version devoted to beam-plasma instability modelling (customisation needed)	Free, NASA GSFC http://ess.gsfc.nasa.gov/inhouse-sw.html http://ess.gsfc.nasa.gov/exchange/contrib/macneice/pic-es3d.html http://ess.gsfc.nasa.gov/exchange/contrib/macneice/pic-tristan.html
TRISTAN	3D	- Electromagnetic PIC - Vectorised and parallel versions	No embedded object	
QUICKSILVER	3D various boundary conditions	- Electromagnetic PIC - Vectorized, parallelized	Devoted to plasma device modelling	Commercial, Sandia (DOE) http://www.sandia.gov/pulspowr/ppeng/qs.html
Lsp	3D	- Electromagnetic PIC - Parallel (MPI) - Some extra physics (field emission, , secondary electron	Devoted to plasma device modelling, in particular interaction with light	Commercial (50 k$ the first year, then 15 or 7 k$) Mision Research Corporation, Albuguerque http://www.mrcabq.com/Services/Particle/Codes/codes.htm

		generation, collisions…)		
MAGIC2D	2D	- electromagnetic PIC		Commercial (9.5 k$ per year) Free demonstration version Mision Research Corporation, Washington http://www.mrcwdc.com/Magic/description.htm
MAGIC3D	3D			
Par-T	3D	- electromagnetic PIC - parallel (MPI)		Not available http://www.astro.phys.ethz.ch/staff/messmer/private/par-T
KEMPO-3D	3D	electromagnetic		Not available Kyoto university http://www.kurasc.kyoto-u.ac.jp/plasma-group/simulation/diele/abstract-e.htm
GCPIC	3D	- electromagnetic PIC - parallel (GCPIC General Concurrent PIC algorithm)		Not available Syracuse University http://www.npac.syr.edu/copywrite/pcw/node177.html
PicUp3D	3D	- electrostatic PIC - some more physics: photo-emission… - floating potential computation	Devoted to spacecraft-plasma interactions modelling (open box boundaries + spacecraft)	Free, ESA http://www.spis.org/spine/picup3d/index.html
PICCHARGE	2D space 3D velocity	electrostatic		Not available (TBC), ESA http://www.spenvis.oma.be/spenvis/help/background/charging/charging.html#PICCHARGE
SILECS	3D	- electrostatic PIC -	Devoted to spacecraft-plasma interactions modelling	Not available, ONERA http://www.estec.esa.nl/wmwww/wma/R_and_D/Spine/SpineM0/pres_jfr2.pdf
NASCAP, NASCAP/LEO, POLAR, NASCAP-2k	3D	- electrostatic, PIC or semi-analytic - multi-grid	“Reference” American codes devoted to spacecraft charging in GEO or LEO	Commercial, most recent codes export restricted, NASA or USAF http://powerweb.grc.nasa.gov/pvsee/software/NASCAP.html http://powerweb.grc.nasa.gov/pvsee/software/nascapleo.html

				http://see.msfc.nasa.gov/ee/model_nascap.html
OSIRIS	3D	- electromagnetic - parallel	Object oriented in FORTRAN 90	Uncertain availability, various universities (Insituto Superior Tecnico, Portugal; UCLA, USA; Ecole Polytechnique, France) http://unseelie.lbl.gov/~ksahearn/ICCS_hpcpast/abstracts/fonseca_abs.pdf
Various codes for particle accelerators modelling		- electromagnetic	Several codes, including 1 or 2 already listed: IMPACT, IMPACT/Marylie, Omega3P, OSIRIS, QuickPIC, Tau3P, VORPAL	Uncertain availability, summary at: http://scidac.nersc.gov/accelerator/software.html http://scidac.nersc.gov/accelerator/
		-		

Annex B: IPICSS URD

The “Investigation of Plasma Induced Charging of Satellite Systems” (IPICSS) Project was conducted by Julien Forest within SPINE community. A User Requirement Document was written in that framework, and its last issue (v1.4 from October 200) is reproduced here as an annex.

The general context of the spacecraft-plasma interactions is very well documented in IPICSS URD, and was thus only sketched in the present SPIS URD. It was much more important to completely reprocess the detail requirements in the new framework of the SPIS software development. The requirements sections of SPIS URD (sections 3 through 8) are thus completely new, with a different organisation. The IPICSS detail requirements are yet reproduced in this annex for reference, since most of their substance must indeed be found in the new SPIS requirements.