RADDOSE-3D: Time and Space Resolved Modelling of Dose in Macromolecular Crystallography

Oliver B. Zeldin, Markus Gerstel and Elspeth F. Garman

Journal of Applied Crystallography (2013), Volume 46, Part 4

Supplementary Material

06.06.2013, describing RADDOSE-3D version 1.0.950

Contents

1 Supplemental Figures

1.1 Figure S1 – RADDOSE-3D object structure overview

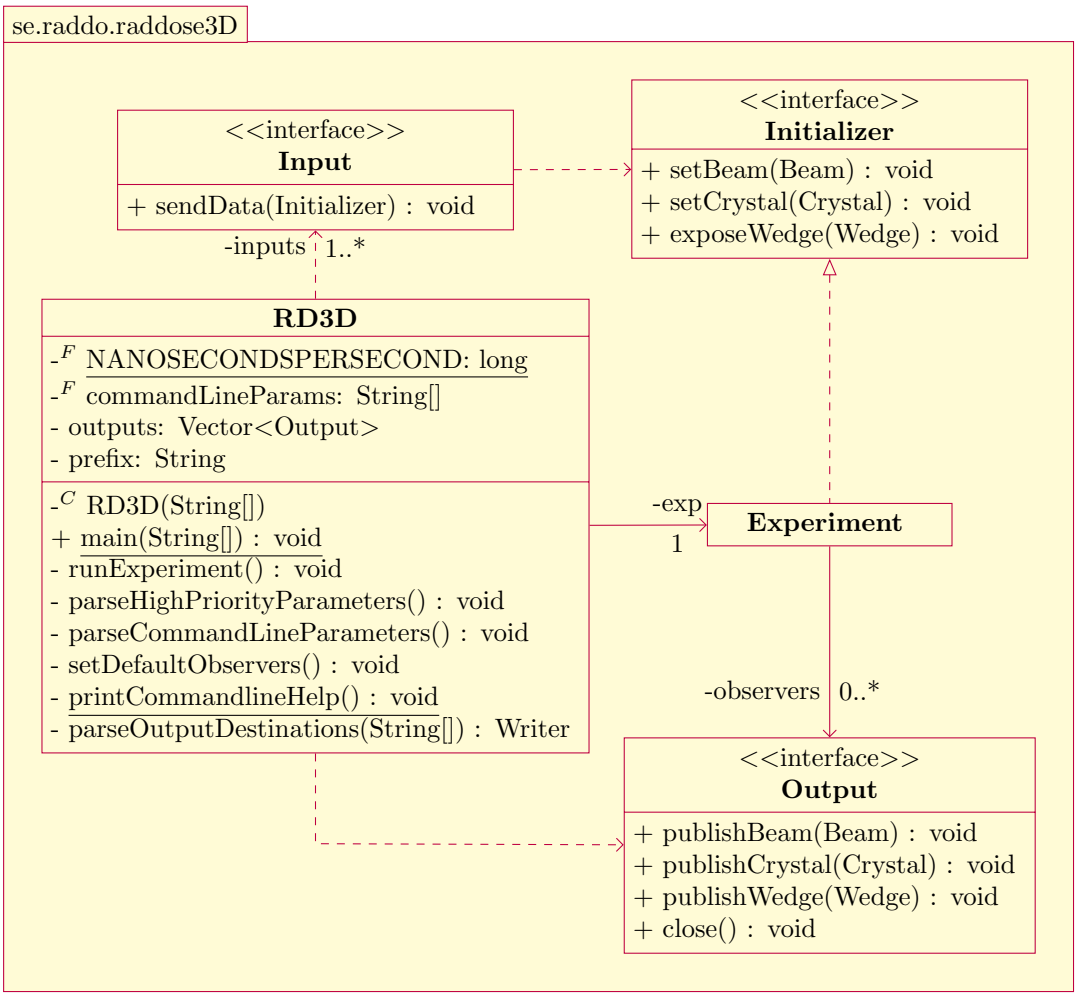

Figure S1: Unified Modeling Language (UML) class diagram of the `RADDOSE-3D` object structure. The main function of `RD3D` is invoked from the command line. Depending on specified command line parameters, `RD3D` creates an `Input` class (Fig. S2), an instance of the `Experiment` class and a number of `Output` classes (Fig. S6) upon launch. The `Output` classes are registered with the `Experiment` class so they can observe the experiment progression (Fig. S5). Once all these classes are set up, `RD3D` triggers the `Input` class to parse the input file and create the relevant `Crystal` (Fig. S3), `Beam` (Fig. S4) and `Wedge` objects, and then to pass these objects to the `Experiment` instance via the `Initializer` interface.

1.2 Figure S2 – RADDOSE-3D processing user input

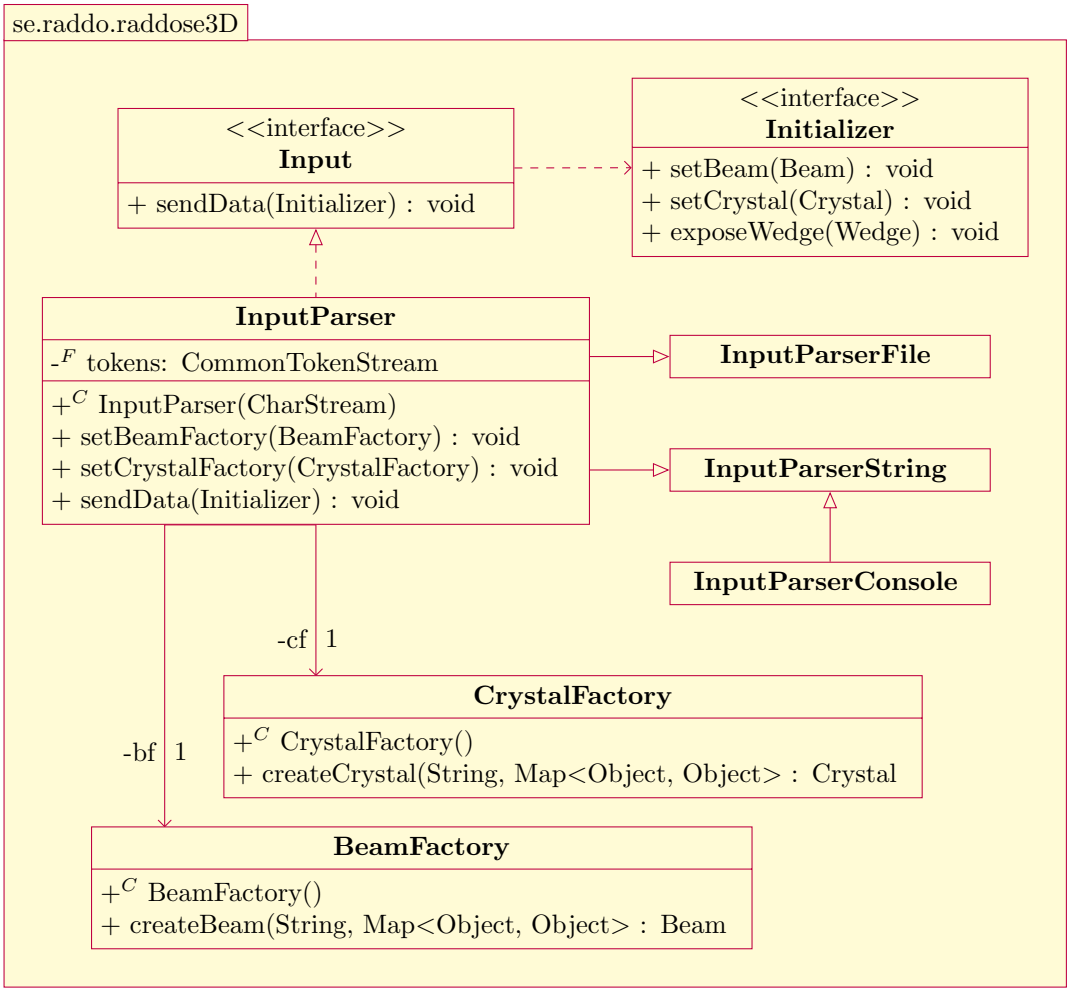

Figure S2: UML class diagram. User input is processed by implementations of the `Input` interface. `RADDOSE-3D` contains four such implementations, all belonging to the group around `InputParser`. In these classes, user input is processed using a parser generated by ANTLR, which understands the syntax detailed in Section 2. The four classes differ in the accepted source of the user input: while `InputParser` itself only accepts a high-level ANTLR token stream, the class `InputParserFile` reads from a file, `InputParserConsole` reads from STDIN and `InputParserString` accepts any `String` object. Internally, the desired property of crystals and beams are collected in Java `Map` data structures, and the creation of the actual `Crystal` and `Beam` classes is delegated to `CrystalFactory` and `BeamFactory` via a flexible, non-specific interface detailed in Figs. S3 and S4. The objects thus produced are handed to the `Initializer` implementation which contains the main program logic.

1.3 Figure S3 – Supporting multiple `Crystal` classes

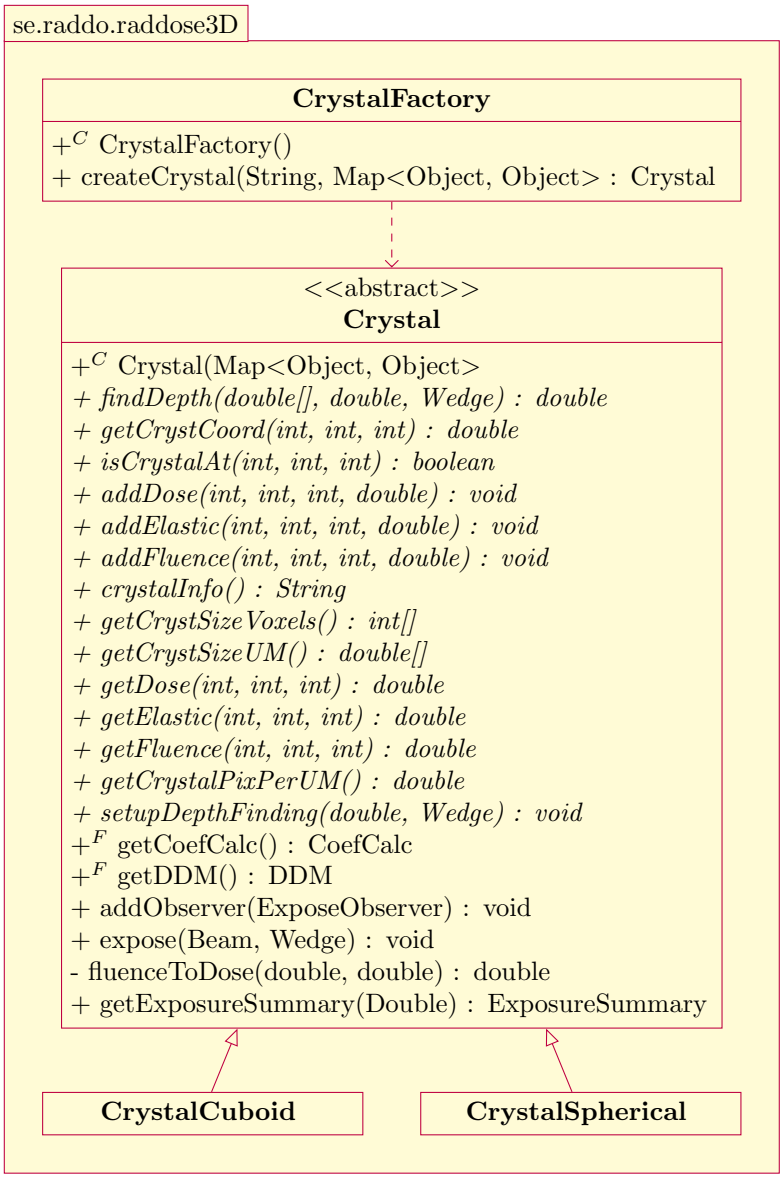

Figure S3: UML class diagram. The abstract class `Crystal` defines a number of common functions that all actual subclasses have to contain. `RADDOSE-3D` contains two subclasses, `CrystalSpherical` and `CrystalCuboid`. The choice of which of these two subclasses is actually used during a run of `RADDOSE-3D` is decided in the class `CrystalFactory`, which receives the parameters from the parser. Subclass implementations are interchangeable: new `Crystal` subclasses can be added easily, and will work with the `RADDOSE-3D` framework and all `Beam` implementations as long as the new `Crystal` subclass contains all the functions that a `Crystal` class requires.

1.4 Figure S4 – Supporting multiple `Beam` classes

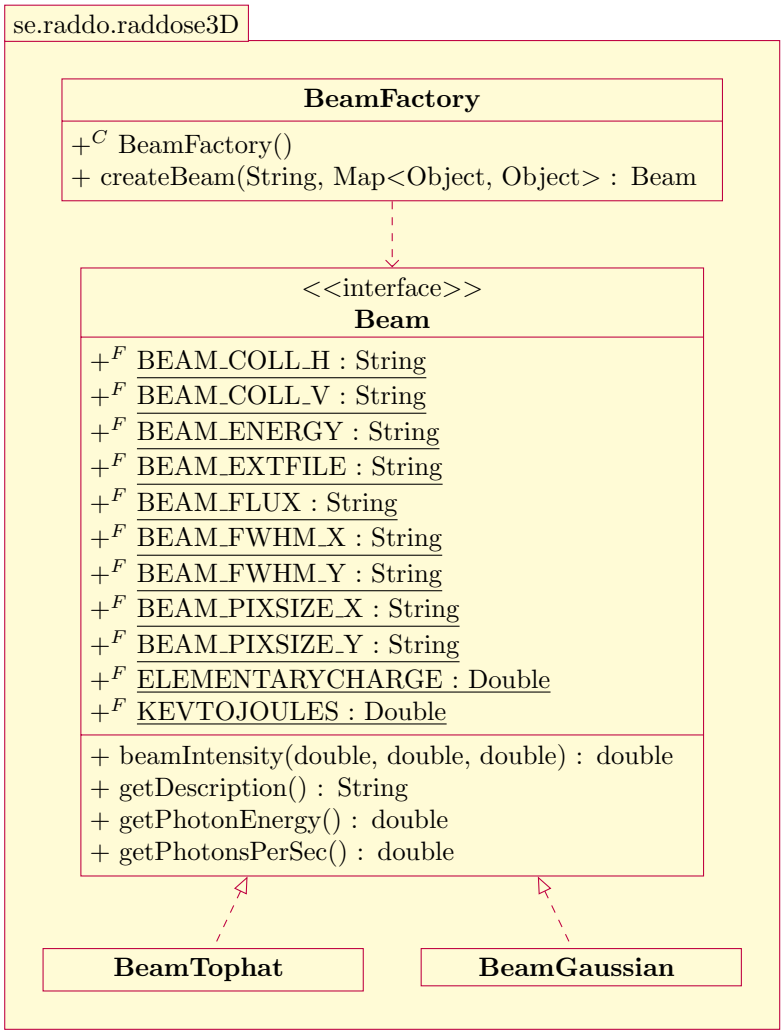

Figure S4: UML class diagram. The interface `Beam` defines a number of common functions that all actual implementations have to contain. `RADDOSE-3D` contains two implementations, `BeamTopHat` and `BeamGaussian`. The choice of which of these two classes is actually used during a run of `RADDOSE-3D` is decided in the class `BeamFactory`, which receives the parameters from the parser. Implementations are interchangeable: new `Beam` classes can be added easily, and will work with the `RADDOSE-3D` framework and all `Crystal` subclasses, as long as the new `Beam` class correctly implements all functions that are required for a `Beam`.

1.5 Figure S5 – Observing the Experiment

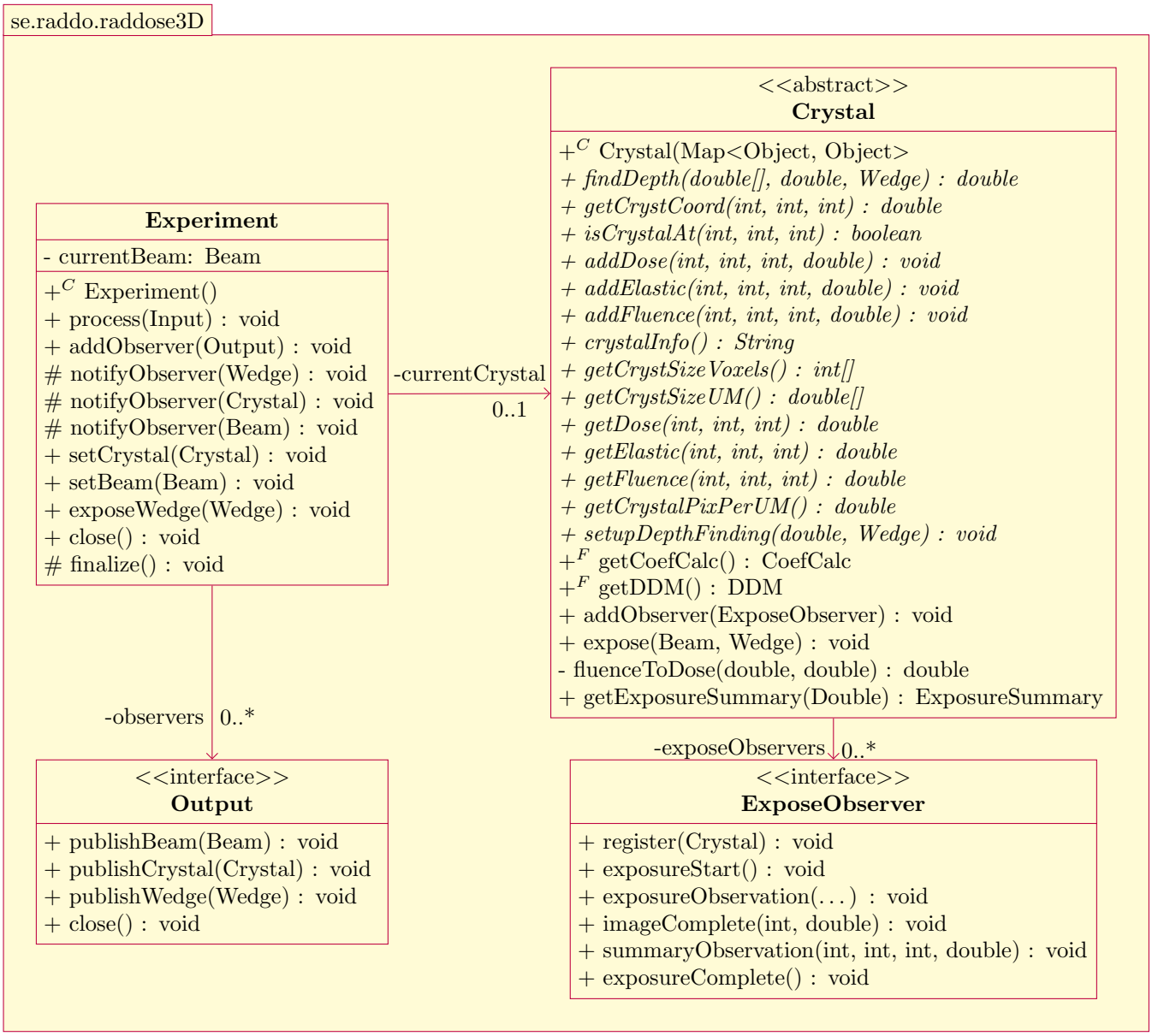

Figure S5: UML class diagram. Classes that implement the `Output` interface can subscribe to an `Experiment` by calling its *addObserver()* method. The `Experiment` class will inform all subscribed objects when a new `Crystal` or `Beam` is passed to it, as well as after each `Wedge` exposure. The final call the observing classes receive is the *close()* method, which commands the `Output` class to flush its buffers and close any open file handlers or equivalents. When a more detailed look into the exposure process is required, an implementation of the `ExposeObserver` class can subscribe to a `Crystal` class. The `Crystal` will inform any subscribed `ExposeObserver` of each individual voxel exposure as well as after each image and at the end of the exposure wedge. The registration of the `ExposeObserver` class to the `Crystal` would usually be initiated in the *publishCrystal()* method of an `Output` class. This `Output` class would then keep a reference to the `ExposeObserver` object and inspect it after a `Wedge` exposure (*publishWedge()*) or during the call to its *close()* method. These object interactions follow the Observer pattern described by Gamma et al. (1994).

[Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: Elements of Reusable Object-Oriented Software. Reading, MA, USA. Addison Wesley.]

1.6 Figure S6 – RADDOSE-3D generating output

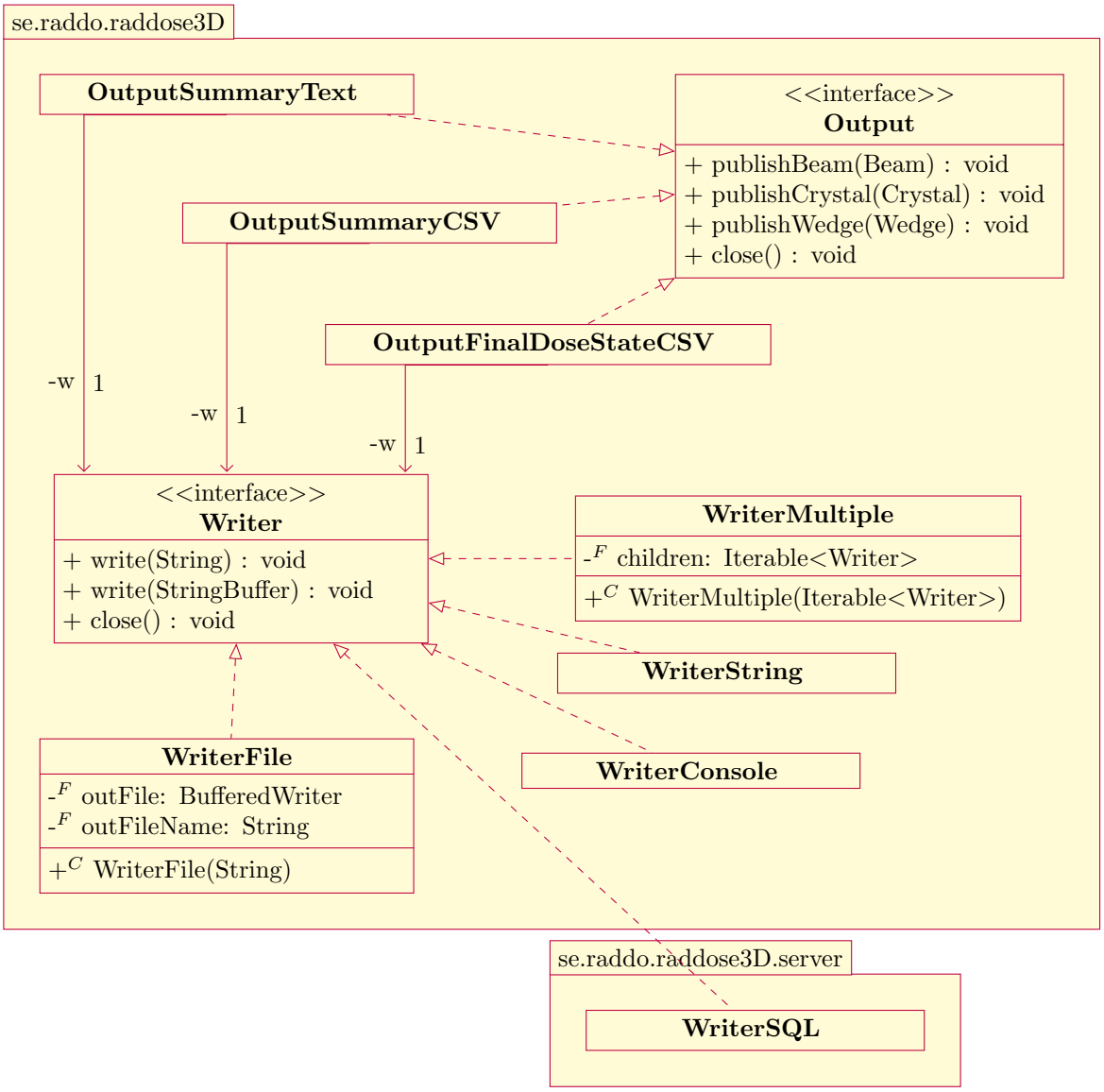

Figure S6: UML class diagram. Output for the user is created in implementations of the `Output` interface. Each implementing class is notified when a new crystal is set up, a beam is defined and after each wedge exposure. Each of the three shown implementations produces different output based on the same simulation. Each `Output` class references a single `Writer` class. This `Writer` class takes care about the actual I/O processes required for presenting the output to the user. Different `Writer` classes allow writing to a file, to the console, or a combination of these (via `WriterMultiple`). A future writer implementation could for example write to a specific field in a graphical user interface. The publically available webservice of `RADDOSE-3D` uses the class `WriterSQL` to write the results directly into an SQL database.

1.7 Figure S7 – Voxel resolution as a function of beam FWHM

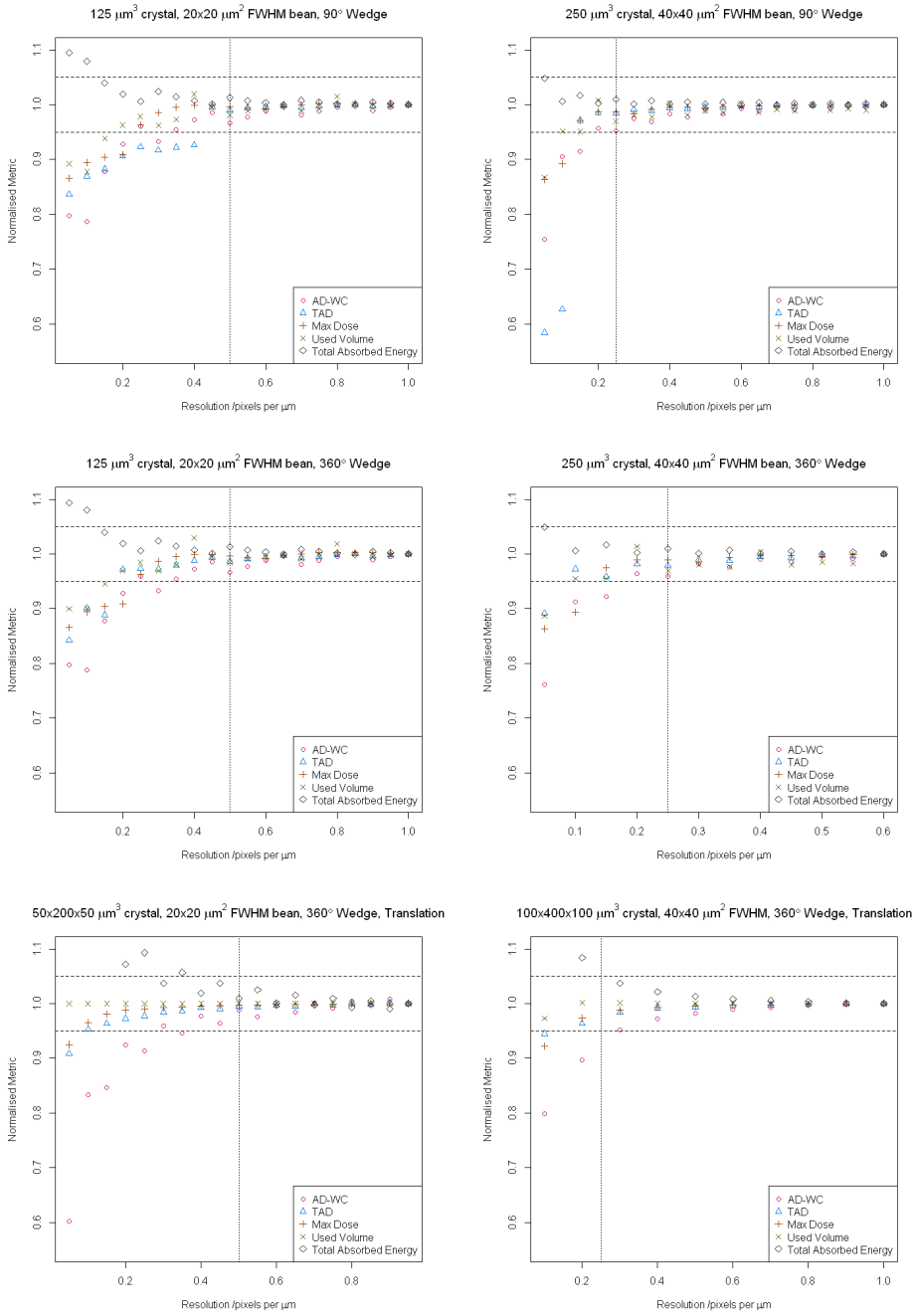

Figure S7: Voxel resolution testing for large (up to 15×10^6 voxels for the $250^3 \mu m^3$ crystal) and smaller crystals, with 20×20 and $40 \times 40 \mu m^2$ FWHM beams (left and right columns). The top row is for a 90° rotation, the middle row is for 360°, and the bottom row is also for 360°, but with a full translation. Note that voxel resolution is quoted in voxels per micron. In all cases, the simulations converge towards their high voxel resolution values at around $1/10^{th}$ of the beam FWHM, shown by the vertical dashed line (0.25 vox/μm for the larger beam, and 0.5 vox/μm for the smaller beam). Horizontal lines have been added at ±5% of the converged values for visual clarity.

2 RADDOSE-3D command reference

RADDOSE-3D can take input from one or more files and/or from standard input (STDIN). Any input will be processed by the InputParser class and the RADDOSE-3D ANTLR parser. This section describes the syntax of accepted input. Advanced users of RADDOSE-3D can create their own input method that need not rely on the InputParser class or the RADDOSE-3D ANTLR parser. This feature will not be covered in this reference.

The simplest use case of RADDOSE-3D will involve only one file describing the entire experiment. In some instances it may be desired to split up the input into a number of files, e.g. one file describing the crystal, one automatically updated file describing the current beam on the beamline, and one file chosen from a set of possible wedge strategies. Each file can contain an arbitrary number (including none) of Crystal, Beam and Wedge block (henceforth called *blocks*). However, splitting up blocks across multiple files is not allowed.

The parser will read the input sequentially, and, when multiple sources are given, one source after the other in the specified order. While the parser may accept Crystal, Beam and Wedge blocks in any order, the exposure of a wedge can only take place if both the crystal and the beam have been set either in an earlier file or before the Wedge block within the same file.

2.1 General syntax considerations

Any keywords specified below are case-insensitive. Upper (**CRYSTAL**), lower (**crystal**) and mixed case (**CrYsTaL**) are equivalent.

The characters **#**, **!** and the character sequence **//** denote the start of a comment. Any text from that position until the end of the current line is ignored.

Tabular and newline characters are treated as white space. They can therefore by freely used to format the file for increased readability.

The order of statements within a Crystal, Beam and Wedge block generally is not relevant. There are two exceptions to this rule: The leading keyword (**CRYSTAL**, **BEAM**, **WEDGE**) **must** be the first keyword of the block. If a keyword is repeated within the same block, then the latter will always override the former.

Every block must be self-contained, e.g. the energy set for the previous Beam is not remembered when setting up the following Beam, and must be repeated.

Numeric values can be given in scientific notation ($2.0\text{e}2 = 2e{+}2 = 200$), negative values may not have a space between the sign ('-') and the value ($-1.9\text{e}-1 = -.19 = -0.19$).

2.2 Crystal block

A Crystal block must begin with the keyword **CRYSTAL**. At least the **TYPE** and **DIMENSION** must be specified. Depending on the chosen **TYPE** further declarations may be required.

2.2.1 TYPE

With the keyword **TYPE** the underlying crystal implementation is chosen. Currently two distinct crystal implementations exist:

TYPE CUBOID defines a solid crystal with a cuboid shape.

TYPE SPHERICAL defines a solid crystal with a spherical shape.

2.2.2 DIMENSION

DIMENSION specifies the size of the crystal. Dimensions are given in micrometres (µm). The keyword **DIMENSION** can take either one or three parameters:

DIMENSION *D* with a single number (see section 2.1) as parameter is used for specifying the crystal dimensions for spherical crystals. The parameter sets the crystal diameter. This syntax cannot be used for cuboid crystals.

DIMENSION *X Y Z* with three numbers as parameters X, Y and Z is used to set the dimensions for cuboid crystals (**TYPE CUBOID**). X defines the length of the crystal orthogonal to both the beam and the goniometer at L=P=0, (see below) Y defines the length along the goniometer axis at L=P=0 and Z defines the length along the beam axis.

If three parameters are given for a spherical crystal (**TYPE SPHERICAL**) the value for X sets the diameter of the crystal while the values of Y and Z are ignored.

2.2.3 PIXELSPERMICRON

PIXELSPERMICRON *F* specifies the resolution of the voxel grid used to represent the crystal in voxels/µm. Defaults to 0.5 voxels/µm.

2.2.4 ANGLEP

ANGLEP *F* sets the angle in the plane of the loop between the crystal Y axis and the goniometer axis. The angle is to be given in degrees, but without the degree symbol (°). The default P ('plane') angle is 0°.

The rotation angle to be applied to the crystal in the plane of the loop (right handed rotation about Z axis applied to all voxels, as shown in figure S8).

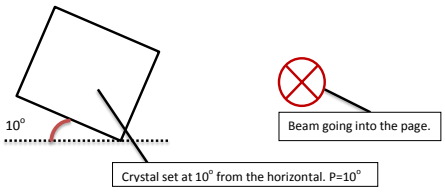

Figure S8: Schematic of **ANGLEP**. Figure courtesy of John Bremridge.

2.2.5 ANGLEL

ANGLEL *F* sets the loop angle between the plane of the crystal loop and the goniometer axis. The angle is to be given in degrees, but without the degree symbol (°). The default L ('loop') angle is 0°.

The rotation angle to be applied to the angle of the crystal in the loop (right handed rotation about X axis applied to all voxels, as shown in figure S9).

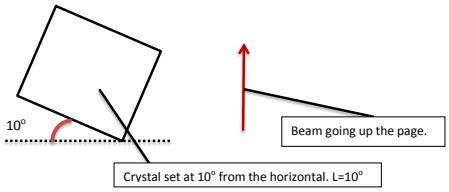

Figure S9: Schematic of **ANGLEL**. Figure courtesy of John Bremridge.

2.2.6 ABSCOEFCALC

This keyword specifies whether the program should use average absorption and attenuation coefficients, or whether it should calculate them from input crystal parameters.

ABSCOEFCALC AVERAGE
ABSCOEFCALC DUMMY

These two commands are equivalent. Each will cause RADDOSE-3D to assume an absorption coefficient of 0.237 mm^{-1} and an attenuation coefficient of 0.281 mm^{-1}. These values are representative of an average crystal at an incident X-ray beam energy of 12.4 keV (1Å). Please see Section 3 in the main paper for more details. Crystal composition keywords will have no effect.

ABSCOEFCALC RD
ABSCOEFCALC RDV2
ABSCOEFCALC RDV3

These three commands are equivalent. RADDOSE-3D will call a previous version of RADDOSE to estimate absorption and attenuation coefficients.

The composition of the crystal has to be described using the keywords **UNITCELL**, **NUMMONOMERS**, **NUMRESIDUES**, **NUMRNA**, **NUMDNA**, **PROTEINHEAVYATOMS**, **SOLVENTHEAVYCONC** and **SOLVENTFRACTION**. The use of these keywords is described in the sections 2.2.7–2.2.14 below.

2.2.7 UNITCELL

This keyword only has an effect when the absorption and attenuation coefficients are estimated using a legacy version of RADDOSE (see section 2.2.6).

UNITCELL *A B C*
UNITCELL *A B C* α β γ

Dimensions and angles of the unit cell a, b, c, α, β, γ

The first three numbers specify the unit cell size in Angstroms. The second three numbers optionally specify the unit cell angles alpha, beta and gamma.

The (optional) angles are to be given in degrees, but without the degree symbol (°). If no angles are specified RADDOSE-3D assumes default angles of 90°.

2.2.8 NUMMONOMERS

This keyword only has an effect when the absorption and attenuation coefficients are estimated using a legacy version of RADDOSE (see section 2.2.6).

NUMMONOMERS *I* specifies the number of monomers in the unit cell. Only integer numbers I should be used. This number should not be confused with the number of monomers in the asymmetric unit.

2.2.9 NUMRESIDUES

This keyword only has an effect when the absorption and attenuation coefficients are estimated using a legacy version of RADDOSE (see section 2.2.6).

NUMRESIDUES *I* specifies the number of amino acid residues per monomer. Only integer numbers I should be used. Using this keyword the number and types of atoms are calculated according to the formula

$$\text{amino acid} = 5C + 1.35N + 1.5O + 8H$$

Sulfur atoms, e.g. from CYS and MET residues, should be added explicitly with the **PROTEINHEAVYATOMS** keyword.

The default value for I is 0.

2.2.10 NUMRNA

This keyword only has an effect when the absorption and attenuation coefficients are estimated using a legacy version of RADDOSE (see section 2.2.6).

NUMRNA *I* specifies the number of RNA nucleotides per monomer. Only integer numbers I should be used. Using this keyword the number and types of atoms are calculated assuming an average nucleotide content defined as

$$\text{mean nucleotide} = 11.25H + 9.5C + 3.75N + 7O + 1P$$

If a more accurate estimate is required, individual atoms may be entered explicitly with the **PROTEINHEAVYATOMS** keyword.

The default value for I is 0.

2.2.11 NUMDNA

This keyword only has an effect when the absorption and attenuation coefficients are estimated using a legacy version of RADDOSE (see section 2.2.6).

NUMDNA *I*

specifies the number of DNA deoxynucleotides per monomer. Only integer numbers I should be used. Using this keyword the number and types of atoms are calculated assuming an average deoxynucleotide content defined as

$$\text{mean nucleotide} = 11.75H + 9.75C + 4N + 6O + 1P$$

If a more accurate estimate is required, individual atoms may be entered explicitly with the **PROTEINHEAVYATOMS** keyword.

The default value for I is 0.

2.2.12 PROTEINHEAVYATOMS

This keyword only has an effect when the absorption and attenuation coefficients are estimated using a legacy version of RADDOSE (see section 2.2.6).

PROTEINHEAVYATOMS *El I (El I (El I ..))* defines a list of atoms to add to the protein part of the absorption. Each species is defined by a two

character string *El* for the elemental symbol, and an integer number I of atoms of that species per monomer.

The command **PROTEINHEAVYATOMS S 10 Se 2** would add 10 sulfur and 2 selenium atoms per monomer.

2.2.13 SOLVENTHEAVYCONC

This keyword only has an effect when the absorption and attenuation coefficients are estimated using a legacy version of RADDOSE (see section 2.2.6).

SOLVENTHEAVYCONC *El I (El I (El I ..))* defines the concentration of elements (not including water) in the solvent in millimoles per litre. Oxygen and lighter elements should not be specified.

The command **SOLVENTHEAVYCONC Na 1000 Cl 1000** specifies 1M sodium chloride in the solvent.

2.2.14 SOLVENTFRACTION

This keyword only has an effect when the absorption and attenuation coefficients are estimated using a legacy version of RADDOSE (see section 2.2.6).

SOLVENTFRACTION *F*

The fraction of the unit cell that is occupied by solvent. If not given explicitly, this value is estimated from **NUMRESIDUES**, **NUMRNA** and **NUMDNA** using 1.35 g/ml for protein, and 2.0 g/ml for RNA and DNA.

2.3 Beam block

A `Beam` block must begin with the keyword **BEAM**. At least the **TYPE** must be specified. Depending on the chosen **TYPE**, further declarations may be required.

2.3.1 TYPE

With the keyword **TYPE**, the underlying beam implementation is chosen. Currently two distinct beam implementations exist:

TYPE TOPHAT defines a beam with uniform flux.

TYPE GAUSSIAN defines a beam with a 2-dimensional Gaussian flux profile. The full-width half-maximum must be specified with the **FWHM** keyword (see section 2.3.3).

2.3.2 FLUX

FLUX *F* specifies the total beam flux in photons per second. The flux parameter F can be specified in scientific notation (e.g. **1.3e12**).

2.3.3 FWHM

FWHM *X Y*

The FWHM of the beam (vertical), (horizontal). Not needed if a Top-Hat beam is used. This defines the X and Y FWHM of the beam respectively in the RADDOSE coordinate system.

2.3.4 ENERGY

ENERGY ***F***
ENERGY ***F*** **KEV**
specifies the incident photon energy in keV. The optional keyword **KEV** can be appended for human readability of the input file.

2.3.5 COLLIMATION

COLLIMATION RECTANGULAR ***X Y***
specifies the horizontal and vertical collimation of the beam. Delimits where the beam has non-zero intensity. This is defined by the slits. For an uncollimated Gaussian beam, set to $\approx 3\times$ FWHM.

2.4 Wedge block

A `Wedge` block must begin with the keyword **WEDGE**.
WEDGE ***A B***
A and B define the start and end angle of the rotation in degrees (°). At 0° the front face of the crystal (X-Y plane) is normal to the beam. Rotation is right handed about the Y axis, as shown in figure S10).

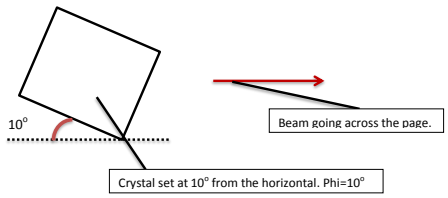

Figure S10: Schematic of angles for **WEDGE**. Figure courtesy of John Bremridge.

2.4.1 EXPOSURETIME

EXPOSURETIME ***F***
specifies the total exposure time for this wedge in seconds.

2.4.2 ANGULARRESOLUTION

ANGULARRESOLUTION ***F***
specifies the angular step size used for wedge iterations in degrees (°). Defaults to 2°.

2.4.3 STARTOFFSET

STARTOFFSET ***X Y Z***
offset translation in µm applied to the crystal relative to the origin (defined as the intersection of the beam and the aligned goniometer axis) for the starting position of the wedge. Defaults to 0 0 0.

2.4.4 TRANSLATEPERDEGREE

TRANSLATEPERDEGREE ***X Y Z***
translation of the goniometer during exposure in µm/° for helical scanning, leading to improvements in dose distribution. Defaults to 0 0 0.

2.4.5 ROTAXBEAMOFFSET

ROTAXBEAMOFFSET *F*

the offset in µm along X (vertical in most set-ups) between the beam axis and the rotation axis. Used to create 'offset' scanning for improvements in dose distribution. Defaults to 0 µm.