Biologically corrected display of isodoses in radiation treatment plans in Eclipse 15.6 – Development of a prototype using ESAPI script

E. Blank1,2), C. Schröder4), M. Grohmann5), M. Krüger1,2), S. Kirschke1,2), P. Grundmann1,2), C. Bergmann1,2), R. Wiener1,3), O. Stoiber1,2), A. Buchali1,2)

*1) Ostprignitz-Ruppiner Gesundheitsdienste GmbH, Neuruppin, Germany*

*2) Medizinische Hochschule Brandenburg, Neuruppin, Germany*

*3) Beuth Hochschule für Technik Berlin, Fachbereich II, Berlin, Germany*

*4) Institut für Radioonkologie, Kantonsspital Winterthur, Winterthur, Swizerland*

*5) Universitätsklinikum Hamburg Eppendorf, Germany*

Corresponding Author:

Dr. rer. nat. Eyck Blank

Ostprignitz-Ruppiner Gesundheitsdienste GmbH

Fehrbelliner Strasse 38

16816 Neuruppin

Germany

e.blank@ruppiner-kliniken.de

**Abstract**

**Background:** In radiation oncology, different fractionation schemes can be compared converting the actual dose into the equivalent dose in 2 Gy fractions (EQD2). The focus of the current project was to create an ESAPI script (Eclipse Scripting API) for this kind of biological dose correction of the 3D dose matrix displayed in the radiation treatment planning system Eclipse (Aria, Varian Medical Systems, A SIEMENS Healthineers Company). Thereby, the goal was to make the biological dose correction topologically visible by conversion of the entire 3D dose distribution, thereby providing an additional tool for plan evaluation.

**Results:** In C#, a script for the biological dose correction of the 3D dose matrix was generated ("EB\_Bio3D"). The script was implemented and successfully tested using real life patient cases.

**Conclusion:** The above mentioned software was developed and implemented. Tests were performed on several patient plans and confirm the correct functioning of the ESAPI script *EB\_Bio3D*. A clinical test run will be started.

**Keywords:** radiation therapy, treatment planning, software, equivalent dose

**Background**

In radiation oncology, the biological effect of the radiation both on the target volume and the organs at risk depends on the dose delivered per fraction. This can by described using the linear-quadratic model [5]. In clinical practice, the equivalent dose in 2 Gy fractions (EQD2) is used to normalize any given fraction dose to a fraction dose of 2 Gy, therefore enabling the comparison of different dose schemes. The focus of this project was to create an ESAPI script (Eclipse Scripting API) for biological dose correction of the 3D dose matrix. Although a solution for manipulating dose matrices for nuclear medicine applications [2] has already been published along with code basics in Python [1], it was not clear if this project could be solved in C# with Eclipse 15.6. Given that the ESAPI libraries for Python and ESAPI are the same [Keranen] and there is an online description of ESAPI [3] the current project was initiated. The idea of this was based on a published Python script for biological conversion of gynecological brachytherapy plans [12].

The focus of this work is the technical aspect of data conversion and display using the above mentioned script. Data on biological correction of dose values [8,10,11] in DVH curves has been published before [9]. The biological correction of DVH curves is mathematically simple, since an organ-specific separation of the dose values is already available and they are already statistically compressed. Due to the monotonic, quasilinear behavior of the linear-quadratic conversion, a reduction of the problem to the DVH conversion is valid [9].

The goal of this project is to make the biological dose correction topologically visible by conversion of the entire 3D dose distribution, thereby providing an additional tool for plan evaluation. This conversion can be of great importance in the planning of re-irradiations. While the use of voxel-based EQD2 (Equivalent Dose 2Gy/Fx) conversion for re-irradiation has been demonstrated with other systems [13,14,15], implementation using ESAPI would be a first.

Debugging support as well as configuration parameter offloading was be realized through Excel.

**Implementation**

***Workflow***

With an in-house ESAPI script *EB\_Bio3D*, a verification plan without radiation fields, which incorporates the biologically corrected EQD2 dose distribution as an *EvaluationDose*-Matrix, is generated from an radiation treatment plan. The detour via a verification plan is necessary since dose matrices in calculated radiation treatment plans are not editable to ensure that unintentional misinterpretations cannot occur.

The main problems of this script are the following:

1. Creation of a verification plan with an *EvaluationDose*-Matrix

2. Conversion of the original dose distribution stored in *RawValue* into an absolute fractional dose distribution

3. Separation of dose points to individual organ structures

4. Organ-specific biological dose correction according to EQD2

5. Reconversion of the EQD2 dose distribution into *RawValues* to be stored back in Eclipse

6. Creation of an in-house code debugging tool using Excel

7. Excel file to store all necessary configuration parameters

Tests of this ESAPI script will be performed on various irradiation schedules.

***Data import***

The advantage of user-specific software based on ESAPI is the practicable handling in Eclipse. Without prior export and import, planning data can be retrieved directly from Eclipse for external processing by means of context-related libraries.

The code for context-related link to the required planning data can be found in *fig. 1.*

184 **// patient and plan context**

185 Patient Pat = context.Patient;

186 PatLName = context.Patient.LastName.ToString();

187 PatFName = context.Patient.FirstName.ToString();

188 PatID = context.Patient.Id.ToString();

189

190 context.Patient.BeginModifications();

191

192 Course eCourse = context.Course;

193

194 ExternalPlanSetup plan = context.ExternalPlanSetup;

195

196 **if** (context.Patient == null || context.StructureSet == null)

197 {

198 MessageBox.Show("Please load a patient, 3D image, and structure

set before running this script.", SCRIPT\_NAME, MessageBoxButton.OKCancel,

MessageBoxImage.Exclamation);

199 **return**;

200 }

***Fig. 1:*** *Context-related data import (lines 185-194)*

The first problem to consider is the interception of the different notations of the structure IDs needed for biological conversion. The use of clinical templates in treatment planning does not guarantee a standardized ID notation, making hard coded structure management error-prone. Goss tumor volume (GTV) IDs, which contain labels for treatment dose as well as anatomical location, are particularly problematic in this regard. For GTV this was solved by reading the structure list beforehand (*fig. 2*). The individual GTV in the structure list are identified by substring analysis.

201 StructureSet ss = context.StructureSet;

202

203 **foreach**(Structure s **in** ss.Structures)

204 {

205 **if** (s.Id.Length>3)

206 {

207 **if** (s.Id == BODY\_ID1 | s.Id == BODY\_ID2) // because

upper-/lowercase

208 {

209 sBody = s.Id;

210 }

211 **if** (s.Id.Substring(0,3) == "GTV")

212 {

213 gtv.Add(s.Id);

214 gtvls.Add(s);

215 }

216 }

217 }

***Fig. 2:*** *Reading the GTV as structure list (lines 201-217)*

*and substring analyse (line 211)*

Another problem to consider during biological conversion are overlapping structures. Although individual organ structures are mutually exclusive, GTV structures may overlap individual organ structures with different radiobiological behavior. GTV are usually considered to have higher  values because of their poorer repair properties. The expansion to generate the planning target volume (PTV) are not considered in the enlarged  values, because here it is assumed that these behave like healthy tissue. In code terms, the dominance of the GTV during conversion is realized by placing them at the end of the processing chain (*fig. 3*). The conversion is carried out twice in the overlapping areas, which leads to longer computing times. However, this was accepted for this ESAPI prototype.

421 **if** (dose != null)

422 {

423 **for** (int zi = 0; zi < dose.ZSize; zi++)

424 {

425 zm[zi] = zi \* dresZ + doZ;

426 dose.GetVoxels(zi, zbuffer); **// here read a dose layer**

427

428 dbi[zi] = zbuffer[90, 54]; **// only for test by debuging,**

**how look the dose values**

429

430 **for** (int yi = 0; yi < dose.YSize; yi++)

431 {

432 ym[yi] = yi \* dresY + doY;

433

434 **for** (int xi = 0; xi < dose.XSize; xi++)

435 {

436 xm[xi] = xi \* dresX + doX; **// calculate carthesian**

**points of matrix points**

**for PointInsideSegment**

437 VV[0] = xm[xi];

438 VV[1] = ym[yi];

439 VV[2] = zm[zi];

440

441 **// IsPointInsideSegment for all entities**

443 **// EQD2 conversion by a/b values**

445 **// imageMatrix und doseMatrix have different**

446 **// IsPointInsideSegment must have carthesian coordinates**

447 **// x, y, z must converted by Xres, Yres, Zres and Origin**

448

449 dbuffer[xi, yi] = dose.VoxelToDoseValue(zbuffer[xi, yi]).Dose \* GD / 100;

450 ebuffer[xi, yi] = 0;

451

452 **// Body**

453 **if** (body.IsPointInsideSegment(VV))

454 {

455 **if** (body != null)

456 {

457 ebuffer[xi, yi] = (dbuffer[xi, yi]/Frakt +

abBody) / (2 + abBody) \* dbuffer[xi, yi];

458 }

459 }

460

461 **// Hirn (brain)**

462 **if** (hirn != null)

463 {

464 **if** (hirn.IsPointInsideSegment(VV))

465 {

466 ebuffer[xi, yi] = (dbuffer[xi, yi]/Frakt +

abHirn) / (2 + abHirn) \* dbuffer[xi, yi];

467 }

468 }

469

. . . . . .

748 **// GTV list (gtvls)**

749 **if** (gtvls != null)

750 {

751 **foreach** (Structure s1 **in** gtvls)

752 {

753 **if** (s1.IsPointInsideSegment(VV))

754 {

755 ebuffer[xi, yi] = (dbuffer[xi, yi]/Frakt

+ abGtv) / (2 + abGtv) \* dbuffer[xi, yi];

756 }

757 }

758 }

759 }

***Fig. 3:*** *End placement of GTV list (lines 749-758) in EQD2 conversation chain (lines 452-758)*

***2.3 Conversion***

Biological dose conversion is performed using the linear quadratic model proposed by Lyman et al [3,4,5,6,7]. The values are organ-specific, which is why a structure query is necessary (*fig. 4*).

. . .

721 **// Darm (bowel / gut)**

722 **if** (darm != null)

723 {

724 **if** (darm.IsPointInsideSegment(VV))

725 {

726 ebuffer[xi, yi] = (dbuffer[xi, yi]/Frakt + abDarm) / (2 + abDarm) \* dbuffer[xi, yi];

727 }

728 }

729

730 **// Rektum (rectum)**

731 **if** (rektum != null)

732 {

733 **if** (rektum.IsPointInsideSegment(VV))

734 {

735 ebuffer[xi, yi] = (dbuffer[xi, yi]/Frakt + abRektum) / (2 + abRektum) \* dbuffer[xi, yi];

736 }

737 }

738

739 **// Blase (bladder)**

740 **if** (blase != null)

741 {

742 **if** (blase.IsPointInsideSegment(VV))

743 {

744 ebuffer[xi, yi] = (dbuffer[xi, yi]/Frakt + abBlase) / (2 + abBlase) \* dbuffer[xi, yi];

745 }

746 }

. . .

***Fig. 4:*** *Organ-specific EQD2 conversion (lines 726, 735, 744)*

*and structure query (lines 724, 733, 742)*

Before the conversion can be performed, it is necessary to consider how the dose matrix is stored in the Eclipse plan. It should be noted that the dose values in the matrix are stored as relative *RawValue*. These must be converted into absolute fractional dose values beforehand (*fig. 5*).

The assignment of the dose points to the individual structures works with Cartesian point coordinates. For this purpose, a conversion of the matrix indices into Cartesian point coordinates (vector *v[ , , ]*) is necessary (*fig. 5*).

. . .

376 **var** dose = plan.Dose;

377

. . .

386 int dSizeIX = dose.XSize;

387 int dSizeIY = dose.YSize;

388 int dSizeIZ = dose.YSize;

389

390 double doX = dose.Origin.x;

391 double doY = dose.Origin.y;

392 double doZ = dose.Origin.z;

393

394 double dsizeX = dose.XSize;

395 double dsizeY = dose.YSize;

396 double dsizeZ = dose.ZSize;

397

398 double dresX = dose.XRes;

399 double dresY = dose.YRes;

400 double dresZ = dose.ZRes;

401

402 int[,] zbuffer = **new** int[dose.XSize, dose.YSize];

403 double[,] dbuffer = **new** double[dose.XSize, dose.YSize];

404 double[,] ebuffer = **new** double[dose.XSize, dose.YSize];

. . .

408 double[] xm = **new** double[dose.XSize];

409 double[] ym = **new** double[dose.YSize];

410 double[] zm = **new** double[dose.ZSize];

. . .

412 EvaluationDose eDose = ePlan.CreateEvaluationDose();

413 ePlan.CopyEvaluationDose(dose); **// to fit dimensions of dose matrices dose und eDose !**

414 double fraktDose = plan.DosePerFraction.Dose;

415 int fractNumber = (int)plan.NumberOfFractions;

416 DoseValue ee = **new** DoseValue(fraktDose, DoseValue.DoseUnit.Gy); **// prepare fraction dose for new ePlan**

417 ePlan.SetPrescription(fractNumber, ee, 1.0); **// here set fraction number and fraction dose of ePlan in ARIA**

418

419 **// whole 3D matrix will be scanned**

420 VVector VV = **new** VVector(0,0,0);

421 **if** (dose != null)

422 {

423 **for** (int zi = 0; zi < dose.ZSize; zi++)

424 {

425 zm[zi] = zi \* dresZ + doZ;

426 dose.GetVoxels(zi, zbuffer); **// here read a dose layer**

427

428 dbi[zi] = zbuffer[90, 54]; **// only for test by debugging, how look the dose values**

429

430 **for** (int yi = 0; yi < dose.YSize; yi++)

431 {

432 ym[yi] = yi \* dresY + doY;

433

434 **for** (int xi = 0; xi < dose.XSize; xi++)

435 {

436 xm[xi] = xi \* dresX + doX; **// calculate carthesian points of matrix points**

**for PointInsideSegment**

437 VV[0] = xm[xi];

438 VV[1] = ym[yi];

439 VV[2] = zm[zi];

. . . . . .

449 dbuffer[xi, yi] = dose.VoxelToDoseValue(zbuffer[xi, yi]).Dose \* GD / 100;

450 ebuffer[xi, yi] = 0;

. . .

***Fig. 5:*** *Conversion the raw values of dose into fractional dose values (lines 426, 449),*

*Buffer matrices zbuffer[,], dbuffer[,], ebuffer[,] (lines 426, 449, 450),*

*Conversion the matrix indices into Carthesian coordinate vectors*

*needed for structure query of dose points (lines 425, 432, 436-439).*

*Adjustment of dose matrix to the DICOM Orign (lines 386-400,425,432, 436)*

The fractional dose values are used to perform the EQD2 conversion, which is performed iteratively through the entire 3D dose matrix.

To avoid double use of the EQD2 formula, multiple buffer matrices are used. The original relative *RawValue* dose values in *zbuffer[,]* are stored as absolute fractional dose values in *dbuffer[,]*. The EQD2 dose values are ultimately located in *ebuffer[,]* (*fig. 5*).

Since ESAPI only allows the dose data to be read out layer by layer, the conversion process is also performed layer by layer with two-dimensional buffer matrices *zbuffer[,]*, *dbuffer[,]* and *ebuffer[,]* (*fig. 6*).

. . .

760 **// all ebuffer reconvert into relative values**

761 ebuffer[xi, yi] = ebuffer[xi, yi] \* 100 / GD;

762 **if** (ebuffer[xi, yi] < 0)

763 {

764 ebuffer[xi, yi] = 0;

765 }

766

767 DoseValue eee = **new** DoseValue(ebuffer[xi, yi], DoseValue.DoseUnit.Percent);

768

769 zbuffer[xi, yi] = (int)eDose.DoseValueToVoxel(eee) ;

770

771 }

772 }

773

774 **// dbu[zi] = ebuffer[90, 54]; // for Excel debugging**

775 **// dbi[zi] = zbuffer[90, 54];**

. . .

785 eDose.SetVoxels(zi, zbuffer);

787 }

***Fig. 6:*** *Back conversion the EQD2-EvaluationDose matrix into RawValues layer by layer (line 761-769)*

*Back writing the converted RawValues into ARIA data base (line 785)*

At the end of the conversion process, the EQD2 values must be converted back to raw dose values. This is done with the function *DoseValueToVoxel()*. Since *SetVoxels()* requires the so-called *DoseValue* type as parameter instead of dose values as double, a small additional step of converting double to *DoseValue* is necessary.

***Excel Debugger***

An important additional help is the possibility to debug the ESAPI code during development time. Since the script works context based, Eclipse must be started each time to test it. In Eclipse, there is no native way to debug as there is under an Integrated Development Environment (IDE). There is the possibility to use a PluginRunner [16,17] to run the script directly from the IDE. However, this requires code intervention. Another possibility is to attach an Excel interface that is often needed in ESAPI scripts anyway (*fig. 8*). This allows variable values to be easily read out for runtime under Eclipse (*fig. 7*).

. . .

**789 //----------------------------------------------------------------------------------**

**790 // call of Debugging Excel**

**791 //----------------------------------------------------------------------------------**

792 /\*

793 **UpdateExcel**(erX.ToString(), erY.ToString(), erZ.ToString(),doX.ToString(), doY.ToString(), doZ.ToString(),

eX.ToString(), eY.ToString(), eZ.ToString(),

. . . . . .

797 dbi[0].ToString(), dbi[1].ToString(), dbi[2].ToString(), dbi[3].ToString(), dbi[4].ToString(), dbi[5].ToString(),

. . . . . .

806 dbi[94].ToString(), dbi[95].ToString(), dbi[96].ToString(), dbi[97].ToString(), dbi[98].ToString(),

dbi[99].ToString() );

807 \*/

. . .

***Fig. 7:*** *Commented Excel-function call, could be inserted everywhere in the code.*

*The parameter list should be adapted (lines 793-806)*

812 **//----------------------------------------------------------------------------------**

813 **// Debugging Excel**

814 **//----------------------------------------------------------------------------------**

815 **private void UpdateExcel**(string S1, string S2, string S3, string S4, string, S5, string S6,

string S11, string S12, string S13,

. . .

string S100, string S101, string S102, string S103, string S104, string S105,

. . .

string S194, string S195, string S196, string S197, string S198, string S199)

829 {

830 Excel.Application oXL = null;

831 Excel.\_Workbook oWB = null;

832 Excel.\_Worksheet oSheet = null;

833

834 **try**

835 {

836 oXL = **new** Microsoft.Office.Interop.Excel.Application();

837 oWB = oXL.Workbooks.Open("Q:/ESAPI/Plugins/EB\_Bio3D/EB\_Bio3D.xlsx");

838 oSheet = String.IsNullOrEmpty("Tab1") ? (Microsoft.Office.Interop.Excel.\_Worksheet)oWB.ActiveSheet :

(Microsoft.Office.Interop.Excel.\_Worksheet)oWB.Worksheets["Tab1"];

840

**//.................................................................................**

840 **// Excel Sheet Test**

841

842 **// read the row counter in Excel**

843 string sReihe = oSheet.Cells[1, 1].Value == null ? "-" :

oSheet.Cells[1, 1].Value.ToString();

844 int iReihe = Convert.ToInt32(sReihe);

845 iReihe = iReihe + 1;

846

847 **// dresX**

848 oSheet.Cells[iReihe, 1] = S1;

849 oSheet.Cells[iReihe, 2] = S2;

850 oSheet.Cells[iReihe, 3] = S3;

851 **// doX**

852 oSheet.Cells[iReihe, 4] = S4;

853 oSheet.Cells[iReihe, 5] = S5;

854 oSheet.Cells[iReihe, 6] = S6;

855 **// dsizeX**

856 oSheet.Cells[iReihe, 8] = S11;

857 oSheet.Cells[iReihe, 9] = S12;

858 oSheet.Cells[iReihe, 10] = S13;

. . . . . .

867 **// curve data**

868 oSheet.Cells[10, 4] = S100;

869 oSheet.Cells[11, 4] = S101;

870 oSheet.Cells[12, 4] = S102;

. . . . . .

974 oSheet.Cells[107, 4] = S197;

975 oSheet.Cells[108, 4] = S198;

976 oSheet.Cells[109, 4] = S199;

977

978 **// write bach the row counter in Excel**

979 sReihe = Convert.ToString(iReihe);

980 oSheet.Cells[1, 1] = sReihe;

981 **// and save Excel**

982 oWB.Save();

983

984 }

985 **// Quit after saving Excel**

986 catch (Exception ex)

987 {

988 MessageBox.Show(ex.ToString());

989 }

990 **finally**

991 {

992 **if** (oWB != null)

993 {

994 oWB.Close(true, null, null);

995 oXL.Quit();

996 }

997

998 }

999 }

***Fig. 8:*** *Excel function for reading and writing into an Excel file (lines 768-956)*

However, some *using* clauses and *Referencies* are required to include an Excel file interface (*fig. 9*). Furthermore, Excel files can also be used as ini or config files. User-specific script configurations can be easily stored in these files without having to hardcode them.

51 **using** System;

52 **using** System.Linq;

53 **using** System.Windows;

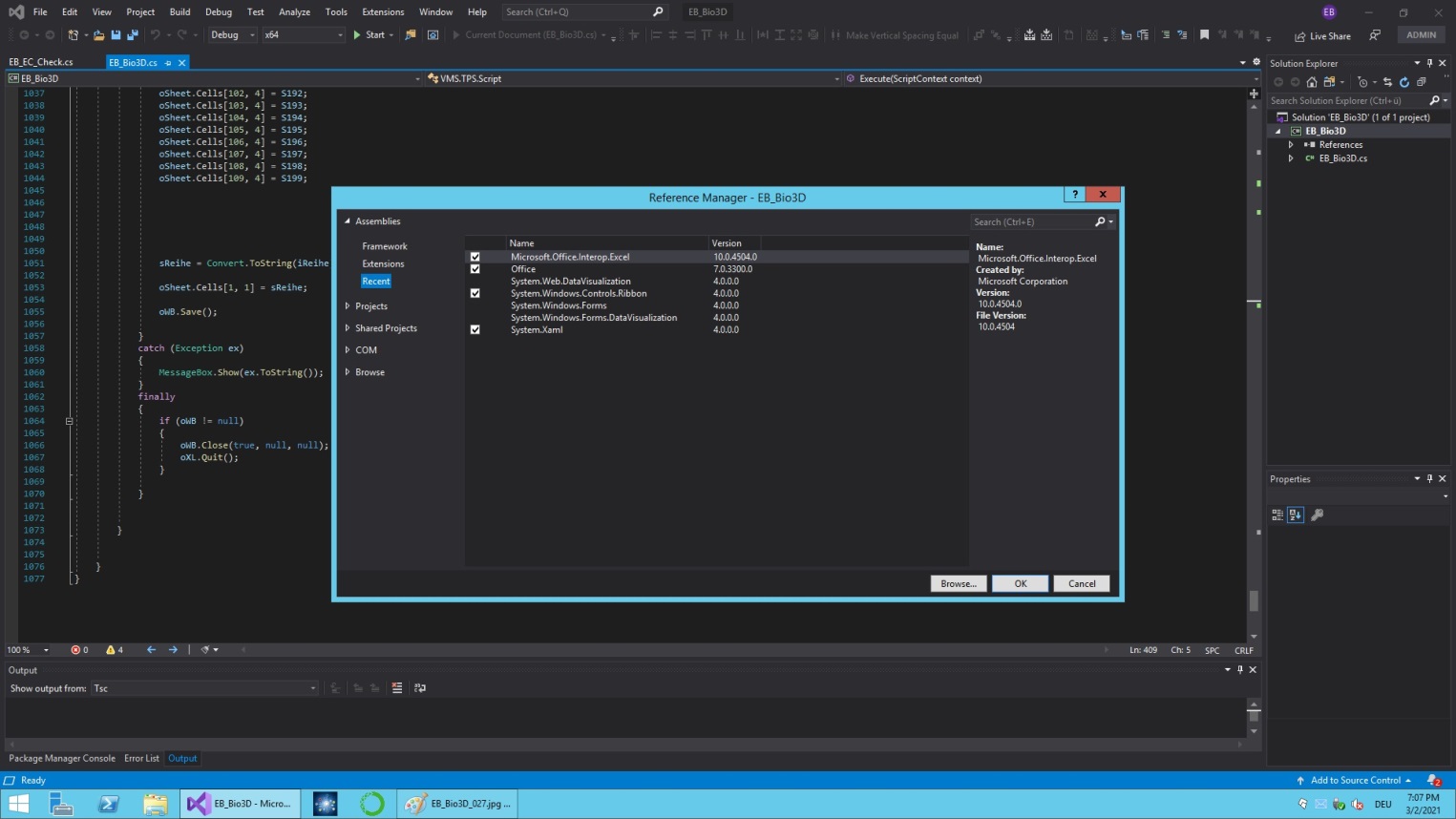
54 **using** System.Collections.Generic;

55 **using** System.Reflection;

56 **using** Excel = Microsoft.Office.Interop.Excel;

57 **using** VMS.TPS.Common.Model.API;

58 **using** VMS.TPS.Common.Model.Types;



***Fig. 9:*** *Using clauses (line 56, upper fig.) and referencies, to embed the Excel Routine into C# Code(lower fig.)*

In the broader context Excel files are advisable under ESAPI, because test results can be stored not only for documentation purposes, but can be processed later effortlessly without script. ESAPI thus only becomes necessary as an interface for later external data collection and evaluation.

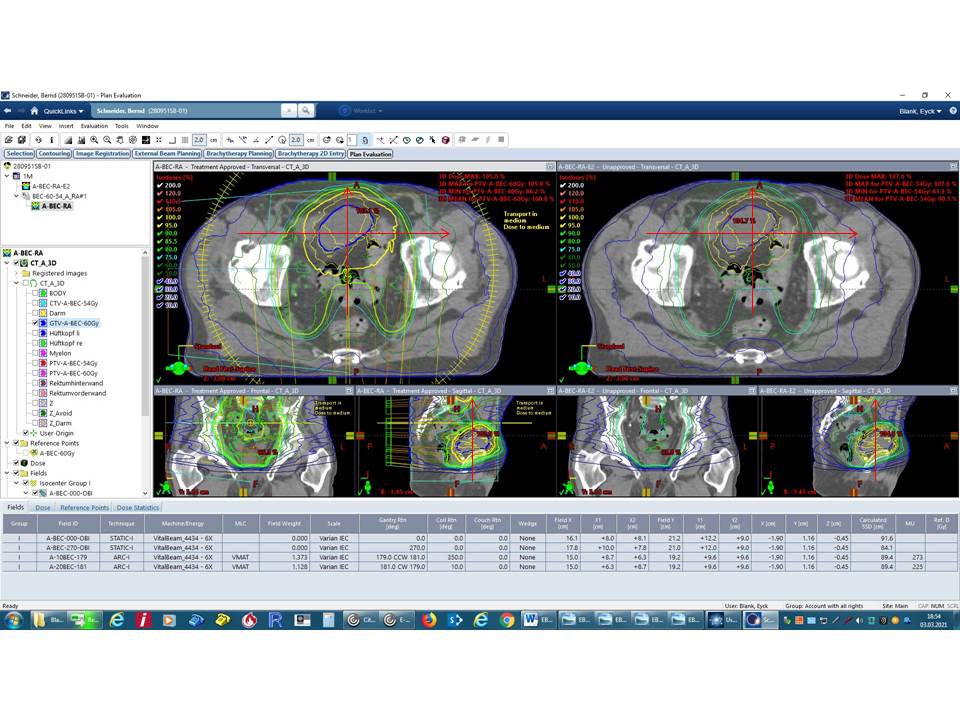
The EB\_Bio3D script will be described in more detail in an ESAPI Cookbook 2021. After publication of the cookbook, the code will also be available on github [18].

**Results**

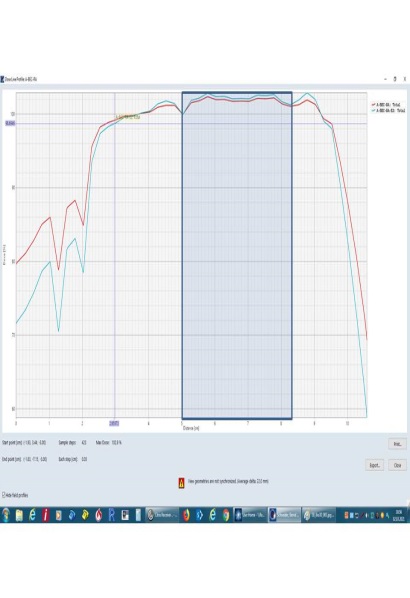
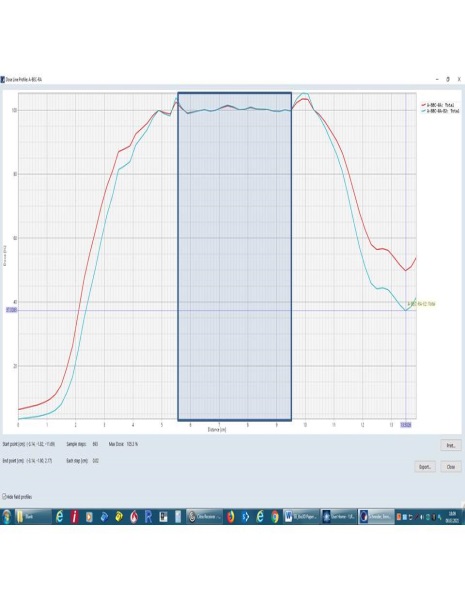
***Patient example***

The function of the script for EQD2 conversion is shown on the following figure as an example of a pelvic irradiation with 60Gy total dose and 2Gy fractional dose (*fig. 10*).

The original plan is shown on the left and the biologically corrected EQD2 dose distribution on the right. The comparison looks plausible. While in the central GTV area (blue structure line) original and EQD2 dose distribution are almost the same ( = 8), EQD2 is significantly reduced in the rest of the area with an  value of 2 and smaller local fractional dose (<2Gy).



***Fig. 10*** *Comparison of original plan (left) and EQD2 converted evaluation plan (right). The red lines mark the comparison profile curves*

***Fig. 11*** *Dose profile comparison original plan (red) and EQD2 converted evaluation plan (blue) transversal(left), pa-ap (middle), caudal-cranial (right). The blue shaded area marks the profile area lying in the GTV. The jags in the left part of the middle curve are air pockets in the rectum*

In *PlanEvaluation*, different profile curves of the original plan and EQD2 plan are compared (*fig. 11*). Plausibility is checked. Attention is paid to the fact that the curves overlap without offset, that there are no dropouts (unconsidered dose points in the EQD2 conversion) and that the organ-specific conversions are confirmed by recalculation.

***Testing of the ESAPI Script***

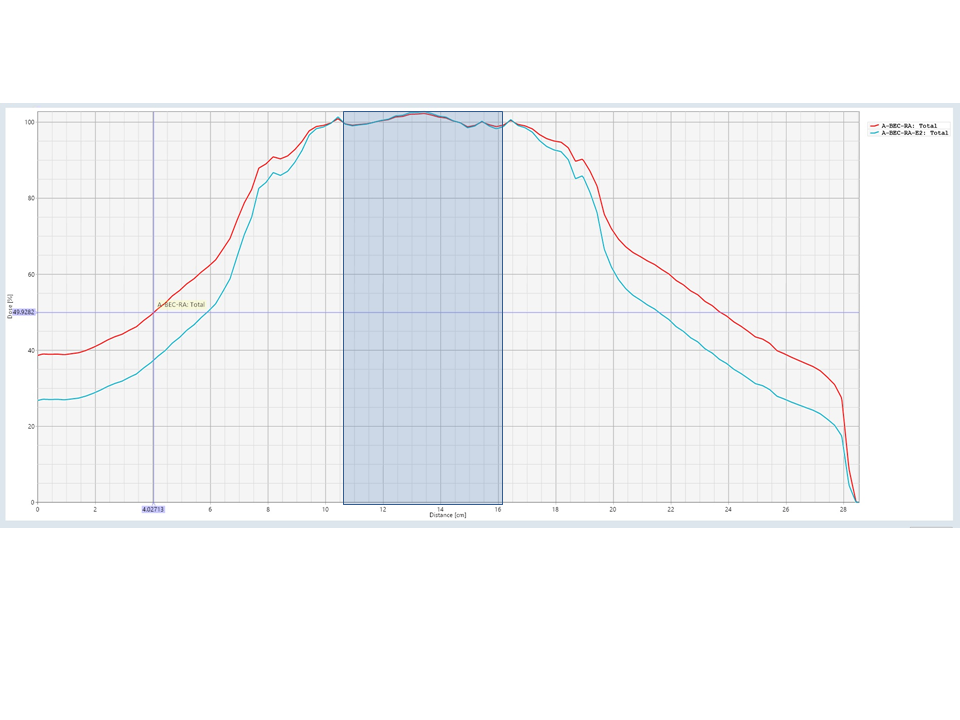
The following describes how the script has been checked for correct operation. The first problem of interest is whether the output dose and evaluation dose distributions have no offset from each other. This error possibility is due to the fact that image (structure) matrix and dose matrix in the code have to be adjusted separately and individually with respect to the DICOM Origins (*fig. 5*).

This origin adjustment is significant for the generation of the metric dose point vectors of the dose matrix indices for organ assignment of the individual dose matrix points.

In the profile curve comparison in all three spatial directions, a superposition of the curve structures is seen as expected (*fig. 11*).

The next test aspect is the detection of the GTV boundaries in the EQD2 dose matrix. Since the GTV is different from the other structures in the conversion, a small jump in the EQD2 dose distribution should be detectable here. A jump is not visible in the profile curves shown at the GTV boundaries (*fig. 11*), since the dose is already approximately equal to the 2Gy prescription dose in the surrounding PTV fringe.

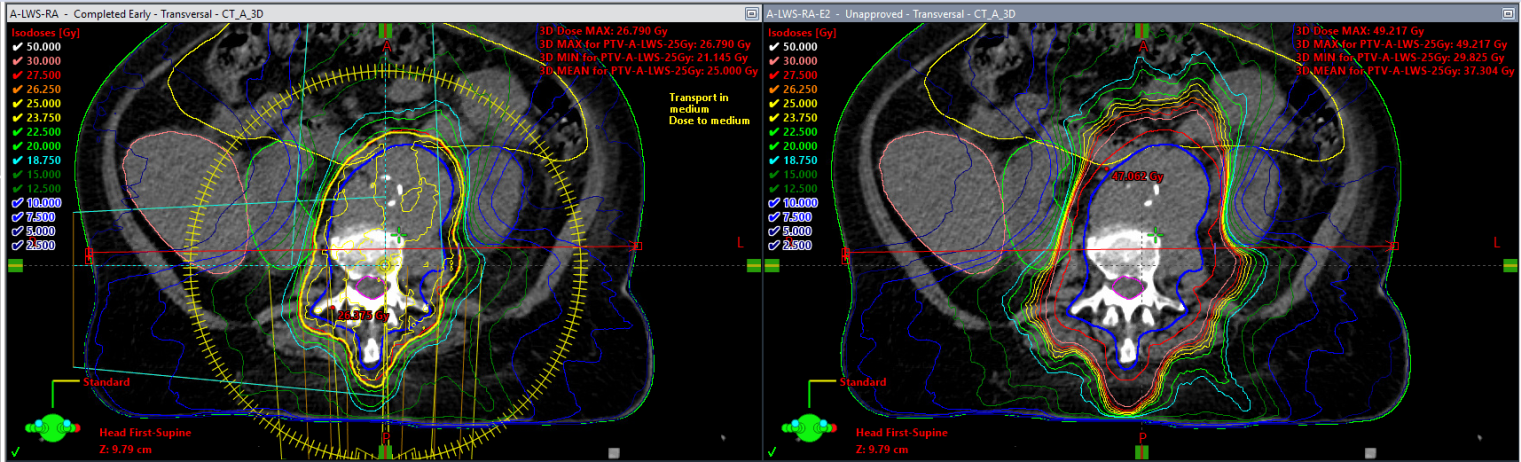
The most important test is the point-by-point recalculation of the biologically corrected EQD2 dose distribution. This test is also best performed in PlanEvaluation and showed good agreement at several test points. This can be easily understood from the graph provided (*fig. 12*).



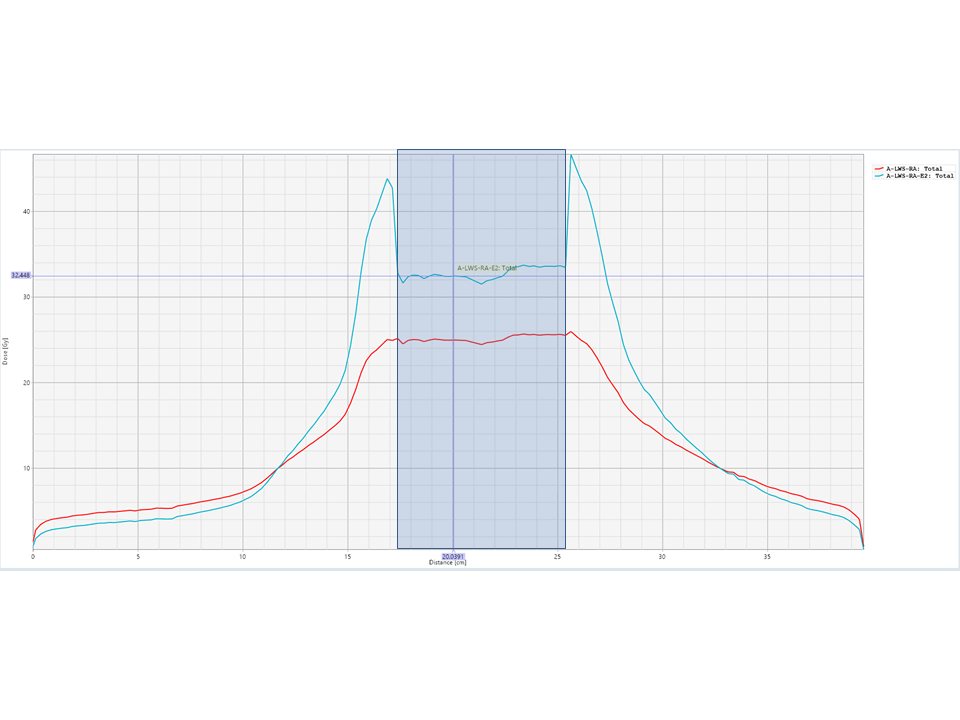
***Fig. 12*** *Enlarged profile curve comparison between original dose (red) and EQD2 dose (blue) for recalculation of the biological dose correction. The shown point in the dose profile shows a dose value of 0.999Gy and lies in a range with = 2*

At a prescribed fractional dose of 2Gy, the dose point in *fig. 12* shows a local nominal dose value of 0.999Gy (49.93%). Recalculated with the -formula, -value of 2 results in a local biological dose of 0.749Gy (37.45%) at this point, which corresponds to the value of the blue curve.

The EQD2 conversion was also tested for different fractionations, e.g. a 5 x 5Gy lumbal spine



***Fig. 13*** *Comparison of original plan (left) and EQD2 converted evaluation plan (right) at 5 x 5Gy fractionation*



***Fig. 14*** *Enlarged profile curve comparison between original dose (red) and EQD2 dose (blue) for recalculation of the biological dose correction of a 5 x 5Gy fractionation*

When comparing the original and EQD2 dose for the 5 x 5Gy fractionation in *fig. 13,* a significantly larger EQD2 dose can be found in the volume surrounding the GTV.

Looking at the dose profiles of original plan and the EQD2 corrected dose (*fig. 14*), there is a steep increase seen for the blue EQD2 curve at the GTV edges due to the different  values of the tissues. Defining  = 8 for the GTV (blue shaded region in *fig. 14*) in this case, the EQD2 calculation can be easily manually checked. For this example, the uncorrected dose inside the GTV (red curve) of 25Gy results in 32.5Gy EQD2 corrected dose.

The most frequent error in the prototype program was the failure to find different structures. This is due to the ID-specific structure assignment and will be corrected in the next program version by reading out and utilizing a structure list.

**Conclusion and Outlook**

These tests were performed on several patient plans and confirm the correct functioning of the ESAPI script *EB\_Bio3D*. A clinical test run will be started. The published script is a prototype to demonstrate the procedure. In the next version, the previously defined values in the code will be stored externally in an Excel file, which will serve as a config file. The current ID-specific structure assignment, also defined in the code, will be automated by a scanned structure list and subsequent substring evaluation.

**References**

*[1] Folkerts, M. M. (2018). PyESAPI: The Python Interface to ESAPI. In J. Pyyry & W. Keranen (Eds.), Varian AP - A handbook for programming in the Varian oncology software ecosystem (pp. 83–102). moz-extension://4327145d-c6a3-4ba2-9014-b3cf61c07c84/enhanced-reader.html?openApp&pdf=https%3A%2F%2Fvarianapis.github.io%2FVarianApiBook.pdf*

*[2] J. Pyrry. (n.d.). Dose calculation for radionuclide therapy. In J. Pyyry & W. Keranen (Eds.), Varian APIs - A handbook for programming in the Varian oncology software ecosystem (pp. 121–127).*

*[3] Varian Medical Systems. (n.d.). Eclipse Scripting API, Online Help (Varian Medical Systems (Ed.); Version 15.5*

*[4] Fowler, J. F. (2006). Practical Time-Dose Evaluations, or How to Stop Worrying and Learn to Love Linear Quadratics. Technical Basis of Radiation Therapy, 3–31.* [*https://doi.org/10.1007/3-540-35665-7\_1*](https://doi.org/10.1007/3-540-35665-7_1)

*[5] Fowler, J. F. (1989). The linear-quadratic formula and progress in fractionated radiotherapy. British Journal of Radiology, 62(740), 679–694.* [*https://doi.org/10.1259/0007-1285-62-740-679*](https://doi.org/10.1259/0007-1285-62-740-679)

*[6] Thames, H. D. (1985). An “incomplete-repair” model for survival after fractionated and continuous irradiations. International Journal of Radiation Biology, 47(3), 319–339.* [*https://doi.org/10.10*](https://doi.org/10.10)

*[7] Lyman, J. T. (1985). Complication probability as assessed from dose-volume histograms. Radiation Research. Supplement, 8, 13–19.* [*https://doi.org/10.2307/3583506*](https://doi.org/10.2307/3583506)

*[8] Voyant, C., & Julian et al., D. (2017). A short synthesis concerning biological effects and equivalent doses in radiotherapy. ArXiv.* [*https://doi.org/10.29328/journal.jro.1001005*](https://doi.org/10.29328/journal.jro.1001005)

*[9] Wheldon, T. E., Deehan, C., Wheldon, E. G., & Barrett, A. (1998). The linear-quadratic transformation of dose-volume histograms in fractionated radiotherapy. Radiotherapy and Oncology, 46(3)*

*[10] Zheng, Y., Yuan, J., Woods, C., Machtay, M., & Wessels, B. (2012). SU‐E‐T‐460: Isoeffective Dose Display (EQD2) for Composite Plan of Radiosurgery and Conventional 3D Radiotherapy. Medical Physics, 39(6), 3810–3811.* [*https://doi.org/10.1118/1.4735549*](https://doi.org/10.1118/1.4735549)

*[11] Jacobs, P., Nelson, A., & Liu, I. (2012). Biological Effective Dose and Tumor Control Probability Modeling using the MIM® Software Suite. Mimsoftware.Com, 693, 3–5.* [*https://www.mimsoftware.com/manager/templates/mimsoftware/resources/whitepapers/BED\_and\_Tumor\_Control\_Probability\_Modeling.pdf*](https://www.mimsoftware.com/manager/templates/mimsoftware/resources/whitepapers/BED_and_Tumor_Control_Probability_Modeling.pdf)

*[12] Faught, A. M., Chino, J. P., Chang, Z., Meltsner, S., Cai, J., Vergalasova, I., Steffey, B., & Craciunescu, O. (2016). Using Varian’s Eclipse Scripting API to Calculate, Add, and Report Biologically Equivalent Doses for Gynecological Brachytherapy and External Beam Radiation Therapy Patients. Brachytherapy, 15(June), S137–S138.* [*https://doi.org/10.1016/j.brachy.2016.04.236*](https://doi.org/10.1016/j.brachy.2016.04.236)

*[13] Jaikuna, T., Khadsiri, P.et al., (2017). Isobio software: Biological dose distribution and biological dose volume histogram from physical dose conversion using linear-quadratic-linear model*. *Journal of Contemporary Brachytherapy, 9(1), 44-51.*

*[14] Singh, G., Kamal, R., Thaper, D., Oinam, A. S., Handa, B., Kumar, V., & Kumar, N. (2020). Voxel based evaluation of sequential radiotherapy treatment plans with different dose fractionation schemes. The British Journal of Radiology, 93(1112), 20200197.* [*https://doi.org/10.1259/bjr.20200197*](https://doi.org/10.1259/bjr.20200197)

*[15] Singh, G., Oinam, A.et al. (2018). Voxel based BED and EQD2Evaluation of the radiotherapy treatment plan. Journal of Medical Physics, 43(3), 155–161.* [*https://doi.org/10.4103/jmp.JMP\_29\_18*](https://doi.org/10.4103/jmp.JMP_29_18)

*[16] Carlos J Anderson. (n.d.). Introducing ESAPI Essentials; Plugin Runner 2.0.0.* [*http://www.carlosjanderson.com/introducing-esapi-essentials/*](http://www.carlosjanderson.com/introducing-esapi-essentials/)

*[17] Castelo, J. (n.d.). How to emulate plugins with executables in ESAPI; Plugin Runner.* [*https://jhmcastelo.medium.com/how-to-emulate-plugins-with-executables-in-esapi-97f5213f4560*](https://jhmcastelo.medium.com/how-to-emulate-plugins-with-executables-in-esapi-97f5213f4560)

*[18] github, ESAPI, EB-DOSIS*

**Availability and requirements**

**Project name:**EB\_Bio3D  
**Project home page:**[https://github.com/EB-DOSIS](https://smex-ctp.trendmicro.com:443/wis/clicktime/v1/query?url=https%3a%2f%2fgithub.com%2fEB%2dDOSIS&umid=fec9130f-fe33-46c6-8a26-e057bf072511&auth=7fe2a32480a9dfc79f9f73dcec4ddd12c050be02-b3f6d6c7a3a0a05c67c87e4fce0010112a409c00)/xxxx   
**Operating system(s):**Windows10  
**Programming language:**C#  
**Other requirements:**Eclipse 15.6 incl. Eclipse Scripting API (VARIAN Medical Systems), MS Excel 2017, MS Visual Studio Community 2019  
**License:** free  
**Any restrictions to use by non-academics:** At first only testing clinically in own department (Radiotherapy of OGD Neuruppin)

**List of abbreviations**

DVH - Dose Volume Histogramm

DICOM - Digital Imaging and Communications in Medicine

ESAPI - Eclipse Scripting API

EQD2 - Equivalent dose in 2 Gy fractions

GTV - Gross Tumor Volume

IDE - Integrated Development Environment

PTV - Planning Target Volume

IDE - Integrated Development Environment

**Declarations**

**Ethics approval and consent to participate**

Not applicable.

**Consent for publication**

Not applicable.

**Availability of data and materials**

See "Availability and requirements".

**Competing interests**

The authors declare that they have no competing interests.

**Funding**

There was no funding.

**Authors' contributions**

All authors contributed to the conception and design. Software preparation, data collection and analysis were performed by Eyck Blank. The first draft of the manuscript was written by Christina Schröder and Eyck Blank and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

**Acknowledgement**

Not applicable.