CVRML conceptual model

Peter Peer[1,2], Victor Segura[1], Franc Solina[2]
[1]CEIT and Tecnun (University of Navarra),
Manuel de Lardizabal 15, 20018 San Sebastian, Spain
[2]Faculty of Computer and Information Science, University of Ljubljana,
Tržaška 25, 1000, Ljubljana, Slovenia
E-mail: peter.peer@fri.uni-lj.si

We have developed our conceptual model in UML [1] (Unified Modeling Language: http://www.uml.org/), which is in general a set of diagramming techniques designed to improve software engineering and requirements capture. UML is the industry standard object-oriented modeling language. In this context, it also allows us to describe experimental methods, results, and subsequent analyses in an implementation-independent manner.

Figure 1 shows the CVRML UML class diagram, which provides a conceptual model of how to integrate automatically obtained computer vision algorithm results with existing data about one 2DGE experiment. It presents the basis for the XML implementation, revealing all the relations between classes and variables. Namely, each box in a diagram represents a class, which in the XML implementation becomes a tag, while their variables become subtags.

On the lower right side of the diagram we can see basic components of the whole proteomics experiment system. Everything starts with sample generation, continues with sample processing, mass spectrometry and ends with analysis of mass spectrometry results. Our proposed model refines the second step in this system, i.e. the sample processing step.

The base class is called 'CVRML' and it encompasses all important information about the experiment from the perspective of computer vision. As such it can be used to identify which information is needed if someone would like to extend existing proteomics or 2DGE experiment models with the CVRML idea. Two extensions, extension of the PEDRo (Proteomics

Experiment Data Repository) [2, 3] model and the AGML (Annotated Gel Markup Language) [4, 5] model, are presented on the CVRML web portal (see URL at end of the document).

The only variable inside this class is *Experiment_description*, which reveals basic logical relations between the gels in the experiment, the hypothesis behind it etc. Physical relations are visible on a lower level, i.e. when we relate the spots between each other.

Our main goal when we were building the model was to present a flexible data-format for computer vision results in 2DGE. Therefore, having in mind the flexibility demand XML was a natural choice as it is presently the consensus choice in most areas. We explain details about the implementation of the model in the XML in the next section, but here we have to explain a term, which we named summary class. In our case all classes that have names in plural and are without variables mark summary classes. From the term itself it follows that such a class summarizes a number of instances of the same subclass. This concept is supported by the XML design goals (http://www.w3.org/TR/2004/REC-xml-20040204/#sec-origin-goals), which also emphasize the importance of human-legibility of XML files. And our hierarchical structure with the concept of summary classes significantly improves the readability. This is especially notable if the generated files are big, which is quite common in proteomics. This concept can also be found in the AGML. In our model a summary class is always associated with at least one instance of the subclass.

The first summary class in the model is 'Gels_2D', which contains all the gels associated with the experiment. Inside it there has to be at least one gel (note that the relation between the two classes is 1 to 1..n). Each gel is represented as an instance of class 'Gel_2D'. In contrast to the AGML file, the CVRML file and consequently the 2DGE experiment described in the file can consist of gels of different samples and even of different sample origins, as also suggested in the PEDRo. Since 'Gel_2D' class is common to all known models (e.g. 'Gel2D' class in PEDRo, 'real' class in AGML and '2D-PAGE' in Gla-PSI (Glasgow – Proteomics Standards Initiative) [6, 7]) it is quite straightforward to integrate the CVRML idea into each of them.

'Gel_2D' class has eight variables: The first one, *Gel_id*, gives the gel identification number. *Image_URI* variable gives the location where the gel image is stored. The location is given by URI (Universal Resource Identifier – a more general form of the standard internet address otherwise known as a URL or Uniform Resource Locator). The name of the experimenter

who produced the ground truth information and the date of its creation are presented by variables *Ground_truth_author* and *Ground_truth_date*. *Sample_origin* variable reveals the tissue used and it can contain all the relevant information from species name downwards. Thus, in general, the application that searches the database can parse the data and get the details about taxonomy, tissue type and cell type, if specified. *Type* describes the experimental condition. For instance, if we are talking about differential proteomics then we could label the gel as Control or Target. Furthermore, if we are talking about gene manipulation then we could label the gel as Wild type (healthy sample) or Knock out (ill sample). On the other hand, if we are interested only in proteome information, i.e. in identification of proteins in a single gel, we can integrate this information as well as our model supports it. *Stain* gives the stain(s) used for identification of protein spots. And finally *Comment* enables us to present additional details about the gel. From the perspective of computer vision this is the information that we would like to have for each gel, while a mapping of it to or from a more exact proteomics experiment system model is straightforward.

The two classes that encompass gels reveal the ground truth only section of our model. The next section of the model represents information that is related to computer vision algorithms only. At this point we would like to stress again that the data gathered about gels by computer vision algorithms inside our file is obtained automatically. Then we can contrast the results of each computer vision algorithm to the ground truth information and/or to other computer vision algorithms applied to the same gels. In this way the comparison is much more objective.

'Cv_algorithms' is a summary class that contains all the computer vision algorithms applied to the gel in question. Each gel can have zero or one relation to 'Cv_algorithms' class, while it has to have at least one instance of 'Cv_algorithm' class. Inside it we have thirteen variables: Similarly as in each gel we give an identifier *Algorithm_id* to each computer vision algorithm. Logical name of the algorithm is written in variable *Algorithm_name*. For each algorithm it is very important to know who developed it (*Authors_of_algorithm*), who implemented it (*Authors_of_implementation*), authors contact information (*Contact_information*) and location where the (in-depth) description of the algorithm can be found (*URI_to_description*). Other information of interest is the date on which the test was performed (*Test_date*), the description of the computer used for testing (*Machine_type*) and three detection times given in mill seconds. *Spot_detection_time* gives

the time needed to detect all the spots (proteins) in the gel, *Representative_spots_detection_time* gives the time needed to detect all representative spots in a pair of gels and *Spots_of_interest_detection_time* gives the time needed to detect all the spots of interest in a pair of gels. The last two new terms are explained in the continuation, with class 'Spot'. Variable *Summary_of_results* gives an overview of the algorithm's efficiency, e.g. the number of true positive detections, false positive detections and false negative detections. *Comment* enables us to present additional details about the algorithm.

The third section of our model encompasses common parts of the ground truth information and computer vision results. It starts with summary class 'Spots'. Thus, this class is used to represent spots identified manually (or interactively) as ground truth information as well as to represent spots identified completely automatically by computer vision algorithms. Each protein spot inside this class is represented by an instance of class 'Spot'. It has six variables: *Spot_id_in_gel* identifies the spot in this gel. *Pixel_x_coordinate* and *Pixel_y_coordinate* give the spot center in pixels, which actually gives the coordinates with the darkest intensity value (*Intensity*) inside the spot. Whenever we talk about pixel values, we can also specify the values in the subpixel accuracy domain. Variable *Probability* gives the probability that this spot is a true spot. When we integrate the ground truth into our file this variable can have only two values: 1 and 0. If there is a spot, we assign to it the probability of 1. But since we are also interested in knowing the position of the spot that is missing in a pairing gel, we can assign the probability of 0 to such spots to indicate the position of where the corresponding spot in the pairing gel should be. Of course, such spots are not real spots, but they integrate very useful additional information into our experiment file.

When we integrate computer vision results into the file the probability can be an arbitrary value from the interval [0..1], while the algorithms can also calculate the probability of each spot being a true spot. *Comment* again enables us to present additional details about the spot. Note that we can easily relate our spot information to its mass spectrometry data written in for example mzXML [8] file, if needed.

Each spot can be marked as a representative spot (*Representative_spot*) or a spot of interest (*Spot_of_interest*) through class 'Annotation_type'. The spot can have many different annotation types (summary class 'Annotation_types'), but each is related to a pairing gel (*Pairing_gel_id*). The repre-

sentative spots are spots which are in both gels well and equally expressed and can serve as an evaluation tool for the registration of two gel images (the process of spatially aligning two images of a scene). On the other hand, the spots of interest are spots with quantitative or qualitative difference in the given gel pair. By quantitative difference we mean that a spot has a corresponding spot in the pairing gel, but the size of this spot is much bigger than the size of the corresponding spot (e.g. at least three times bigger in 3D). And by qualitative difference we mean that a corresponding spot is missing in a pairing gel. These spots are important because after the registration we want to identify the spots on which we want to perform mass spectrometry. According to this interpretation the name *Spot_of_interest* is logical.

Each gel in a pair has a set of spots of interest related to the pairing gel, but we do not have any redundant data, i.e. the same data is never written in the file twice. When we write the representative spots into our file, we write it only for the image that served as the basic image for the comparison, normally for Control or Wild type (healthy sample) gel image. But this is not necessary; in general, the basic image reflects control, healthy or previous state of the subject (e.g. you can also compare two Knock out (ill sample) gel images). In this way we do not have any redundant data in the file, but we can simply relate parts of information in the file.

For each spot we have three general marking possibilities: A spot can be marked as a representative spot or a spot of interest. Based on definitions of these two terms it is obvious that we are talking about exclusive or operation: a representative spot cannot be a spot of interest or vice versa. These two marking possibilities are explicitly stated in our model, but there is also one implicit. Namely, if the information inside the file does not reveal a relation with this spot then this means that no annotation type is associated with this spot, neither explicitly (it could be written inside this spot information section of our file) nor implicitly (in general, it could be a representative spot written inside a pairing gel information section of our file). Such a spot is treated simply as a protein spot. This information is also extremely important, while the first goal of computer vision applied to gels is to identify all protein spots in each gel.

If we mark a spot as a representative spot then we also must give proper corresponding spot information (an instance of class 'Corresponding_spot'). When we integrate the ground truth information in the file, we prefer to have only one true corresponding spot, but in the case of automatic processing this is in some cases unfortunately only wishful thinking. That is

why we can integrate more corresponding spots (see summary class 'Corresponding_spots') inside one instance of 'Annotation_type' class, but we can order them according to the probability that given corresponding spot is a true corresponding spot (*Prabability*). Each corresponding spot is identified with a number assigned to it in the related gel (*Spot_id_in_gel*). If we mark the spot as a spot of interest then it follows from our definition of variable *Spot_of_interest* that such a spot can have corresponding spot(s) or not.

A spot can also be modeled in 2D or 3D. Thus, a 'Spot' class can be related to two subclasses: 'Models_2D' and 'Models_3D'. Since here we have only two subclasses, we do not need a summary class. In 2D we can model a spot with a circle ('Circle'), where we have to specify its radius (*Radius*). Remember, we do not write redundant data into our file, so we do not give the center location again. It is already given in the instance of the class 'Spot'. We can model a spot also with a rectangle ('Rectangle'), where we have to give the bounding box (*Size_in_x*, *Size_in_y*). Another model can be an ellipse ('Ellipse'), where we have to give the size of both axis (*Size_of_big_axis*, *Size_of_small_axis*) and the angle for which the ellipse is rotated (*Rotation_angle*). The last 2D model is described with a set of boundary points (summary class 'Boundary_points'), where we have to present an ordered list (e.g. in the mathematical positive direction) of boundary points. Each boundary point ('Boundary_point') is given with coordinates in pixels (*Pixel_x_coordinate*, *Pixel_y_coordinate*). To all mentioned models we can also associate more precise size of the spot's surface (*Spot_surface*) than is that, which can be calculated from the information about each model.

In 3D we can model the spot with the Gaussian model ('Gaussian_model'), where we have to specify the two remaining parameters for such model, i.e. the standard deviations (*Standard_deviation_in_x*, *Standard_deviation_in_y*). Other parameters are already given in the instance of the class 'Spot': the spot's center and its intensity. Similarly as in 2D, in 3D we can associate to each model more precise size of the spot's volume (*Spot_volume*). If there will be a need to introduce new spot models in the future, we can easily integrate them into the CVRML conceptual model.

URL

The CVRML web portal:
http://www.lrv.fri.uni-lj.si/~peterp/CVRML/CVRML.htm.

References

[1] Rumbaugh, J., Jacobson, I. & Booch, G. *The Unified Modeling Language reference manual* (Addison-Wesley, Reading, MA, USA, 1999).

[2] Taylor, C.F. et al. A systematic approach to modeling, capturing, and disseminating proteomics experimental data. *Nat. Biotechnol.* **21**, 247–254 (2003).

[3] Garwood, K. et al. PEDRo: A database for storing, searching and disseminating experimental proteomics data. *BMC Genomics* **5** (2004).

[4] Stanislaus, R., Jiang, L. H., Swartz, M., Arthur, J. & Almeida J. S. An XML standard for the dissemination of annotated 2D gel electrophoresis data complemented with mass spectrometry results. *BMC Bioinformatics* **5** (2004).

[5] Stanislaus, R., Chen, C., Franklin, J., Arthur, J. & Almeida J.S. AGML central: web based gel proteomic infrastructure. *Bioinformatics* **21,** 1754–1757 (2005).

[6] Jones, A., Wastling, J. & Hunt, E. Proposal for a standard representation of two-dimensional gel electrophoresis data. *Comp. Funct. Genom.* **4**, 492–501 (2003).

[7] Jones, A., Hunt, E., Wastling, J.M., Pizarro, A. & Stoeckert, C.J. Jr. An object model and database for functional genomics. *Bioinformatics* **20**, 1583–1590 (2004).

[8] Pedrioli, P.G.A. et al. A common open representation of mass spectrometry data and its application to proteomics research. *Nat. Biotechnol.* **22**, 1459–1466 (2004).

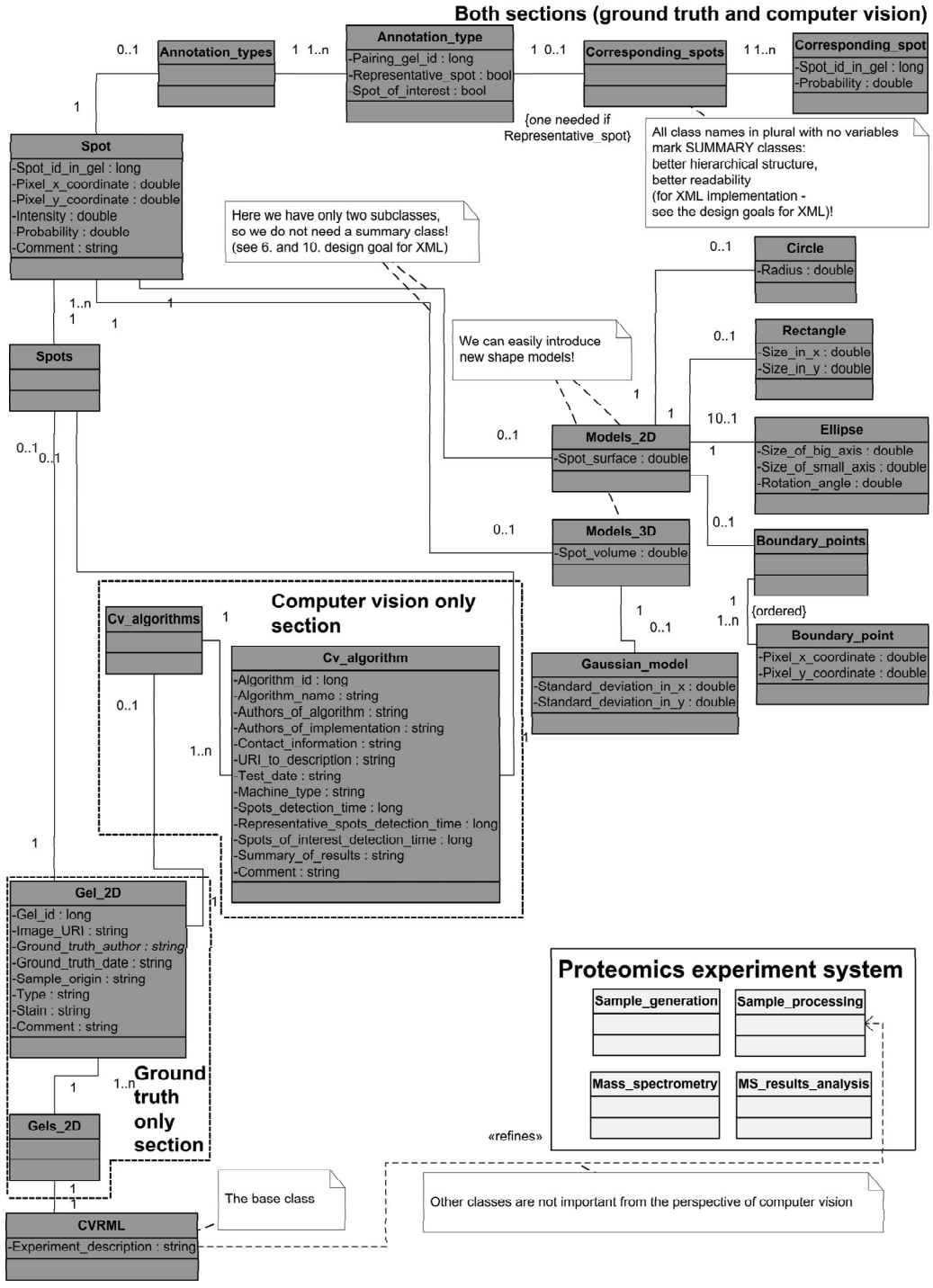

Figure 1: The CVRML conceptual model presented as the UML class diagram. See text for in-depth explanation.