Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche

ISTI Technical Reports

A Graphical User Interface for Medical Image Analysis with Declarative Spatial Logic - Cognitive and Memory Load Evaluation

Giovanna Broccia, CNR-ISTI, Pisa, Italy
Vincenzo Ciancia, CNR-ISTI, Pisa, Italy
Diego Latella, CNR-ISTI, Pisa, Italy
Mieke Massink, CNR-ISTI, Pisa, Italy

ISTI-TR-2021/012

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche

A Graphical User Interface for Medical Image Analysis with Declarative Spatial Logic - Cognitive and Memory Load Evaluation

Broccia G.; Ciancia V.; Latella D.; Massink M.

ISTI-TR-2021/012

Abstract

Logic based (semi-)automatic contouring in Medical Imaging has shown to be a very promising and versatile technique that can potentially greatly facilitate the work of different professionals in this domain while supporting explainability, easy replicability and exchange of medical image analysis methods. In such a context there is a clear need of a prototype Graphical User Interface (GUI) support for professionals which is usable, understandable and which reduces unnecessary cognitive load to the minimum, so that the focus of attention can remain on the main, critical, tasks such as image segmentation in support of planning of radiotherapy. In this paper we introduce a first proposal for a graphical user interface for the segmentation of medical images via the spatial logic based analyser VoxLogicA. Since both the logic approach to image analysis and its application in medical imaging are completely new, this is the first step in an iterative development process that will involve various analysis and development techniques, including empirical research and formal analysis. In the current work we analyse the GUI with a focus on the cognitive and memory load aspects which are critical in this domain of application.

User-Centred design, Cognitive evaluation, Medical image analysis, Spatial logic, Spatial model checking

Citation

Broccia G.; Ciancia V.; Latella D.; Massink M. *A Graphical User Interface for Medical Image Analysis with Declarative Spatial Logic - Cognitive and Memory Load Evaluation* ISTI Technical Reports 2021/012. DOI: 10.32079/ISTI-TR-2021/012

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Area della Ricerca CNR di Pisa
Via G. Moruzzi 1
56124 Pisa Italy
http://www.isti.cnr.it

A Graphical User Interface for Medical Image Analysis with Declarative Spatial Logic - Cognitive and Memory Load Evaluation

Giovanna Broccia[1], Vincenzo Ciancia[1], and Diego Latella[1] and Mieke Massink[1]

Consiglio Nazionale delle Ricerche, Istituto di Scienze e Tecnologie dell'Informazione 'A. Faedo', Pisa, Italy
{giovanna.broccia,vincenzo.ciancia, diego.latella, mieke.massink}@isti.cnr.it

Abstract. Logic based (semi-)automatic contouring in Medical Imaging has shown to be a very promising and versatile technique that can potentially greatly facilitate the work of different professionals in this domain while supporting explainability, easy replicability and exchange of medical image analysis methods. In such a context there is a clear need of a prototype Graphical User Interface (GUI) support for professionals which is usable, understandable and which reduces unnecessary cognitive load to the minimum, so that the focus of attention can remain on the main, critical, tasks such as image segmentation in support of planning of radiotherapy. In this paper we introduce a first proposal for a graphical user interface for the segmentation of medical images via the spatial logic based analyser VoxLogicA. Since both the logic approach to image analysis and its application in medical imaging are completely new, this is the first step in an iterative development process that will involve various analysis and development techniques, including empirical research and formal analysis. In the current work we analyse the GUI with a focus on the cognitive and memory load aspects which are critical in this domain of application.

Keywords: User-Centred Design, Cognitive Evaluation, Medical Image Analysis, Spatial Logic, Spatial Model Checking

1 Introduction and Related Work

Since the discovery of X-ray radiation in 1895, the field of medical imaging has grown up as a scientific discipline. The analysis of patient data, through the acquisition of a variety of medical images ranging from Magnetic Resonance (MR) to X-ray and Computed Tomography (CT) scans, offers many opportunities for diagnosis, therapy planning, and therapy assessment [51]. Several classes of computational methods for the analysis and visualisation of medical images are available in the literature, among which we mention:

- *Image enhancement*: the removal of image distortions (e.g. noise or background anomalies) or the improvement of other relevant properties (e.g. image contours or light) can help the diagnosis and the segmentation of malignant regions [46, 63];
- *Image segmentation*: the identification of the contours of a region of interest or a pattern (see Section 2.1 for an introduction to medical image segmentation);
- *Image registration* [36]: the spatial alignment to achieve anatomical agreement, has emerged as one of the key technologies in medical image computing with applications in diagnosis, therapy, and surgery [52];
- *Quantification*: the calculation of geometrical properties of an anatomical structure from the acquired images (e.g. volume, diameter, curvature), can enable early diagnosis or can help in the therapy planning [1, 58]
- *Visualisation*: the 2-D and 3-D rendering of image data and virtual models of anatomical structures.
- *Computer-aided detection*: the classification of areas in images, based on the presence of signs of specific diseases [29]

Over the years, several techniques have been applied to medical image analysis. Recent advances in machine learning, especially with regard to deep learning, are helping to identify, classify, and quantify patterns in medical images [54]. For instance, of particular relevance to the present work are the case studies on melanoma recognition and brain tumour segmentation. In [24] deep learning techniques have been used for melanoma recognition in dermoscopic images. A number of deep learning approaches have been applied to brain tumour identification and segmentation [38].

In [32] Guarracino et al. propose an algorithm for the automatic segmentation of skin lesion in dermoscopic images; the algorithm has been implemented in Matlab for future comparisons[1].

Recently developed *spatial model checking* techniques have been successfully applied as well to medical image analysis. In [4, 6, 8] spatial logic and spatial model checking have been used for contouring of various kinds of brain tissues and brain tumours. In [7] spatial model checking techniques have been used for the contouring of 2-D images of nevi.

Despite all the cited approaches show good results in the recognition and segmentation of patterns within medical images and, more in general, in the analysis of medical images, the spatial logic approach has the unique advantage of explainability, replicability and exchangeability. The latter consists in the possibility of experts to exchange their methods of segmentation in the form of concise, domain oriented, human readable logical specifications of their methods. In fact, the structure of a procedure specified in spatial logic can be explained, enhanced to contain new observations, and discussed between domain experts. Moreover, mathematically formalised, unambiguous semantics permit the procedure to be replicated with other implementations.

[1] http://www.na.icar.cnr.it/ maddalena.l/SDIplus.html

Within the domain of medical image analysis, there appears to be very little research on the design of suitable graphical user interfaces for systems that support automated analysis procedures, despite the critical and cognitively demanding nature of the work. Professionals that analyse medical images on a regular basis need to manage a variety of information of very different nature such as case information, medical records, dictation systems, reporting facilities, literature search, access to databases and version control, to mention a few [53]. Moreover, examining medical images is a high volume of repetitive work for professionals requiring high precision. This could lead to some signs indicating disease being missed [37] or wrongly interpreted, with potentially critical consequences for the selection of the adequate therapy for the patient. For this reason, it is important that distraction of attention, cognitive load and memory load due to the interaction with the GUI are kept to the minimum. Additionally, the integration of novel and more interactive approaches to medical imaging, such as those based on spatial logic and model checking or the execution of user programs and interactive calibration of parameters, pose further challenges to the interface design. This requires an adequate support for forms of multi-tasking with minimal impact on the focus and on the already demanding cognitive load of domain experts. There is a need for the development of systems based on an improved understanding of users' cognitive processes, including their reasoning and decision making [40].

To the best of our knowledge, systems that support the development of new analysis methods and that automatically can perform such new analysis and show the results on a set of images in (nearly) real time are uncommon. This feature becomes particularly useful when, for instance, a researcher who developed a new analysis method wants to test it on a greater number of cases to check its soundness and stability, or when a clinician wants to analyse more than one image referring to the same patient over time, in order to check the disease progression or regression.

Additionally, the existing medical user interfaces often provide a number of features that are not always relevant for the specific tasks of different classes of users and that could be confusing and cause unnecessary cognitive (over)load [56, 39]. There is a need for a user interface that can be adapted to each type of task [31] and that, as a minimum, complies with the well-known Nielsen heuristics for the design of usable user interface [47].

Finally, although a great number of research efforts have focused on measurement of outcomes of the introduction of information systems in a health care environment, far less work has been conducted in characterising the effects of using these information systems on the cognitive processes involved in the user's interaction with the system [40]. A number of issues found in the design and implementation of interactive systems have their roots in the lack of understanding of the cognitive processes of users [57]. Most of the approaches used to *cognitively* evaluate a system rely on users' opinions, such as collections of questionnaire data and interviews with end-users. For instance, in [41] a methodology for assessing the usability of medical systems is presented. The technique

consists in gathering a dataset containing the transcription of physicians as they "think aloud" in interacting with a system, along with videos recording their interaction. Even if these techniques offer a way of assessing users' needs, they could not identify the needs of all classes of users. Moreover, these methods rely on users' recall of their past experience in using an information system that is often limited and inaccurate [41].

Other cognitive evaluation techniques are performed by the systems' designers in the early stages of design before empirical user testing is possible; among these the most used is the cognitive walkthrough. In the cognitive walkthrough, an analyst chooses a specific task from the set of tasks that the interface is intended to support and determines one or more correct sequences of actions for that task. The technique is used to identify problems with a user interface and to suggest reasons for these problems [42].

Still others address the cognitive processes involved in the interaction with a system, however, these studies focus more on the identification of errors that may occur rather than on usability evaluation. This field of study has its origin in Human Reliability Assessment (HRA) techniques, where nevertheless there was no representation of human cognitive processes within the model of the interaction. The increasing use of formal methods within the study of interactive systems [28] leads to the formal description of expected users behaviour and the formal analysis of human errors. More recently, formal methods techniques have been applied to modelling human behaviour and reasoning (see for example [18–20]) and to the user-centred design of interactive systems (see for example [34, 35]).

`VoxLogicA` is a spatial model-checker that describes spatial properties in high-level specifications written in a logic language providing additional and derived domain oriented operators[2]. Through such specifications the spatial model-checker can automatically and efficiently identify spatial patterns and structures of interest in medical images. Currently the model checker is used by command line invocation, and the results can be visualised with an external image viewer. This situation is far from ideal from a usability point of view, in particular when aiming at a wider uptake of the approach by domain experts, researchers that are not necessarily computer scientists, and data analysts. In order to fully exploit the potential of this promising analysis approach, a better support to the various user groups in the form of a suitable GUI is a prerequisite.

In the present paper, we present the first prototype of a GUI that supports the analysis of medical images through the development and use of spatial logic specifications. The interface we propose is not specific for a single analysis method, though in the current paper it is used with `VoxLogicA`. With the system one can visualise the results of the analyses performed with the spatial model-checker over a dataset of (one or more) images; edit and run a specification to enhance the analysis of selected images; search images in the dataset according to some conditions on numerical indexes (e.g., similarity scores) computed by

[2] `VoxLogicA` is available at https://github.com/vincenzoml/VoxLogicA

the specification; and compare the results of a specification in different images by opening the embedded viewer in the same window.

In this paper, we present a first analysis of the cognitive aspects involved in using the `VoxLogicA` GUI. In particular, we show a theoretical, as opposed to empirical, cognitive evaluation of the interface by comparing the plausible users' cognitive efforts during four use cases, each one analysed in two options: one option with the GUI and the other option command-line based, i.e. without the GUI. We show that using the `VoxLogicA` GUI can lead to a considerable reduction of both the users' cognitive and mnemonic load. We here focus on these two options because the spatial logic approach is very different from other existing segmentation techniques.

The GUI prototype we propose is adaptable, which means that users can show or hide relevant or irrelevant features for their actual purposes. Only the features adopted are shown so that the interaction with the system is not cognitively overwhelming. Moreover, the GUI gives the users suitable feedback and labels to recognise the actions that can be performed in an adequate manner.

In summary, the main contributions of the current paper are the following:

- Presentation of a brief survey of current practise of segmentation of brain lesions in medical images
- Identification of typical tasks in this domain and the identification of suitable use cases
- Presentation of a first prototype of a GUI for spatial logic based methods for image segmentation
- Theoretical cognitive analysis of the proposed GUI for four different domain specific use cases

The remainder of the paper is organised in the following way. Section 2 provides the relevant background on medical image segmentation, related current support tools, typical user tasks and the theory of cognitive load analysis. Section 3 provides an overview of the spatial logic approach to medical image analysis. Section 4 introduces the prototype of the GUI. In Section 5 various typical use cases are analysed from the perspective of cognitive load and memory load. Section 6 concludes the paper and provides an outlook for further research.

2 Background

In this section, we provide the relevant background on various aspects of our work. Section 2.1 gives a brief introduction to medical image segmentation. Section 2.2 surveys some of the commonly used systems that support medical image segmentation. Section 2.3 presents some typical tasks that users perform in this domain. Section 2.4 presents the cognitive systems involved in the interaction between users and systems. Section 2.5 presents a formal model that describes the human behaviour during the interaction with a system.

2.1 Medical Image Segmentation

Segmentation is the process of dividing an image into different meaningful segments. In medical imaging, these segments often correspond to different tissues, organs, diseases, or other relevant structures [30]. Segmentation methods can be divided into three categories (each with its pros and cons): manual, semi-automatic, and automatic.

The first category, the manual segmentation, is performed by an expert, namely a radiologist or a specialised physician: the expert can manually encircle the interested segment or he/she can annotate the voxels of interest in the image. The segmentation performed by an expert is usually regarded as the golden standard and called *ground truth*. However, this method is prone to intra and inter-observer variability and it is highly time consuming.

The semi-automatic method tries to solve some of the limitations encountered in the manual segmentation by assisting the expert with algorithms (e.g. in 3D images by extending the manual segmentation to all slices when the segmentation has been performed slice-by-slice). Several variations exist in the semi-automatic methods: some can assist the physician before or during the manual segmentation, some can finalise the segmentation itself. However, the variability encountered in the first category can still be present since semi-automatic methods still depend on the manual segmentation and the algorithm settings.

For what concerns automatic segmentation, there are a number of variations of this method, all having in common that they do not rely on user interaction. Most of the automatic methods are learning-based and rely on deep learning: the segmentation is automatically performed by a neural network previously trained with labelled data. One of the advantages of this method is that it is relatively fast (once that the network has been trained). However, a plethora of labelled data are required, as well as a long training time. Furthermore, the segmentations used for the training data still have to be produced using a manual or semi-automatic method.

Non-learning-based methods rely on properties of the image and region of interest that are used to perform the segmentation task. Among these methods, recently developed spatial model checking techniques have been applied to image segmentation using specifications written in a logical language to describe spatial properties and to automatically identify patterns and structures of interest. As in the case of learning methods, these techniques depend on a single observer, meaning that the constructed specification will have the same bias as the observer. However, the segmentation procedure supports explainability and easy applicability, as well as being computationally efficient.

An advantage of automatic segmentation methods is that they produce segmentations that are consistent and reproducible, that does not mean that the segmentation is necessarily accurate, but that the procedure produces always the same results. As such, the errors that the method could make in the segmentation are systematic errors instead of incidental errors as in the case of manual or semi-automatic methods [55]. Usually, during their development, automatic seg-

mentations are compared with the ground truth segmentation performed manually.

Be it performed manually or (semi-) automatically, the segmentation procedure is done on an image that we will call *base image*, namely the input image over which the analysis is to be performed. Though for some kind of medical images there exist a single base image – see for instance dermoscopic images captured as 2D images through dermoscopy (a specialised technique of high-resolution imaging of the skin) –, for other kinds of images the analysis can be performed on different base images referring to the same case. In the case of MRI scans, for instance, there exist different images according to the MRI sequence used[3] (e.g. Flair, T1, T2); these different base images are essentially different ways to appear of the same MRI scan. The segmentation procedure produces then a resulting image that can be seen as a *layer* of, or overlay on, the base image.

2.2 Brief Survey of Commonly Used Systems for Medical Image Segmentation

Among the existing systems for the analysis of medical images, we identify three main categories of software developed with a focus on one main feature: viewing of medical images, support to (new) images analysis techniques, and the handling of images datasets. All these kinds of software tools support, to some extent, the analysis of medical images and each of them may show some overlap with the others.

The first category of systems – software tools mainly devoted to viewing – are commonly known as DICOM viewer, where DICOM stands for Digital Imaging and Communications in Medicine and it is the leading standard for image data management in medical applications [44] (see [33] for a comprehensive evaluation of software with advanced functionality for medical images visualisation). A DICOM application aims at storing information in the PACS (Picture Archiving and Communication System) about the image analysis, along with patient details, and when necessary, to view and interpret and (possibly) modify the medical images retrieved from the PACS. Among the DICOM viewers OsiriX[4] is one of the most widely used systems; its commercial version (OsiriX MD), is certified for medical use. OsiriX offers a complete integration with any PACS server, post-processing techniques in 2D and 3D, and techniques for 3D and 4D navigation. Moreover, it is possible to compare two studies by opening them in two different windows.

The systems mainly devoted to the analysis of medical images are particularly useful when conducting research (see [62] for a sampling of software from commercial, government, and research devoted to image analysis). Being able not only to analyse images but also to visualise them, this kind of systems can

[3] MRI sequence is a particular setting of radiofrequency pulses and gradients, resulting in a particular image appearance

[4] https://www.osirix-viewer.com/

as well be used as a DICOM viewer. 3D Slicer[5] belongs to such a group of tools. This software tool is not approved for clinical use but is designed specifically for research purposes. Analyses provided by Slicer include segmentation, registration, and various quantification metrics. Moreover, this system allows researchers to develop new analysis methods in Python and C++: a full code environment is provided where any Python package can be installed and combined with built-in features. Slicer has a built-in Python console and can act as a Jupyter Notebook kernel[6] with remote rendering capabilities.

For what concerns the systems able to handle a dataset of images, the majority of them deal with Deep Learning (DL). Among these, IBM Visual Insights and Google's AutoML Vision are web-based interfaces used to configure and train built-in DL models that can be thus, used as well by users with limited skills in deep learning technologies, to analyse images and video streams for classification and object detection. The functionalities of these tools are still limited to just a few prearranged DL models specific for a limited number of problems. IBM Visual Insights[7] offers a number of features. It predicts whether an image belongs to one or more classes of images based on overall image contents. Users can label, typically manually, the contents of an image, an uploaded video or a live video stream based on user-defined data labels. One can also segment objects in an image and label them or annotate parts of a video where a specific action is taking place. Moreover, it is possible to improve the accuracy of an existing model for object detection by using the auto label function, which uses pre-existing labels in the dataset to generate new ones. The updated dataset can be used to train a more precise model. Finally, one can improve the model using data augmentation: images or video frames in the dataset are *augmented* using filters, such as blur or rotate, to create new versions of them. A new dataset with the augmented images is created that can be used to train a more precise model. A dataset can be a group of images, videos, or both. DICOM images are converted to PNG files for storage and can then be labelled or augmented like any other image.

As far as we know, there do not exist other logic-based tools for medical image segmentation. Therefore a direct comparison with existing systems is not feasible.

2.3 Class of users and tasks in automatic medical image segmentation

Automatic medical image segmentation is an activity that could interest different classes of users, each of which with a main purpose and a set of tasks. Here, we focus on three of such classes: physicians, developers, and researchers.

[5] https://www.slicer.org/

[6] Jupyter Notebook is a web-based interactive computational environment for creating notebook documents. Jupyter notebooks are built upon several popular open-source libraries.

[7] https://www.ibm.com/docs/en/mvi/1.2.0?topic=overview

Physicians are mainly interested in clinical purposes, namely segmenting tissues, organs or illnesses (such as tumours) to eventually recognise a disease reliably and at an early stage. Developers and researchers are more interested in supporting clinical purposes, in order to help clinicians or domain experts in efficiently identifying possible illnesses.

Physicians, for instance, might be interested in viewing and analysing regions of interest (e.g. brain tumour segmentation) identified by an automatic method and that can be shown as a semi-transparent overlay or layer over the original image to evaluate the validity of the method; they could be as well interested in comparing the results of different segmentation methods in order to evaluate the best one; furthermore, they might be interested in analysing the same layers identified by an automatic method in different images of a dataset (e.g. MRI scans of the same patient over time).

On their part, developers might be interested in designing and developing new methods to automatically segment some regions of interest or in running the specification developed by themselves over a dataset in order to testing/refining it.

Researchers, instead, might be interested in analysing a dataset of similar images (such as a dataset of MRI scans) with an automatic method, in order to evaluate the accuracy of the analysis by comparing the automatic segmentation and the ground truth; or they could be interested in comparing the effect of different thresholds in an existing segmentation method. Also, they could be interested in developing their own specification in order to analyse a given dataset.

2.4 Cognitive Systems Involved in the Interaction

During the interaction with a system, one of the most relevant cognitive resources for users is their working memory (WM), a cognitive system responsible for the transient holding and processing of pieces of information. Indeed, in order to accomplish a task, users may need to retrieve information from their WM useful to complete (part of) the task. There is a limit to the amount of information that can be held in WM, generally referred to as *memory span*. In [45], psychologist George Miller suggested that WM has a memory span of seven items plus or minus two. Such items can be organised in higher-order cognitive representation called *chunks* (e.g. when a phone number has to be remembered and digits are gathered together in groups). We call *Memory Load* (ML) the amount of information needed to complete a task (or part of it). Namely, the number of chunks that users need to retrieve from their WM and use during the interaction with a system.

Several models have been proposed to explain how WM works, the most influential is the *multi-component model* proposed by Baddeley and Hitch in 1974 [2]. One important assumption in the multi-component theory is that information in WM decays over time, unless this is prevented by rehearsal. Regarding the nature of such decay, one of the most elaborate theories is the "Time-Based Resource Sharing Model" [5]. According to this theory, items stored in WM are

subject to processing and maintenance activities that use the same cognitive resource, namely the attention. When the processing activities do not require attention, such a resource is addressed to the maintenance activities to refresh the memory items. When attention is drawn away from maintenance activities, items in WM suffer from a time-related decay. Barrouillet et al. [5] define an indicator, called *cognitive load* (CL) that, given a task, measures the temporal density of attentional demand for the task. Specifically, CL gives a measure of the total amount of time during which maintenance of items in WM is impeded, and thus it provides a measure of how much a task is cognitively demanding.

2.5 Formal Model of Human Interaction

In the literature, there are a number of formal models describing the interaction between users and systems. Some of these models are based on a mathematical specification and implementation which supports only simulation; others make use of formal methods and provide an executable model which can be also subject to a range of formal analyses (see [12] for a detailed description of the state of the art on the cognitive models). In the current work, we will use the model presented in [15] that provides a formal description of the cognitive mechanisms involved in the interaction, useful for computing the users' cognitive and memory load while interacting with a GUI such as the one we will present in Section 4. The algorithm underlying the cognitive model has been validated against data gathered from a test conducted with real users engaged in a multitasking interaction with an interactive system [14].

A "task" is modelled as a '::'-separated sequence of subtasks; each subtask is further decomposed into a sequence of basic tasks (simpler actions that can not be further decomposed). Such a partition of tasks into two levels (subtasks and basic tasks) fits with the granularity of the use cases we will analyse in Section 5. Each task is characterised by a measure of how much it is cognitively demanding, namely the cognitive load.

Each subtask composing the task is a (possibly empty) sequence of basic tasks. Between any two basic tasks some time may pass, which correspond to the time required by the system to process the received input. Such time between two basic tasks is called *delay* and it is modelled as a positive real number. Moreover, each basic task is equipped with two additional parameters: the duration and the difficulty (namely the cognitive difficulty), also modelled as positive real number.

Given a set of actions A, a set of interface states S, and a set of information Inf, a basic task is defined as a tuple $\langle j, s, a, k, t, d, \delta \rangle$ and it is represented as:

$$j \mid s \implies a \mid k \textbf{ duration } t \textbf{ difficulty } d \textbf{ delay } \delta$$

where:

- $j, k \in Inf \cup \{noInfo\}$
- $s \in S$
- $a \in A \cup \{noAction\}$
- $\delta \in \mathbb{R}_{\geq 0}$

– $d, t \in \mathbb{R}_{>0}$

It indicates that when the system is in state s and users have inside their WM the information j, they can perform the action a and replace the information j with the information k in their WM; such a basic task has duration t and difficulty d and it is enabled – and thus it can be executed – if and only if the delay δ is elapsed ($\delta = 0$). In [15] it is assumed that users always perceive the state of the system by observing its interface.

Basic tasks can be of two kinds : a *user action* to be performed on the system, and a *cognitive basic task* carried out by users with no involvement of the system.

In the first case, the basic task specifies the action to perform on the system and eventually the information to update in the users' working memory; for instance, the basic task of opening the directory `dataset01` has the form [8]:

$$dataset01 \mid \texttt{datasets} \implies openDir \mid noInfo$$

and means that when the system is on state `datasets` (supposing that the state of the system corresponds to the current directory) and users have in their working memory the information about the name of the directory they are looking for (`dataset01`), they can open the directory and remove the information about its name from their working memory.

In the second case, the basic task only specifies that the item to update in the memory must be the mental plan of the user, and the action to perform must be equal to *noAction* (since there is no involvement of the system). For instance if users have to find the name of the directory `dataset01` they have not to perform any kind of action on the system – which remains on the state `datasets` – but they have to insert in their working memory the information about the directory's name; such basic cognitive task has the form:

$$noInfo \mid \texttt{datasets} \implies noAction \mid dataset01$$

For what concerns the cognitive load, in [15] a formula for estimating CL is presented that is a modified version of that presented in [5], taking into account the time between two basic tasks. The CL is computed on a subtask basis. In the present paper we take the sum of the CLs of the subtasks as a measure of the CL of the entire task.

Given a subtask ST, CL of ST is computed as the ratio of the *difficulty factor* of the subtask – namely a measure of temporal density of difficulty of the subtask – (defined in Equation 2), by the *total duration* of the subtask ST (defined in Equation 3):

$$CL_{ST} = \frac{DF_{ST}}{TD_{ST}} \tag{1}$$

where:

[8] For the sake of simplicity, we omit in these examples the parameters **duration**, **difficulty**, and **delay**.

– DF_{ST} denotes the difficulty factor of the subtask ST, computed as:

$$DF_{ST} = \sum_{k=0}^{N-1} duration_{BT,k} \times difficulty_{BT,k} \tag{2}$$

– TD_{ST} denotes the total duration of the subtask ST, computed as:

$$TD_{ST} = \sum_{k=0}^{N-1} duration_{BT,k} + delay_{BT,k} \tag{3}$$

where $duration_{BT,k}$ and $difficulty_{BT,k}$ represent respectively the duration and the difficulty of the k-th basic task of ST, and N is the number of basic tasks composing subtask ST.

Using this theory and, in particular, the estimation of the cognitive and memory load, is especially useful at the early stages of a system design, to better understand the effort necessary to operate the system when a full implementation of the system is not yet available. Moreover, the analysis can also be used retrospectively, to analyse already implemented systems and complement results from user studies. In [13], for instance, the theory has been used to check whether a design solution for an infusion pump could be adopted to reduce the memory load.

In the present paper, we use the theory to analyse a number of use cases in order to check whether some design and implementation decisions could reduce the memory and cognitive load.

3 The Spatial Model Checker VoxLogicA

Spatial model checking [21, 22, 4, 9] is a novel variant of model checking in which properties of physical space, expressed in a spatial logic, can be checked automatically for a spatial model. Typical spatial models may be represented as graphs. Images composed of pixels can be seen as particular forms of regular graphs, or grids, where each node in the graph represents a pixel (in 2D images) or voxel (in 3D images). Pixels or voxels can have particular features, such as their luminosity or intensity or their colour. These basic features can be used in the logical properties. Examples of properties can be simple ones, such as find all pixels with a luminosity above a certain threshold, or more complex ones, such as find all bright pixels that are surrounded by pixels that are at at most 10 mm from a dark pixel. In the following we provide a brief overview of the spatial model checker VoxLogicA.

3.1 ImgQL: the Spatial Query Language of VoxLogicA

In `VoxLogicA` spatial properties are defined as high-level specifications. Through such specifications, written in a logic language, a spatial model checker can automatically and efficiently identify in 3D voxel-based (medical) images spatial patterns and structures of interest. The input language of `VoxLogicA` [9] is ImgQL

(*Image Query Language*). ImgQL [4, 9] is a spatial logic language developed for the analysis of medical images, based on SLCS (*Spatial Logic for Closure Spaces*) [21, 22]. `VoxLogicA` is a *global* spatial model-checker in the sense that, given a model $\mathcal{M}$ (i.e. a digital image) and a formula Φ, it computes the set $[\![\Phi]\!]^{\mathcal{M}}$ of all points in the image that satisfy Φ. Such a set of points are represented by a boolean image, namely a model of the same dimension of $\mathcal{M}$ whose points are assigned the value true if the corresponding voxel satisfies Φ and false otherwise. Moreover, in `VoxLogicA` one can also obtain a *numerical* image whose points are assigned the numerical value computed for the verification of a formula Φ on the corresponding voxel of $\mathcal{M}$, as we will see below in the case, for instance, of the texture analysis operator $\triangle\!\!\!\!\triangle$. Below we provide an informal account of the language, referring to [9, 7, 23] for its formal definition. The syntax of ImgQL is the following, where p stands for an atomic predicate (e.g. the assertion that the level of attribute 'intensity' of the voxel at hand is below to a certain threshold):

$$\Phi ::= p \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \rho\Phi_1[\Phi_2] \mid \mathcal{D}^I\Phi \mid \triangle\!\!\!\!\triangle$$

where:

- Negation ($\neg$) and conjunction ($\wedge$) are classical boolean operators;
- The *Reachability* operator $\rho\Phi_1[\Phi_2]$ is satisfied by a voxel x in an image (a model $\mathcal{M}$) if there is a path π of *adjacent*[9] voxels in $\mathcal{M}$ starting from $x = \pi(0)$ and leading to a voxel $\pi(l)$ that satisfies Φ_1 and $\pi(j)$ satisfies Φ_2 for all j such that $0 < j < l$; in other words, it is satisfied by all the voxels that lay on a path leading to a voxel satisfying Φ_1 and composed exclusively of voxels satisfying Φ_2 (except for x).
- The *Distance* operator $\mathcal{D}^I\Phi$ is satisfied by all the voxels that are at a distance falling in interval I from a voxel that satisfies Φ;
- The *Texture Analysis* operator $\triangle\!\!\!\!\triangle$ is satisfied by a point x if the correlation between the value distribution of a specific attribute in a region of interest around x and that of an attribute in the area specified by a logic formula satisfies a given constraint; the distributions are represented by the histograms of the two areas; the attributes of interest, the radius of the sphere characterising the region of interest, the features of the histograms and the constraint are given as parameters of the operator $\triangle\!\!\!\!\triangle$; for more details on this operator we refer the reader to [7]

Classical derived operators are defined in the usual way. In addition, there are a number of more specific derived operators:

- The *Near* operator $\mathcal{N}\Phi$ is satisfied by a voxel x in $\mathcal{M}$ that can reach *in at most one step* a voxel y in $\mathcal{M}$ satisfying Φ;

[9] The number of adjacent voxels depends on the adjacency relation chosen. If the *orthogonal adjacency* relation is chosen, each voxel is adjacent to the set of voxels which it shares an edge with; otherwise, if the *orthodiagonal adjacency* relation is chosen, each voxel is adjacent to the set of voxels which it shares both an edge and a vertex with.

– The *Touch* operator $touch(\Phi_1, \Phi_2)$ is satisfied by all the voxels that satisfy Φ_1 and that lay on a path that eventually leads to a voxel satisfying Φ_2 passing only by voxels that satisfy Φ_1;
– The *Grow* operator $grow(\Phi_1, \Phi_2)$ is satisfied by all the voxels that satisfy Φ_1 or that start a path composed of voxels that all satisfy Φ_2 and that leads to a voxel satisfying Φ_1 in the last step.
– The *Smoothen* operator $smoothen(r, \Phi_1)$ is satisfied by all the voxels that are at a distance of less than r from a voxel that is at least at distance r from voxels that do not satisfy Φ_1.

Fig. 1 and Fig. 2a show some simple examples of some of the above mentioned operators.

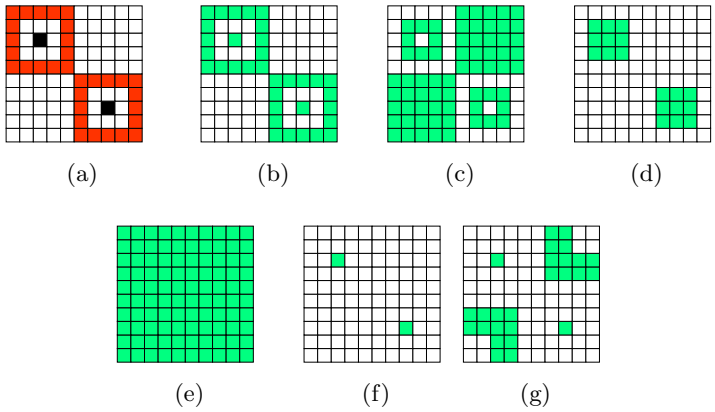


Fig. 1: An example model (1a); the points shown in green are those satisfying $black \vee red$ (1b), $\neg(black \vee red)$ (1c), $\mathcal{N} black$ (1d), $\rho\ red[white]$ (1e), $black \wedge \rho\ red[white]$ (1f), and $\mathcal{D}^{[2,3]} red$ (1g).

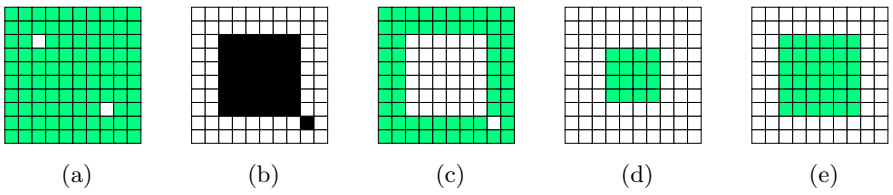
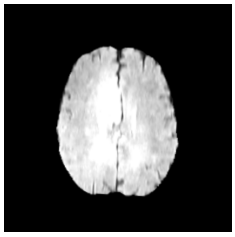
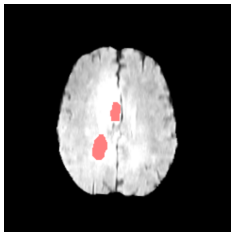


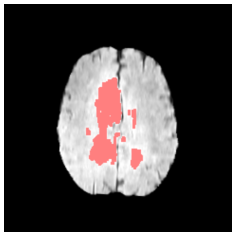
Fig. 2: With reference to the model in Figure 1a, Figure 2a shows in green the points satisfying *grow*(*red*, *white*). In Figure 2c (2d, 2e, respectively) the points of the model shown in Figure 2b that satisfy $\neg black$ ($\mathcal{D}^{\geq 2}(\neg black)$, $\mathcal{D}^{<2}(\mathcal{D}^{\geq 2}(\neg black))$—i.e. *smoothen*(2,*black*)—respectively) are shown in green.



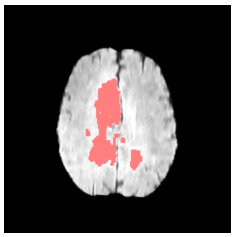
(a) Axial 2D slice of Brats17_2013_2_1



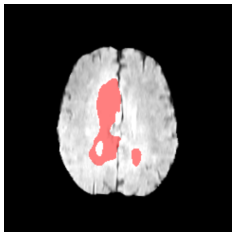
(b) Points satisfying `hI` (line 4 of Spec. 1)



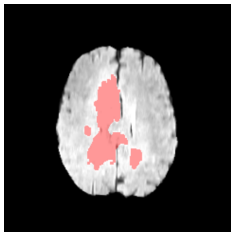
(c) Points satisfying `vI` (line 5 of Spec. 1)



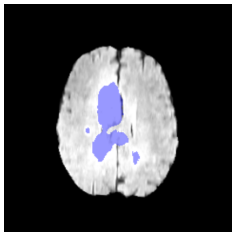
(d) Points satisfying `growTum` (line 8 of Spec. 1)



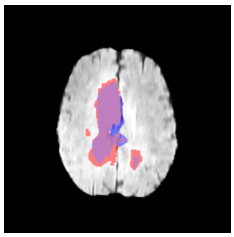
(e) Points satisfying `tumStatCC` (line 10 of Spec. 1)



(f) Points satisfying `gtv` (line 11 of Spec. 1)



(g) Manual segmentation



(h) Comparison between `gtv` (red) and manual segmentation (blue). Overlapping area is purple.

Fig. 3: Illustration of the segmentation procedure presented in Specification 1 of image `Brats17_2013_2_1`, FLAIR, axial 2D slice at $X = 155$, $Y = 117$ and $Z = 97$ (fig. 3a).

ImgQL Specification 1: Tumour segmentation method (excerpt)

```
1  let background = touch(intensity <. 0.1, border)
2  let brain = !background
3  let pflair = percentiles(flair,brain,0)
4  let hI = pflair >. 0.95
5  let vI = pflair >. 0.86
6  let hyperIntense = flt(5.0,hI)
7  let veryIntense = flt(2.0,vI)
8  let growTum = grow(hyperIntense,veryIntense)
9  let tumSim = similarTo(5,growTum,flair)
10 let tumStatCC = flt(2.0,(tumSim >. 0.6))
11 let gtv= grow(growTum,tumStatCC)
12 save "gtv.nii.gz" gtv
13 print "00_dice_gtv" diceM(gtv)
```

`VoxLogicA` also offers the possibility of defining in ImgQL and saving a number of similarity indexes, to compare the level of similarity between two segmentations of the same image, in particular, the similarity between a manual segmentation (ground truth) and the automatic `VoxLogicA` segmentation. Functions and predicates are defined in `VoxLogicA` in the usual way.

Specification 1 shows the main part of the ImgQL segmentation procedure used to identify Glioblastoma multiforme (GBM), the most common brain tumour whose segmentation is crucial for one of the first-line treatments, radiotherapy. For simplicity, loading operations of images are not shown in the specification. In the excerpt, the base image is referred to as 'flair'. The intensity of each voxel in 'flair' is referred to as 'intensity'. Figure 3 shows some of the intermediate phases of the procedure presented in Specification 1.

Lines 1-2 define the background as the region of all voxels, with intensity less than 0.1, that touches the border of the image, and the brain as the complement of the background. The predefined property 'border' is satisfied by all voxels at the border of the image. The sub-expression $intensity < .0.1$ is satisfied by all voxels of the image with an intensity below 0.1. The percentiles operator in line 3 assigns the percentile rank of the intensity of those voxels in the base image 'flair' that are part of the brain. Based on these percentiles, hyper-intense and very-intense points are identified that satisfy `hI` (line 4) and `vI` (line5), shown in red in Fig. 3b and in Fig. 3c, respectively. In lines 6-7 the hyper-intense and very-intense points are filtered, thus removing noise, and considering only areas of a certain relevant size. In line 8 the areas of hyper-intense points are extended via the grow operator. The points satisfying `growTum` are shown in red in Fig. 3d. In line 9 the similarity operator is used to assign to all voxels a texture similarity score with respect to the area identified by `growTum`. In line 10 this operator is used to find those voxels that have a high cross correlation coefficient and thus are likely part of the tumour. The result is shown in Fig. 3e. In line 11, the voxels

that are identified as part of the whole tumour are those that satisfy `growTum` extended with those that are statistically similar to it via the grow operator. Points that satisfy `gtv` are shown in red in Fig. 3f. As an example, in line 12 the points satisfying `gtv` are saved as a resulting image that can be further analysed as an overlay on the base image. Other layers can be saved in a similar way for any of the intermediate definitions in the specification, such as `veryIntense`, `growTum` and so on. In line 13 the Dice similarity index between `gtv` and the manual segmentation is computed and saved.

The specification has been validated in [9] using the 2017 BraTS dataset [43] containing magnetic resonance imaging (MRI) scans of 210 patients affected by GBM. The interested reader is referred to [9] for a detailed description of the approach and the specification. `VoxLogicA` can save intermediate and final results. The resulting images can be visualised as *layers* on the base image.

3.2 The VoxLogicA Directories Hierarchy

A `VoxLogicA` release consists of an executable file accompanied by a number of libraries, that must reside in the same directory as the executable. In order to run `VoxLogicA` from the command-line, users need first to identify the main executable, residing in the tool directory. The name of the executable is "VoxLogicA" on Linux and mac-OS systems, and "VoxLogicA.exe" on Windows systems.

The hierarchy of the `VoxLogicA` directory depends, of course, on how users create their sub-directories and on how they save the analysis results. A well-organised method is to have, inside the VoxLogicA directory, a sub-directory for specifications, a sub-directory for datasets and a sub-directory for the results, in addition to the libraries and the script/executable file for running the tool.

Going more in detail in such hierarchy:

- The `specifications` directory contains a sub-directory for each specification describing the analysis to be executed.
- The `datasets` directory contains a sub-directory for each dataset analysed, each of which contains a further sub-directory for each item in the dataset, containing the image over which the analysis is executed and the associated ground truth segmentation (i.e. the segmentation made by a human expert). In some cases, these sub-directories could contain a variable number of images to analyse referring to the same item in the dataset, according to the way the images are taken – e.g. in the case of brain tumour segmentation, there are different images according to the MRI sequence used[10] (e.g. Flair, T1, T2); we will refer to these images as "base images".
- The `results` directory contains a sub-directory for each specification used, inside of which there is a number of further sub-directories for each dataset analysed through *that* specification. Inside this latter directory there is a

[10] MRI sequence is a particular setting of radiofrequency pulses and gradients, resulting in a particular image appearance.

sub-directory for each "session" (namely for each different analysis executed with the same specification over the same dataset). The session directory contains a sub-directory for each item in the dataset, in turn containing the resulting images saved during the analysis, and a .csv file containing the printed values for each similarity index defined in the specification for each item in the dataset.

Figure 12 in Appendix shows an example of the `VoxLogicA` directories hierarchy.

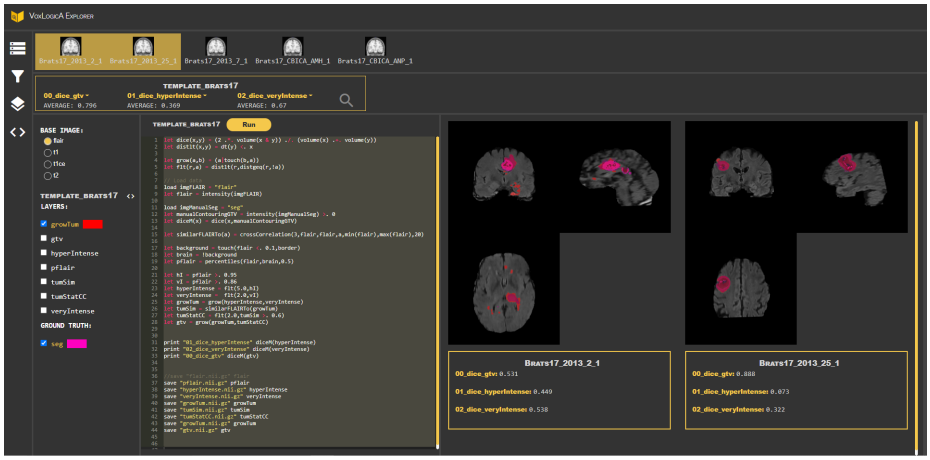


Fig. 4: VoxLogicA Graphical User Interface

4 A Graphical User Interface for VoxLogicA

Graphical user interfaces are being increasingly used to provide users of formal analysis and simulation tools with a friendly and visual approach to specifying all input parameters, as well as reading related results, and increased configuration flexibility [26].

Here, we present a prototype of a GUI for `VoxLogicA`, whose aim is to give appropriate support to images formal analysis, managing complicated tasks (possibly demanding from a cognitive and mnemonic point of view) and a large amount of information. It is a prototype implementation that can be used for theoretical analysis – as those presented in Section 5 – and for further empirical analysis with real users in future work, when the pandemic condition will allow this again.

4.1 GUI functional description

Figure 4 shows the complete user interface. Below we describe its elements. The name of each item also introduces the name of the respective GUI element.

Dataset row A section on the top row of the window shows the dataset of images (see Figure 5); to analyse one or more images more in detail, users can open/close by clicking on the thumbnail images. This action will open/close an embedded DICOM viewer for each image in the *work space* that we will present below, and will change the thumbnail images background colour to yellow.

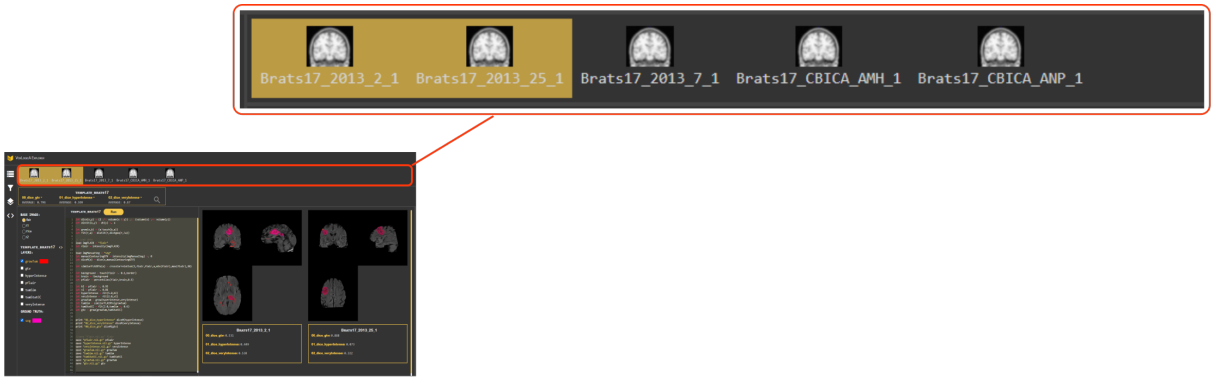


Fig. 5: Detail on dataset row

Indexes row A section immediately below the dataset row (see Figure 6) shows the information about the similarity indexes computed by a given ImgQL specification (the specification name is written at the top of the section) for the dataset shown in the row above. In Figure 6 the specification name is "TEMPLATE_BRATS17" and the names of the indexes are "00_dice_gtv", "01_dice_hyperintense", and "02_dice_veryIntense".

For each index the average value, the standard deviation, and the minimum and the maximum value w.r.t the ground truth are computed. Users can select one of these computations from the drop-down menu appearing by clicking one of the indexes.

Moreover, in this section one can look for items in the dataset with specific characteristics by clicking on the search icon to the right of the indexes names. On clicking the icon a box is opened where users can change the research settings for each similarity index that can be above, below or between some values set by the user (see Figure 7a); each condition can be inverted by clicking on the "invert" box; by setting more than one condition they will be combined in conjunction.

If the user sets incorrectly one of the search parameters (e.g. a user forgets to set a value), the system displays an error message in natural language with an indication of the nature of the error and a suggestion on how to solve it (see Figure 8). The number of items found is shown to the right of the icon that, when activated, changes its colour to light blue, as it can be seen

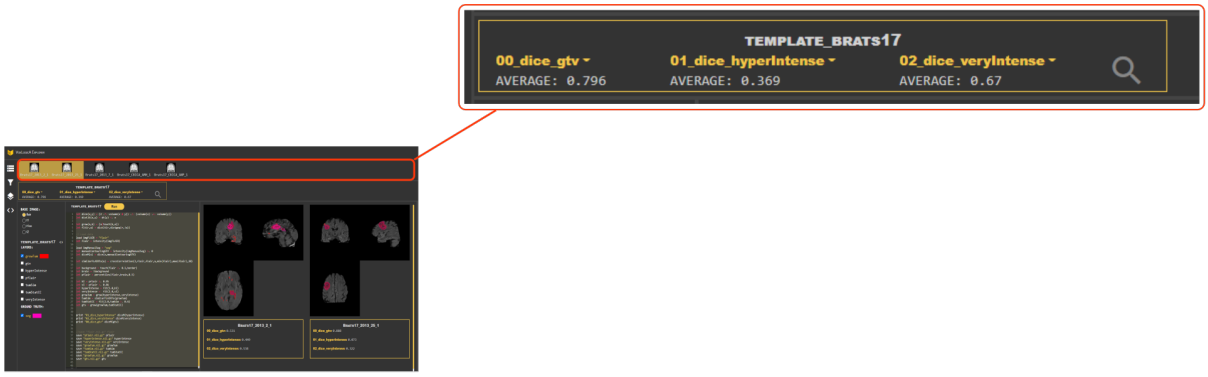
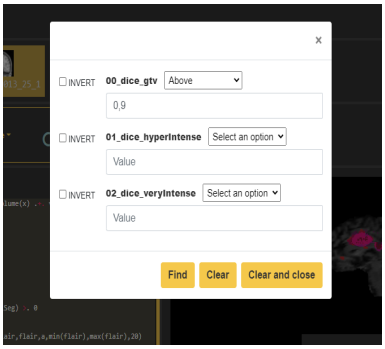
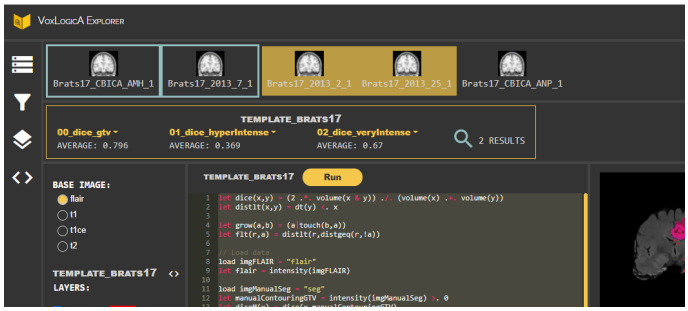


Fig. 6: Detail on indexes row



(a) Search box in the VoxLogicA GUI



(b) Search results in the VoxLogicA GUI

Fig. 7: Details on search behaviour

in Figure 7b. A border of the same colour identifies the items found in the dataset row. Users can clear and reset the search settings from the search box.

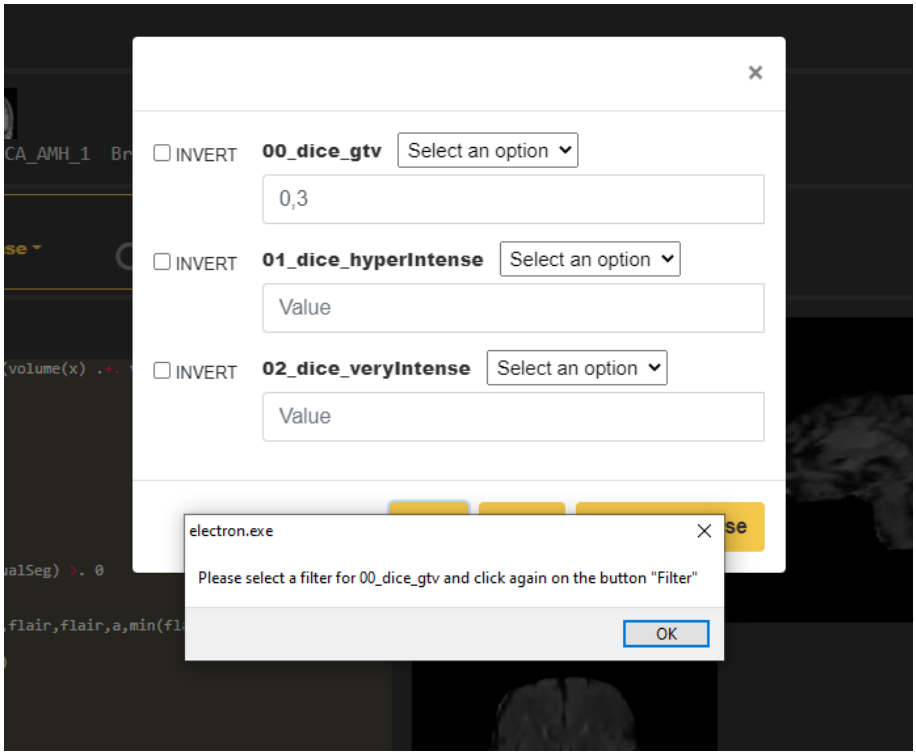


Fig. 8: Example of error message

Layers column A section in the bottom-left column of the window shows the list of base images and the list of overlays saved with a given ImgQL specification; the title "BASE IMAGE" precedes the list of base images, the name of the specification used to save the overlays precedes the list of overlays (see Section 2.1 for further details on base images). Recall that overlays are included in the specification (see Section 3). Figure 9 shows a detail of the column, where the base images are "flair", "t1", "t1ce", and "t2", the name of the specification is "TEMPLATE_BRATS17", and the overlays are those saved by the specification, namely "growTum", "gtv", "hyperIntense", "pflair", "tumSim", "tumStatCC", and "veryIntense".
By clicking on one of the base images, users activate it: when they will click a case from the dataset row, the system will display the activated base image in the viewer referring to the chosen case in the work space (presented below). It is not possible to select more than one base image, neither to deselect all

of them. The list of base images can be hidden/shown by clicking on "BASE IMAGE:".

Moreover, by clicking one or more overlays, users can activate/deactivate them: they will be opened (if not already opened) as layers of the base image in the work space. The system automatically selects a different colour for each layer. The list of overlays can be hidden/shown by clicking on the name of the specification, that is located above the list ("TEMPLATE_BRATS17" in Figure 9). Furthermore, by clicking the code icon to the right of the specification's name, one can show/hide the specification code on the *code* column (presented below). If present, here it is possible to open the ground truth as a layer, shown after the title "GROUND TRUTH" ("seg" in Figure 9)

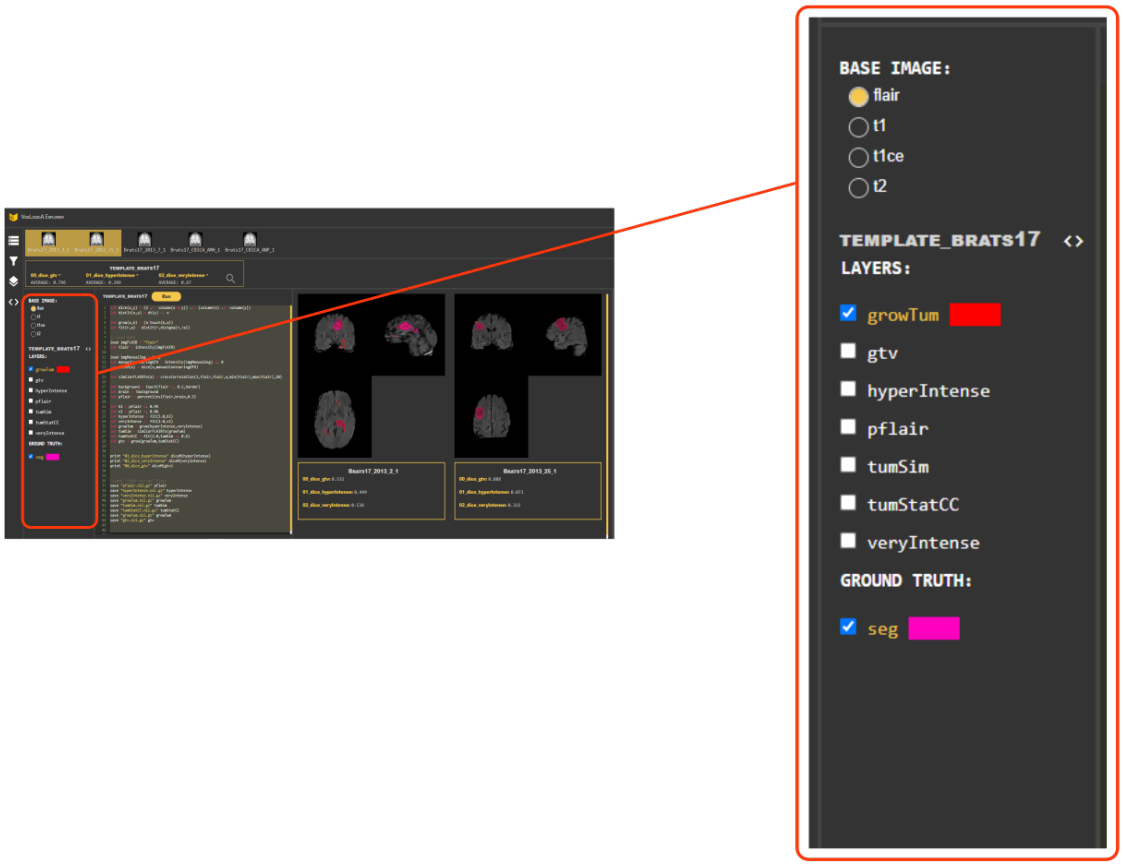


Fig. 9: Detail on layers column

Code column A section in the bottom-middle column of the window shows the ImgQL code of a specification (whose name is written just above the code) through an embedded code editor. Users can edit the code and run the modified specification on the database's open cases (i.e. those opened in the work space) by clicking on the button "Run"; if no cases are open, the button is not active. As long as the new analysis is running, the GUI shows a box containing a progress bar and the analysis log (see Figure 10a). In case of error, the log window shows its details. The resulting layers (computed only over the opened cases) are shown in the layers column (see Figure 10b). The indexes computed with the new specification are updated in the indexes row.

Work space A section in the bottom-right column of the window displays the open cases in the dataset (instantiated with the active base image) and the active overlays through embedded image viewers. The colour of an overlay corresponds to the colour indicated at the name of the overlay in the overlays column. The order in which overlays are opened as layers on the base image depends on the order in which users select them. Overlays automatically set transparency in order to make the layers opened previously visible. The colours (and layers) order is maintained for each further opening of a new case from the dataset row. To change such an order users need to close and reopen the overlay they want to visualise as the first layer. When many cases are opened, the work space becomes *scrollable* in order to avoid reducing the visual size of images. Recall that filters can be used to identify relevant images to display.

Below each image viewer a box is shown with the information about the open case: its name and the values for the computed similarity indexes.

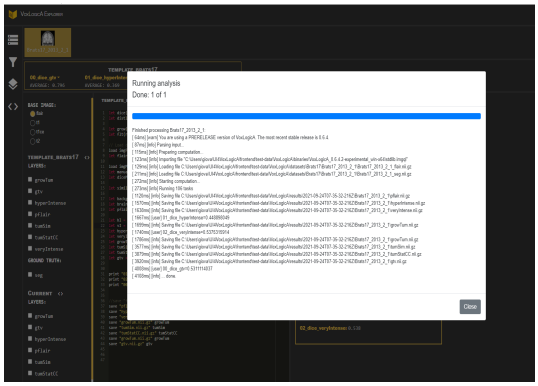
Icons column A column on the left side of the window shows four icons by clicking on which one can show or hide some of the sections in the window (see Figure 11). Specifically, the four icons as they appear from the top of the column open respectively the dataset row, the indexes row, the layers column, and the code column.

4.2 GUI User-Centred Design

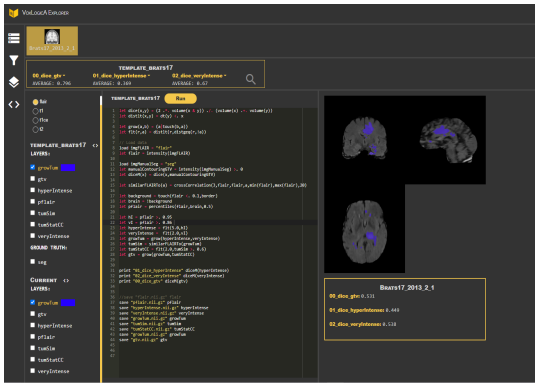
User-centred design (UCD)[11] is a process in which, at each stage of the design process, great attention is posed to usability goals, user characteristics, environment, and tasks [61]. We used this framework for the development of the GUI, posing at the centre of the design process not only the need to develop an interface able to adequately support a number of tasks but also the need to successfully comply with a usability inspection.

For what concerns the support to the tasks we identified a combination of GUI elements motivated by the various uses that we think the users of our system will perform: examining patients images and overlays, comparing results of

[11] Also called human-centred design or user-driven development



(a) Progress bar and analysis log



(b) Resulting layers

Fig. 10: Details on analysis behaviour

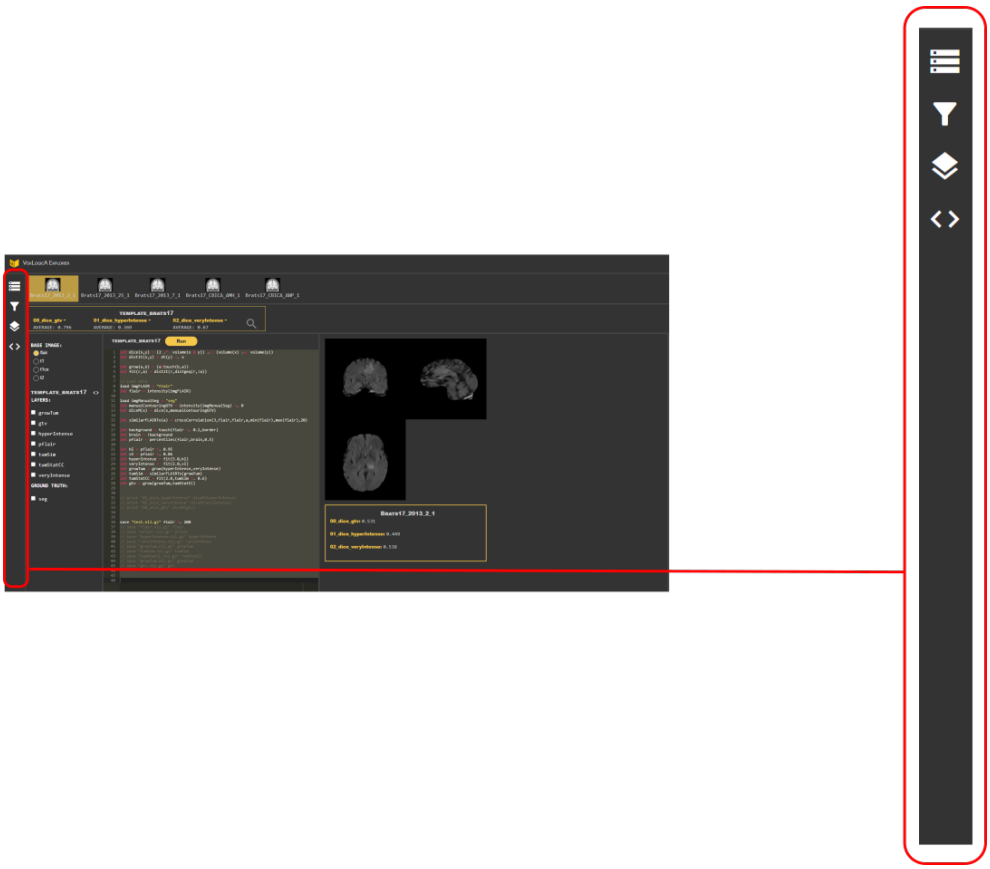


Fig. 11: Detail on icons column

different patients, designing new segmentation methods, comparing segmentation methods, comparing the effect of different thresholds. Each of those uses is envisioned and performed by the various classes of users that we identified and presented in Section 2.3. The features needed to support the tasks – dataset handling, multiple DICOM viewers, an ImgQL editor, and the spatial model checker – have been organised in a grid layout. In this way, all the classes of users can handle their tasks and display the features they need on a single window. With the help of the icons in the icons column, they can as well hide sections that are not relevant for their current task.

For what concerns the usability inspection we performed a heuristic evaluation using the ten heuristics proposed by Nielsen in [47].

Below we address Nielsen's heuristics and how our GUI faces each of them.

1. **Visibility of the system status:** a system should always inform users about its status through appropriate and timely feedback
   In the GUI we present, every time the system changes its status, users are informed through a visual change: e.g. when users click a button in the dataset row, the button background becomes yellow and a DICOM viewer is opened in the work space; every time users click on an overlay in the layers column, the layer name becomes yellow and its box is checked; every time users click on an icon anywhere in the GUI window, the corresponding section is shown or hidden.
2. **Match between system and the real world:** a system has to speak the users' language: use words, phrases, and concepts familiar to users, rather than internal jargon.
   In the `VoxLogicA` GUI each behaviour is described using natural language and elements used are common elements in the application domain: e.g. error messages are in natural language; in the search box, the logical operators and the conditions on the indexes values are expressed in natural language ("invert", "above", and "below" rather than "not", "¿", and "¡"). For what concerns the icons we used, they have a form which corresponds to actual objects in the real world or to well-established forms, with which users are familiar. Moreover, we provide all of the icons with a title, which means that when the user keeps the mouse over them, their name appears.
3. **User control and freedom:** a system has to provide an evident "emergency exit" to leave the unwanted actions (i.e. actions performed by mistake) without having to wait for an extended process.
   Within the `VoxLogicA` GUI each action can be quickly inverted: e.g. if users open dataset cases or layers by mistake they can close them by clicking again on the corresponding button. For what concerns the launch of an analysis, which is, by the way, the longest GUI process, we plan to implement a button with which users can stop the run.
4. **Consistency and standards:** a system has to be consistent with both the platform (namely similar actions performed on the system should produce similar outcomes) and the industry conventions (namely names, words and actions should be common to those of the domain).

For what concerns the consistency of the GUI, in our system similar actions produce similar results, e.g. one can open and close a GUI element by clicking on a button or an icon: icons open/close row and column, buttons in the dataset row open/close DICOM viewers in the work space, buttons in the input/output column open/close layers or base images in the work space; moreover, the colour of active elements (namely those that have opened an element) is consistent according to the type of element: icons are white, buttons are yellow. For what concerns the consistency with industry conventions, the GUI follows those conventions; e.g. we use check-boxes for multiple choices and radio buttons for single choice.

5. **Error prevention:** a systems has to prevent problems from occurring.
   Though best efforts have been made in the design and implementation phase of this first prototype to prevent errors, further formal and empirical analysis is required to verify correctness properties. Results on this aspect will be treated in forthcoming work[12].
6. **Recognition rather than recall:** the system has to minimise the user's memory load by making elements, actions, and options visible.
   The `VoxLogicA` GUI has been designed in order to allow users to find information in a single window without having to remember them or look for them in the file system. More specifically a memory load evaluation has been performed for selected use cases and it is presented in Section 5.
7. **Flexibility and efficiency of use:** a system has to provide shortcuts that may speed up the interaction for experienced users, such that the design can cater to both inexperienced and experienced users.
   More than considering users with different levels of experience, the GUI we propose takes into account different classes of users that might be interested in different tasks (and thus need different GUI configurations). Therefore we propose some simplifications in the interface appearance that can speed up the completion of a number of tasks (e.g. users interested only in using the DICOM viewer to analyse a resulting image can close the code editor that could distract them from their current task and lower their performance).
8. **Aesthetic and minimalist design:** a system should contain only relevant and needed information.
   Our GUI not only contains just the elements that we consider relevant for the tasks of selected classes of users, but it is also adaptable: users can show or hide features for their purposes so that the system shows only those effectively adopted. Indeed, users can show/hide each section in the application window by using the four icons on the icons column (except for the work space, which is always present). The interaction with the system thus has a minimal impact on the cognitive load and also reduces possible confusion to the minimum.
   For what concerns the aesthetic aspects, we organised the GUI elements in a grid system, a layout that users are used to deal with since, historically, it

[12] As soon as the pandemic condition would allow, we plan to conduct a usability test with real users to find potential errors.

has been first used to arrange handwriting on paper and then in publishing. Having low complexity, this layout is perceived as having low complexity by the users [59, 60, 10]. Moreover, the different sizes and positions of the sections convey a hierarchy between elements that helps users understanding the role of each section [25, 49].
We selected a palette of a very limited number of colours to convey a message of simplicity. The GUI has a dark theme in order to reduce the luminosity emitted by the device screen, while still meeting the colour contrast ratios (we leave the appearance customisation for future development). Moreover, the dark theme helps improve visual ergonomics by reducing eye strain, adjusting brightness to current lighting conditions, and facilitating screen use in dark environments [27]. The contrast between background and text is a factor that influences the users' attention [3] and it is required by Web Content Accessibility Guideline (WCAG), one of the web accessibility guidelines published by the Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C). WCAG level AAA (the highest level) requires a contrast ratio of at least 7:1 for normal text and 4.5:1 for large text. Our interface uses white elements on a dark grey background, and it has a contrast ratio of more than 12:1[13]. Moreover, in order to help users to spot easily interactive elements and to highlight the active elements in the GUI, we use a palette of two colours (yellow and light blue), making the system consistent.

9. **Help users recognise, diagnose, and recover from errors:** a system should present error messages expressed in natural language indicating precisely the problem (with no error codes) and suggesting a solution.
   At the moment, the GUI we present can show users two kinds of error messages: one is related to search settings and the other one is related to running analysis. In both cases, the GUI shows users a message in natural language indicating how to fix the problem (see Figure 8).
10. **Help and documentation:** A system should provide the necessary documentation to help users understanding the system functioning. The best system is that which does not need explanation.
    At the moment, we do not provide any documentation to help users understanding the GUI. We plan to implement a wizard in order to explain to users how to complete their tasks on the GUI.

4.3 GUI Implementation

The `VoxLogicA` prototype user interface has been implemented using HTML, CSS and JavaScript and it runs as a desktop application through Electron [14], an open-source software framework which allows for the development of desktop GUI applications using web technologies by combining the Chromium rendering engine and the Node.js runtime.

[13] Computed online with the contrast ratio checker at https://webaim.org/resources/contrastchecker/
[14] https://www.electronjs.org/

In particular, the desktop version has been preferred for security and performance reasons. As regards security, since the interface needs to process and use datasets of medical images, it is preferable not to share sensitive data on the web. For what concerns performance, using the system from desktop ensures better performances, also considering the GPU-based version of `VoxLogicA` that is currently under development [16].

5 Theoretical Cognitive Evaluation

In this section, we present an evaluation of the `VoxLogicA` GUI based on the cognitive processes underlying the interaction with the system. In particular, we show how the use of a GUI can reduce the cognitive efforts of users by computing and comparing the cognitive and memory load for pairs of use cases whose goal is the same, one performed with the GUI and one performed without it i.e. using the different components of the system, such as a DICOM viewer and spreadsheets from the command line. Being the first GUI prototype that supports a combination of novel features (i.e. the analysis of medical images via spatial logic, the visualisation of the results of the analysis performed over a dataset of (one or more) images, the edit and running of a specification to enhance the analysis of selected images, and the filtering of dataset images according to some conditions on the similarity indexes) a direct comparison of the cognitive and memory load of the GUI prototype proposed in the current work with the cognitive and memory load of alternative systems was not feasible. We therefore focused on a comparison between the use of the features from command line and their use when offered via an integrated GUI.

In order to evaluate how much demanding each subtask is, both from the cognitive point of view and from the mnemonic one, and in order to compare the pairs of use cases we will follow the method presented in [15].

In [15] each basic task is characterised by the three parameters we have already recalled in Section 2.5, namely duration, difficulty and delay. While the delay is a parameter that depends on the system for the way it responds to input, duration and difficulty are subjective parameters that depend respectively on how a user performs an action (i.e. how long he/she takes to perform an action) and the user's perception of the cognitive difficulty of an action. Moreover, among these three parameters, delay and duration are easily measurable, while the user's perception of difficulty is not. Although there is an objective difference when comparing the intrinsic difficulty level of some activities – clicking on a button is undoubtedly easier from a cognitive point of view than writing a command on a shell since the latter requires more mental effort in retrieving from the memory the right sequence to be written – giving a numerical and precise value to rate the perceived cognitive difficulty of the activities one wants to analyse can be complicated, especially considering that different classes of users might have different perceptions of some actions.

A common way to rate the difficulty of a set of actions is to use a user's estimate. In [17, 50] users are asked to rate a set of actions using a cognitive load

scale provided by Paas and Van Merrienboer. However, as already mentioned, we could not conduct a user test yet due to pandemic conditions. Therefore, we performed a theoretical evaluation, using the Paas mental-effort rating scale [48], that is a modified version of the Bratfisch et al. scale [11] for measuring perceived task difficulty. The scale is composed of the 9 categories presented below, each of which is assigned a numerical value from 1 to 9 in the following way:

(1) very, very low mental effort
(2) very low mental effort
(3) low mental effort
(4) rather low mental effort
(5) neither low nor high mental effort
(6) rather high mental effort
(7) high mental effort
(8) very high mental effort
(9) very, very high mental effort

We rank the actions we analysed in the theoretical evaluation in terms of the Paas categories, using plausible values for each class of users presented in Section 2.3 (see Table 1). We do the same for what concerns the durations. Making these plausible values explicit makes it possible to discuss them and use them as a base for comparison when pandemic conditions will allow us to perform and gather empirical data based on tests involving actual users.

Below we present 4 different use cases. We analyse each of them in their two options: performed using the GUI or using command line actions. We describe each use case as a list of subtasks (we will refer to each of them as *Subtask X*, where X is the number that identifies it), which in turn are composed of one or more basic tasks (we will refer to each of them as *Basic Task X y*, where X is the number identifying the subtask to which the basic task belong to, and y is the letter which identifies the basic task). For each basic task, we present the duration parameter expressed in seconds and the difficulty parameter expressed in terms of the plausible values presented in Table 1 for each class of users. We omit in the list of basic tasks the delay parameter which is always equal to 0. In some of the analysed use cases, a number of subtasks have to be repeated until the desired goal has been achieved. We identify such subtasks with a * beside their description.

For each use case, we provide a comparison between the two options. Moreover, we present a table showing the cognitive load and the memory load (CL and ML respectively) for each subtask (ST) in both options of each use case. CL has been computed with Eq. 1, using for the duration and the difficulty of each basic task the values presented in the list of subtasks.

5.1 Patient data inspection

Use case description A physician wants to check the `gtv` identified in Specification 1 at line 11 on a single patient. The dataset over which the analysis has been performed is composed of a single item, the MRI scans of a single patient.

ACTIONS	PLAUSIBLE MENTAL EFFORT		
	Researcher	**Physician**	**Developer**
save a specification on a code editor	very, very low	low	very, very low
find a line in a specification	very, very low	very, very low	very, very low
open a directory	very, very low	very, very low	very, very low
open an application (e.g. code editor, shell, DICOM viewer)	very low	very low	very, very low
search a file (e.g. image, document) in a file system	very low	very low	very low
open an image with a DICOM viewer	low	very low	low
interact with a GUI (e.g. press buttons, scroll items, open menu)	very low	very low	very low
open a csv file with a spreadsheet	very low	very low	very low
make calculation on spreadsheet	very low	low	low
open an image as a layer with a DICOM viewer	low	very low	low
remember file/directory name	low	low	low
write shell commands	rather high	very, very high	neither low nor high
edit a value in a specification	rather high	high	rather high
evaluate a layer automatically identified	very high	rather high	very high
compare two layers opened in two separate DICOM viewers	very high	rather high	very high
write a specification	very high	very, very high	rather high

Table 1: Table showing plausible mental effort for the analysed actions for each class of users.

Option 1 – without GUI

1. Open the base image:
   (a) remember the name of the directory where all datasets are filed [**duration:** 5 **difficulty:** 3]
   (b) search and open the `datasets` directory [**duration:** 12 **difficulty:** 2]
   (c) remember the name of the dataset analysed [**duration:** 5 **difficulty:** 3]
   (d) search and open the directory related to the dataset analysed [**duration:** 12 **difficulty:** 2]
   (e) open the directory related to the dataset item [**duration:** 2 **difficulty:** 1]
   (f) remember the name of the base image used for the analysis [**duration:** 5 **difficulty:** 3]
   (g) search and open the base image with a DICOM viewer [**duration:** 12 **difficulty:** 2]
2. Open the layer (`gtv`):
   (a) remember the name of the directory where all the results are stored (`results` directory) [**duration:** 5 **difficulty:** 3]
   (b) search and open the `results` directory [**duration:** 12 **difficulty:** 2]
   (c) remember the name of the specification used for the analysis [**duration:** 5 **difficulty:** 3]
   (d) search and open the directory related to the specification [**duration:** 12 **difficulty:** 2]

(e) remember the name of the dataset analysed [**duration:** 5 **difficulty:** 3]
(f) search and open the directory related to the dataset [**duration:** 12 **difficulty:** 2]
(g) search and open the directory related to the last session [**duration:** 12 **difficulty:** 2]
(h) open the directory related to the dataset item [**duration:** 2 **difficulty:** 1]
(i) search and open the `gtv` image as a layer of the image opened in Basic Task 1g [**duration:** 12 **difficulty:** 2]

3. Analyse the layer (`gtv`):
   (a) Evaluate the layer opened in a nifti format image viewer [**duration:** d **difficulty:** 6]

Option 2 – with GUI

1. Open the base image:
   (a) open the `VoxLogicA` GUI [**duration:** 2 **difficulty:** 2]
   (b) remember the name of the base image used for the analysis [**duration:** 5 **difficulty:** 3]
   (c) click the button corresponding to the base image recalled in Basic Task 1b in the input/output column [**duration:** 2 **difficulty:** 2]]
2. Open the layer (`gtv`):
   (a) click the button corresponding to the layer `gtv` in the input/output column [**duration:** 2 **difficulty:** 2]
3. Analyse the layer (`gtv`):
   (a) close the code column by clicking the corresponding icon in the icons column [**duration:** 2 **difficulty:** 2]
   (b) evaluate the layer opened in the DICOM viewer in the work space [**duration:** d **difficulty:** 6]

Use case options comparison and evaluation The main difference between the two options regards the memory load involved to complete both Subtask 1 (open the base image) and Subtask 2 (open the `gtv` layer). As Table 2 shows, the memory load of users completing these tasks without the support of the GUI is twice as high as that of users using the GUI. This is mainly due to the extra effort that is required to remember the names of files in the command line based setting. There is no significant difference in the cognitive load.

Representing Subtask 1 of Option 1 with the notation presented in [15] and recalled in Section 2.5, we have[15]:

[15] For the sake of simplicity, we omit the parameters **duration**, **difficulty**, and **delay** since in this case we focus on memory load and not on cognitive load (for the computation of which we need the parameters associated with each basic task).

$$noInfo \mid \texttt{VoxLogicA} \implies noAction \mid datasetsName$$
$$datasetsName \mid \texttt{VoxLogicA} \implies openDir \mid noInfo$$
$$noInfo \mid \texttt{datasets} \implies noAction \mid datasetName$$
$$datasetName \mid \texttt{datasets} \implies openDir \mid noInfo$$
$$noInfo \mid \texttt{dataset01} \implies openDir \mid noInfo$$
$$noInfo \mid \texttt{item01} \implies noAction \mid baseImageName$$
$$baseImageName \mid \texttt{item01} \implies openDir \mid noInfo$$

where the state of the system is represented by the current state of the user's personal computer (namely the current directory or the application currently used). The first two basic tasks mean that the user is in the directory `VoxLogicA`, he/she remembers the name of the directory where all datasets are filed and he/she stores this item (*datasetsName*) in his/her WM; he/she then uses this piece of information to search and open the right directory. This is done for each couple of basic tasks where the user has to retrieve from WM the piece of information regarding the directory (or the file) to open and he/she opens it. In Basic Task 1e of Option 1 it is not necessary to remember the name of the directory to open since the dataset is composed of a single item, and the user does not need to remember the name of such item to open its directory. As it can be noted, the user has to store in and retrieve from his/her working memory 3 different items (*datasetsName*, *datasetName*, and *baseImageName*). These pieces of information have to be maintained in WM through rehearsal (namely by continuously focusing attention on them) from the beginning of the interaction with the system until they are used; this could cause a memory overload during the interaction assuming that WM can contain 7 ± 2 items.

The memory load evaluation led us to design a GUI where all the elements are declared explicitly with labels or titles: e.g. the base images are preceded by the title "BASE IMAGE", the layers are preceded by the title "LAYERS". In this way, users do not need to retrieve from their WM the information about the position and the function of the GUI elements, they just need to retrieve this information by observing the GUI.

For what concerns Subtask 3 (analyse the layer), in Option 2 users can close the code column to make the GUI less distracting (basic task 3a) and increasing the space for image inspection; this action influences the CL of the entire subtask, however, we need to consider that such basic task requires a negligible cognitive effort and that entails a lowering of the cognitive effort required for the evaluation of the overlay (basic task 3b). Therefore we can consider the two options as equally demanding. In order to evaluate whether a layer has been correctly identified by the ImgQL specification, the physician needs to activate some cognitive mechanisms and these cognitive mechanisms are the same, be the analysis performed on a DICOM viewer embedded in the GUI or on an external DICOM viewer. It is important to note that the analysis of the memory load of Subtask 3 is simplified: we assume that users would retrieve a single piece of information from their WM to complete Basic Task 3b. Actually, to evaluate a layer identified by an ImgQL specification, users need to activate higher cognitive mechanisms, retrieving knowledge from their long term memory. Moreover,

we use for both options a symbolic parameter d representing the time necessary to evaluate a layer. This abstraction does not affect our comparison since we make the same assumption for both options.

	Option 1 w/o GUI		**Option 2** with GUI	
Subtasks of Use Case 5.1:	CL	ML	CL	ML
(1) Open the base image	2.2	3	2.5	1
(2) Open the layer (`gtv`)	2.3	3	2	0
(3) Analyse the layer (`gtv`)	6	1	6	1
TOT.	10.5	5	10.5	2

Table 2: Table showing the cognitive and memory load for Use Case 5.1 – Patient data inspection.

As Table 2 shows, there is a significant difference between the two options for what concerns the memory load: the amount of information necessary to complete Option 1 is twice the amount of information necessary to complete Option 2. For what concerns the cognitive load, instead, there are no differences between the two options: as mentioned before, to analyse a layer on a DICOM viewer is an activity equally demanding, be it performed using the GUI or not. We plan to investigate more advanced features in the `VoxLogicA` GUI to further reduce the cognitive load as well.

5.2 Analysing a Patient's Disease Evolution

Use case description A physician wants to check the progression or regression over time of the GBM of a single patient. The data set is composed of the MRI scans of the same patient at two different points in time. The physician wants to analyse the GBM based on the segmentation result of the image in layer `gtv` identified in Specification 1 at line 11.

Option 1 – without GUI

1. Open the base image (repeat for the 2 images in the dataset):
   (a) remember the name of the directory where all datasets are filed [**duration:** 5 **difficulty:** 3]
   (b) search and open the `datasets` directory [**duration:** 12 **difficulty:** 2]
   (c) remember the name of the dataset analysed [**duration:** 5 **difficulty:** 3]
   (d) search and open the directory related to the dataset analysed [**duration:** 12 **difficulty:** 2]
   (e) remember the name of a dataset item not already opened (namely one of the images in the dataset) [**duration:** 5 **difficulty:** 3]

(f) search and open the directory related to the dataset item recalled in Basic Task 1e [**duration:** 12 **difficulty:** 2]
(g) remember the name of the base image used for the analysis [**duration:** 5 **difficulty:** 3]
(h) search and open the base image with a nifti format image viewer [**duration:** 12 **difficulty:** 2]

2. Open the layer (`gtv`) (repeat for the 2 images in the dataset):
   (a) remember the name of the directory where all results are filed [**duration:** 5 **difficulty:** 3]
   (b) search and open the `results` directory [**duration:** 12 **difficulty:** 2]
   (c) remember the name of the specification used for the analysis [**duration:** 5 **difficulty:** 3]
   (d) search and open the directory related to the specification [**duration:** 12 **difficulty:** 2]
   (e) remember the name of the dataset analysed [**duration:** 5 **difficulty:** 3]
   (f) search and open the directory related to the dataset [**duration:** 12 **difficulty:** 2]
   (g) search and open the directory related to the last session [**duration:** 12 **difficulty:** 2]
   (h) remember the name of the dataset item whose base image has been opened in Basic Task 1h [**duration:** 5 **difficulty:** 3]
   (i) search and open the directory related to the dataset item recalled in Basic Task 2h [**duration:** 12 **difficulty:** 2]
   (j) search and open the `gtv` image as a layer of the image opened in Basic Task 1h [**duration:** 12 **difficulty:** 2]
3. Compare the images:
   (a) Evaluate the layers opened in two nifti format image viewers [**duration:** d **difficulty:** 6]

Option 2 – with GUI

1. Open the base image (all 2 images at once):
   (a) open the `VoxLogicA` GUI [**duration:** 2 **difficulty:** 2]
   (b) remember the name of the base image used for the analysis [**duration:** 5 **difficulty:** 3]
   (c) click the button corresponding to the base image recalled in Basic Task 1b in the layers column [**duration:** 2 **difficulty:** 2]
2. Open the layer (`gtv`) (all 2 images at once):
   (a) click the button corresponding to the layer `gtv` in the layers column [**duration:** 2 **difficulty:** 2]
3. Compare the images:
   (a) Evaluate the layers opened in two nifti format image viewers in the work space [**duration:** d **difficulty:** 6]

Use case options comparison and evaluation As in Use Case 5.1, the main difference between the two options regards the memory load necessary to complete Subtask 1 and Subtask 2 (respectively open the base image and open the layer). These two subtasks are almost the same as those in Use Case 5.1, except for the fact that here they have to be repeated twice: once for each dataset item. In addition to this, here the user needs to store and retrieve one additional piece of information regarding the dataset item already opened. As Table 3 shows, the memory load to complete Option 1 is almost 9 times higher than the memory load necessary to complete Option 2.

The high ML in Option 1 led us to reason about a GUI where a single action affects multiple elements: namely, when the user clicks on a base image or a layer, this action affects all the dataset items opened (or that will be opened) in the work space. This behaviour reduces the number of steps users need to perform to check a base image or a layer on multiple items and, therefore, also their cognitive effort. Indeed, as Table 3 shows in this use case the cognitive load too is lower in Option 2 respect in Option 1.

For what concerns Subtask 3 (compare the images), here too the cognitive load necessary to complete the subtask is almost the same in both options. The main difference regards the DICOM viewers: while an external viewer could be more sophisticated and allow users to perform more accurate analysis (e.g. measuring portions of the MRI scans), using the embedded viewers in the GUI would support the comparison by *automatically* opening the viewers side by side in the workspace, without further steps in the interaction. However, it must be investigated, through the user test, if this class of users has additional preferences on the positioning of the viewers, or possibilities to zoom in to inspect particular aspects more in detail.

	Option 1 w/o GUI		**Option 2** with GUI	
Subtasks of Use Case 5.2	CL	ML	CL	ML
(1) Open the base image	2.3 (×2)	4 (×2)	2.5	1
(2) Open the layer (`gtv`)	2.2 (×2)	4 (×2)	2	0
(3) Compare the images	6	1	6	1
TOT.	15	17	10.5	2

Table 3: Table showing the cognitive and memory load for Use Case 5.2 – Analysing a Patient's Disease Evolution.

5.3 Parameters calibration of an existing ImgQL specification

Use case description A researcher wants to calibrate the parameter used in the ImgQL Specification 1 at line 4 to find the optimal value for `hI` in a dataset

of 10 MRI scans. In order to evaluate the used value, he/she has to check the resulting layer saved with the specification. The `VoxLogicA` directories structure is as the one shown in Figure 12.

Option 1 – without GUI

1. Change the parameter: *
   (a) open Specification 1 on a code editor [**duration:** 2 **difficulty:** 2]
   (b) find line 4 in Specification 1 [**duration:** 2 **difficulty:** 1]
   (c) edit the value [**duration:** 30 **difficulty:** 6]
   (d) save the specification [**duration:** 1 **difficulty:** 1]
2. Run the specification: *
   (a) open a shell [**duration:** 2 **difficulty:** 2]
   (b) remember and write the command to go to the `VoxLogicA` directory [**duration:** 5 **difficulty:** 6]
   (c) remember and write the command to run the specification [**duration:** 5 **difficulty:** 6]
3. Open the base image (repeat for the 10 images in the dataset): *
   (a) remember the name of the directory where all datasets are filed [**duration:** 5 **difficulty:** 3]
   (b) search and open the `datasets` directory [**duration:** 12 **difficulty:** 2]
   (c) remember the name of the dataset analysed [**duration:** 5 **difficulty:** 3]
   (d) search and open the directory related to the dataset analysed [**duration:** 12 **difficulty:** 2]
   (e) remember the name of a dataset item not already opened (namely one of the images in the dataset) [**duration:** 5 **difficulty:** 3]
   (f) search and open the directory related to the dataset item recalled in Basic Task 3e [**duration:** 12 **difficulty:** 2]
   (g) remember the name of the base image used for the analysis [**duration:** 5 **difficulty:** 3]
   (h) search and open the base image with a nifti format image viewer [**duration:** 12 **difficulty:** 3]
4. Open the resulting layer (`hI`) (repeat for the 10 images in the dataset): *
   (a) remember the name of the directory where all results are filed [**duration:** 5 **difficulty:** 3]
   (b) search and open the `results` directory [**duration:** 12 **difficulty:** 2]
   (c) remember the name of the specification just run [**duration:** 5 **difficulty:** 3]
   (d) search and open the directory related to the specification [**duration:** 12 **difficulty:** 2]
   (e) remember the name of the dataset analysed [**duration:** 5 **difficulty:** 3]
   (f) search and open the directory related to the dataset [**duration:** 12 **difficulty:** 2]
   (g) search and open the directory related to the last session [**duration:** 12 **difficulty:** 2]

(h) remember the name of the dataset item whose base image has been opened in Basic Task 3h [**duration:** 5 **difficulty:** 3]
(i) search and open the directory related to the dataset item recalled in Basic Task 4h [**duration:** 12 **difficulty:** 2]
(j) search and open the `hI` image as a layer of the image opened in Basic Task 3h [**duration:** 12 **difficulty:** 3]

5. Analyse the resulting layer (`hI`) (repeat for the 10 images in the dataset): *
   (a) evaluate the result [**duration:** d **difficulty:** 8]

Option 2 – with GUI

1. Change the parameter: *
   (a) open the `VoxLogicA` GUI [**duration:** 2 **difficulty:** 2]
   (b) find line 4 in Specification 1 in the code column [**duration:** 2 **difficulty:** 1]
   (c) edit the value [**duration:** 30 **difficulty:** 6]
2. Run the specification: *
   (a) open a dataset item from the dataset row (repeat for the 10 images in the dataset) [**duration:** 2 **difficulty:** 2]
   (b) click the button "run" on the GUI [**duration:** 2 **difficulty:** 2]
3. Open the base image (all 10 images at once): *
   (a) remember the name of the base image used for the analysis [**duration:** 5 **difficulty:** 3]
   (b) click the button corresponding to the base image recalled in Basic Task 3a in the input/output column [**duration:** 2 **difficulty:** 2]
4. Open the resulting layer (`hI`) (all 10 images at once): *
   (a) click the button corresponding to the layer `hI` in the input/output column [**duration:** 2 **difficulty:** 2]
5. Analyse the resulting layer (`hI`) (repeat for the 10 images in the dataset): *
   (a) evaluate the result [**duration:** d **difficulty:** 8]

Use case options comparison and evaluation In both options, some subtasks remain cognitively demanding, such as Subtask 1 (change the parameter) and Subtask 5 (analyse the resulting layer), which in both options has to be repeated for the 10 images composing the dataset. For what concerns the latter (Subtask 5), as already said for Use Case 5.1, the user needs to activate the same higher cognitive mechanisms, be the analysis performed on the GUI or not. Moreover, since the user here is a researcher (that hence is not necessarily extremely well-trained to deal with analysing medical images), his/her cognitive load rises to 8 (see Table 4). From this subtask, users have to evaluate the quality of the resulting layers: if they consider them not satisfactory, they have to repeat all subtasks until the results are close enough to what they expect. Even if each repetition influences the users' cognitive efforts, Table 4 shows the total cognitive and memory load related to a single loop.

As regards Subtask 1, the only differences between the two options concern the opening of the specification on different supports (on a code editor embedded

on the GUI for Option 2 and on an external code editor for Option 1) and the saving of the edited specification (that is not necessary on the GUI but it is required in the external code editor). Both differences do not significantly affect the cognitive load of the entire subtask, since the cognitive load of the basic task concerning the opening of the support is the same in both options, and the cognitive load of the basic task concerning the saving of the specification is minimal.

For what concerns Subtask 2 (run the specification), Option 1 is more demanding than Option 2 from both cognitive and mnemonic points of view (see Table 4). Without the support of the GUI, the user needs to remember a number of shell commands in order to run the specification; this activity is not only more cognitively demanding, but it also requires to retrieve information from WM. Moreover, if users employ different operating systems (OS) the situation is even more complicated, since they have to remember different commands for each OS used. With the GUI, instead, even if to run the analysis users have to first open in the work space each dataset item on which they want to perform the analysis (10 items in this case), this activity is almost effortless from a cognitive point of view and totally costless from a mnemonic point of view. Moreover, the mere execution method is extremely simplified: considering that, analysing medical images is an already cognitively demanding task, we designed a GUI where the analysis execution is simply performed by clicking a button. To further reduce the cognitive load of this subtask, we plan to design and implement new methods to select simultaneously multiple dataset items (e.g. a button to select all cases or allow keyboard shortcuts).

As in Use Case 5.1 and in Use Case 5.2, here too Subtask 3 (open the base image) and Subtask 4 (open the resulting layer) of Option 1, are more demanding from a memory point of view. However, in this case the user has to repeat such subtasks for the 10 items in the dataset. This implies a higher memory load and could lead users to fail: remember a name and then recall it from the working memory could seem an easy procedure, however, when this procedure is repeated several times, this could lead to error since the human memory is not infallible. For instance, a user could open the layer corresponding to one dataset item on the base image related to another dataset item. Moreover, this makes the cognitive load of such subtasks of Option 1 10 times higher than their counterpart of Option 2 (see Table 4).

The memory and cognitive load evaluation of such a use case led us to reason about the concept of dataset and base images: when using `VoxLogicA` from the command line, users could confuse the concept of the dataset in input with the concept of the base image, since when an analysis is performed on a dataset, the same base image is implicitly used for each image in the dataset. Therefore, we distinguish in the GUI the two concepts, placing the data on the dataset row and base images on the layers column.

	Option 1 w/o GUI		**Option 2** with GUI	
Subtasks Use Case 5.3:	CL	ML	CL	ML
(1) Change the parameter *	5.3	1	5.5	1
(2) Run the specification *	5.3	2	2	0
(3) Open the base image *	2.5 (×10)	4	2.7	1
(4) Open the resulting layer `hI` *	2.6 (×10)	4	2	0
(5) Analyse the resulting layer `hI` (10 times) *	8 (×10)	1	8 (×10)	1
TOT.	141.6	12	92.2	3

Table 4: Table showing the cognitive and memory load for Use Case 5.3 – Parameters calibration of an existing ImgQL specification.

5.4 Development of an ImgQL specification

Use case description A developer wants to develop a specification for the segmentation of brain lesions of a set of 10 MRI scans of patients affected by GBM, in order to automatically segment the tumour. The `VoxLogicA` directories structure is as the one shown in Figure 12. The similarity index used to evaluate the accuracy of the analysis is the Dice index (the most relevant and commonly used in the literature) computed in comparison w.r.t the manual segmentation (ground truth). The developer has to compare the automatic and manual segmentations for all the dataset items with a Dice index below 0.8 to check where the specification shows lower precision and try to improve it.

Option 1 – without GUI

1. Write a specification:
   (a) open a code editor [**duration:** 2 **difficulty:** 1]
   (b) write the specification [**duration:** d **difficulty:** 6]
   (c) save the specification [**duration:** 1 **difficulty:** 1]
2. Run the specification: *
   (a) open a shell [**duration:** 2 **difficulty:** 1]
   (b) remember and write the command to go on the `VoxLogicA` directory [**duration:** 5 **difficulty:** 5]
   (c) remember and write the command to run the specification [**duration:** 5 **difficulty:** 5]
3. Analyse the results for the similarity indexes: *
   (a) remember the name of the results directory [**duration:** 5 **difficulty:** 3]
   (b) search and open the `results` directory [**duration:** 12 **difficulty:** 2]
   (c) remember the name of the specification just run [**duration:** 5 **difficulty:** 3]
   (d) search and open the directory related to the specification [**duration:** 12 **difficulty:** 2]
   (e) remember the name of the dataset analysed [**duration:** 5 **difficulty:** 3]

(f) search and open the directory related to the dataset [**duration:** 12 **difficulty:** 2]
(g) search and open the directory related to the last session [**duration:** 12 **difficulty:** 2]
(h) search and open with a spreadsheet the .csv file containing the values for the similarity indexes saved during the analysis for each dataset item analysed [**duration:** 12 **difficulty:** 2]
(i) compute the average value of the Dice index for all the dataset items analysed [**duration:** 20 **difficulty:** 2]
(j) filter the results to find the items with a Dice index below 0.8 [**duration:** 20 **difficulty:** 2]

4. Open the base image (repeat for the k images identified in basic task 3j): *
   (a) remember the name of the directory where all datasets are filed [**duration:** 5 **difficulty:** 3]
   (b) search and open the `datasets` directory [**duration:** 12 **difficulty:** 2]
   (c) remember the name of the dataset analysed [**duration:** 5 **difficulty:** 3]
   (d) search and open the directory related to the dataset analysed [**duration:** 12 **difficulty:** 2]
   (e) remember the name of a dataset item not already opened (namely one of the images in the dataset) [**duration:** 5 **difficulty:** 3]
   (f) search and open the directory related to the dataset item recalled in Basic Task 4e [**duration:** 12 **difficulty:** 2]
   (g) remember the name of the base image used for the analysis [**duration:** 5 **difficulty:** 3]
   (h) search and open the base image with a nifti format image viewer [**duration:** 12 **difficulty:** 3]
5. Open the layer (`gtv`) (repeat for the k images identified in basic task 3j): *
   (a) remember the name of the directory where all results are filed [**duration:** 5 **difficulty:** 3]
   (b) search and open the `results` directory [**duration:** 12 **difficulty:** 2]
   (c) remember the name of the specification just run [**duration:** 5 **difficulty:** 3]
   (d) search and open the directory related to the specification [**duration:** 12 **difficulty:** 2]
   (e) remember the name of the dataset analysed [**duration:** 5 **difficulty:** 3]
   (f) search and open the directory related to the dataset [**duration:** 12 **difficulty:** 2]
   (g) search and open the directory related to the last session [**duration:** 12 **difficulty:** 2]
   (h) remember the name of the dataset item whose base image has been opened in Basic Task 4h [**duration:** 5 **difficulty:** 3]
   (i) search and open the directory related to the dataset item recalled in Basic Task 5h [**duration:** 12 **difficulty:** 2]
   (j) search and open the `gtv` image as a layer of the image opened in Basic Task 4h [**duration:** 12 **difficulty:** 3]

6. Open the ground truth (repeat for the k images identified in basic task 3j): *
   (a) remember the name of the directory where all datasets are filed [**duration:** 5 **difficulty:** 3]
   (b) search and open the `datasets` directory [**duration:** 12 **difficulty:** 2]
   (c) remember the name of the dataset analysed [**duration:** 5 **difficulty:** 3]
   (d) search and open the directory related to the dataset analysed [**duration:** 12 **difficulty:** 2]
   (e) remember the name of a dataset item not already opened (namely one of the images in the dataset) [**duration:** 5 **difficulty:** 3]
   (f) search and open the directory related to the dataset item recalled in Basic Task 6e [**duration:**12 **difficulty:** 2]
   (g) search and open the ground truth image as a layer of the image opened in Basic Task 4h [**duration:** 12 **difficulty:** 3]
7. Analyse the automatic segmentation (`gtv`) (repeat for the k items identified in basic task 3j): *
   (a) analyse `gtv` and compare it with the ground truth in order to evaluate where they do not correspond [**duration:** t **difficulty:** 8]
8. Revise the specification: *
   (a) open the code editor with the specification [**duration:** 2 **difficulty:** 1]
   (b) edit the specification [**duration:** f **difficulty:** 6]
   (c) save the specification [**duration:** 1 **difficulty:** 1]

Option 2 – with GUI

1. Write a specification:
   (a) open the GUI [**duration:** 2 **difficulty:** 2]
   (b) write the specification on the embedded code editor [**duration:** d **difficulty:** 6]
2. Run the specification: *
   (a) open a dataset item from the dataset row (repeat for the 10 images in the dataset) [**duration:** 2 **difficulty:** 2]
   (b) click the button "run" on the GUI [**duration:** 2 **difficulty:** 2]
3. Analyse the results for the similarity indexes: *
   (a) check the average value of the Dice index for all the item in the indexes row on the GUI [**duration:** 2 **difficulty:** 2]
   (b) look for the items of the dataset analysed with a Dice index below 0.8 using the search box in the GUI [**duration:** 20 **difficulty:** 2]
4. Open the base image (all the k images identified in basic task 3b at once): *
   (a) remember the name of the base image used for the analysis [**duration:** 5 **difficulty:** 3]
   (b) click the button corresponding to the base image in the input/output column [**duration:** 2 **difficulty:** 2]
5. Open the layer (`gtv`) (all the k images identified in basic task 3b at once): *
   (a) click the button corresponding to the `gtv` in the input/output column [**duration:** 2 **difficulty:** 2]

6. Open the ground truth (all the k images identified in basic task 3b at once): *
   (a) click the button corresponding to the ground truth in the input/output column [**duration:** 2 **difficulty:** 2]
7. Analyse the automatic segmentation (`gtv`) (repeat for the k items identified in basic task 3b): *
   (a) analyse `gtv` and compare it with the ground truth in order to evaluate where they do not correspond [**duration:** t **difficulty:** 8]
8. Revise the specification: *
   (a) edit the specification on the code column [**duration:** f **difficulty:** 6]

Use case options comparison and evaluation In both options, Subtask 1 is highly demanding for what concerns the cognitive load since users have to remember how to write the specification. Moreover, basic task 1b could be divided into further basic tasks if we were to consider it as a sequence of atomic steps for each line of code. For the sake of simplicity we consider it as a single basic task where users have to retrieve a single information on how to write the code. It is worth to note that in option 2, Basic Task 1a (open the `VoxLogicA` GUI) is common to the entire use case: namely users do not need to open different GUI/systems (such as code editor, directories, command shell, or image viewers) to complete the use case. Representing such a subtask with the notation in [15] and recalled in Section 2.5 of the present paper, we have:

$$\begin{gathered} noInfo \mid \texttt{desktop} \implies openCodeEditor \mid noInfo \\ \textbf{duration } 2 \textbf{ difficulty } 2 \textbf{ delay } 0 \\ howTo \mid \texttt{codeEditor} \implies writeCode \mid noInfo \\ \textbf{duration } d \textbf{ difficulty } 10 \textbf{ delay } 0 \\ noInfo \mid \texttt{codeEditor} \implies saveTheCode \mid noInfo \\ \textbf{duration } 1 \textbf{ difficulty } 1 \textbf{ delay } 0 \end{gathered}$$

where the information $howTo$ represents the info on how to write the specification and the duration d is a symbolic parameter representing the time the developer needs to write a specification which we assume to be at least several minutes; since this duration can be assumed to be the same in both options (with and without GUI), using a symbolic parameter does not change the computation of the cognitive load for both options of Subtask 1. As in Use Case 5.3, here the only differences between the two options regard the support to be opened and the saving of the specification and both these differences do not affect the cognitive load of the entire subtask.

As the computation of the CL for both options of such subtask shows (see Table 4), the higher the value of d, the closer the CL gets to about 6, with the value of d in any case at least in the order of several minutes. We use such approximation to compute the total cognitive load of both options. We make the same assumption about the cognitive load of Subtask 8 in Option 1.

Subtask 8 (revise the specification) as well is cognitively demanding in both options since it involves the same higher cognitive mechanisms be it performed

on the GUI or not. In this case too, we use symbolic parameter f for the duration of the basic task for both options. We still can observe a high CL in the subtasks dealing with the writing or editing of the ImgQL specification. This indicates a further place where the GUI can be improved to reduce cognitive load. We plan to investigate GUI options that might facilitate these activities in a future version, such as templates to help users to write spatial logic formulas.

In this use case, in addition to open the base image (Subtask 4) and open the layer (Subtask 5), activities which are as those presented in the previous use cases, users need also to open the ground truth (Subtask 6). All these subtasks must be repeated for the k images with a Dice index below 0.8 identified in Basic Task 3b. As Table 4 shows, these repetitions increase the cognitive load of the subtasks. As for the previous use cases, Option 1 of all these subtasks related to the opening of an image (be it a base image, a resulting layer, or the manual segmentation), requires the memorisation of a number of information regarding the name of the directories or of the files to open, that could lead users to memory overload.

For what concerns Subtask 3 (analyse the results for the similarity indexes), Option 1 could require some cognitive effort in setting the file to open with a spreadsheet and in remembering the steps to compute the average and to order the values. To lower the cognitive and mnemonic effort, we designed a GUI where users can check the values for each similarity index and select the dataset items according to the conditions they set on such values, just by clicking on some selected buttons on the GUI. In this way, users do not need to remember how to set the file and how to make computations on the data. It is true that an external spreadsheet is more sophisticated and allows users to perform more accurate analysis on data, however, in the actual state of the system, we consider the operations available in the GUI those necessary to complete several tasks in the domain of medical image analysis. We plan to investigate more advanced features in the user test we will perform when the pandemic conditions will allow us.

From Subtask 3 users have to evaluate how the results are in terms of both similarity indexes and resulting images; if they consider the results not satisfactory in terms of accuracy of the analysis, they have to repeat subtasks from 2 to 6 until the results are close to what they expect.

Finally, Subtask 7 (analyse the automatic segmentation), in both options, remains cognitively demanding since users need to make a cognitive effort to compare the automatic and manual segmentations. This becomes even more complicated since users need to repeat the subtask for the k elements with a Dice index below 0.8. Again, we use a symbolic parameter f for the duration of the basic task. Also in this case further GUI features could be introduced to facilitate the assessment of the difference between two segmentations of the same image. For example one could highlight the difference between segmentations in the viewer or toggle between showing and not showing the overlays on the base image to better appreciate where the segmentation could be further improved.

	Option 1 w/o GUI		**Option 2** with GUI	
Subtasks Use Case 5.4:	CL	ML	CL	ML
(1) Write a specification	$\frac{3+6d}{3+d}$	1	$\frac{4+6d}{2+d}$	1
(2) Run the specification *	4.3	2	2	0
(3) Analyse the results *	2.1	5	2	0
(4) Open the base image *	2.3 ($\times k$)	4 ($\times k$)	2.7	1
(5) Open the layer * (`gtv`)	2.6 ($\times k$)	4 ($\times k$)	2	0
(6) Open the ground truth *	2.4 ($\times k$)	3 ($\times k$)	2	0
(7) Analyse the automatic segmentation * (`gtv`)	8 ($\times k$)	1 ($\times k$)	8 ($\times k$)	1
(8) Revise the specification *	$\frac{3+6f}{3+f}$	1	6	1
TOT.	18.4 + 15.3×k	9 + 12×k	22.7 + 8×k	4

Table 5: Table showing the cognitive and memory load for Use Case 5.4 – Development of a ImgQL specification.

6 Conclusion

We presented a prototype of a novel graphical user interface for various user classes in the domain of medical image segmentation. The GUI facilitates the use of the spatial model checker `VoxLogicA` for the design of novel analysis methods using a declarative, high-level, logical specification language with domain oriented operators. We presented a theoretical cognitive evaluation of this prototype interface to obtain a first impression of the cognitive load and memory load of the GUI. This is an important aspect because the results of medical image analysis are informing therapy planning, such as radiotherapy, which is obviously a very critical task. Furthermore, the analysis tasks themselves require high precision and concentration of the user for a sustained amount of time. Although `VoxLogicA` has shown to be very promising in analysing medical images, the command-line process is not interactive, and the analysis risks to be too slow and involving to be used by a larger number of users.

The GUI we propose aims at supporting the analysis of medical images by combining not only DICOM viewers, but also a way to analyse images using a spatial logic, that includes domain oriented operators, and a system to manage datasets for medical images segmentation. This way the GUI presents a mix of tools, some of which novel and some already in use in this domain. The system allows users to automatically analyse several images simultaneously in real-time, as well as to visualise the results of the analysis on more than one image at the same time. Moreover, also all intermediate results can be visualised so that the

development of a novel or existing analysis method can be checked step by step also by non-programmers. This may greatly facilitate the uptake of this novel approach in this domain and could open the way for domain experts, that are not necessarily programmers, to develop their own segmentation methods in a transparent, concise and explainable way, using their expertise, and to exchange and discuss their methods with their peers.

Being a new system, the first usage of the `VoxLogicA` GUI may lead initially to a relatively high cognitive and memory load while users are learning to use it in the right way. To overcome this problem, we plan to implement a wizard to teach/show users how to use the GUI. Moreover, we take for granted that, after a brief period of training, users will be able to use the system with a general lower cognitive effort.

In the present paper, we evaluated the user interface in a theoretical way, using the method presented in [15] and validated in [14]. We showed the advantage of using the GUI especially from a memory point of view: having all the necessary information easily accessible makes the interaction easier and quicker. The method will also be used as a guidance to identify further places in the GUI that can be improved and in future extensions of the interface; for instance, the GUI could be extended with templates that could help users in activities such as writing and editing spatial logic specifications, further reducing the cognitive load of such activities.

As soon as the pandemic condition will allow it, we plan to perform usability tests. These usability tests will be submitted to three classes of users (i.e. developers, physicians and researchers) after a training period. Users will be directly observed while performing a number of tasks representative for their class. Moreover, after the test we will collect their feedback on the GUI with a questionnaire and their evaluation on the perceived difficulties for the actions performed. The study will take place in the presence of moderators that will observe users during the execution of the tasks, annotate behaviour, comments, and hesitations, as well as record completion times for each task.

References

1. Aël Chetelat, G., Baron, J.C.: Early diagnosis of alzheimer's disease: contribution of structural neuroimaging. Neuroimage 18(2), 525–541 (2003)
2. Baddeley, A.D., Hitch, G.: Working memory. In: Psychology of learning and motivation, vol. 8, pp. 47–89. Elsevier (1974)
3. Baik, M., Suk, H.J., Lee, J., Choi, K.: Investigation of eye-catching colors using eye tracking. In: Human Vision and Electronic Imaging XVIII. vol. 8651, p. 86510W. International Society for Optics and Photonics (2013)
4. Banci Buonamici, F., Belmonte, G., Ciancia, V., Latella, D., Massink, M.: Spatial logics and model checking for medical imaging. Int. J. Softw. Tools Technol. Transf. 22(2), 195–217 (2020), https://doi.org/10.1007/s10009-019-00511-9
5. Barrouillet, P., Bernardin, S., Camos, V.: Time constraints and resource sharing in adults' working memory spans. Journal of Experimental Psychology: General 133(1), 83 (2004)

6. Belmonte, G., Ciancia, V., Latella, D., Massink, M., Biondi, M., De Otto, G., Nardone, V., Rubino, G., Vanzi, E., Banci Buonamici, F.: A topological method for automatic segmentation of glioblastoma in mr flair for radiotherapy-esmrmb 2017. In: 34th annual scientific meeting. Magnetic Resonance Materials in Physics, Biology and Medicine. vol. 30, p. 437 (2017)
7. Belmonte, G., Broccia, G., Ciancia, V., Latella, D., Massink, M.: Feasibility of spatial model checking for nevus segmentation. In: 2021 IEEE/ACM 9th International Conference on Formal Methods in Software Engineering (FormaliSE). pp. 1–12 (2021)
8. Belmonte, G., Ciancia, V., Latella, D., Massink, M.: Innovating medical image analysis via spatial logics. In: From Software Engineering to Formal Methods and Tools, and Back, pp. 85–109. Springer (2019)
9. Belmonte, G., Ciancia, V., Latella, D., Massink, M.: Voxlogica: A spatial model checker for declarative image analysis. In: Vojnar, T., Zhang, L. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11427, pp. 281–298. Springer (2019), https://doi.org/10.1007/978-3-030-17462-0_16
10. Bonsiepe, G.: A method of quantifying order in typographic design. Visible Language 2(3), 203–220 (1968)
11. Bratfisch, O., et al.: Perceived item-difficulty in three tests of intellectual performance capacity. (1972)
12. Broccia, G.: A Formal Framework for Modelling and Analysing Safety-Critical Human Multitasking. Ph.D. thesis, University of Pisa, Department of Computer Science (2019)
13. Broccia, G., Masci, P., Milazzo, P.: Modeling and analysis of human memory load in multitasking scenarios: Late-breaking results. In: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems. pp. 1–7 (2018)
14. Broccia, G., Milazzo, P., Belviso, C., Montiel, C.B.: Validation of a simulation algorithm for safety-critical human multitasking. In: International Symposium on Formal Methods. pp. 99–113. Springer (2019)
15. Broccia, G., Milazzo, P., Ölveczky, P.C.: Formal modeling and analysis of safety-critical human multitasking. Innovations in Systems and Software Engineering 15(3), 169–190 (2019)
16. Bussi, L., Ciancia, V., Gadducci, F.: Towards a spatial model checker on gpu. In: International Conference on Formal Techniques for Distributed Objects, Components, and Systems. pp. 188–196. Springer (2021)
17. Çakiroğlu, Ü., Suiçmez, S.S., Kurtoğlu, Y.B., Sari, A., Yildiz, S., Öztürk, M.: Exploring perceived cognitive load in learning programming via scratch. Research in Learning Technology 26 (2018)
18. Cerone, A.: Towards a cognitive architecture for the formal analysis of human behaviour and learning. In: Federation of International Conferences on Software Technologies: Applications and Foundations. pp. 216–232. Springer (2018)
19. Cerone, A.: Behaviour and reasoning description language (brdl). In: International Conference on Software Engineering and Formal Methods. pp. 137–153. Springer (2019)
20. Cerone, A., Pluck, G.: A formal model for emulating the generation of human knowledge in semantic memory. In: International Symposium: From Data to Models and Back. pp. 104–122. Springer (2020)

21. Ciancia, V., Latella, D., Loreti, M., Massink, M.: Specifying and verifying properties of space. In: Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8705, pp. 222–235. Springer (2014)
22. Ciancia, V., Latella, D., Loreti, M., Massink, M.: Model Checking Spatial Logics for Closure Spaces. Logical Methods in Computer Science Volume 12, Issue 4 (Oct 2016), http://lmcs.episciences.org/2067
23. Ciancia, V., Latella, D., Loreti, M., Massink, M.: Model checking spatial logics for closure spaces. Log. Methods Comput. Sci. 12(4) (2016), https://doi.org/10.2168/LMCS-12(4:2)2016
24. Codella, N.C., Nguyen, Q.B., Pankanti, S., Gutman, D.A., Helba, B., Halpern, A.C., Smith, J.R.: Deep learning ensembles for melanoma recognition in dermoscopy images. IBM Journal of Research and Development 61(4/5), 5–1 (2017)
25. Comber, T., Maltby, J.R.: Screen complexity and user design preference in windows applications. In: Proceedings of OZCHI. vol. 94 (1994)
26. Depcik, C., Assanis, D.N.: Graphical user interfaces in an engineering educational environment. Computer Applications in Engineering Education 13(1), 48–59 (2005)
27. Design, M.: Dark theme, https://material.io/design/color/dark-theme.html
28. Dix, A.J.: Formal methods for interactive systems, vol. 16. Academic Press London (1991)
29. Doi, K.: Computer-aided diagnosis in medical imaging: historical review, current status and future potential. Computerized medical imaging and graphics 31(4-5), 198–211 (2007)
30. Forouzanfar, M., Forghani, N., Teshnehlab, M.: Parameter optimization of improved fuzzy c-means clustering algorithm for brain mr image segmentation. Engineering Applications of Artificial Intelligence 23(2), 160–168 (2010)
31. Gambino, O., Rundo, L., Cannella, V., Vitabile, S., Pirrone, R.: A framework for data-driven adaptive gui generation based on dicom. Journal of biomedical informatics 88, 37–52 (2018)
32. Guarracino, M.R., Maddalena, L.: Sdi+: A novel algorithm for segmenting dermoscopic images. IEEE journal of biomedical and health informatics 23(2), 481–488 (2018)
33. Haak, D., Page, C.E., Deserno, T.M.: A survey of dicom viewer software to integrate clinical research and medical imaging. Journal of digital imaging 29(2), 206–215 (2016)
34. Harrison, M.D., Masci, P., Campos, J.C.: Formal modelling as a component of user centred design. In: Federation of International Conferences on Software Technologies: Applications and Foundations. pp. 274–289. Springer (2018)
35. Harrison, M.D., Masci, P., Campos, J.C.: Balancing the formal and the informal in user-centred design. Interacting with Computers 33(1), 55–72 (2021)
36. Hill, D.L., Batchelor, P.G., Holden, M., Hawkes, D.J.: Medical image registration. Physics in medicine & biology 46(3), R1 (2001)
37. Journal, B.M.: Insufficient evidence that ai breast cancer screening is accurate enough to replace human scrutiny (2021), https://medicalxpress.com/news/2021-09-insufficient-evidence-ai-breast-cancer.html
38. Kamnitsas, K., Bai, W., Ferrante, E., McDonagh, S., Sinclair, M., Pawlowski, N., Rajchl, M., Lee, M., Kainz, B., Rueckert, D., et al.: Ensembles of multiple models and architectures for robust brain tumour segmentation. In: International MICCAI brainlesion workshop. pp. 450–462. Springer (2017)

39. Kauweloa, K.I., Gutierrez, A.N., Stathakis, S., Papanikolaou, N., Mavroidis, P.: A graphical user interface (gui) toolkit for the calculation of three-dimensional (3d) multi-phase biological effective dose (bed) distributions including statistical analyses. Computer methods and programs in biomedicine 131, 1–12 (2016)
40. Kushniruk, A.W., Patel, V.L.: Cognitive evaluation of decision making processes and assessment of information technology in medicine. International journal of medical informatics 51(2-3), 83–90 (1998)
41. Kushniruk, A., Patel, V.: Cognitive computer-based video analysis: its application in assessing the usability of medical systems. Medinfo. MEDINFO 8, 1566–1569 (1995)
42. Lewis, C., Wharton, C.: Cognitive walkthroughs. In: Handbook of human-computer interaction, pp. 717–732. Elsevier (1997)
43. Menze, B.H.e.a.: The multimodal brain tumor image segmentation benchmark (brats). IEEE Transactions on Medical Imaging 34(10), 1993–2024 (2015)
44. Mildenberger, P., Eichelberg, M., Martin, E.: Introduction to the dicom standard. European radiology 12(4), 920–927 (2002)
45. Miller, G.A.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological review 63(2), 81 (1956)
46. Min, A., Kyu, Z.M.: Mri images enhancement and tumor segmentation for brain. In: 2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT). pp. 270–275. IEEE (2017)
47. Nielsen, J.: Usability engineering. Morgan Kaufmann (1994)
48. Paas, F.G., Van Merriënboer, J.J.: The efficiency of instructional conditions: An approach to combine mental effort and performance measures. Human factors 35(4), 737–743 (1993)
49. Parush, A., Nadir, R., Shtub, A.: Evaluating the layout of graphical user interface screens: Validation of a numerical computerized model. International Journal of Human-Computer Interaction 10(4), 343–360 (1998)
50. Renkl, A., Atkinson, R.K.: Structuring the transition from example study to problem solving in cognitive skill acquisition: A cognitive load perspective. Educational psychologist 38(1), 15–22 (2003)
51. Ritter, F., Boskamp, T., Homeyer, A., Laue, H., Schwier, M., Link, F., Peitgen, H.O.: Medical image analysis. IEEE pulse 2(6), 60–70 (2011)
52. Rueckert, D., Schnabel, J.A.: Medical image registration. In: Biomedical image processing, pp. 131–154. Springer (2010)
53. Sharma, A., Wang, K., Siegel, E.: Radiologist digital workspace use and preference: a survey-based study. Journal of digital imaging 30(6), 687–694 (2017)
54. Shen, D., Wu, G., Suk, H.I.: Deep learning in medical image analysis. Annual review of biomedical engineering 19, 221–248 (2017)
55. Starmans, M.P., van der Voort, S.R., Tovar, J.M.C., Veenland, J.F., Klein, S., Niessen, W.J.: Radiomics: Data mining using quantitative medical image features. In: Handbook of medical image computing and computer assisted intervention, pp. 429–456. Elsevier (2020)
56. Sweller, J.: Cognitive load during problem solving: Effects on learning. Cognitive science 12(2), 257–285 (1988)
57. Tang, P.C., Patel, V.L.: Major issues in user interface design for health professional workstations: summary and recommendations. International journal of bio-medical computing 34(1-4), 139–148 (1994)
58. Toosy, A., Werring, D., Orrell, R., Howard, R., King, M., Barker, G., Miller, D., Thompson, A.: Diffusion tensor imaging detects corticospinal tract involvement at

multiple levels in amyotrophic lateral sclerosis. Journal of Neurology, Neurosurgery & Psychiatry 74(9), 1250–1257 (2003)
59. Tullis, T.S.: An evaluation of alphanumeric, graphic, and color information displays. Human Factors 23(5), 541–550 (1981)
60. Tullis, T.S.: The formatting of alphanumeric displays: A review and analysis. Human Factors 25(6), 657–682 (1983)
61. W3C: Notes on user centered design process (ucd) (2004), https://www.w3.org/WAI/redesign/ucd
62. Withey, D.J., Koles, Z.J.: Medical image segmentation: Methods and software. In: 2007 Joint Meeting of the 6th International Symposium on Noninvasive Functional Source Imaging of the Brain and Heart and the International Conference on Functional Biomedical Imaging. pp. 140–143. IEEE (2007)
63. Yasmin, M., Sharif, M., Masood, S., Raza, M., Mohsin, S.: Brain image enhancement-a survey. World Applied Sciences Journal 17(9), 1192–1204 (2012)

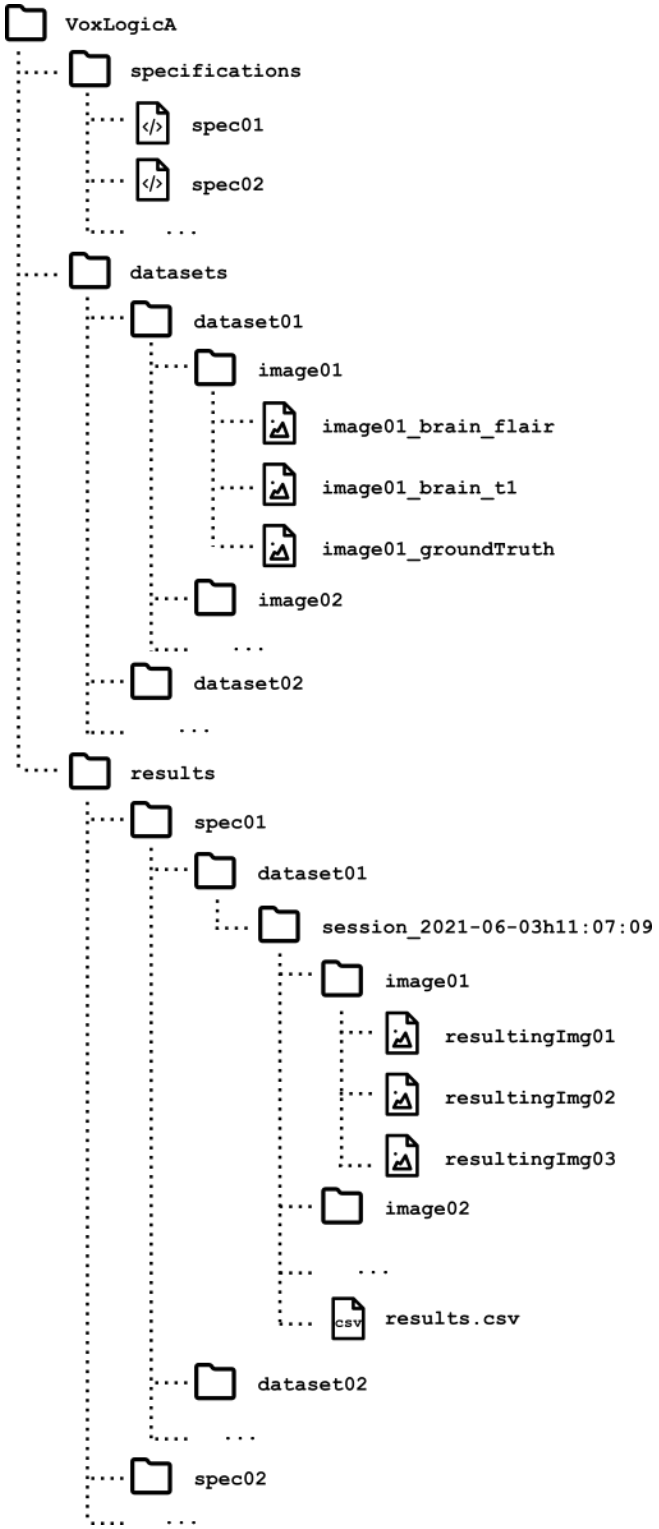


Fig. 12: Example of the VoxLogicA directories hierarchy