ORIGINAL RESEARCH ARTICLE
published: 28 August 2009
doi: 10.3389/neuro.11.027.2009

CamBAfx: workflow design, implementation and application for neuroimaging

***Cinly Ooi*[1,2]**, *Edward T. Bullmore*[1,2]*, *Alle-Meije Wink*[3]*, *Levent Sendur*[1,2]*, *Anna Barnes*[1,2]*, *Sophie Achard*[1,2]*, *John Aspden*[4]*, *Sanja Abbott*[2]*, *Shigang Yue*[5]*, *Manfred Kitzbichler*[2]*, *David Meunier*[1,2]*, *Voichita Maxim*[1,2]*, *Raymond Salvador*[2]*, *Julian Henty*[1,2]*, *Roger Tait*[1,2]*, *Naresh Subramaniam*[2]* and *John Suckling*[1,2]**

[1] *Brain Mapping Unit, Department of Psychiatry, University of Cambridge, Cambridge, UK*
[2] *Behavioural and Clinical Neuroscience Institute, University of Cambridge, Cambridge, UK*
[3] *Imaging Sciences Department, MRC Clinical Sciences Centre, Imperial College, London, UK*
[4] *J.L. Aspden Ltd, Cambridge, UK*
[5] *Department of Computing and Informatics, University of Lincoln, Lincoln, UK*

Edited by:
John Van Horn, University of California, USA

Reviewed by:
Ivo Dinov, University of California, USA
John Van Horn, University of California, USA

****Correspondence:***
Cinly Ooi, Brain Mapping Unit, Herchel Smith Building for Brain And Mind Sciences, University Forvie Site, Robinson Way, Cambridge CB2 0SZ, UK.
e-mail: co224@cam.ac.uk

CamBAfx is a workflow application designed for both researchers who use workflows to process data (consumers) and those who design them (designers). It provides a front-end (user interface) optimized for data processing designed in a way familiar to consumers. The back-end uses a pipeline model to represent workflows since this is a common and useful metaphor used by designers and is easy to manipulate compared to other representations like programming scripts. As an Eclipse Rich Client Platform application, CamBAfx's pipelines and functions can be bundled with the software or downloaded post-installation. The user interface contains all the workflow facilities expected by consumers. Using the Eclipse Extension Mechanism designers are encouraged to customize CamBAfx for their own pipelines. CamBAfx wraps a workflow facility around neuroinformatics software without modification. CamBAfx's design, licensing and Eclipse Branding Mechanism allow it to be used as the user interface for other software, facilitating exchange of innovative computational tools between originating labs.

Keywords: workflow, eclipse, rich client platform, batch processing, camba, open source, pipeline

INTRODUCTION

Workflows are the combination of pipelines (i.e. modules representing individual programs with connecting pipes representing data transfer from one module to another) and data control systems that coordinate data processing on local or distributed computer architectures. Neuroimaging brings together two broad scientific constituencies: the design and implementation of workflows and the application of these workflows to brain imaging datasets. Correspondingly, the demands made upon workflow-based software change according to circumstances.

Conceptually, workflows are a useful way to gain traction over complex data analysis tasks. By decomposing the workflow into constituent parts, the problem is reduced to the creation and maintenance of small, simple programs that can be reused across workflows. To workflow designers (*designers*), the development environment should offer uncomplicated integration of their programs into existing pipelines, quick construction of new pipelines from existing modules and facilities for rapid testing, validation and deployment of workflows.

For those who apply workflows (*consumers*), the small effect sizes and large between-subject variance associated with most neuroimaging techniques call for a simple system for entering data into the workflow at low error rates and data control systems that emphasize high dataset throughput. Ideally, all workflows should follow a common ontology and it should be possible to use the same workflows with different data control systems without modification.

Whilst workflows are important tools for both designers and consumers in their own right, they also form a vital line of communication within the neuroimaging community to disseminate new algorithms or pipelines, optimised module parameters and standardised procedures. A workflow represents the collective wisdom on how to perform a data analysis task and documents the process allowing experience to be reused, transferred, and consolidated.

A review of workflow applications reveals that the majority of existing workflow environments are effective at modifying pipelines, but not optimised for processing large amounts of data. CamBAfx is an Eclipse (International Business Machines, 2006) Rich Client Platform (RCP, McAffer and Lemieux, 2005) based workflow application that provides both a front-end (user interface) optimized for data processing and a back-end pipeline model to facilitate creation and manipulation of pipelines. Additionally, it offers the flexibility of using different process strategies (for example, single machine scripting or grid-based computing) within the same pipeline description.

We begin with a brief overview of alternative workflow applications providing context for the objectives of CamBAfx. The operation of CamBAfx from the viewpoint of consumers is then considered showing how the user interface helps in the analysis of their data. For designers, discussion is orientated towards the delivery of the pipelines and workflows to consumers through the deployment of Eclipse-based facilities. Examples of delivering pipelines from a variety of neuroinformatics packages are given and concluding remarks made on future directions.

WORKFLOW ENVIRONMENTS

Workflows are normally visualised as pipelines, i.e., a collection of modules with pipes to represent the data flow from output ports of one module to the input ports of another. Traditionally, Visual Pipeline Editors (VPEs) are used to manipulate pipelines. VPEs represent pipelines graphically, usually with boxes as modules, lines as pipes, and small shapes inside the module box as input and output ports. Users modify workflows by manipulating this graphical representation, such as adding modules or re-routing pipes. Commercially available software offers workflow capability in two different ways: either specialised for workflow operations (National Instruments' LabView)[1], with a VPE as the main user interface and programming interface for module creation and different data processing strategies, or as extensions to existing programming languages (Simulink)[2] that provide VPEs and programming interfaces for modules to accommodate pipelines.

The LONI Pipeline (Rex et al., 2000) looks and behaves like a traditional Visual Pipeline Editor. To enter data, consumers click on input ports which then request single values or a list of values. Batch processing is achieved by asking the input port of a module to interpret a list of values one-at-a-time instead of all-at-once. For batch-processing, LONI Pipeline offers the run-on-machine method, including via a script containing the individual processing instructions, as well as grid processing. It uses Extensible Markup Language (XML, Bray et al., 2008) to describe the pipeline as a combination of modules, connections, ports and data. Conveniently, meta-data about modules such as their creators and the software suite to which the modules belong can also be stored. LONI Pipeline modules may be downloaded separately to augment the main package.

Fiswidgets (Fissell et al., 2003) visualizes its pipeline as a linear stack without pipes or ports. Modules need not be activated in the order the visual representation implies. Clicking on modules brings up a module window that asks for data and parameters. Fiswidgets' modules are defined in Java or in XML and describe the layout of the module window. For batch-processing a visual programming approach is adopted with loop structures for iteration within the pipeline. Inside the module windows, symbols define inputs and outputs. During data processing the symbols are substituted with the corresponding values from a lookup table. Fiswidgets distributes modules as part of the main software.

BrainVISA (Cointepas et al., 2001) has a collection of workflows each with a "configuration page" that presents a workflow as a tree of modules. Important module parameters can be attached as leaves to the module in the tree, others in an associated detail page. Batch processing is initiated by duplicating the "configuration page" for each dataset. BrainVISA's pipeline is implemented in the form of Python scripts. Pipelines are delivered in toolboxes bundled with the software or downloaded post-installation. The toolbox itself is a directory of configuration files, binary files, text files, help files and python scripts. BrainVISA has an optional database for managing datasets that uses a data ontology and provides software for conversion between images file formats.

In summary, based on the applications' look-and-feel both LONI and FisWidgets give strong emphasis to pipeline manipulation while softwares like BrainVISA prioritise clear data entry. Finally, an established way to deliver workflow-based software is to write a custom user interface for each workflow; some program interfaces in FSL (Smith et al., 2004) and SPM (Friston et al., 1995) fall into this category.

CamBAfx is a user interface for neuroinformatics software designed to support multiple pipelines and to provide the facilities needed to support workflow operation; namely, data management and batch processing. The philosophy is to provide the shortest possible bridge between designers and consumers, iteratively improving processing with pipelines via software development and practical experience. CamBAfx aims to provide resource in equal measure to both constituencies.

CamBAfx

OBJECTIVES

Workflows evolve as algorithms are developed and applications become more demanding. A Workflow environment must therefore be able to maintain flexibility for development while being able to include new applications without modification and maintain a consistent user interface across all pipelines. Thus, a major objective in the design of CamBAfx is to provide for consumers' needs at the front-end, while exploiting the flexibility of workflows at the back-end in order to deliver the pipeline assembly capability for designers. As expectations change, the environment should be flexible enough to refocus these different aspects from front-end to back-end and vice-versa.

The user interface practices a minimalist philosophy: the initial download is a complete, ready-to-use package but only contains those functions that are needed immediately to get started. Consumers customise the interface as dictated by their needs.

Generic functions to manage pipelines and data are provided. Designers are encouraged to make their pipelines more attractive by adding supporting functions.

The environment should reuse existing industrial-grade software and follow existing and de facto standards and practices. Availability of an Integrated Development Environment (IDE) that supports day-to-day programming work such as debugging, version control and automation of mundane tasks greatly improves developers' productivity.

FRONT-END: RESOURCES FOR WORKFLOW CONSUMERS

Our observations indicate that normal practice for workflow consumers is to maintain a library of workflows. Once a workflow has been demonstrated as robust and capable, its composition and parameters are infrequently reconfigured suggesting that it would not be appropriate to focus on workflow manipulation capability for these users. Instead, the biggest workload undertaken by consumers is to enter specific data instances into the workflow and to ensure the data is valid to maximize the success rate of processing. Thus, the front-end of CamBAfx has as its most important undertaking the acceptance and validation of data entered by consumers. Careful validation of the data reduces the number of problematic datasets in a multi-subject dataset, but cannot completely eliminate them. The problems that then arise are corrected between repeats of batch processing. The challenge is to design a system that accommodates multiple repeats, but

[1]http://www.ni.com/labview/

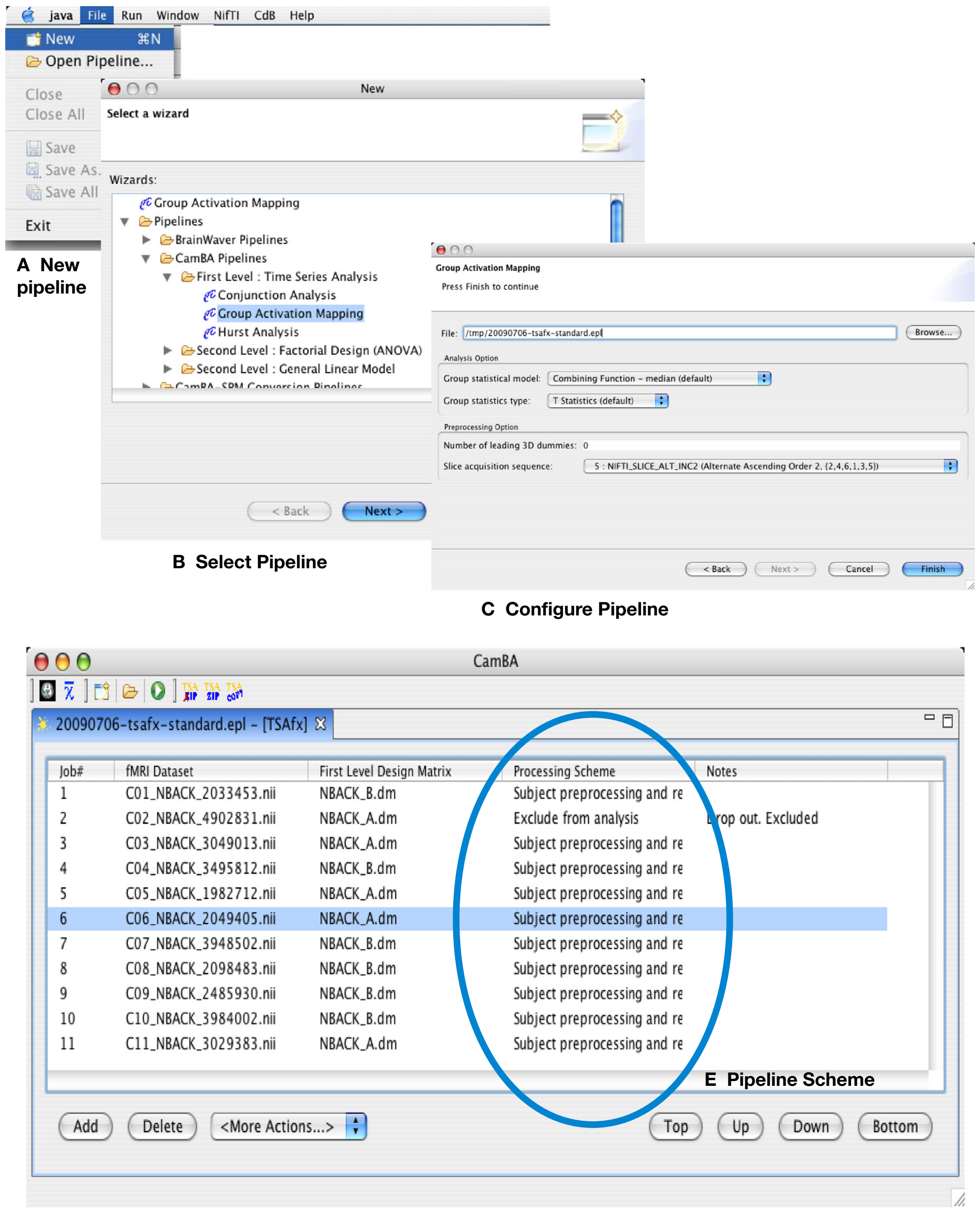
[2]http://www.mathworks.co.uk/products/simulink/

reduces unnecessary reprocessing of datasets already successfully processed. This overall process maps well onto a traditional software usage pattern:

(1) select a workflow and configure it
(2) enter the data into the interface
(3) run the processing in batch mode

Selecting and configuring workflows

In CamBAfx, the process starts by selecting a pipeline from a library of pipelines using a New Wizard (**Figures 1A,B**). A pipeline-specific wizard (**Figure 1C**) is then used to guide the configuration of the pipeline, including a review of the important module parameters and requests to supply values to parameters that cannot have default values. CamBAfx requires pipeline designers to guarantee that the



File: /tmp/20090706-tsafx-standard.epl

Group statistical model: Combining Function – median (default)

Group statistics type: T Statistics (default)

Number of leading 3D dummies: 0

Slice acquisition sequence: 5 : NIFTI_SLICE_ALT_INC2 (Alternate Ascending Order 2, {2,4,6,1,3,5})

20090706-tsafx-standard.epl – [TSAfx]

Job#	fMRI Dataset	First Level Design Matrix	Processing Scheme	Notes
1	C01_NBACK_2033453.nii	NBACK_B.dm	Subject preprocessing and re	
2	C02_NBACK_4902831.nii	NBACK_A.dm	Exclude from analysis	rop out. Excluded
3	C03_NBACK_3049013.nii	NBACK_A.dm	Subject preprocessing and re	
4	C04_NBACK_3495812.nii	NBACK_B.dm	Subject preprocessing and re	
5	C05_NBACK_1982712.nii	NBACK_A.dm	Subject preprocessing and re	
6	C06_NBACK_2049405.nii	NBACK_A.dm	Subject preprocessing and re	
7	C07_NBACK_3948502.nii	NBACK_B.dm	Subject preprocessing and re	
8	C08_NBACK_2098483.nii	NBACK_B.dm	Subject preprocessing and re	
9	C09_NBACK_2485930.nii	NBACK_B.dm	Subject preprocessing and re	
10	C10_NBACK_3984002.nii	NBACK_B.dm	Subject preprocessing and re	
11	C11_NBACK_3029383.nii	NBACK_A.dm	Subject preprocessing and re	

FIGURE 1 | Steps in operation by consumers to select, create and modify workflows. See text for details.

pipeline created at the end of this process is valid and immediately useable.

Data entry via the interface

The pipeline itself is not graphically represented. Instead an Input Table (**Figure 1D**) is presented where all the data necessary for batch processing is specified. The Input Table is customised to the workflow, although there is consistency across the instances of the interface for each pipeline. In general, each row refers to the data for a particular imaging dataset. A table cell only displays the appropriate interactive element determined by the pipeline to solicit data (e.g. text boxes, drop-down lists of choices, file and directory selection dialogs). If the data required is a list, then a new table with one column is presented with the same interactive element facilities as the Input Table. If there are two or more list-based data required, they can each use a separate table or share a multi-column table.

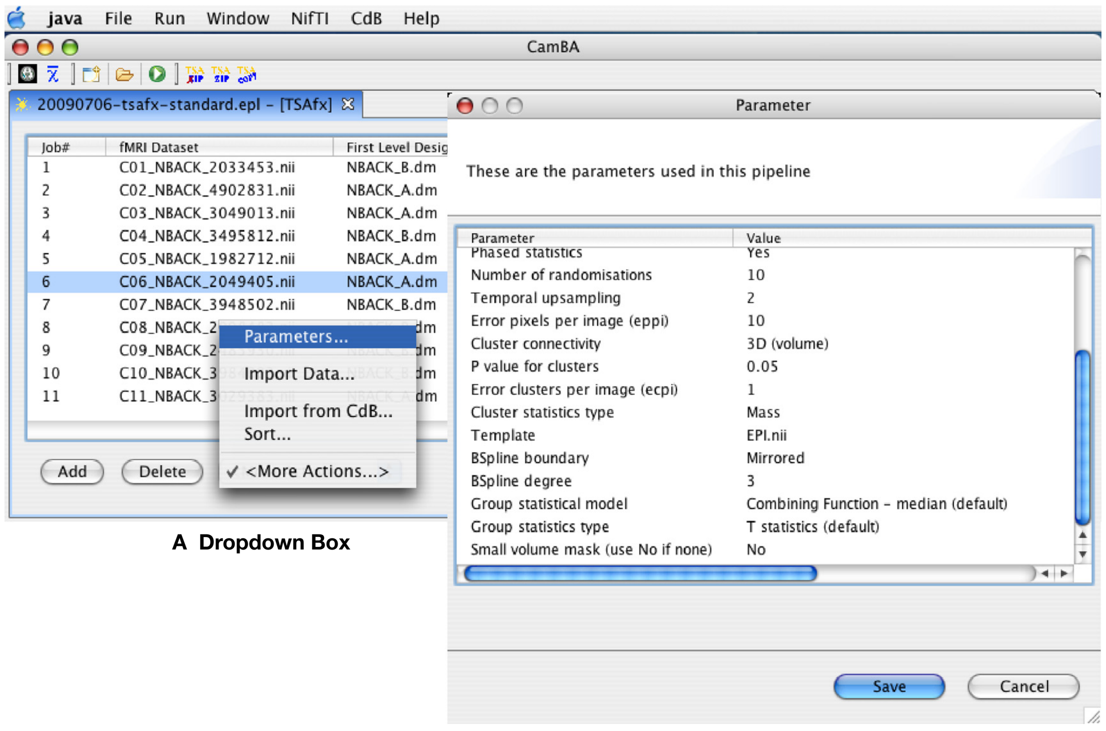
To improve the chances for successful data processing the table cells accept or reject data following input. This can be as simple as rejecting letters when numbers are expected or enforcing specific restrictions imposed by the pipeline, such as minimum and maximum values or lengths. Error messages, possibly containing a message from the pipeline designer, are displayed to the user where available. The Input Table additionally contains a free-text cell entitled "Notes" where annotations can be made about the dataset.

Associated with each dataset is a "Pipeline Schemes" (**Figure 1E**). This is a drop-down list with preconfigured schemes that define the precise list of modules activated in the processing of that dataset. By default, the two schemes that bound the possible processing are available; namely, one that activates all modules and another that entirely bypasses all the modules. Pipeline designers can add new schemes that activate only part of the pipeline and in doing so lead to more efficient analyses of datasets that have been partially processed previously.

A drop-down box below the Input Table (**Figure 2A**) is used to host functions that work on the Input Table as a whole. A function to copy data from another instance of the same pipeline is available. Pipeline designers can add pipeline-specific functions into this drop-down box. The table of parameters (**Figure 2B**) can also be invoked from here. Parameters are variables for modules that remain constant throughout processing of the datasets (e.g. a spatial smoothing kernel). In keeping with the philosophy of a pipeline-centric view, this table shows all parameters for all modules. It uses a two column format with one parameter per row. The first column contains the parameter name and the second its value. The table offers the same interactive elements and validation facilities as the Input Table. For parameters that must share the same value, only one will be listed and any modification here is propagated to all parameters.

Batch mode processing

Once data entry is complete, the workflow is initiated via the "Run Wizard" (**Figures 3A,B,C**). Here additional information required by the data processing engine, such as the summary output directory name, will be requested. Currently, the data processing engine operates by script generation and execution.

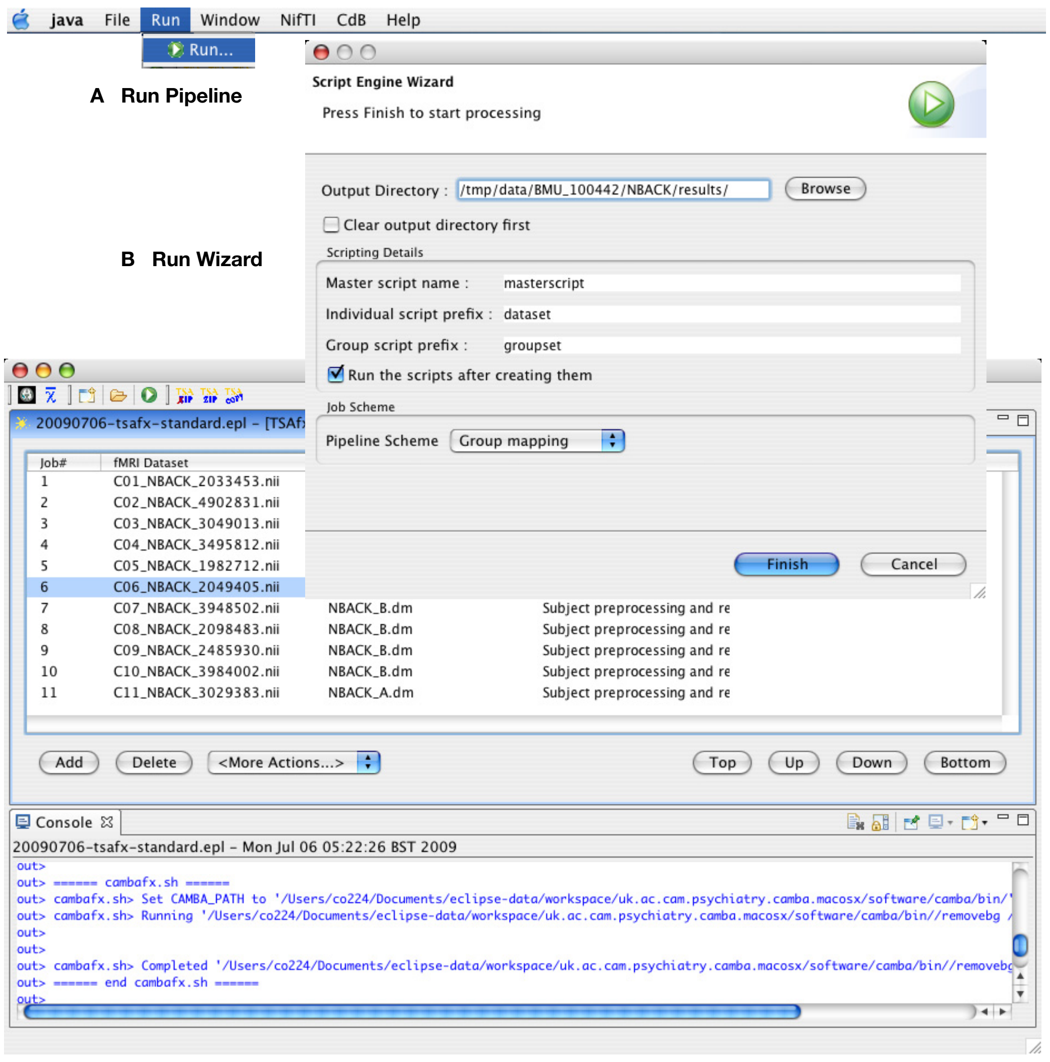


Job#	fMRI Dataset	First Level Desig
1	C01_NBACK_2033453.nii	NBACK_B.dm
2	C02_NBACK_4902831.nii	NBACK_A.dm
3	C03_NBACK_3049013.nii	NBACK_A.dm
4	C04_NBACK_3495812.nii	NBACK_B.dm
5	C05_NBACK_1982712.nii	NBACK_A.dm
6	C06_NBACK_2049405.nii	NBACK_A.dm
7	C07_NBACK_3948502.nii	NBACK_B.dm
8	C08_NBACK_2	dm
9	C09_NBACK_2	dm
10	C10_NBACK_3	dm
11	C11_NBACK_3	dm

These are the parameters used in this pipeline

Parameter	Value
Phased statistics	Yes
Number of randomisations	10
Temporal upsampling	2
Error pixels per image (eppi)	10
Cluster connectivity	3D (volume)
P value for clusters	0.05
Error clusters per image (ecpi)	1
Cluster statistics type	Mass
Template	EPI.nii
BSpline boundary	Mirrored
BSpline degree	3
Group statistical model	Combining Function – median (default)
Group statistics type	T statistics (default)
Small volume mask (use No if none)	No

FIGURE 2 | Steps in operation by consumers to modify parameters for the workflow. See text for details.



Script Engine Wizard

Press Finish to start processing

Output Directory : /tmp/data/BMU_100442/NBACK/results/

☐ Clear output directory first

Scripting Details

Master script name : masterscript

Individual script prefix : dataset

Group script prefix : groupset

☑ Run the scripts after creating them

Job Scheme

Pipeline Scheme: Group mapping

20090706-tsafx-standard.epl – [TSAfx

Job#	fMRI Dataset		
1	C01_NBACK_2033453.nii		
2	C02_NBACK_4902831.nii		
3	C03_NBACK_3049013.nii		
4	C04_NBACK_3495812.nii		
5	C05_NBACK_1982712.nii		
6	C06_NBACK_2049405.nii		
7	C07_NBACK_3948502.nii	NBACK_B.dm	Subject preprocessing and re
8	C08_NBACK_2098483.nii	NBACK_B.dm	Subject preprocessing and re
9	C09_NBACK_2485930.nii	NBACK_B.dm	Subject preprocessing and re
10	C10_NBACK_3984002.nii	NBACK_B.dm	Subject preprocessing and re
11	C11_NBACK_3029383.nii	NBACK_A.dm	Subject preprocessing and re

```
20090706-tsafx-standard.epl – Mon Jul 06 05:22:26 BST 2009
out>
out> ====== cambafx.sh ======
out> cambafx.sh> Set CAMBA_PATH to '/Users/co224/Documents/eclipse-data/workspace/uk.ac.cam.psychiatry.camba.macosx/software/camba/bin/'
out> cambafx.sh> Running '/Users/co224/Documents/eclipse-data/workspace/uk.ac.cam.psychiatry.camba.macosx/software/camba/bin//removebg /
out>
out>
out> cambafx.sh> Completed '/Users/co224/Documents/eclipse-data/workspace/uk.ac.cam.psychiatry.camba.macosx/software/camba/bin//removebg
out> ====== end cambafx.sh ======
out>
```

FIGURE 3 | Steps in operation by consumers to run the workflow. See text for details.

Other practical issues

CamBAfx is a self-extracting archive available for download[3] containing both CamBAfx, the workflow environment, and a set of pipelines based on modules of the CamBA software (Suckling and Bullmore, 2004; Suckling et al., 2006). Also included are supporting functions such as functions to copy the results of one pipeline as the input to another. New pipelines and functions are delivered post-installation as plug-ins that are downloaded, dropped into the original installation and included into the distribution following a restart of the software. Most plug-ins orientated towards consumers modify the user interface to advertise their availability.

BACK-END: RESOURCES FOR WORKFLOW DESIGNERS

Out of the box, CamBAfx has all the generic facilities needed to manage workflows. For all pipelines, CamBAfx provides all the expected facilities to configure pipelines as well as collect, collate and batch-process datasets that together form the workflow. However, since the Eclipse Extension Mechanism (EEM, Bolour, 2003) gives access to the user interface and allows them to contribute new functions, CamBAfx plug-ins customize the user interface to support the specific processing requirements of each pipeline and implement support facilities such as data imports from other pipelines.

Pipeline features

All information CamBAfx needs is contained in the pipeline file, written in XML, with three sections: Pipeline, Input Data and Preferences. The Pipeline section represents the pipeline as a collection of modules and connections. The modules are further decomposed into variables (i.e. installation specific values), parameters, input and output ports, and how to invoke the program. Almost everything describing the pipeline is in XML except for complex data manipulation, such as generating the command

[3]http://www-bmu.psychiatry.cam.ac.uk/software/

line instructions, where Java program code is used in the form of a BeanShell Script[4]. Variables, parameters, input and output ports all carry datatype information (e.g., integer or string) and include restrictions on the data. All pipeline components can have variations on, for example, datatypes, modules (input or standard) and ports (data or signals). They start with a XML element with the same name, but with an attribute that identifies the variant. The attached XML leaf elements change according to the variant. The Input Data section simply contains a description of each dataset as displayed by the Input Table. The Preference section contains optional information about the pipeline such as pipeline schemes and a list of linked parameters that should share the same value.

Steps are taken to make the pipeline simpler and easier to understand: First, looping constructs, normally used to effect batch processing, but complicating data flow, are eliminated by insisting that each dataset is processed through the complete workflow from beginning to end and that each input port can only have one connection. Second, uncertainty about whether an input port needs to be connected is removed by insisting that all ports need to be connected. To satisfy this, and to show where datasets enter the pipeline, each pipeline has one (and only one) input module responsible for communication with the outside world.

Data standard and datatype hierarchy

For effective data exchange between modules, CamBAfx has a Data Standard for all datatypes it uses that defines the file format and the meta-data it must provide. For example, functional magnetic resonance imaging data (fMRI) is in 4D NifTI (Cox et al., 2004) single file format and must carry the sequence in which the slices of the three-dimensional volume were acquired, which is encoded as the `slice_code` meta-data. This approach guarantees the exact content available to designers for writing modules. In return, the output from a module should also satisfy this standard and the designer is responsible for converting data to and from the data format their program expects. Adhering to this data standard means data can be easily exchanged between modules. Designers only have to convert their data to one other format, i.e. to the data standard only and not all possible data formats they might encounter. Although CamBAfx is organised to validate data against the data standard following input, this is postponed until CamBAfx develops the appropriate editors to edit the data in situ as consumers prefer to be able to do this if their data fail validation.

Datatypes are organized into a hierarchy, with each datatype having only one parent and children must carry all data inherited from its parent as well as optional data of its own. A special equivalence is used to define a unidirectional relationship between two datatypes that do not share a common ancestry. This data hierarchy tree is used to prevent incompatible data transfer between modules in the VPE by restricting connection of output ports to input ports that expect the same datatype or its parents.

New pipeline wizards

A new pipeline can be created by cloning, i.e., loading the pipeline into the user interface and then saving it under a new name. This approach may, however, also copy unwanted details from the old pipelines, such as the specific dataset names and modifications to the pipeline. Therefore in CamBAfx, the preferred approach is to create pipelines using a New Pipeline Wizard where the new pipeline is cloned from a clean copy of the parent pipeline and can be manipulated if necessary before being presented to consumers.

CamBAfx DESIGN AND ARCHITECTURE

Eclipse and eclipse rich client platform

CamBAfx is an Eclipse Rich Client Platform (RCP, McAffer and Lemieux, 2005) application. Eclipse[5] (International Business Machines, 2006) was originally created as an IDE with an extension mechanism (Eclipse Extension Mechanism, EEM, also known as Eclipse Plug-in Architecture, Bolour, 2003) designed to integrate development tools. The EEM is a way of extending an Eclipse-aware program. A program that supports extensions publishes an extension point and its expectation. Interested parties then provide extension(s) that latch on to this extension point. Extensions can provide configuration information or program code or both and together with their supporting data, such as icons and programs, are packaged into plug-ins.

Eclipse itself is designed as a collection of plug-ins, with the exception of a small kernel that starts up and bootstraps the EEM. After bootstrapping, the EEM discovers and manages all the installed plug-ins. It then searches the command that invoked it, and if necessary a configuration file, to find the master application. This is read through the EEM and executed. In the original design there was only one master application: the Eclipse IDE. However, the Eclipse Extension Mechanism proved sufficiently useful as a platform for development of standard programs that it was exploited by the Rich Client Platform (RCP) project. The RCP project allows other applications, such as CamBAfx, to be the master application.

All RCP applications are programs built using the EEM, and all share a common architecture and plumbing. RCP developers simply write the missing part, i.e. the program code specific to their project and insert it into the RCP framework.

CamBAfx as a RCP application

CamBAfx, like all RCP applications, is actually a collection of plug-ins. For example, all CamBA's command line programs, pipelines and supporting functions are encapsulated into Eclipse plug-ins and managed through the EEM. Tasks such as creating a New Pipeline Wizard are performed by extending CamBAfx using EEM.

The Eclipse extension point *org.eclipse.core.runtime.applications*, is the only mandatory extension point allowing CamBAfx to be invoked as a master application. CamBAfx also uses other Eclipse extension points such as *org.eclipse.ui.editors* for the main Input Table and *org.eclipse.ui.actionSets* to add menu and toolbars items. CamBAfx also defines its own extension points including the *org.genericfx.ui.inputtable.taskagents* extension point which adds items to the Input Table's drop-down box. Extension points, such as *org.genericfx.data.hierarchy* which define the datatype hierarchy customize CamBAfx for designers. CamBAfx also provides a special generic New Pipeline Wizard for the *org.eclipse.ui.newWizards* extension point eliminating the need to write generic wizards for pipelines by using instead CamBAfx's *org.genericfx.ui.base.newWizards* extension

[4]http://www.beanshell.org/

[5]http://www.eclipse.org/

point to read in the pipeline from a file. Part of CamBAfx, such as new items for the Input Table drop-down box, are constructed by extending its own extension points. All extension points, either those of Eclipse or CamBAfx, are available to downstream developers who can also define their own.

Developing for CamBAfx

As standard Eclipse plug-ins, CamBAfx and its plug-ins are developed using Eclipse's Plug-in Development Environment (PDE, Melhem and Glozic, 2003) that is designed specifically to develop, test and integrate plug-ins with their intended application. CamBAfx provides an editor, integrated into the IDE, for development and testing of pipelines. This editor has the Input Table and a rudimentary VPE. CamBAfx has two data processing engines: traditional batch processing controlled directly by the program itself and a version that writes and then executes the processing steps via scripts. Both are callable from the IDE via its Run Wizard.

Eclipse also makes available supporting software facilities, such as an update mechanism and help browser. It provides tools for CamBAfx such as the Graphical Editor Framework[6] (GEF, Hudson, 2004) which is the basis of CamBAfx's VPE.

Developers "pick-and-mix" CamBAfx plug-ins for their applications. Architecturally, there are three major parts: Pipeline, Input Table and Data Processing Engine (**Figure 4**). These three parts are kept independent of each other with minimum communication between them. Conceptually, the software is developed in three layers (**Figure 5**): At the bottom is GenericFX, a complete generic pipeline application; BrainFX is the middle layer that customizes GenericFX for neuroinformatics applications by defining the data hierarchy, data standard and some commonly used routines, such as NifTI data conversion. CamBAfx is the top layer and contains only CamBA-specific pipelines and functionalities. Third party developers who do not need CamBA can create their applications from either GenericFX or BrainFX. The same Eclipse Branding Mechanism (Eidsness and Rapicault, 2004) that defines CamBA's own About Dialog, splash screen and icons can be used to brand other applications.

[6]http://www.eclipse.org/gef/

IMPLEMENTATION OF PIPELINES

CamBA ANALYSIS PIPELINES

CamBA is software for the analysis of neuroimaging data. The initial download contains a number of pipelines available for first-level (within-subject) and second-level (between-subject) analysis for which CamBAfx provides customised interfaces. The CamBAfx application running CamBA pipelines has been widely used in the analysis of functional and structural MRI (examples include: Chamberlain et al., 2008, 2009; Habets et al., 2008; Menzies et al., 2008; Wink et al., 2008).

CamBA's first-level analysis pipelines' main purpose is to generate maps that summarise responses or signal properties from raw 4D fMRI. For example, a "time-series analysis pipeline" pre-processes the data removing subject movement related artefacts followed by response estimation with the general linear model. The resulting effect maps are mapped into a standard stereotactic space in readiness for second-level pipelines.

Consumers start by choosing the "group activation mapping" pipeline from the library of pipelines (**Figures 1A,B**). Its pipeline wizard (**Figure 1C**) can configure the pipeline to perform housekeeping tasks to meet the Data Standard, such as inserting the correct `slice_code` into the fMRI 4D data and removing unwanted 3D scans from the start of the data. The Input Table (**Figure 1D**) asks for the fMRI data and the design matrix file. Its Pipeline Schemes are carefully selected to activate parts of the pipeline according to the specified usage of the pipeline.

At the second level, pipelines that offer flexibility in choosing different statistical models present a more difficult challenge for parameter configuration, with many parameters dependent on others. The pipeline can be invalidated if the wrong combination of parameter values is chosen. The corresponding New Pipeline Wizard therefore guides consumers by changing the display according to the model required. At pipeline creation, the available parameter values are screened to remove incompatibilities. The Wizard adds,

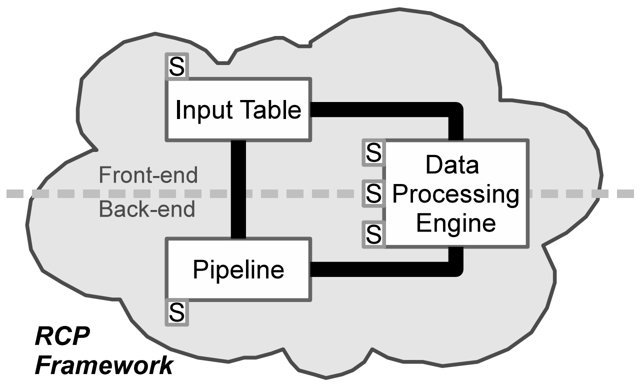


FIGURE 4 | Conceptually, CamBAfx application is constructed from of three components: Pipeline, Input Table and Data Processing Engine, integrated inside the Eclipse Rich Client Platform (RCP) framework. Minimal communication between each component allows components to be replaced or removed. The separation between front-end (user interface) and back-end can be reconfigured as needs change. "S" denotes supporting functions which can be attached to the application as plug-ins.

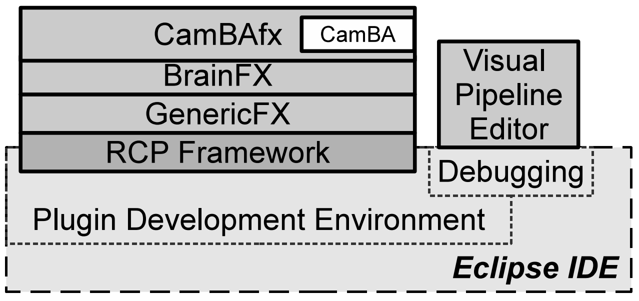


FIGURE 5 | Architecturally, CamBAfx is organized in three layers (Solid boxes): CamBAfx, BrainFX and GenericFX. CamBA is unmodified and completely contained inside a plug-in in the CamBAfx layer. They are all built on top of the Eclipse RCP Framework and developed inside Eclipse Integrated Development Environment's (IDE) Plug-in Development Environment (dashed boxes). CamBAfx extends the Eclipse IDE with a Visual Pipeline Editor to visualize the pipelines during their development and testing.

on request, new ports and connections to the pipeline that represent additional variables. These variables also appear on the Input Table as additional columns. The majority of the Input Table columns are programmed to accept numbers only and where appropriate are further restricted to a small range of values. In effect, the wizard creates different variations of pipelines for the consumers. All second-level pipelines insert an item into the drop-down box below the Input Table that can import results from first-level pipelines.

In general, data generated by one software suite cannot be used by another because the data are stored as a different data type. The most common data type mismatch is 32 bit and 64 bit floating-point data and therefore CamBAfx provides a pipeline to convert data between these formats. Additional information for performing data type conversions from specific software suite is available inside the Help Browser bundled with the core CamBAfx download.

For first-level pipelines, the repetitive entering of data is assisted by a supporting function for automatically reading data into the Input Table from a directory-based data organization. Following download and installation, it adds itself to the drop-down box of the Input Table. Another download adds a menu item to extract statistics from data in predefined regions-of-interest (anatomical or identified by statistical testing). Finally, users can download a menu item that modifies the NifTI header data in batch mode and checks that the modification satisfies the data standard.

IMPLEMENTATION OF FSL TRACK-BASED SPATIAL STATISTICS

To illustrate the flexibility of the CamBAfx approach, a plug-in (TBSSfx) is available which repackages the tract-based spatial statistics (TBSS, Smith et al., 2006) software for diffusion tensor image analysis, available as part of the FSL package. Since TBSS is part of the FSL pipeline, licensing restrictions require a separate download of FSL[7]. In brief, TBSS is a five step process:

(1) Input data is organised into a directory. Pre-processing software relocates input data into a subdirectory.
(2) If there is a target image that defines the stereotactic space of the analysis, copy and rename into the subdirectory. The target image cannot be copied until step 1 is completed.
(3) The analysis software is executed.
(4) A design matrix and a contrast file are created and further analysis takes place.
(5) Call a collection of programs to perform voxel-wise statistical analysis.

There are a number of restrictions on these steps, particularly with regard to the order in which they are conducted. Furthermore, construction of the design matrix is interactive and unconstrained.

TBSSfx is a collection of plug-ins with a plug-in used to host the FSL archive, which the consumers download separately. TBSSfx simplifies data entry and automates the processing ensuring compliance with the restrictions on the processing steps. For example, during pipeline creation, TBSSfx asks the user to name the number of conditions (columns) for the design matrix and to specify the contrast file and then validates this against the format of the design matrix. The contrast file is defined at this stage (and not later in the pipeline) to guarantee that the pipeline created is configured correctly. In the Input Table consumers enter the image data filename in the first row with subsequent columns only accepting numerical data corresponding to the design matrix.

[7]http://www.fmrib.ox.ac.uk/fsl/

The Run Wizard asks for an output directory, which is cleaned and populated with hard links to the actual data for speed and economy of resources as well as ensuring that the original data are preserved. The design matrix file and contrast file are then created with filenames constructed to maintain the list orders from the Input Table. The processing script manages data processing in a way consistent with the original TBSS process.

DISCUSSION

CamBAfx is an application that presents workflows according the needs of users: designers or consumers. The initial download consists of the basic program only. New functionalities and pipelines can be added post-installation maintaining the installation to a size adequate for local needs. This is made possible by the EEM which manages plug-ins for consumers.

The overall organisation is as Input Table, Pipeline Configuration and Data Processing Engine. The Input Table presents the full view of the datasets, allows users to take notes and fine tune the actual processing of individual datasets. Both Input Table and Parameter Table validate and reject invalid data. These are all designed to improve the chances of successful data processing.

CamBAfx packages neuroinformatics software, without modification, inside plug-ins. Other CamBAfx plug-ins provide the branding, the pipelines and their New Pipeline Wizards as well as supporting functions. Pipelines are organized into directories and each pipeline comes with its own customized wizards.

The back-end's aim is to deliver workflows to the user. It uses the traditional pipeline view of the workflow making modifications straightforward. Facilities like data hierarchy, data standards and pipeline simplification strategies are designed to assist pipeline construction and improve readability. Pipelines are written in XML for human-readability and can be manipulated programmatically.

For developers, CamBAfx supplies a generic set of functions for their pipelines. However, customization of CamBAfx is encouraged by developing supporting facilities. These supporting functions have access to the user interface via Eclipse or CamBAfx extension points.

Organising software in a consistent manner facilitates construction of new pipelines from modules originating from different software packages and is an important design objective for CamBAfx. Analysis software is not merely repackaged, rather consumers and designers can integrate tools to generate custom workflows or undertake optimisation of pipelines through systematic comparison of modules.

Using Eclipse RCP technology means that CamBAfx uses industrial standard architecture reducing development time and ensuring that the underlying technology is constantly updated and improved. Eclipse-based tools can be incorporated easily and CamBAfx can integrate with other Eclipse programs. Eclipse's PDE is a useful aid for developing CamBAfx and its plug-ins. CamBAfx's extensions for Eclipse IDE allows plug-in integration to be debugged and tested using PDE. The source code is organized in a logical and flexible manner to maximize reuse potential. Workflow applications can be developed from BrainFX or GenericFX if CamBA is not needed.

CamBAfx is released under the terms of General Public License (GPL, Free Software Foundation, 2007) and specifically allows designers to integrate their pipelines before shipping. This removes problems associated with consumers having to download pipelines and workflow applications separately and following instructions to integrate them to form the final application.

GenericFX, BrainFX, CamBAfx and TBSSfx can be downloaded from SourceForge.net[8] or NITRC[9].

CONCLUSION

CamBAfx is a workflow application designed to be the user interface that services consumers' needs in the front-end by guiding them throughout the whole process from pipeline creation, through data entry and validation, to data processing. At the back-end, workflow creation and manipulation are made easier by adopting a pipeline model complete with a strategy to understand and use a data standard and data hierarchy as well as facilities to manipulate these pipelines. Out of the box, CamBAfx provides all the generic facilities expected of a workflow application for any pipeline although, uniquely, designers are encouraged to customize CamBAfx for their own pipelines. CamBAfx is built as an Eclipse RCP application and benefits from industrial standard architecture and modern software facilities, such as supporting post-installation modification. EEM makes CamBAfx highly flexible, configurable and extensible. Designers use it to customise CamBAfx for their pipelines, to insert supporting functions and to access the user interface. Moreover, by selecting components from CamBAfx and with the help of Eclipse Branding Mechanism, new workflow applications can be created. The availability of PDE, designed to support Eclipse plug-in developments, improves CamBAfx designers' productivity.

[8]http://sourceforge.net/projects/camba/

[9]http://www.nitrc.org/projects/camba/

FUTURE WORK

New versions of CamBAfx will use EEM more extensively. Small utility programs are being developed to check that the CamBAfx instance is error free. The current XML pipeline descriptor can contain two or more ways to describe the same data. This will be reduced to one as part of the effort to rationalise the XML descriptors. The new XML will use XML Namespace (Bray et al., 2006) and support XML Schema (Fallside and Walmsley, 2004) validation. Meta-data such as the author's name and email, are managed centrally using the Resource Description Framework (RDF, Beckett, 2004), removing duplication and simplifying updates. RDF also stores the relationship between meta-data.

ACKNOWLEDGEMENTS

This neuroinformatics research was supported by a Human Brain Project grant from the National Institute of Mental Health and the National Institute of Biomedical Imaging and Bioengineering. EB is employed 50% by GlaxoSmithKline and 50% by the University of Cambridge. The work was conducted in the MRC/Wellcome Trust Behavioural & Clinical Neurosciences Institute, Cambridge UK. This project was awarded an IBM Eclipse Innovation Award 2003.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at http://www.frontiersin.org/neuroinformatics/paper/10.3389/neuro.11/027.2009/

REFERENCES

Beckett, D. (ed.) (2004). RDF/XML Syntax Specification (Revised). W3C Recommendation 10 February 2004. Available at: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/. RDF Interest Group, http://www.w3.org/RDF/.

Bolour, A. (2003). Notes on the Eclipse Plug-in Architecture (Eclipse Corner Article). Available at: http://www.eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html.

Bray, T., Hollander, D., Layman, A., and Tobin, R. (eds) (2006). Namespaces in XML 1.0, 2nd Edn. W3C Recommendation 16 August 2006. Available at: http://www.w3.org/TR/2006/RECxml-names-20060816/. XML Core Working Group, http://www.w3.org/XML/Core/.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. (2008). Extensible Markup Language (XML) 1.0, 5th Edn. W3C Recommendation 26 November 2008. Available at: http://www.w3.org/TR/REC-xml/. XML Core Working Group, http://www.w3.org/XML/Core/.

Chamberlain, S. R., Hampshire, A., Müller, U., Rubia, K., Del Campo, N., Craig, K., Regenthal, R., Suckling, J., Roiser, J. P., Grant, J. E., Bullmore, E. T., Robbins, T. W., and Sahakian, B. J. (2009). Atomoxetine modulates right inferior frontal activation during inhibitory control: a pharmacological functional magnetic resonance imaging study. *Biol. Psychiatry* 65, 550–555.

Chamberlain, S. R., Menzies, L., Hampshire, A., Suckling, J., Fineberg, N. A., del Campo, N., Aitken, M., Craig, K., Owen, A. M., Bullmore, E. T., Robbins, T. W., and Sahakian, B. J. (2008). Orbitofrontal dysfunction in patients with obsessive-compulsive disorder and their unaffected relatives. *Science* 321, 421–422.

Cointepas, Y., Mangin, J.-F., Garnero, L., Poline, J.-B., and Benali, H. (2001). BrainVISA: software platform for visualization and analysis of multimodality brain data. *Neuroimage* 13, S98. Available at: http://www.brainvisa.info/.

Cox, R. W., Ashbourner, J., Breman, H., Fissell, K., Haselgrove, C., Holmes, C. J., Lancaster, J. L., Rex, D. E., Smith, S. M., Woodward, J. B., and Strother, S. C. (2004). A (Sort of) New Image Data Format Standard: NIfTI-1. 10th Annual Meeting of the Organization for Human Brain Mapping (OHBM 2004), Budapest, Hungary, June 13–17. Available at: http://nifti.nimh.nih.gov/nifti-1/documentation/hbm_nifti_2004.pdf.

Eidsness, A., and Rapicault, P. (2004). Branding Your Application (Eclipse Corner Article). Available at: http://www.eclipse.org/articles/Article-Branding/branding-your-application.html.

Fallside, D. C., and Walmsley, P. (eds) (2004). XML Schema Part 0, Primer, 2nd Edn. W3C Recommendation 28 October 2004. Available at: http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/. XML Schema Working Group, http://www.w3.org/XML/Schema/.

Fissell, K., Tseytlin, E., Cunningham, D., Iyer, K., Carter, C. S., Schneider, W., and Cohen, J. D. (2003). Fiswidgets: a graphical computing environment for neuroimaging analysis. *Neuroinformatics* 1, 111–125. Available at: http://grommit.lrdc.pitt.edu/.

Free Software Foundation (2007). GNU General Public License. Available at: http://www.gnu.org/licenses/gpl.html.

Friston, K. J., Holmes, A. P., Worsley, K. J., Poline, J. B., Frith, C., and Frackowia, R. S. J. (1995). Statistical parametric maps in functional imaging: a general linear approach. *Hum. Brain Mapp.* 1995, 189–210. Available at: http://www.fil.ion.ucl.ac.uk/spm/.

Habets, P., Krabbendam, L., Hofman, P., Suckling, J., Oderwald, F., Bullmore, E., Woodruff, P., Van Os, J., and Marcelis, M. (2008). Cognitive performance and grey matter density in psychosis: functional relevance of a structural endophenotype. *Neuropsychobiology* 58, 128–137.

Hudson, R. (2004). The Graphical Editing Framework. EclipseCon 2–5 February 2004, Anaheim, CA, USA. Available at: http://www.eclipsecon.org/2004/EclipseCon_2004_TechnicalTrackPresentations/47_Hudson.pdf.

International Business Machines (2006). Eclipse Platform Technical Overview (Eclipse White Paper). Available at: http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.pdf.

McAffer, J., and Lemieux, J.-M. (2005). Eclipse Rich Client Platform – Designing, Coding and Packaging Java Applications (Addison-Wesley Professional). RCP website, Available at: http://wiki.eclipse.org/index.php/Rich_Client_Platform.

Melhem, W., and Glozic, D. (2003). PDE Does Plug-ins (Eclipse Corner Article). Available at: http://www.eclipse.org/articles/Article-PDE-does-plugins/PDE-intro.html. PDE project website at http://www.eclipse.org/pde/.

Menzies, L., Williams, G., Chamberlain, S., Ooi, C., Fineberg, N., Suckling, J., Sahakian, B., Robbins, T., and Bullmore, E. (2008). White matter abnormalities in patients with obsessive-compulsive disorder and their first-degree relatives. *Am. J. Psychiatry* 165, 1308–1315.

Rex, D. E., Ma, J. Q., and Toga, A. W. (2000). The LONI pipeline processing environment. *Neuroimage* 19, 1033–1048. Available at: http://pipeline.loni.ucla.edu/.

Smith, S. M., Jenkinson, M., Johansen-Berg, H., Rueckert, D., Nichols, T. E., Mackay, C. E., Watkins, K. E., Ciccarelli, O., Cader, M. Z., Matthews, P. M., and Behrens, T. E. J. (2006). Tract-based spatial statistics: voxelwise analysis of multi-subject diffusion data. *Neuroimage* 31, 1487–1505. Available at: http://www.fmrib.ox.ac.uk/fsl/tbss/index.html. Implementation described her is based on the version in 2007: http://web.archive.org/web/20070703045209/http://www.fmrib.ox.ac.uk/fsl/tbss/index.html.

Smith, S. M., Jenkinson, M., Woolrich, M. W., Beckmann, C. F., Behrens, T. E. J., Johansen-Berg, H., Bannister, P. R., De Luca, M., Drobnjak, I., Flitney, D. E., Niazy, R., Saunders, J., Vickers, J., Zhang, Y., De Stefano, N., Brady, J. M., and Matthews, P. M. (2004). Advances in functional and structural MR image analysis and implementation as FSL. *Neuroimage* 23, 208–219. Available at: http://www.fmrib.ox.ac.uk/fsl/.

Suckling, J., and Bullmore, E. (2004). Permutation tests for factorially designed neuroimaging experiments. *Hum. Brain Mapp.* 22, 193–205.

Suckling, M., Davis, M., Ooi, C., Wink, A. M., Fadili, J., Salvador, R., Welchew, D., Sendur, L., Maxim, V., and Bullmore, E. (2006). Permutation testing of orthogonal, factorial effects in a language processing experiment using fMRI. *Hum. Brain Mapp.* 27, 425–433.

Wink, A. M., Bullmore, E., Barnes, A., Bernard, F., and Suckling, J. (2008). Monofractal and multifractal dynamics of low frequency endogenous brain oscillations in functional MRI. *Hum. Brain Mapp.* 29, 791–801.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 31 March 2009; paper pending published: 26 June 2009; accepted: 02 August 2009; published online: 28 August 2009.

*Citation: Ooi C, Bullmore ET, Wink A-M, Sendur L, Barnes A, Achard S, Aspden J, Abbott S, Yue S, Kitzbichler M, Meunier D, Maxim V, Salvador R, Henty J, Tait R, Subramaniam N and Suckling J (2009) CamBAfx: workflow design, implementation and application for neuroimaging. Front. Neuroinform. **3**:27. doi: 10.3389/neuro.11.027.2009*