Πτυχιακή Εργασία

Ανάπτυξη της παραμετρικής γραφικής διεπαφής χρήστη της αστρονομικής υπηρεσίας Remote Interface for Science Analysis

The Remote Interface for Science Analysis (RISA) parameter GUI development

Σπουδάστρια: Ειρήνη Κομνηνού
ΣΤΕΦ - Τμ. Ηλεκτρολογίας
Τεχνολογικό Εκπαιδευτικό Ιδρυμα Κρήτης

Επιβλέπων καθηγητής: Δρ. Ευστράτιος Γεωργίου
ΣΤΕΦ - Τμ. Ηλεκτρολογίας
Τεχνολογικό Εκπαιδευτικό Ιδρυμα Κρήτης

Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης (ΤΕΙΚ)

Μάρτιος 2010

CONTENTS

Part I

INTRODUCTION

1

INTRODUCTION

1.1 αντί προλόγου

Το Σεπτέμβριο του 2008, η γράφουσα ξεκίνησε την πρακτική της άσκηση στο Ευρωπαϊκό Κέντρο Αστρονομίας και Διαστημικής (European Space Astronomy Centre - ESAC) της Ευρωπαϊκής Υπηρεσίας Διαστήματος (European Space Agency - ESA) το οποίο εδρεύει στην Μαδρίτη. Παράλληλα αναπτύχθηκε ένα νέο λογισμικό στα πλαίσια της πτυχιακής εργασίας, υπό την εποπτεία του Δρ. Ευστράτιου Γεωργίου (ΤΕΙ Κρήτης). Για την εκπόνηση του τεχνικού μέρους της πτυχιακής εργασίας χρησιμοποιήθηκε η γλώσσα προγραμματισμού Θαα[1], η οποία μπορεί να θεωρηθεί πιθανότατα ως η πιο διαδεδομένη αντικειμενοστραφής γλώσσα προγραμματισμού σήμερα. Κατά τη διάρκεια της 6μηνης περιόδου πρακτικής άσκησης αποκομίσθηκαν νέες γνώσεις και δεξιότητες στον τομέα επιστήμης και τεχνολογίας υπολογιστών κυρίως, με την καθοδήγηση των Δρ. Κάρλος Γκαμπριέλ (ESAC) και Αϊτόρ Ιμπάρρα (ESAC).

Έχοντας υπόβαθρο Ηλεκτρολόγου μηχανικού με έμφαση στην μελέτη και ανάπτυξη υλικού και με εμπειρία σε δομημένο προγραμματισμό (γλώσσα C) μόνο, η διαδικασία εκμάθησης και χρήσης μιας αντικειμενοστραφούς γλώσσας προγραμματισμού όπως η Java αποτέλεσε πρόκληση δεδομένου του περιορισμένου χρονικού περιθωρίου που προσφέρει η περίοδος πρακτικής άσκησης.

Ως μέλος της ομάδας του Επιστημονικού Κέντρου Επιχειρίσεων (Science Operations Centre) του δορυφόρου XMM-Newton,

[1] Η Θαα είναι γλώσσα προγραμματισμού που αναπτύχθηκε το 1995 απ΄την εταιρία Συν Μιςροσψστεμς σαν δομικό στοιχείο της πλατφόρμας Θαα. Η σύνταξη της γλώσσας προέρχεται απ΄την σύνταξη των γλωσσών ῝ · ῝++ με αρκετές απλοποιήσεις και λιγότερες λειτουργίες χαμηλού επιπέδου (λοω λεελ). Οι εφαρμογές Θαα μεταφράζονται μέσω του ςομπιλερ σε σύνολα εντολών ονόματι ςλασς και μπορούν να τρέξουν σε οποιοδήποτε Θαα ῞ιρτυαλ Μαςηινε (Θ῞Μ)

οι κύριες υποχρεώσεις μου περιελάμβαναν μεταξυ άλλων την μελέτη και ανάπτυξη μιας νέας παραμετρικής γραφικής διεπαφής χρήστη για την αστρονομική υπηρεσία Remote Interface for Science Analysis (RISA), και αποτελεί την πτυχιακή μου εργασία. Η νέα γραφική διεπαφή χρήστη συμβάλλει στην εύκολη διαχείριση των δυνατοτήτων του RISA από έμπειρους και άπειρους χρήστες, κάθιστόντας την εφαρμογή πιο ελκυστική και προσιτή σε ένα ευρύ φάσμα επιστημόνων του χώρου.

Η υπηρεσία RISA που άρχισε να αναπτύσσεται απ'την ESA πρόσφατα, αποτελείται από ένα σύστημα client/server που έχει την δυνατότητα να προσφέρει όλες τις λειτουργίες του ήδη υπάρχοντος αλλά πεπαλαιωμένου Science Analysis Software (SAS)[2] καθιστόντας εφικτή την χρησιμοποίηση όλων των δυνατοτήτων του SAS χωρίς να απαιτείται τοπική εγκατάσταση του προγράμματος.

Το RISA έχει πολύ μικρές απαιτήσεις πόρων συστήματος και μπορεί να δουλέψει με σύνδεση διαδικτύου οποιασδήποτε ταχύτητας. Λαμβάνοντας υπόψιν τους πόρους[3] που απαιτεί το SAS σε σύγκριση με τις απαιτήσεις του RISA, η ESA θεωρεί επιβεβλημένη την αντικατάσταση του SAS απ'το RISA.

Οι πηγές που χρησιμοποιήθηκαν για την ανάπτυξη της παρούσας εργασίας ήταν αρχικά μερικές επιστημονικές εργασίες σαν εισαγωγή, με κύρια πηγή πληροφοριών το διαδίκτυο. Η διαδικασία έρευνας μέσω διαδικτύου υπήρξε πολύ διδακτική, οδήγησε την γράφουσα στην ανάπτυξη ερευνητικής και ανεξαρτητης σκέψης

[2] Το SAS αναπτύχθηκε σαν συνοδευτικό λογισμικό της αποστολής XMM-Newton. Αποτελεί το λογισμικό απλοποίησης δεδομένων της αποστολής και σχεδιάστηκε με σκοπό να μετατρέπει τα ανεπεξέργαστα δεδομένα των παρατηρήσεων του δορυφόρου σε ορθά βαθμονομημένα και επεξεργασμένα επιστημονικά δεδομένα έτοιμα προς ανάλυση. Για παράδειγμα το SAS προσφέρει στους αστρονόμους εικόνες, φασματικές αναλύσεις, καμπύλες φωτός κ.ά. Χρησιμοποιείται απ'ολους τους αστρονόμους που πραγματεύονται με παρατηρήσεις ακτίνων X και επιλέγουν να χρησιμοποιήσουν τα δεδομένα του δορυφόρου XMM-Newton για την έρευνά τους

[3] Αφένός απαιτείται χρονοβόρα και πολύπλοκη διαδικασία εγκατάστασης του κυρίως προγράμματος και των συνοδευτικών υποπρογραμμάτων απ'τον χρήστη. Επίσης απαιτείται μεγάλη υπολογιστική ισχύ κατα τις διαδικασίες ανάλυσης των παρατηρήσεων του δορυφορου. Αφέταίρου απαιτούνται αυξημένοι πόροι για την ανάπτυξη και συντήρηση της εφαρμογής SAS με αποτέλεσμα την αύξηση του συνολικού προϋπολογισμού της αποστολής XMM-Newton

και μεθοδολογίας εργασίας.

Ο σκοπός της παρούσας εργασίας είναι να παρουσιαστεί η τεχνολογία, η μεθοδολογία (συμπεριλαμβανομένου του πηγαίου κώδικα) και η φιλοσοφία που χρησιμοποιήθηκε για την ανάπτυξη της νέας γραφικής διεπαφής του RISA. Κάθε κομμάτι πηγαίου κώδικα περιέχει σχόλια με λεπτομερείς επεξηγήσεις τα οποία καθοδηγούν τον μέσο αναγνώστη που έχει βασικό υπόβαθρο θεωρίας προγραμματισμού. Γιάυτό το λόγο, η εργασία χωρίζεται σε δύο μέρη: Το κυρίως κείμενο που δίδει δομικές πληροφορίες και επεξήγηση της μεθοδολογίας που ακολουθήθηκε και ο πηγαίος κώδικας –μαζί με τα συνοδευτικά σχόλια– που βρίσκεται στα παραρτήματα.
Προτείνεται να γίνει παράλληλη ανάγνωση του κυρίως κειμένου και των παραρτημάτων που αντιστοιχούν σε κάθε κεφάλαιο.

1.2 GENERAL INTRODUCTION

In September 2008, the author of this thesis started working as a trainee at the European Space Astronomy Centre, the European Space Agency centre for astronomy and space science. During the six month placement period and under the tutorship of Dr. Carlos Gabriel (ESAC) and Aitor Ibarra (ESAC) , new skills and aspects of computer technology were mastered. Furthermore a new piece of software was developed serving as the author's thesis project, under the tutorship of Dr. Efstratios Georgiou (TEIC). *Java*[4] –probably the most popular object oriented language as we speak– was used for the developement of the source code of this project.

Having an Electrical engineering background with emphasis on hardware analysis & design and experience in imperative programming (C programming language) but no experience in any object oriented programming languages, learning and manipulating a high level object oriented language like *Java* was a challenge, given the tight time frame.

Working as part of the XMM-Newton Science Operations Centre software development team, the author's main responsibilities included developing a new parameter Graphical User Interface (GUI) for the Remote Interface for Science Analysis (RISA) astrophysical service as her thesis project, making RISA practical and easy to use, thus more appealing to the end user. RISA is a client/server interface offering all the functionalities available in Science Analysis System[5] (SAS), making it possible for anyone to uti-

[4] Java is a programming language originally developed at Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture.

[5] SAS was developed as part of the XMM-Newton's mission infrastructure. It is XMM-Newton's data reduction software designed to convert raw data to scientifically useful products, like calibrated event lists, images, spectra, lightcurves, source lists, etc. It's used by all X-ray astronomers who need to utilize XMM-Newton data for scientific research and analysis.

lize all SAS functions remotely without the need of a local installation. RISA requires minimal processing power and other computer resources (RAM etc) and will work with any type of internet connection. Given the very low resource requirements of RISA and the fact that developing and maintaining SAS is resource consuming[6], RISA is scheduled to substitute SAS soon. This thesis project will be implemented to RISA on the following version release.

A few papers served as references initially but the real knowledge derived almost entirely from the internet, with the ESAC tutors acting more like advisors rather than reference points. This approach was helpful and didactic, leading to a new more self sufficient way of thinking and working even though the learning curve became quite steep and the development process more time consuming. It also proved that the internet is a vast source of information, providing high level information to people ranging from hobbyists, to students and professionals.

This document aims in describing the technology, methodology, technical aspects and work philosophy applied that led to the successful development of the new RISA parameter GUI, i.e this thesis project. The entire source code can be found in the Appendices on the last chapter of the thesis. Each *Java* class contains explanation notes wherever needed, able to guide any programmer with basic *Java* syntax knowledge. Thus the main text contains program structure information and methodology explanation rather than detailed explanations of the source code itself. Parallel reading of the thesis text along with the source code –included in the Appendices– is recommended by the author.

[6] From the end user's point of view: Performing a local installation of SAS is time consuming, running SAS is an intensive task requiring many computer resources. From the developers' point of view: SAS requires manpower, appropriate hardware and a corresponding budget, thus raising the overall XMM-Newton mission maintenance costs

1.3 INTRODUCTION TO XMM-NEWTON

XMM-Newton (Figure 1.1) is the most sensitive X-ray satellite ever built, and with a total length of 10 meters, the largest satellite ever launched by ESA. It has been operating as an open observatory since the beginning of 2000, providing X-ray scientific data through three X-ray imaging cameras and two X-ray spectrometers, as well as visible and UV images through an optical telescope. The large amount of data collected by XMM-Newton is due to its unprecedented effective area in the X-ray domain in combination with the simultaneous operation of all its instruments. The latter is not only of quantitative, but also of eminent qualitative importance given the intrinsically variable nature of most of the phenomena associated with cosmic X-ray generation.

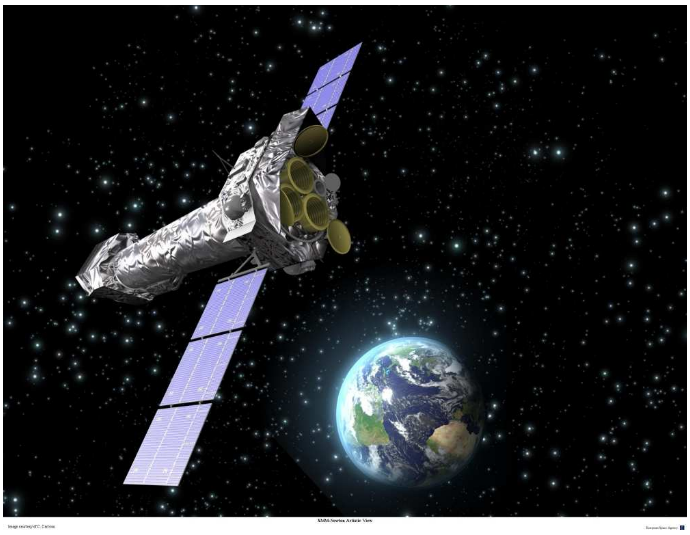


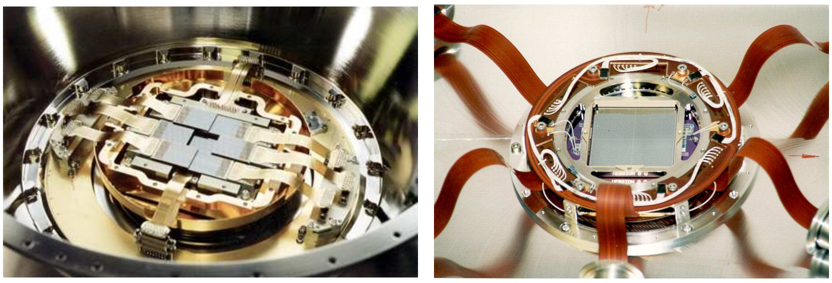
Figure 1.1: XMM-Newton - Artist’s impression

1.3.1 *X-ray telescope optics*

The XMM-Newton X-ray telescope composed of three barrel-shaped mirror modules, each with 58 ”Wolter-II” type concentric gold-coated electroformed nickel mirror shells, covering through grazing incidence reflection the spectral range of [0.1-12]keV. The large total collecting area of 4300 cm2 at 1.5 keV is complemented by moderate angular resolution of ca. 5” Full Width Half Maximum (FWHM) (14” Half Energy Width (HEW)).

1.3.2 *X-ray cameras*

XMM-Newton carries three X-ray imaging CCD cameras: the European Photon Imaging Cameras (EPIC) each of them in the focal plane of one of the X-ray telescopes. The cameras are of two different types, one of them using a new type of CCD (pn) (Figure 1.2b) especially developed for X-ray missions. Since all three work in single-photon register mode, and can register also energy and arrival time of each incoming X-ray photon, they provide simultaneously moderate spectroscopic and timing capabilities. Different operating modes can be used to optimize the observations according to the brightness of the target and the purpose of the observation.



(a) EPIC-MOS CCDs

(b) EPIC-pn CCDs

Figure 1.2: EPIC instruments

1.3.3 *X-ray high resolution spectroscopy*

Behind two of the three X-ray telescopes, diffraction gratings (Figure 1.3) intercept around 50% of the incoming light, dispersing by reflection onto the CCDs (Figure 1.4) of the Reflection Grating Spectrometer (RGS) with a spectral resolution (λ / $\Delta\lambda$) of around 200 in first order dispersion in the soft X-ray domain ([0.3-2.4] keV).

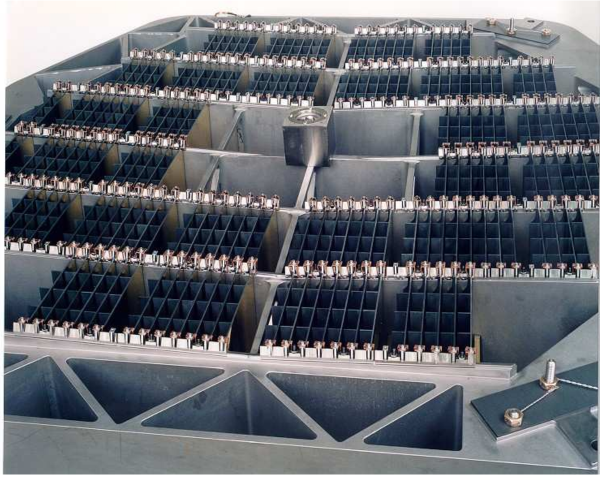
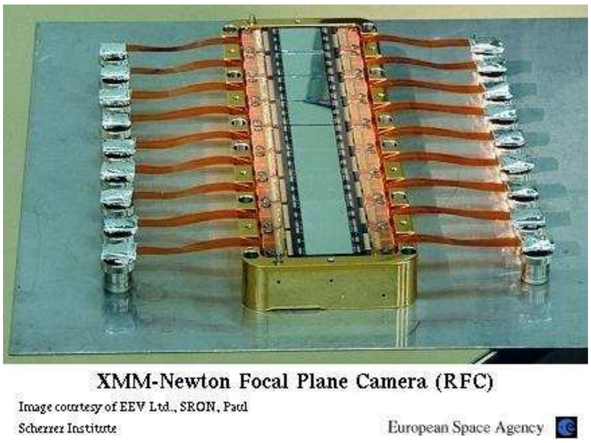


Figure 1.3: The Reflection Grating Array (RGA)



XMM-Newton Focal Plane Camera (RFC)

Image courtesy of EEV Ltd., SRON, Paul Scherrer Institute

European Space Agency

Figure 1.4: The RGS Focal Plane Camera (RFC)

1.3.4 *The Optical Monitor (OM)*

Co-aligned with the X-ray telescopes, the OM (Figure 1.5) gives XMM-Newton a multi-wavelength capability, operating in the range [1600-6600] Å. The OM camera can work also in photon counting mode, providing therefore time-resolved information in addition to visible and UV images. It offers a FOV of 17′ and a limiting sensitivity (5σ) of 20.7 mag for an integration time of 1000 seconds. Additional optical and UV grisms provide XMM-Newton with a moderate spectral resolution, also in this range.

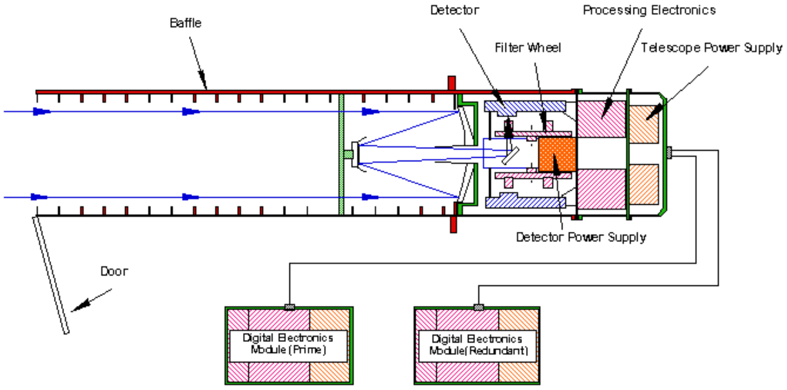
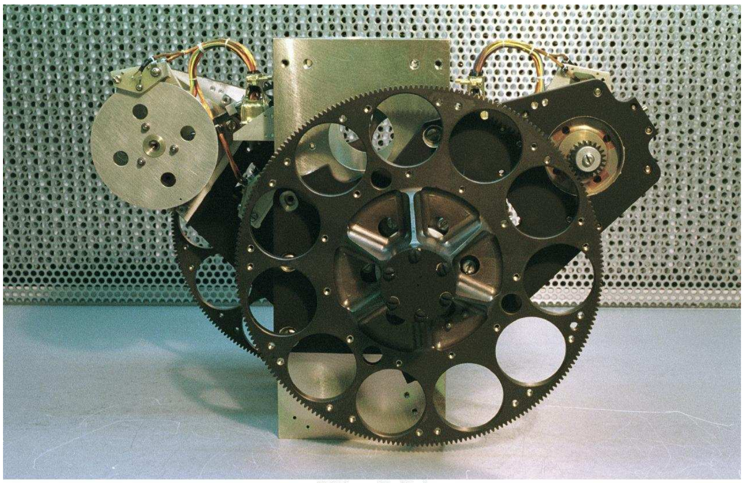


Figure 1.5: A schematic of the Optical Monitor



OM Telescope Filter Wheel

Image courtesy of Mullard Space Science Laboratory, UK.

European Space Agency

Figure 1.6: The OM Telescope Filter Wheel

Part II

XMM-NEWTON DATA ANALYSIS

2

SAS AND RISA

2.1 A BRIEF INTRODUCTION TO THE SCIENCE ANALYSIS SYSTEM (SAS)

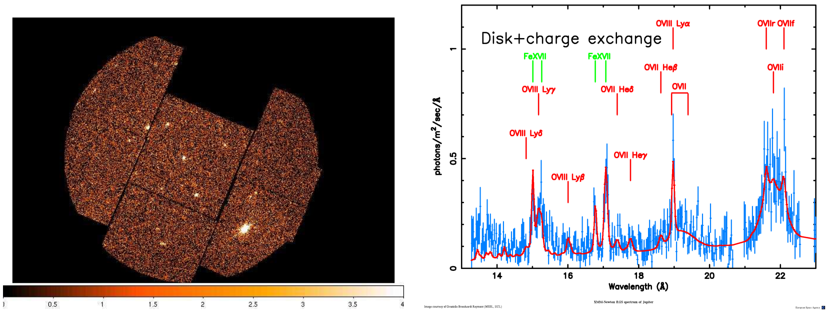
The Science Analysis System (SAS) is the XMM-Newton data reduction software able to convert raw data to scientifically useful products, like calibrated event lists, images, spectra, lightcurves, source lists, etc. It is used regularly by almost 2000 astronomers and has helped to do the scientific analysis by the vast majority of more than 2200 refereed scientific publications based on XMM-Newton data so far.

Mainly coded in C++ and Fortran 90, SAS is a complex software application dealing with data from diverse X-ray and optical / UV instruments, designed to run on a standalone computer, and heavily dependent on third party software and libraries.

To process XMM-Newton raw data using SAS and convert them to scientifically useful products, the end user has to perform the following steps:

```
- Download and install SAS and calibration Database.

- Download and install third party software (perl,
  cfitsio, ds9, grace...).

- Search and download the data from the XMM-Newton
  archive.  (http://xmm.esac.esa.int/xsa/)

- Process the data locally, create clean event files.

- Derive images, spectra, light curves and source
  lists.  (See figures below)
```

Figures 2.1a, 2.1b and 2.2 seen below contain some of the SAS products:



(a) XMM-Newton image taken from the MOS1 camera – Credits: Nuria Fonseca and ESA

(b) XMM-Newton RGS spectrum of Jupiter – Credits: G.Branduardi-Raymont and ESA

Figure 2.1: SAS products

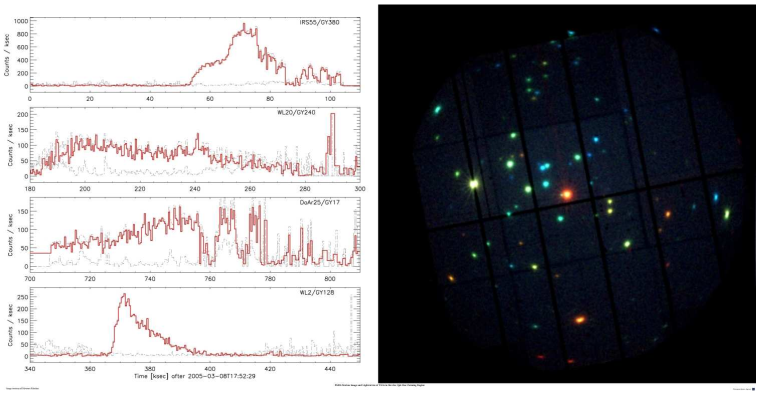


Figure 2.2: XMM-Newton EPIC-pn Lightcurves and Image of YSOs in the ρ Oph Star Forming Region – Credits: S.Sciortino and ESA

SAS binaries are distributed for several OSs and flavors, which have to be produced, forced by the high dependency from internal libraries. This requires more manpower, increased cost and time between SAS releases. A brief introduction to the new SAS approach is following.

2.2 RISA: THE NEW, VERSATILE APPROACH TO DATA REDUCTION AND ANALYSIS

The Remote Interface for Science Analysis (RISA) is an interface, under development, to all SAS functionalities through a Client / Server application running SAS workflows in a Grid Architecture. It is developed in Java and makes use of AJAX and SOAP technologies for web interfacing as well as message exchange between the Server and the Client applications. GridWay [1] is used as a Grid metascheduler[2]. You can see the RISA flow of work in Figure 2.3.

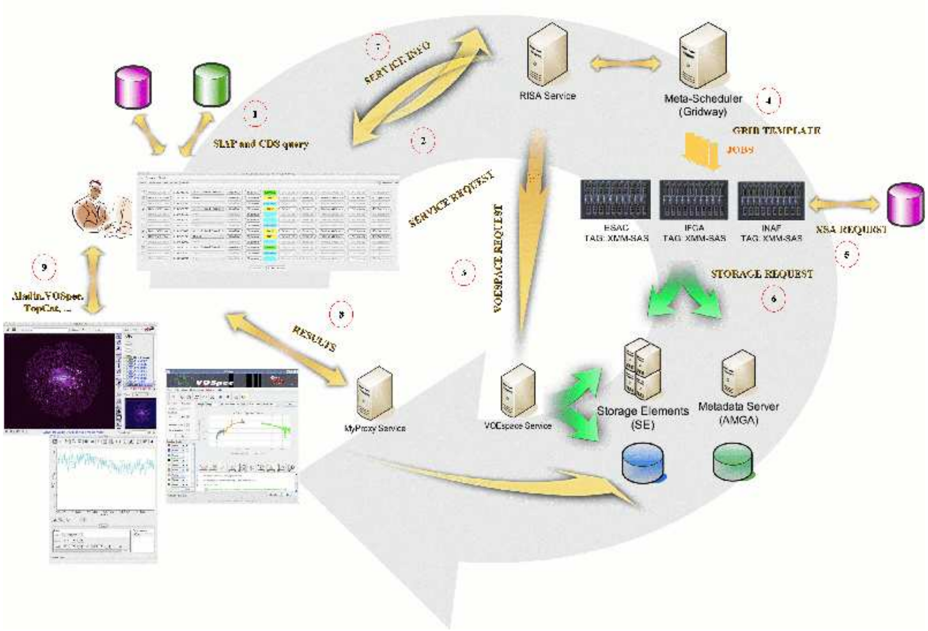


Figure 2.3: RISA flow of work. Each of the steps is explained right below

[1] GridWay (www.gridway.org) is an open source meta-scheduling technology that enables large-scale, secure, reliable and efficient sharing of computing resources (clusters, computing farms, servers, supercomputers...), managed by different Distributed Resource Management (DRM) systems, such as Sun Grid Engine, Condor, PBS, Load Sharing Facility..., within a single organization (enterprise grid) or scattered across several administrative domains (partner or supply-chain grid).

[2] Meta-scheduling or Super scheduling is a computer software technique of optimizing computational workloads by combining an organization's multiple Distributed Resource Managers into a single aggregated view, allowing batch jobs to be directed to the best location for execution.

The following points explain in detail the RISA flow of work:

1 SIAP query to retrieve information about the XMM-Newton observation (Observation Start Time, Exposure Time, Instrument Modes). CDS query to resolve astronomical names.

2 Service request. The client serializes the RISA input workflow together with the parameters corresponding to each workflow.

3 The RISA service contacts the VOEspace service to create data node(s), where the results will be stored once the observation(s) are reduced on the Grid. The VOEspace service creates the data node(s) and returns a set of GridFTP URL (endpoints) which RISA can use for uploading the results.

4 RISA sends the job(s) to the Grid through DRMAA OGF standard (Grid- Way implementation) specifying the output file(s) to be stored at the end- point(s) that VOEspace returned in the previous step.

5 Each node in the GRID makes a requests to the XMM-Newton Science Archive (XSA) to retrieve the XMM-Newton ODF (Observation Data File) data set, corresponding to a given observation ID, using the AIOClient (ESAC archive team tool to automatically download XMM-Newton data). application.

6 Once the data processing has finished, the results are taken to the Storage Element (GridFTP endpoints returned by VOEspace) automatically by GridWay. The RISA service then knows that the observations have been processed and informs the user (job DONE).

7 The RISA service sends information to the Client about the job status.

8 The user clicks a button for downloading results which connects to MyProxy service to download temporary credentials (needed to interact with the Grid Storage Elements) and performs the copy using GridFTP.

9 The results can be displayed with Aladin / VOSpec / TopCat / ... opened through PLASTIC interface.

Due to this approach, RISA is able to:

- Avoid or at least reduce the Operating System libraries dependencies and compilers evolution[3] -- Virtualization (SAS on demand)[4].
- Reduce the installation of third party software on the user's side.
- Easily increase the computing resources -- Scalability[5].
- Make use of Virtual Organization infrastructures in a Grid Computing environment.
- Leave a legacy application for a long scientific mission[6].

2.2.1 *RISA advantages*

The RISA webservice allows the remote execution of a SAS Workflow in a Grid environment. This design has the following advantages:

- No need for a local SAS installation or local maintenance of calibration files or third party software is required.
- Easy and Fast way of accessing large amounts of data (e.g. no download required).
- User's platform independent SAS execution.
- No need for large number of SAS integration platforms.
- Easy access to large computing resources thanks to the Grid technology (virtual organizations).

[3] Maintaining SAS will be much easier. For the time being there are 16 SAS versions produced, corresponding to 16 OSs, every time a new SAS release is made. With RISA, only one SAS release needs to be produced, corresponding to the server OS.

[4] With the use of system virtual machines, the end user is able to install and use the available SAS release under the user's computer OS.

[5] RISA can be run on one or more GRIDs thus increasing the processing power if necessary.

[6] RISA requires minimal maintenance and will be able to produce scientific products from the XMM-Newton data even many years after the end of the mission.

Future developments and improvements of this design will focus on the following points:

- Standardization of SAS Workflows[7].
- Access to all SAS functionalities (SAS parameters)[8].
- SAS error handling.
- General VO type access to SAS routines.
- Storage data products: MySpace, VOSpace[9].
- On the fly generator of Virtual Observatory compliant products.

[7] A SAS workflow is a chain of calls to several tasks performing a certain type of data reduction.

[8] RISA provides a categorized view of all SAS functionalities as well as a graphical interface for each SAS task's parameters, enabling the end user to easily and fully parametrise the final SAS product according to the user's needs.

[9] VOSpace is an International Virtual Observatory Alliance (IVOA) standard describing the interfaces to implement for a Virtual Observatory (VO) distributed storage service. It is just the visible part of the implementation which specifies how VO agents and applications can store and exchange data in a standard way. It can be considered as an access point, through a SOAP Web Service, to a distributed storage network.

2.2.2 *Limitations of the current RISA implementation*

Even though RISA is a pioneering concept providing the astrophysicists and astronomers with a very handy SAS implementation via the web, the initial RISA parameter GUI (Figure 2.4) was less user friendly due to its very basic design.

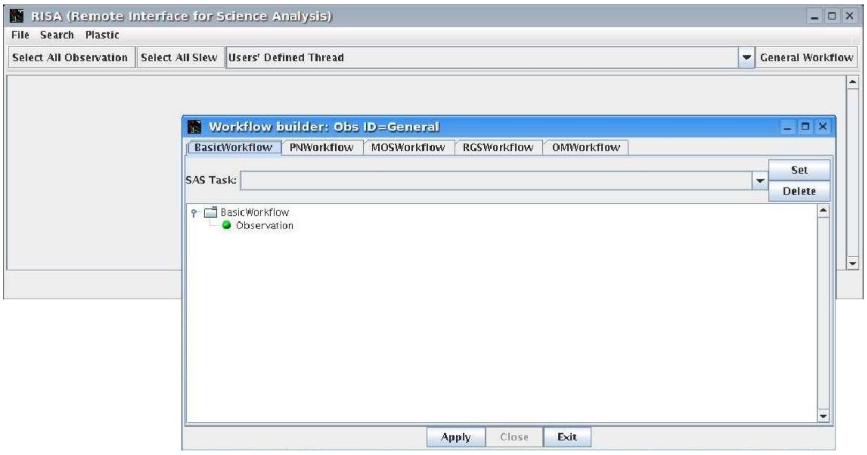


Figure 2.4: The old RISA parameter GUI

The need for a user friendly parameter GUI (Graphical User Interface), similar to the one implemented in SAS was recognized from the very beginning. It should be able to select SAS tasks and set or change the default values of the task parameters using a graphical interface similar to the one encountered in SAS which is very familiar to the X-ray scientists that deal with data analysis.

The following chapter will introduce the new RISA parameter GUI concept which constitutes this thesis, developed during the period 09/2008 - 03/2009.

Part III

THESIS PROJECT

3

THE STUDENT'S THESIS PROJECT

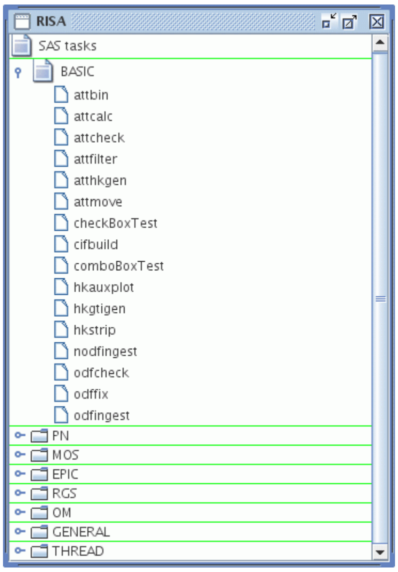
3.1 THE NEW RISA PARAMETER GUI

The aim of this thesis project was to improve the functionality and usability of RISA by implementing a new parameter Graphical User Interface. Since SAS is a well known and widely used application for X-ray astronomy, the SAS GUI served as a guidline and therefore greatly influenced the overall RISA GUI philosophy.

In general, producing a simple yet practical GUI simplifies the usage of a program, thus making it more appealing to all possible end users. When it comes to scientific applications, the existence of a clear, simple, practical GUI is essential since the vast majority of scientists counts reliability and practicallity over appearance.

3.1.1 *The new parameter GUI structure*

At first, a search for new graphical interface concepts was done. The pros and cons were discussed and the final decision on how the new RISA interface would look was taken. The new RISA parameter GUI consists of a simple tree containing nodes and leafs. Nodes are categorized based on the respective XMM-Newton satellite instruments -for example EPIC, MOS, PN, RGS- or by the scope of the task -i.e GENERAL and THREAD- and leafs contain all the tasks that are applicable for the respective instrument data as seen in Figure 3.1. Whereas the original SAS GUI (Figure 3.2), which is rarely used due to its low practicality, contains all the tasks listed uncategorized.



RISA

SAS tasks

BASIC

- attbin
- attcalc
- attcheck
- attfilter
- atthkgen
- attmove
- checkBoxTest
- cifbuild
- comboBoxTest
- hkauxplot
- hkgtigen
- hkstrip
- nodfingest
- odfcheck
- odffix
- odfingest

PN

MOS

EPIC

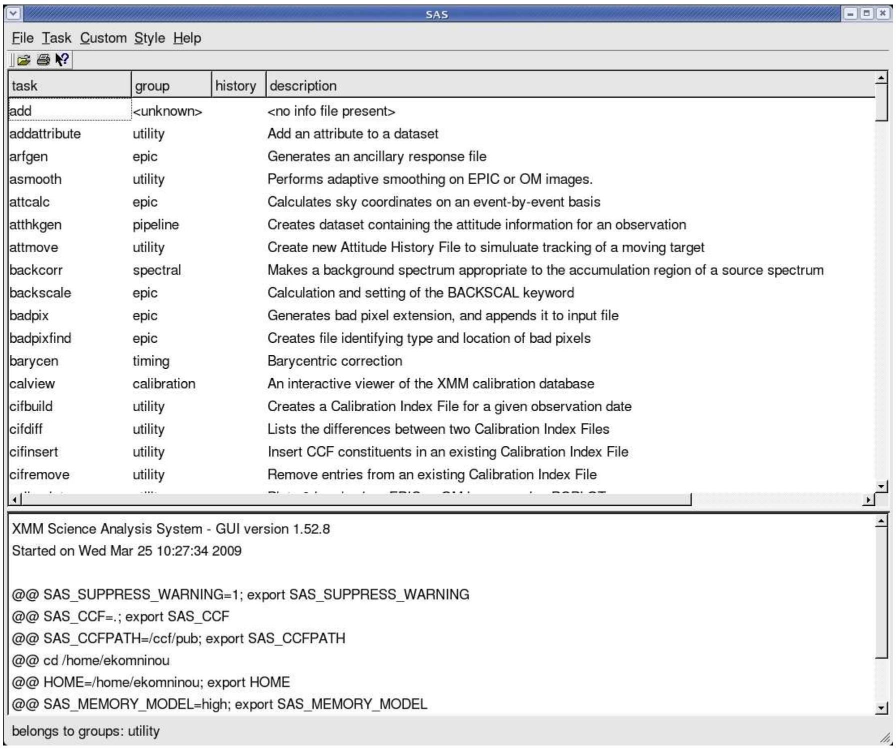
RGS

OM

GENERAL

THREAD

Figure 3.1: The **new** RISA task selector



task	group	history	description
add	<unknown>		<no info file present>
addattribute	utility		Add an attribute to a dataset
arfgen	epic		Generates an ancillary response file
asmooth	utility		Performs adaptive smoothing on EPIC or OM images.
attcalc	epic		Calculates sky coordinates on an event-by-event basis
atthkgen	pipeline		Creates dataset containing the attitude information for an observation
attmove	utility		Create new Attitude History File to simuluate tracking of a moving target
backcorr	spectral		Makes a background spectrum appropriate to the accumulation region of a source spectrum
backscale	epic		Calculation and setting of the BACKSCAL keyword
badpix	epic		Generates bad pixel extension, and appends it to input file
badpixfind	epic		Creates file identifying type and location of bad pixels
barycen	timing		Barycentric correction
calview	calibration		An interactive viewer of the XMM calibration database
cifbuild	utility		Creates a Calibration Index File for a given observation date
cifdiff	utility		Lists the differences between two Calibration Index Files
cifinsert	utility		Insert CCF constituents in an existing Calibration Index File
cifremove	utility		Remove entries from an existing Calibration Index File

```
XMM Science Analysis System - GUI version 1.52.8
Started on Wed Mar 25 10:27:34 2009

@@ SAS_SUPPRESS_WARNING=1; export SAS_SUPPRESS_WARNING
@@ SAS_CCF=.; export SAS_CCF
@@ SAS_CCFPATH=/ccf/pub; export SAS_CCFPATH
@@ cd /home/ekomninou
@@ HOME=/home/ekomninou; export HOME
@@ SAS_MEMORY_MODEL=high; export SAS_MEMORY_MODEL
```

belongs to groups: utility

Figure 3.2: The SAS task selector

Whenever the user clicks on any task of the RISA task selector, a new window pops up containing all the parameters of this task. It is easily noticeable that the new RISA parameter GUI has a very strong resemblence to the classical

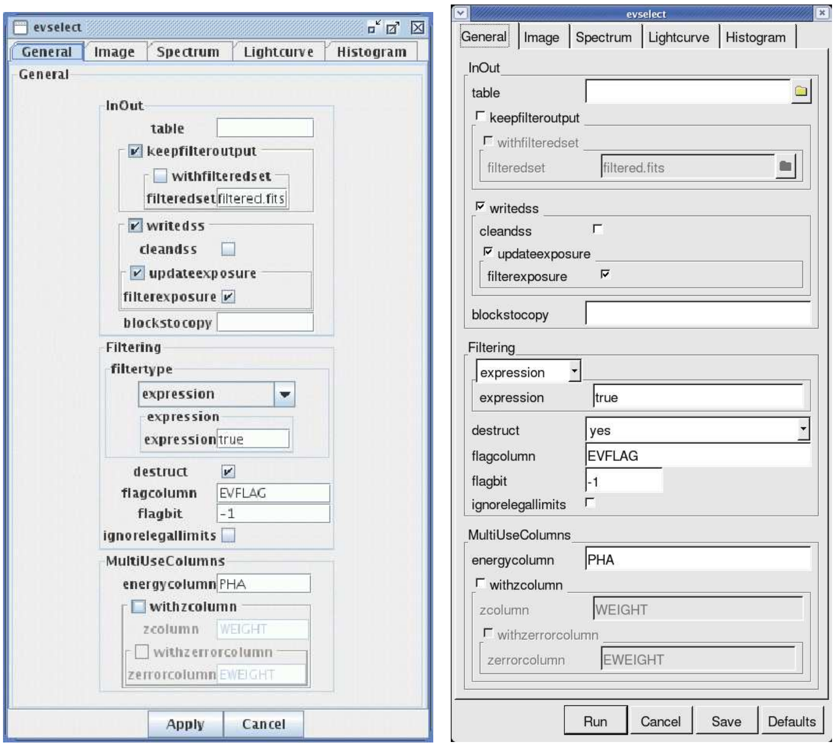
SAS GUI -which is considered a standard in the X-ray astronomical community- and all parameters are categorised depending on their function. Whereas the old RISA parameter GUI contains all the parameters listed uncategorized. In the next page, you can see a SAS task example and its RISA counterpart. The *evselect*[1] task window of the new RISA parameter GUI compared to the SAS one (Figure 3.3) and the evselect task window of the old RISA parameter GUI (Figure 3.4).

[1] *Evselect* is of central significance among the suite of individual SAS tasks. It serves two complementary purposes:

1. Filter event list data controlled by user-specified selection criteria. [Calibrated event list files consist of calibrated ODF (Observation Data File) sets, which are used as input for further detailed scientific analysis. ODFs are raw observation data produced by the respecive XMM-Newton scientific instruments. Calibration means to apply all the transformations, which allow to convert instrument quantities into physical ones].

2. Extract images, spectra, and time series from the filtered event list obtained in the first step. The filtered list is selectable for output as well.

The availability of both features within a single task allows the end user to generate useful products from an event list (e.g create a rates file in the energy band 0.5 - 3 key from the X-ray source lying within a circle of 10 arcsec radius in the center of the FOV (Field Of View) in a convenient manner.



General

InOut
- table
- ☑ keepfilteroutput
- □ withfilteredset
- filteredset filtered.fits
- ☑ writedss
- cleandss □
- ☑ updateexposure
- filterexposure ☑
- blockstocopy

Filtering
- filtertype
- expression
- expression true
- destruct ☑
- flagcolumn EVFLAG
- flagbit -1
- ignorelegallimits □

MultiUseColumns
- energycolumn PHA
- □ withzcolumn
- zcolumn WEIGHT
- □ withzerrorcolumn
- zerrorcolumn EWEIGHT

(a) The **new** RISA evselect task window

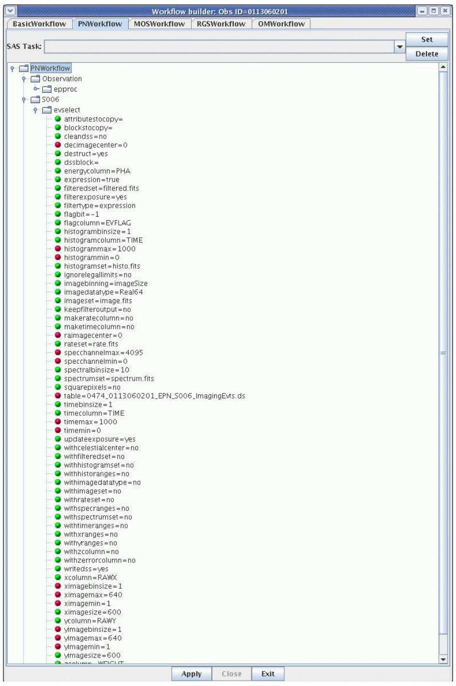
InOut
- table
- □ keepfilteroutput
- □ withfilteredset
- filteredset filtered.fits
- ☑ writedss
- cleandss □
- ☑ updateexposure
- filterexposure ☑
- blockstocopy

Filtering
- expression
- expression true
- destruct yes
- flagcolumn EVFLAG
- flagbit -1
- ignorelegallimits □

MultiUseColumns
- energycolumn PHA
- □ withzcolumn
- zcolumn WEIGHT
- □ withzerrorcolumn
- zerrorcolumn EWEIGHT

(b) The SAS evselect task window

Figure 3.3: RISA vs SAS task windows



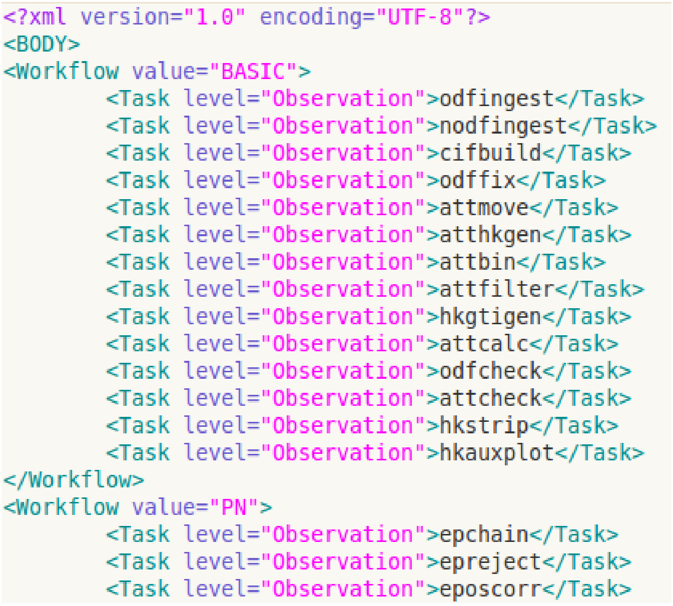
```
PNWorkflow
  Observation
    epproc
  S006
    evselect
      attributestocopy=
      blockstocopy=
      cleandss=no
      decimagecenter=0
      destruct=yes
      dssblock=
      energycolumn=PHA
      expression=true
      filteredset=filtered.fits
      filterexposure=yes
      filtertype=expression
      flagbit=-1
      flagcolumn=EVFLAG
      histogrambinsize=1
      histogramcolumn=TIME
      histogrammax=1000
      histogrammin=0
      histogramset=histo.fits
      ignorelegallimits=no
      imagebinning=imageSize
      imagedatatype=Real64
      imageset=image.fits
      keepfilteroutput=no
      makeratecolumn=no
      maketimecolumn=no
      raimagecenter=0
      rateset=rate.fits
      specchannelmax=4095
      specchannelmin=0
      spectralbinsize=10
      spectrumset=spectrum.fits
      squarepixels=no
      table=0474_0113060201_EPN_S006_ImagingEvts.ds
      timebinsize=1
      timecolumn=TIME
      timemax=1000
      timemin=0
      updateexposure=yes
      withcelestialcenter=no
      withfilteredset=no
      withhistogramset=no
      withhistoranges=no
      withimagedatatype=no
      withimageset=no
      withrateset=no
      withspecranges=no
      withspectrumset=no
      withtimeranges=no
      withxranges=no
      withyranges=no
      withzcolumn=no
      withzerrorcolumn=no
      writedss=yes
      xcolumn=RAWX
      ximagebinsize=1
      ximagemax=640
      ximagemin=1
      ximagesize=600
      ycolumn=RAWY
      yimagebinsize=1
      yimagemax=640
      yimagemin=1
      yimagesize=600
      zcolumn=WEIGHT
```

Figure 3.4: The **old** RISA evselect task window

The new RISA parameter GUI resemblance with the classical SAS one is easily noted in the screenshots above (Figure 3.3 (a) and (b)). The reason of designing a GUI identical to the original SAS one was simple: For the past 10 years this interface has served the researchers very well and it has become a well known standard among astronomers / astrophysicists dealing with X-ray astronomy like mentioned above. With simplicity and practicallity in mind, there was no reason for altering or completely changing the very satisfactory SAS GUI for something more sophisticated or modern in appearence.

3.1.2 *The RISA task tree*

The SAS task data is included in an XML file containing all the available *workflows* as well as the corresponding *tasks* of each workflow (Figure 3.5).



```
<?xml version="1.0" encoding="UTF-8"?>
<BODY>
<Workflow value="BASIC">
        <Task level="Observation">odfingest</Task>
        <Task level="Observation">nodfingest</Task>
        <Task level="Observation">cifbuild</Task>
        <Task level="Observation">odffix</Task>
        <Task level="Observation">attmove</Task>
        <Task level="Observation">atthkgen</Task>
        <Task level="Observation">attbin</Task>
        <Task level="Observation">attfilter</Task>
        <Task level="Observation">hkgtigen</Task>
        <Task level="Observation">attcalc</Task>
        <Task level="Observation">odfcheck</Task>
        <Task level="Observation">attcheck</Task>
        <Task level="Observation">hkstrip</Task>
        <Task level="Observation">hkauxplot</Task>
</Workflow>
<Workflow value="PN">
        <Task level="Observation">epchain</Task>
        <Task level="Observation">epreject</Task>
        <Task level="Observation">eposcorr</Task>
```

Figure 3.5: SAS task data in XML format (see *Appendix A* for full XML code)

Therefore, a *parser* class called TaskParser[2] had to be developed that is be able to read the XML data and pass it to the graphical interface, using tools from the JDOM[3] API containing the well known SAX[4] among others.

Also, a *Tree interface* class was developed, namely TaskTree[5]. This interface is able to read the SAS XML parsed data and pass them as individual or grouped nodes in a tree using the *JTree*[6] *Java* class. This *Tree interface* is flexible enough since it's able to dynamically change the tree

[2] see *Appendix B* for full Java code

[3] `http://www.jdom.org/` Application Programming Interface (API) – JDOM provides a complete, Java-based solution for accessing, manipulating, and outputting XML data from Java code.

[4] `http://www.saxproject.org/` – SAX is the Simple API for XML, originally a Java-only API. SAX was the first widely adopted API for XML in Java, and is a "de facto" standard.

[5] see *Appendix C* for full Java code

[6] A control that displays a set of hierarchical data as an outline

contents whenever the actual XML file needs to be altered for some reason (e.g new SAS tasks released, obsolete SAS tasks removed by the SAS developing team).

The combination of parsing the SAS tasks XML file with passing the parsed data to the *Tree* interface resulted in a new simple yet effective *task selector* containing all SAS tasks categorized based on the respective XMM-Newton instrument and in alphabetical order (instruments and tasks starting with *a* are first, instruments and tasks starting with *z* are last).

After constructing the new RISA *task selector* interface, the second thing that had to be taken care of was to construct a *parser* which would be able to read the SAS graphical interface files and *dynamically*[7] produce *modal windows*[8] similar to the SAS ones.

It should be noted that SAS offers approximately 170 tasks[9] (and rising) for all kinds of astrophysical data analysis, i.e 170+ different windows containing a combination of 6 types of widgets (also known as controls) depending on the task. Producing more than 170 different static GUIs is out of the question. It can be a very time and resource consuming procedure that's not sophisticated at all thus leaving us with the choice of writing code that will be able to produce such a variety of windows *dynamically*. This was a big challenge for a novice *Java* programmer –it can be challenging even for an experienced programmer– that

[7] Each SAS GUI is composed dynamically based on two files' contents: the respective *.lyt* file contents which define the final graphical interface output and the *.par* file contents which set the default values of all parameters of each SAS task. This is done via numerous loops that "decide" which widget should be produced and where should it be put depending on the content of the respective task. Both *.lyt* and *.par* files are heritage of SAS and are used for producing all the existing GUIs of SAS.

[8] A modal window is a child window that requires the user to interact with it before they can return to operating the parent application, thus preventing the workflow on the application main window. Modal windows are often called heavy windows or modal dialogs because the window is often used to display a dialog box.

[9] A list of all tasks can be found at `http://xmm.vilspa.esa.es/sas/6.5.0/doc/packagelist.html`

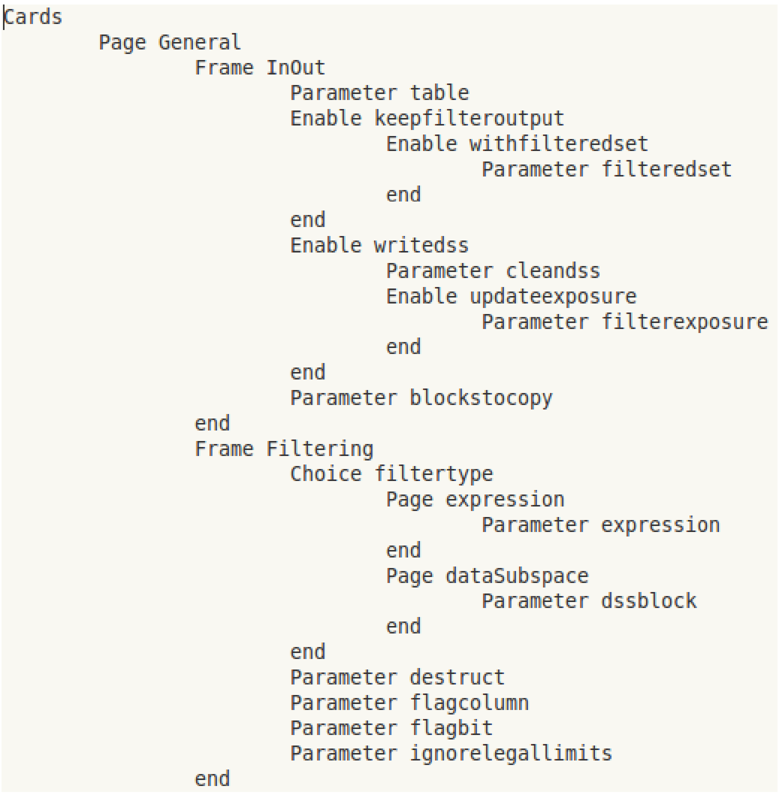
the student had to face.

3.1.3 *Producing windows dynamically*

The SAS graphical interface is originally generated by parsing and graphically interpreting two types of documents using the *Qt toolkit*[10] : *.lyt* and *.par* files. Each SAS task has a corresponding *.lyt* and *.par* file. Both file types were developed during the early '90s as part of the SAS graphical interface infrastructure. The first file type is used for setting the graphical interface of each SAS task window by indicating the right widgets and parameters that should be contained in each task and where they should be placed. The latter contains information regarding the default values of each parameter contained in the respective SAS task window. In SAS both files are read through a parser written in C++ and interpreted into windows containing a layout that's composed of a set of widgets –depending on the chosen task– where each widget includes the default parameters of this particular task.

Since RISA is been developed as a client / server application with many elements and complex procedures running in the background, the choice of *Java* as the main programming language was essential, therefore making it impossible to use the original C++ SAS parsers for generating the original SAS graphical interfaces. This means that a new approach had to be followed for reading the data included in the *.lyt* and *.par* files of each SAS task and generating a fully functional SAS task window using the RISA infrastructure. In order to be able to read the corresponding *.lyt* and *.par* files of each SAS task, new *Java* code had to be developed which would be able to:

[10] `http://qt.nokia.com/` Qt (pronounced as the English word "cute") is a cross-platform application development framework, widely used for the development of GUI programs (in which case it is known as a widget toolkit), and also used for developing non-GUI programs such as console tools and servers. Qt is most notably used in Adobe Photoshop Album, Google Earth, KDE, Opera, OPIE, Skype, Qt Extended, VLC media player and VirtualBox.

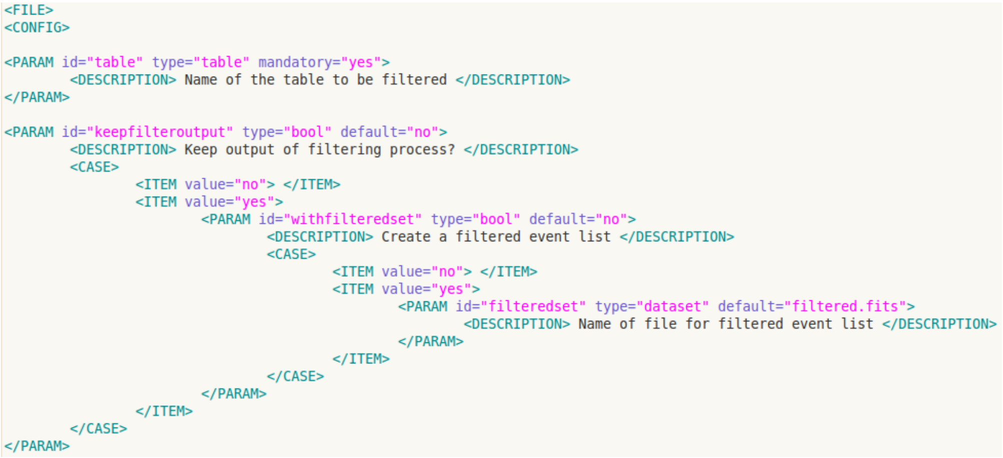


```
Cards
        Page General
                Frame InOut
                        Parameter table
                        Enable keepfilteroutput
                                Enable withfilteredset
                                        Parameter filteredset
                                end
                        end
                        Enable writedss
                                Parameter cleandss
                                Enable updateexposure
                                        Parameter filterexposure
                                end
                        end
                        Parameter blockstocopy
                end
                Frame Filtering
                        Choice filtertype
                                Page expression
                                        Parameter expression
                                end
                                Page dataSubspace
                                        Parameter dssblock
                                end
                        end
                        Parameter destruct
                        Parameter flagcolumn
                        Parameter flagbit
                        Parameter ignorelegallimits
                end
```

Figure 3.6: Sample code of a *.lyt* file (See *Appendix P* for full code)

```
1 read each .par file and temporarily store the included
    default values of each parameter.

2 read the corresponding .lyt file.

3 create a layout (=various widgets produced dynamically
    depending on the .lyt contents) which includes
    all the parameters of the corresponding SAS task.

4 create a window (=container) that will contain the
    layout

5 add the layout to the window.
```

At first, a baseline was constructed consisting of a dozen new *Java* classes approximately. Initially this new set of classes was able to produce a basic GUI, including at least a few widgets in each window.



```
<FILE>
<CONFIG>

<PARAM id="table" type="table" mandatory="yes">
        <DESCRIPTION> Name of the table to be filtered </DESCRIPTION>
</PARAM>

<PARAM id="keepfilteroutput" type="bool" default="no">
        <DESCRIPTION> Keep output of filtering process? </DESCRIPTION>
        <CASE>
                <ITEM value="no"> </ITEM>
                <ITEM value="yes">
                        <PARAM id="withfilteredset" type="bool" default="no">
                                <DESCRIPTION> Create a filtered event list </DESCRIPTION>
                                <CASE>
                                        <ITEM value="no"> </ITEM>
                                        <ITEM value="yes">
                                                <PARAM id="filteredset" type="dataset" default="filtered.fits">
                                                        <DESCRIPTION> Name of file for filtered event list </DESCRIPTION>
                                                </PARAM>
                                        </ITEM>
                                </CASE>
                        </PARAM>
                </ITEM>
        </CASE>
</PARAM>
```

Figure 3.7: Sample code of a *.par* file (See *Appendix O* for full code) – note the XML resemblance

The *.par* files are being processed using two classes, namely Param[11] and Tasks[12]. Param parses each *.par*'s file contents whereas Tasks sorts the parsed content of each *.par* file. The *.lyt* files are being parsed using the LYTReader[13] class .

Lots of new additions, improvements and corrections had to be done on the initial set of classes, in order for the RISA tasks' GUI to be considered fully functional. Many widgets were malfunctioning, others overlapped each other, others weren't visible at all and others were situated off frame. Therefore it was crucial to ensure that the *.lyt* and *.par* files would be parsed correctly resulting in correctly formed windows, fully functional drop down menus, checkboxes and so on, the window constraints would be properly set in order to add each widget to the correct position, the methods and elements in the source code would be correctly set.

After going through the *Java* code, it was obvious that LYTReader was only able to read the vary basic *.lyt* files (developed only for testing the project functionality) correctly since they don't contain complex expressions like normal SAS *.lyt* files do. Therefore reforming the LYTReader was the first priority

[11] see *Appendix D* for full Java code
[12] see *Appendix E* for full Java code
[13] see *Appendix F* for full Java code

3.1.4 *Improving the source code*

After rewriting some parts of the LYTReader code and setting the layout constraints correctly the overlapping widgets problem was solved, but the missing widgets issue as well as blank windows and so on still had to be taken care of since these issues were mainly caused by the way the *.lyt* files were parsed. Therefore the method of reading and parsing the *.lyt* files had to be revised and improved so that blank spaces and blank lines will be omitted, comments will be ignored and words between quotes will be considered as one string.

The main problem was that while reading each *.lyt* line by line, the line contents were extracted and buffered by using blank spaces as string split indicators even though the content of each line contained more complex strings like *"some text and whitespaces"* or comment (#) characters, tab spaces, whitespaces and so on. This forced us to search for a different way of splitting the components of each line before extraction. The regular expressions feature of Java was used which is powerful and flexible enough to ignore comments (#), tab spaces, whitespaces and read entire phrases as one string therefore conserving the original format of each task's GUI included in the respective *.lyt* file.

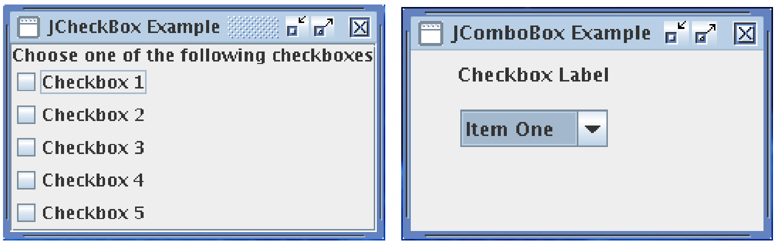
Like mentioned above, each *.lyt* (and the corresponding *.par*) file is being parsed using a fairly complex class (LYTReader) containing some of the powerful features of *Java* such as Lists, Iterators, BufferedReaders, FileReaders, Regular expressions, ActionListeners and many others.

The working philosophy behind this particular class is the following:

Firstly the *.lyt* file is read line by line using a FileReader. Once a line is buffered using a BufferedReader, the corresponding *.par* data is also buffered. If the line contains one or more keywords specifically set by the programmer, the corresponding method is triggered therefore producing the right widget frame.
The same line is read once more by the aforementioned reg-

ular expressions method, scanning for regular expressions that the programmer has set like *"Text and whitespaces"* for example. Once the regular expression has been located, it is corrected by subtracting the quotes or any other non necessary symbol(s) and is then passed along with the already processed *.lyt* and *.par* data to the corresponding widget where the full widget is finally added to the overall layout container.
If a line starts or contains a # (comment symbol), the regular expression method makes sure to omit that line and proceeds to the next one.

Like mentioned above, there are 6 different types of widgets available for the RISA parameter GUI depending on the functions that a SAS task can provide to the end user. Namely: BuildPanel[14], BuildParameter[15], BuildTab[16], BuildTabPage[17], BuildComboBox[18], BuildCheckBox[19] are the classes that produce panels, text fields and labels, tabs, tab pages, drop down menus and check boxes respectively. All 6 classes are heavily based on the Swing[20] API elements.



(a) An example of a Swing checkbox widget

(b) An example of a Swing drop down menu widget

Figure 3.8: Swing widgets

[14] see *Appendix G* for full java code
[15] see *Appendix H* for full java code
[16] see *Appendix I* for full java code
[17] see *Appendix J* for full java code
[18] see *Appendix K* for full java code
[19] see *Appendix L* for full java code
[20] Swing is a widget toolkit for Java. It is part of Sun Microsystems' Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs. `http://java.sun.com/javase/6/docs/technotes/guides/swing/`

There are quite a few implementations of Swing available. *Jigloo*[21] was eventually chosen, a user friendly Swing implementation containing a GUI that allows the programmer to construct any kind of static graphical interface by dragging and dropping widgets on a panel. When dragging and dropping widgets on the main panel, *Jigloo* generates the GUI *Java* code automatically therefore letting the programmer focus on coding the functionality behind the GUI rather than the GUI itself. Unfortunately this feature was not useful for the RISA parameter GUI project since the only standard feature used in all 170+ RISA implementations of the SAS tasks is a container where all widgets lay on, the rest of the features get to be produced dynamically like mentioned above. This added to the overall difficulty of the project but was very didactic in terms of the *Java* programming potentials and coding techniques.

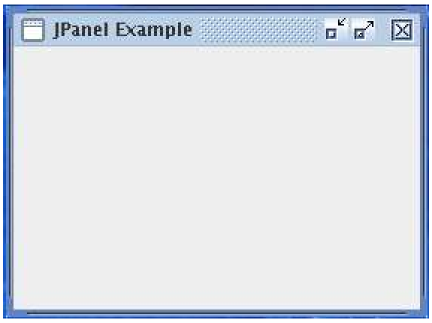


Figure 3.9: JPanel, a widget used as a "frame" surrounding other Swing controls

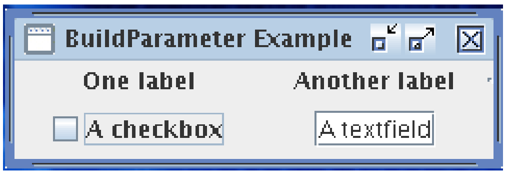


Figure 3.10: A BuildParameter widget - it consists of JLabels, JTextFields and JCheckBoxes

Three of the aforementioned widget classes, BuildParameter, BuildPanel and BuildTabPage are included in the LYTReader's methods. Every time a *.lyt* line is parsed, one of these three classes is called depending on the type of

[21] www.cloudgarden.com/jigloo/

data the line contains. Information is passed to the class and the corresponding widget is formed. Each widget is placed in the layout, with respect to the constraints set, and when the last line of the *.lyt* file is read and the layout is complete it is then added to the main frame thus forming the task window.

Furthermore BuildPanel includes another three widget classes –BuildParameter, BuildCheckBox and BuildComboBox– in its methods. Everytime BuildPanel is called from a LYTReader method, it constructs the appropriate panel based on the data buffered from reading each *.lyt* line and passes it to the overall layout list containing the sum of widgets produced for assembling the GUI corresponding to the chosen task.

Also BuildPanel, BuildCheckBox, BuildComboBox and BuildTab implement an interface[22] called LayoutPane[23].

A minor detail that had to be taken into consideration – which makes a noticeable difference though– was the scrollbar. Each window is able to be resized depending on the end user's needs, therefore a scrollbar has to be present once the window becomes smaller than the default value set by the programmer. This allows the window contents to be fully accessible no matter of the size of the window itself.

At first adding a scrollbar to the tasks' GUI wasn't feasible because of the Swing's API design constraints. In order for a scrollbar to be added to a window, the programmer has to set a valid viewport that will be accessible by the scrollbar. But the Swing API applies some constraints on each control. One of these constraints actually doesn't allow a container to be hosted inside another container.
Initially both the *window* and the *layout* were registered as

[22] An interface in computer science is a set of named operations that can be invoked by clients. In the Java programming language, an interface is an abstract type that is used to specify an interface (in the generic sense of the term) that classes must implement. In programming languages, an abstract type is a type in a nominative type system which is declared by the programmer –where a nominative type system actually is a major class of type system, in which compatibility and equivalence of data types is determined by explicit declarations and/or the name of the types.

[23] See *Appendix M* for full java code

containers, thus making it impossible to incorporate the scrollbar to the *layout* and the *layout* to the *window*. For that reason the code had to be slightly altered in order to meet Swing's API criteria regarding scrollbars. The overall procedure was not overly complicated, still finding out the exact Swing constraints and making the proper code adjustments was a challenge.

Before the end of the traineeship period in March 2009 the RISA parameter GUI was fully functional, it displayed all the elements of each SAS task and only a few minor flaws had to be sorted out like correcting the spacing between the window frame and each layout etc in order for the GUI to be considered complete. Adjusting the layout constraints slightly –the GridBagConstraints[24] class is mainly used throughout the entire project– solved this issue.

The last step that had to be taken was to come up with a way to pass all the information included in each window (labels, textfields and their content, checkboxes and their boolean status, tabs and their contents etc) in an XML file which will contain all the untouched parameter values including the new values entered by the end user.

Passing information from the task window to an XML file was a slightly complicated procedure. A new class was developed called PassWindowReader[25] which is able to extract all useful data off of a task window and store them in an XML file with the help of another very helpful characteristic found in *Java*, the *Java logging API*[26]. For

[24] The GridBagConstraints class specifies constraints for components that are laid out using the GridBagLayout class. `http://java.sun.com/j2se/1.5.0/docs/api/java/awt/GridBagConstraints.html`

[25] see *Appendix N* for full Java code

[26] The logging API is part of J2SE as of JDK 1.5, and it ships with the JDK. It is designed to let a Java program, servlet, applet, EJB, etc. produce messages of interest to end users, system administrators, field engineers, and software developers. Especially in production situations, where things can't be run in a debugger, or if doing so masks the problem that is occurring (because it is timing related, for example), such logs are frequently the greatest (and sometimes the only) source of information about a running program.

that reason a Logger[27] and a Handler[28] were utilized. The Logger was correctly set both in terms of syntax and logging level (amount of information to be logged) and the Handler was instructed to export all logged messages to a specific *.xml* file called *RISAlog.xml*. Finally, a listener was introduced to each window's `Apply` button, which would pass all the information included in the current window to any tool that could handle and extract this XML data set (e.g the Logger).

What still needs to be finished is the output XML syntax of the Logger, containing the parameters of the task the end-user is using in a format compatible to the SAS XML parser. This can be done by tweaking the current XML logger used. The optimized XML logging system will be added to the new RISA configuration shortly.

[27] A Logger object is used to log messages for a specific system or application component. Loggers are normally named, using a hierarchical dot-separated namespace. Logger names can be arbitrary strings, but they should normally be based on the package name or class name of the logged component, such as java.net or javax.swing. `http://java.sun.com/j2se/1.5.0/docs/api/java/util/logging/Logger.html`

[28] A Handler object takes log messages from a Logger and exports them. It might for example, write them to a console or write them to a file, or send them to a network logging service, or forward them to an OS log and so on. `http://java.sun.com/j2se/1.5.0/docs/api/java/util/logging/Handler.html`

3.1.5 *Architecture visualization*

A good way to visualize a system's architectural blueprints in software engineering is to utilize the UML (Unified Modeling Language)[29]. UML data models provide an overview of the system architecture and element interconnections (e.g class interconnections), which offers a very helpful insight of the overall system. For that reason a project UML diagram was constructed (Figure 3.11) which depicts the project class' interconnections. On this diagram you can see how classes depend on each other by calling one another while the code is executed. Also you can see which classes depend on the LayoutPane interface. That way you can get a good idea of the way all classes are linked with each other which gives a good overview of the project's source code architecture.

[29] The Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. It is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as programming language statements and so on

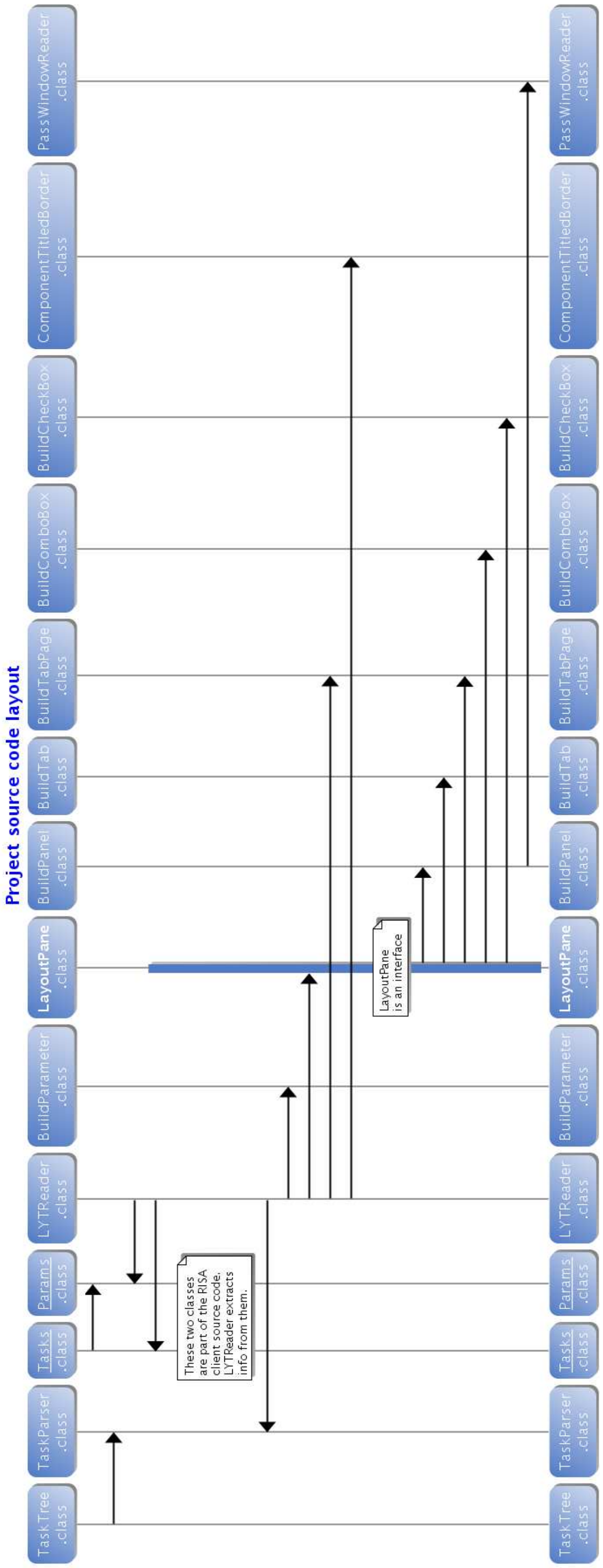


Figure 3.11: The UML data model of the project

3.2 RESULTS AND FUTURE WORK

By the end of the placement period, a new Graphical User Interface was produced for manipulating all the SAS tasks that are available via the RISA service. The new RISA parameter GUI was built with simplicity, practicallity and reliability in mind, making it easier for the end user to control all the parameters and tasks provided in RISA. Since RISA will substitute SAS soon, developing a practical and easy to use parameter GUI was a crucial step for helping RISA evolve into the next standard for the X-ray astronomy community which utilizes XMM-Newton data for scientific research and analysis.

3.2.1 *Scientific importance*

Apart from developing a new parameter GUI for the RISA application, which improved its usability, the current thesis work gave RISA another strong feature: flexibility. The new GUI was developed in such a way that enables RISA to dynamically produce any kind of SAS task GUI, as long as a corresponding *.lyt* and *.par* file is provided for generating the new task's GUI. So once a new task is developed and implemented in SAS, the corresponding RISA application will be able to automatically generate the new task GUI on the fly, without the need of any alterations or patches to the actual RISA source code, just by gaining read-only access to the new task's corresponding *.lyt* and *.par* file.

Given the flexibility and the significant improvement of the actual parameter GUI which makes RISA very user friendly, RISA is constantly gaining ground among the astronomical / astrophysical X-ray research community. This combined with the advanced technologies RISA utilizes (client-server interface, GRID technology, multiplatform and architecture-independent application etc) makes RISA a very appealing solution for X-ray astronomers / astrophysicists. It is also a very appealing concept for other applications to follow, proving that web-technologies in combination with the power of object-oriented programming languages like *Java* and advanced technologies like GRID computing can bring exquisite results such as very powerful applications working under low tech hardware, low bandwidth Internet connections but delivering high quality results nevertheless.

Therefore, while this thesis project is far from being considered a world class breakthrough in software technology, it is a valuable component greatly improving the usability of a largely growing and very promising astrophysical application.

3.2.2 *Hardware technology used*

The entire thesis code was developed under Linux using low end hardware. In particular the main tool of work was an affordable Asus EeePC 900 netbook with the following specs: Intel Celeron 900MHz CPU, 1Gb RAM, 8Gb Solid State Disk, Intel GMA 950 GPU operating under *Xandros Linux Light*[30] OS.

Despite the low end hardware technology included in this particular type of personal computers, the processing power was more than enough for the job. For that reason this thesis can also be considered as a technological demonstration of the ability to produce very satisfactory results using high end software technology combined with low end hardware technology which is nowadays considered satisfactory mainly for surfing the internet (ASUS calls the EeePC a "mobile Internet device").

3.2.3 *Software technology used*

The entire thesis source code was developed using open-source software tools. As stated above, the OS used was a Debian based distribution of Linux. The IDE (Integrated Development Environment) used was *Eclipse*[31] *Ganymede 3.4*. The *JVM* (Java Virtual Machine) runtime used was the one included in the *JDK* (Java Developer's Kit) *1.5.0*.

[30] Debian-based OS developed by the Xandros company `www.xandros.com`

[31] `www.eclipse.org`

Furthermore, apart from the technical side, open-source software was also used for the documentation of the thesis work. LaTeX[32] was used for typesetting the text, *Texmaker*[33] was used for writing the thesis text, *Ubuntu*[34]*9.04* was the OS under which the documentation part was realized.

Therefore this thesis also underlines the importance of open-source Operating Systems and software tools. A good example of this is the fact that *Eclipse* and *Netbeans*[35] are dominant in the developers' world, major technology / research centres like ESAC use Linux for all the desktops and servers on site, LaTeXis widely used for documenting scientific or technological research results in universities and research centres and so on.

Sophisticated open-source software allows users and developers to make use of all the available resources such as CPU power and RAM quantity in a very efficient way. An example which proves the dominance of Unix-based Operating Systems is the fact that any *Windows Vista* 32bit version has a 3Gb RAM max limit (64bit advanced versions like Vista Business can support up to 128Gb of RAM), *Windows 7* 32bit version has a 4Gb RAM max limit (64bit versions can support up to 192Gb of RAM)[36]. Linux distributions face a similar restriction unless the end-user installs the Physical Address Extension (PAE) aware kernel[37], which is a very simple and straightforward procedure if you use apt-get or similar package managers. That way a 32bit OS can actually address and handle its RAM like 64Bit OSs do hence overcoming all RAM limitations.

[32] LaTeXis a document markup language and document preparation system for the TeX typesetting program. It is most widely used by mathematicians, scientists, engineers, philosophers, economists and other scholars in academia and the commercial world, and other professionals.

[33] Texmaker is a cross-platform open source LaTeXeditor that integrates many tools needed to develop documents with LaTeX, in just one application. `http://www.xm1math.net/texmaker/`

[34] `www.canonical.com`

[35] `www.netbeans.org`

[36] More information at `http://news.softpedia.com/news/Windows-7-Maximum-Supported-RAM-192-GB-RAM-119101.shtml`

[37] More information at `http://www.cyberciti.biz/faq/ubuntu-linux-4gb-ram-limitation-solution/`

3.2.4 *Remaining work*

By the end of the traineeship period, the development of the source code of the project was almost complete with only a few minor details to be taken care of. Once the coding was finished, there were a few ideas about further improving the final product. They were originally scheduled to be implemented within the next few months after the end of the traineeship period before or while working on the documentation of the thesis project.

Due to an unfortunate incident in mid-2009, the thesis work saved in the author's computers and backup hardware was lost since all computer hardware was stolen. This led to a major delay in the completion of the project.

This major delay led to documenting the thesis work based on very few resources like a few screenshots of the final product, non-functional pieces of *Java* code and so on while following a long and time-consuming procedure of gathering every lost piece of information again. Furthermore, given the fact that a very satisfactory postgraduate opportunity came up, starting on March 2010, the time remaining in order to improve or expand any features of the thesis product was very little making it very difficult to implement all the planned improvements and new features.

Some of the improvements that are required in order for the final product to be considered complete:

1 Complete the XML logging system so that it will pass all the necessary information to an *.xml* log file following the XML syntax required by the SAS parser in order to process the document correctly.

2 Deal with the GUI window contents refresh bug. Some widgets are visible only after refreshing the window contents more than once. – Solution: The window contents should be refreshed at least once right after the frame being generated.

3 Deal with the window frame decoration bug. The first time any task window pops up, the window frame decoration follows the Operating System's theme

colours and style. Normally the frame decoration is standard and independent of the OS theme colours and style. – Solution: Unknown for the time being.

4 Deal with the frame default dimensions bug. Some times frames appear smaller or much bigger instead of being adjusted on the frame contents. – Solution: Force the frame to be resized depending on the window contents.

5 Deal with the RISA tree mouse listener improvement. For the time being whenever the cursor is situated on a task leaf, any mouse button will trigger the action of the task window to pop up. – Solution: Set the mouse listener trigger to work only when left-clicking on the leaf once (or twice depending on the end-users' preference).

New features that should be implemented to the final product:

1 Add a (possibly numbering) sequence leading the end user through the correct steps before proceeding to requesting certain SAS products[38].

2 Add explanatory notes right next to each task leaf on the task tree. That way a new or inexperienced user will know what each task is capable of.

3 Add explanatory pop up tabs of each widget's elements, a very helpful feature for new SAS users who haven't read the SAS manual extensively yet and a good reminder for the experienced SAS users.

3.2.5 *Lessons learned*

The overall experience of developing this thesis which included working at the European Space Agency on a soft-

[38] Each XMM-Newton data set needs to be processed using some particular preparatory SAS tasks before it can be considered ready for scientific analysis. Therefore there should be some kind of guide leading the inexperienced user through the first few preparatory steps before proceeding to any kind of scientific analysis. This can be done by following a numbered pattern or possibly different techniques which indicate the correct task sequence that has to be followed before proceeding to any other tasks which conduct analysis and data extraction.

ware based project, trying to develop part of an astrophysical application, was more than just didactic. It involved several new concepts and ways of work such as learning an object oriented programming language in a very tight timeframe, working independently, working on a multinational professional (i.e non academic) environment, improvement of proactive skills, communication skills and working on a multidisciplinary field.

From the programming point of view, new knowledge on object oriented programming and particularly in *Java* was acquired.
A new way of thinking was also acquired, leading to acquiring a broader technological spectrum and a deeper understanding of software engineering mentality.
From the working methodology point of view, a more independent way of working and thinking was developed by extracting and validating information and knowledge from various sources whenever needed as well as experimenting on and testing new techniques with little or no guidance from the tutors.

Overall the knowledge obtained and the experience acquired throughout the process of developing this thesis contributed to a deeper understanding of different fields such as software engineering and astronomy, as well as acquiring a more liberal way of thinking.

BIBLIOGRAPHY

[1] Aitor Ibarra, Ignacio de la Calle, Carlos Gabriel, Jesús Salgado, Pedro Osuna, *XMM-Newton Science Analysis Software: How to bring new technologies to long-life satellite missions*. ADASS XVII 2008.

[2] Aitor Ibarra, Ignacio de la Calle, Daniel Tapiador, Nora Loiseau, Carlos Gabriel, Jesús Salgado, Pedro Osuna, *Remote Interface to Science Analysis Tools for Grid Architecture: The XMM-Newton SAS Case*. ADASS XVI 2006.

[3] Carlos Gabriel, Matteo Guainazzi, Leo Metcalfe, *The ESA XMM-Newton Science Operations Centre: Making Basic Space Science Available to the Whole Scientific World*.

[4] *XMM-Newton Image Gallery:*
http://xmm.esac.esa.int/external/xmm_science/gallery/public/index.php

[5] *XMM-Newton: A Technical Description:*
http://xmm.esac.esa.int/external/xmm_user_support/documentation/technical/

[6] *XMM-Newton Science Analysis System*
http://xmm.vilspa.esa.es/sas/6.5.0/doc/index.html

[7] Matteo Guainazzi
An introduction to XMM-Newton data analysis and the SAS grand-scheme http://xmm.esa.int/external/xmm_data_analysis/sas_workshops/sas_ws9_files/presentations/Matteo_SAS_introduction.pdf

[8] *Enabling a robust VOSpace based on iRODS for astronomers*
https://www.irods.org/index.php/VOSpace

[9] *JDK^{TM}5.0 Documentation*
http://java.sun.com/j2se/1.5.0/docs/

[10] *Java examples (example source code)*
http://www.java2s.com/

[11] *Wikipedia - The Free Encyclopedia*
`http://en.wikipedia.org/wiki/Java_(programming_language)`

[12] *Wikipedia - The Free Encyclopedia*
`http://en.wikipedia.org/wiki/GridWay`

[13] *Wikipedia - The Free Encyclopedia*
`http://en.wikipedia.org/wiki/Modal_window`

[14] *Wikipedia - The Free Encyclopedia*
`http://en.wikipedia.org/wiki/Interface_(computer_science)`

[15] *Wikipedia - The Free Encyclopedia*
`http://en.wikipedia.org/wiki/Unified_Modeling_Language`

[16] *Wikipedia - The Free Encyclopedia*
`http://en.wikipedia.org/wiki/LaTeX`

Part IV

APPENDICES

A

SAS TASKS XML FILE

```xml
<?xml version="1.0" encoding="UTF–8"?>
<BODY>
<Workflow value="BASIC">
  <Task level="Observation">odfingest</Task>
  <Task level="Observation">nodfingest</Task>
  <Task level="Observation">cifbuild</Task>
  <Task level="Observation">odffix</Task>
  <Task level="Observation">attmove</Task>
  <Task level="Observation">atthkgen</Task>
  <Task level="Observation">attbin</Task>
  <Task level="Observation">attfilter</Task>
  <Task level="Observation">hkgtigen</Task>
  <Task level="Observation">attcalc</Task>
  <Task level="Observation">odfcheck</Task>
  <Task level="Observation">attcheck</Task>
  <Task level="Observation">hkstrip</Task>
  <Task level="Observation">hkauxplot</Task>
</Workflow>
<Workflow value="PN">
  <Task level="Observation">epchain</Task>
  <Task level="Observation">epreject</Task>
  <Task level="Observation">eposcorr</Task>
  <Task level="Observation">epauxcomb</Task>
  <Task level="Observation">epframes</Task>
  <Task level="Observation">epevents</Task>
  <Task level="Observation">epexposure</Task>
  <Task level="Observation">epproc</Task>
  <Task level="Observation">epiclccorr</Task>
</Workflow>
<Workflow value="MOS">
  <Task level="Observation">emchain</Task>
  <Task level="Observation">emframes</Task>
  <Task level="Observation">emeventsproj</Task>
  <Task level="Observation">emevents</Task>
  <Task level="Observation">emenergy</Task>
  <Task level="Observation">emdiag</Task>
  <Task level="Observation">embadpixfind</Task>
  <Task level="Observation">emproc</Task>
</Workflow>
```

```
<Workflow value="EPIC">
  <Task level="Instrument">emldetect</Task>
  <Task level="Instrument">epatplot</Task>
  <Task level="Instrument">ebadpixupdate</Task>
  <Task level="Instrument">badpixfind</Task>
  <Task level="Instrument">gtimerge</Task>
  <Task level="Instrument">eboxdetect</Task>
  <Task level="Instrument">arfgen</Task>
  <Task level="Instrument">eexpchipmap</Task>
  <Task level="Instrument">esplinemap</Task>
  <Task level="Instrument">rmfgen</Task>
  <Task level="Instrument">backscale</Task>
  <Task level="Instrument">edetect_chain</Task>
  <Task level="Instrument">especget</Task>
  <Task level="Instrument">eregionanalyse</Task>
  <Task level="Instrument">etimeget</Task>
  <Task level="Instrument">echeckregion</Task>
  <Task level="Instrument">eexpmap</Task>
  <Task level="Instrument">lccorr</Task>
  <Task level="Instrument">specadd</Task>
  <Task level="Instrument">movecalc</Task>
  <Task level="Instrument">lcplot</Task>
  <Task level="Instrument">gtialign</Task>
  <Task level="Instrument">ewavelet</Task>
  <Task level="Instrument">evalcorr</Task>
  <Task level="Instrument">especplot</Task>
  <Task level="Instrument">esensmap</Task>
  <Task level="Instrument">emask</Task>
  <Task level="Instrument">elcplot</Task>
  <Task level="Instrument">efftplot</Task>
  <Task level="Instrument">inthist</Task>
  <Task level="Instrument">implot</Task>
  <Task level="Instrument">evarimgen</Task>
  <Task level="Instrument">esrcselect</Task>
  <Task level="Instrument">esrcmerge</Task>
  <Task level="Instrument">esrcfetch</Task>
  <Task level="Instrument">esources</Task>
  <Task level="Instrument">esky2det</Task>
  <Task level="Instrument">emosaic</Task>
  <Task level="Instrument">ekstest</Task>
  <Task level="Instrument">dpssflag</Task>
  <Task level="Instrument">evproject</Task>
  <Task level="Instrument">elcbuild</Task>
  <Task level="Instrument">region</Task>
  <Task level="Instrument">colimplot</Task>
  <Task level="Instrument">evigweight</Task>
  <Task level="Instrument">esensitivity</Task>
  <Task level="Instrument">econvolverprep</Task>
  <Task level="Instrument">sensitivity</Task>
  <Task level="Instrument">flspec</Task>
  <Task level="Instrument">flmask</Task>
  <Task level="Instrument">srcdisplay</Task>
  <Task level="Instrument">backcorr</Task>
</Workflow>
<Workflow value="RGS">
  <Task level="Observation">rgsoffsetcalc</Task>
  <Task level="Observation">rgsbkgmodel</Task>
  <Task level="Observation">rgsbkgmodelTest</Task>
  <Task level="Observation">rgsevconvert</Task>
  <Task level="Instrument">rgsspectrum</Task>
  <Task level="Instrument">rgssources</Task>
  <Task level="Instrument">rgsrmfgen</Task>
  <Task level="Instrument">rgsmcrgen</Task>
  <Task level="Instrument">rgsregions</Task>
  <Task level="Instrument">rgslinepos</Task>
  <Task level="Observation">rgsframes</Task>
  <Task level="Instrument">rgsfluxer</Task>
  <Task level="Observation">rgsfilter</Task>
  <Task level="Observation">rgsevents</Task>
  <Task level="Observation">rgsenergy</Task>
  <Task level="Observation">rgscombine</Task>
  <Task level="Observation">rgsbadpix</Task>
  <Task level="Instrument">rgsspecplot</Task>
  <Task level="Instrument">rgsimplot</Task>
  <Task level="Observation">rgsauxtable</Task>
  <Task level="Observation">rgsangles</Task>
  <Task level="Observation">rgssuperrmf</Task>
  <Task level="Observation">rgsproc</Task>
  <Task level="Instrument">rgslccorr</Task>
  <Task level="Observation">rgsprods</Task>
  <Task level="Observation">htrframes</Task>
</Workflow>
<Workflow value="OM">
  <Task level="Observation">omthlcplot</Task>
  <Task level="Observation">omgrismplot</Task>
  <Task level="Observation">omflatindex</Task>
  <Task level="Observation">omcomb</Task>
  <Task level="Observation">omthconv</Task>
  <Task level="Observation">omsrclistcomb</Task>
  <Task level="Observation">omregion</Task>
  <Task level="Observation">omprep</Task>
  <Task level="Observation">ommosaic</Task>
  <Task level="Observation">ommodmap</Task>
  <Task level="Observation">ommag</Task>
  <Task level="Observation">omlcbuild</Task>
  <Task level="Observation">omgrism</Task>
  <Task level="Observation">omgprep</Task>
  <Task level="Observation">omflatgen</Task>
  <Task level="Observation">omflatfield</Task>
  <Task level="Observation">omfastshift</Task>
  <Task level="Observation">omfastflat</Task>
  <Task level="Observation">omdrifthist</Task>
  <Task level="Observation">omdetect</Task>
  <Task level="Observation">omcosflag</Task>
  <Task level="Observation">omatt</Task>
  <Task level="Observation">omphkgen</Task>
  <Task level="Observation">omslewchain</Task>
  <Task level="Observation">omsource</Task>
  <Task level="Observation">omichain</Task>
  <Task level="Observation">omgsource</Task>
  <Task level="Observation">omgchain</Task>
  <Task level="Observation">omfchain</Task>
  <Task level="Observation">rudiframetime</Task>
  <Task level="Observation">movecalc</Task>
  <Task level="Observation">lcplot</Task>
  <Task level="Observation">gtialign</Task>
  <Task level="Observation">implot</Task>
  <Task level="Observation">colimplot</Task>
  <Task level="Observation">srcdisplay</Task>
</Workflow>
<Workflow value="GENERAL">
  <Task level="Instrument">merge</Task>
  <Task level="Instrument">evselect</Task>
  <Task level="Instrument">dstoplot</Task>
  <Task level="Instrument">dsplot</Task>
  <Task level="Instrument">badpix</Task>
  <Task level="Instrument">phasecalc</Task>
  <Task level="Instrument">tabgtigen</Task>
  <Task level="Instrument">orbit</Task>
  <Task level="Instrument">colsmooth</Task>
  <Task level="Instrument">asmooth</Task>
  <Task level="Instrument">calview</Task>
  <Task level="Instrument">timeappend</Task>
  <Task level="Instrument">evlistcomb</Task>
  <Task level="Instrument">statsget</Task>
</Workflow>
<Workflow value="Thread">
  <Task level="Observation">epic_event_thread</Task>
  <Task level="Instrument">epic_lightcurve_thread</Task>
  <Task level="Instrument">epic_spectrum_thread</Task>
  <Task level="Instrument">epic_edetectchain_thread</Task>
  <Task level="Instrument">epic_analysis_thread</Task>
  <Task level="Observation">rgs_thread</Task>
    <Task level="Observation">epic_slew_thread</Task>
</Workflow>
</BODY>
```

B

TASKPARSER.JAVA

```java
package layoutCreator;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.input.DOMBuilder;
import org.jdom.input.SAXBuilder;
import com.sun.org.apache.xerces.internal.parsers.
    DOMParser;

public class TaskParser {

  SAXBuilder builder;
  Document doc;
  Element xmlRootElement;
  static Element child;
  static Element ch;
  static Element root;
  static Element childe;


  private HashMap<String,List<String>> _myMap = new
      HashMap<String,List<String>>(); //Create a
      HashMap containing a mapping of 2 types of
      elements: a string and a list of strings

  public void readXMLFile(String file)
  {
     System.out.println("Reading ");
     file = "SASData_v2.xml"; //Set the file that's
         about to be parsed
           DOMBuilder builder = new DOMBuilder();
           DOMParser parser = new DOMParser();

           // Read the entire document into memory
```

```java
try {
  parser.parse(file);
  org.w3c.dom.Document domDoc  = parser.getDocument();
        org.jdom.Document jdomDoc = builder.build(domDoc);
        Element root = jdomDoc.getRootElement();
        final List<?> allChildren = root.getChildren(); //Create a list of all the parsed words

        Iterator<?> itr = allChildren.iterator(); //Add an iterator to allChildren

      while (itr.hasNext())
         {
        List<String> taskList =new ArrayList<String>();
        Element child = (Element) itr.next(); //Return the next element
        child.getAttributeValue("value"); //Get each workflow name

        final List<?> childe = child.getChildren(); //Add each workflow name to childe
        Iterator<?> it = ((List<?>) childe).iterator(); //Add an iterator to childe

        while (it.hasNext())
        {
          Element ch = (Element) it.next(); //Return the next element
          ch.getText(); //Get each task name
          ch.getAttributeValue("level"); //Get the task level (Observation or Instrument)
          taskList.add(ch.getText()); //Add the task name to taskList
          Collections.sort(taskList); //Sort taskList by alphabetical order
          System.out.println("Tasks "+ch.getText() );
        }

        _myMap.put(child.getAttributeValue("value"),taskList); //Add each workflow and its corresponding tasks in the HashMap
         }
}
catch (Exception e) {
    }
  }

  public HashMap<String,List<String>> getInfo()
  {
    System.out.println("My Map: "+_myMap);
    return _myMap; //Return the contents of _myMap
  }
}
```

C

TASKTREE.JAVA

```java
import java.awt.Color;
import java.awt.Dimension;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.HashMap;
import java.util.List;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JRootPane;
import javax.swing.JScrollPane;
import javax.swing.JTree;
import javax.swing.UIManager;
import javax.swing.event.TreeSelectionListener;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.MutableTreeNode;
import javax.swing.tree.TreeSelectionModel;
import layoutCreator.LYTReader;
import layoutCreator.TaskParser;

public class TaskTree {

  DefaultMutableTreeNode parent = null;
  static JTree tree = null;

  public TaskTree()
  {
    //Read the parsed info
    TaskParser myParse = new TaskParser();
    myParse.readXMLFile("SASData_v2.xml");

      //Create a tree, tree nodes and a frame
      DefaultMutableTreeNode basic = new
         DefaultMutableTreeNode("BASIC");
      DefaultMutableTreeNode nBasic;

      DefaultMutableTreeNode pn = new
         DefaultMutableTreeNode("PN");
        DefaultMutableTreeNode nPn;

        DefaultMutableTreeNode mos = new
           DefaultMutableTreeNode("MOS");
        DefaultMutableTreeNode nMos;

        DefaultMutableTreeNode epic = new
           DefaultMutableTreeNode("EPIC");
        DefaultMutableTreeNode nEpic;

        DefaultMutableTreeNode rgs = new
           DefaultMutableTreeNode("RGS");
        DefaultMutableTreeNode nRgs;

        DefaultMutableTreeNode om = new
           DefaultMutableTreeNode("OM");
        DefaultMutableTreeNode nOm;

        DefaultMutableTreeNode gen = new
           DefaultMutableTreeNode("GENERAL");
        DefaultMutableTreeNode nGen;

        DefaultMutableTreeNode thread = new
           DefaultMutableTreeNode("THREAD");
        DefaultMutableTreeNode nThread;

    //Create a HashMap containing a mapping of 2
       types of elements: a string and a list of
       strings
    final HashMap<String,List<String>> info =
       myParse.getInfo();

    //Extract each workflow separately
     for (int k = 0 ; k < info.get("BASIC").size()
        ; k++)
      {
       nBasic = new DefaultMutableTreeNode(info.get
          ("BASIC").get(k));
       basic.add(nBasic);
      }

     for (int k = 0 ; k < info.get("PN").size() ; k
        ++)
      {
        nPn = new DefaultMutableTreeNode(info.get("
           PN").get(k));
        pn.add(nPn);
      }

     for (int k = 0 ; k < info.get("MOS").size() ;
        k++)
      {
           nMos = new DefaultMutableTreeNode(info.get("MOS").get(k));
           mos.add(nMos);
        }

       for (int k = 0 ; k < info.get("EPIC").size() ; k++)
        {
           nEpic = new DefaultMutableTreeNode(info.get("EPIC").get(k));
           epic.add(nEpic);
        }

       for (int k = 0 ; k < info.get("RGS").size() ; k++)
        {
           nRgs = new DefaultMutableTreeNode(info.get("RGS").get(k));
           rgs.add(nRgs);
        }

       for (int k = 0 ; k < info.get("OM").size() ; k++)
        {
            nOm = new DefaultMutableTreeNode(info.get("OM").get(k));
          om.add(nOm);
        }

       for (int k = 0 ; k < info.get("GENERAL").size() ; k++)
        {
            nGen = new DefaultMutableTreeNode(info.get("GENERAL").get(k));
          gen.add(nGen);
        }

       for (int k = 0 ; k < info.get("Thread").size() ; k++)
        {
            nThread = new DefaultMutableTreeNode(info.get("Thread").get(k));
          thread.add(nThread);
        }


    //Build the tree and add the extracted data in it
    parent = new DefaultMutableTreeNode("SAS tasks", true);
      parent.add(basic);
      parent.add(pn);
      parent.add(mos);
      parent.add(epic);
      parent.add(rgs);
      parent.add(om);
      parent.add(gen);
      parent.add(thread);

    //Improve the looks of the window containing the tree
    tree = new JTree(parent);
    UIManager.put("Tree.line", Color.GREEN);
    UIManager.put("Tree.openIcon",new ImageIcon("config/postM.png"));
    tree.putClientProperty("JTree.lineStyle", "Horizontal");
    tree.updateUI();

    //Create a frame
    final JFrame frame = new JFrame("RISA");

    frame.add(tree);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLocationRelativeTo(null);
    Dimension minimumSize = new Dimension();
        minimumSize.setSize(180,210);
    frame.setUndecorated(true);
        frame.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);
        frame.setMinimumSize(minimumSize);
        frame.pack();
        frame.setVisible(true);
        frame.setLocale(new java.util.Locale("en", "US"));


      //Create a scrollbar whenever needed
      JScrollPane scrollPane = new JScrollPane();
      scrollPane.getViewport().add(tree);
      frame.add(scrollPane);

      //Mouse listener
      tree.addMouseListener(new MouseAdapter() {
          public void mouseClicked(MouseEvent e) {
            doMouseClicked(e);
          }
        });
  }

    //Create a main and instruct it to "read" the ListenerTest class
    public static void main(String[] args) {

      new TaskTree();

      }

    //Mouse listener function. It which will give
       output only when a tree leaf is clicked with
       any mouse button
    /*Possible improvement: Change this to "left
       mouse button only".*/
      static void doMouseClicked(MouseEvent e) {

      DefaultMutableTreeNode node = (
         DefaultMutableTreeNode)
      tree.getLastSelectedPathComponent();

          if (node == null) return;

          Object nodeInfo = node.getUserObject();

          if (node.isLeaf()) {

        //Read and show the windows (based on the .
           lyt and .par files of SAS) corresponding
           to each SAS task
          LYTReader reader = new LYTReader();

          boolean flag = reader.open(nodeInfo.
             toString());
          if (flag)
          {
             reader.parse();
          }

          reader.show();

            }
      }
  }
```

D

PARAMS.JAVA

```java
package sasTask;

import java.util.HashMap;
import java.util.TreeMap;
import java.util.logging.Level;
import java.util.logging.Logger;

import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.apache.xerces.parsers.*;


public class Params { //This class parses all the info included in each .par file

  private TreeMap<String,String> params = null;
  private HashMap<String,String> mandatoryParams = new HashMap<String,String>();
  private HashMap<String,String> _paramType = new HashMap<String,String>();

  public Params()
  {
    params = new TreeMap<String,String>(); // Constructs a new, empty map of strings, sorted according to the keys' natural order
  }

  public TreeMap<String,String> readSASParam(String SASTask)
  {
    String file = ("config/"+SASTask+".par"); // Load the .par file
    //Initiate parsing
    DOMParser parser = new DOMParser();  //Xerces-specific parser class

    try {
      // Read the entire .par file into memory
```

```
parser.parse(file.toString());

org.w3c.dom.Document domDoc  = parser.
   getDocument();

params = new TreeMap<String,String>(); //
   Constructs a new, empty map of strings,
   sorted according to the keys' natural
   order containing the parameters of each.
   par

NodeList nodes = domDoc.getElementsByTagName(
   "PARAM"); //Returns a NodeList of all the
   Elements in document order with a given
   tag name (PARAM) and are contained in the
   document

for (int i = 0; i < nodes.getLength(); i++) {
       Node element = (Node) nodes.item(i);

       String attribute ="";
       if (element.getAttributes().
          getNamedItem("default") == null)
         attribute = "";
       else
         attribute = element.getAttributes().
            getNamedItem("default").
            getNodeValue();

     params.put(element.getAttributes().
        getNamedItem("id").getNodeValue(),
         attribute);

     if (element.getAttributes().getNamedItem
        ("mandatory") != null)
       mandatoryParams.put(element.
          getAttributes().getNamedItem("id").
          getNodeValue(),"true"); //Add the
          values "element+attributes+id+
          idvalue" and "true" to the
          mandatoryParams HashMap
     else
       mandatoryParams.put(element.
          getAttributes().getNamedItem("id").
          getNodeValue(),"false"); //Add the
          values "element+attributes+id+
          idvalue" and "false" to the
          mandatoryParams HashMap

     _paramType.put(element.getAttributes().
        getNamedItem("id").getNodeValue(),
            element.getAttributes().getNamedItem(
            "type").getNodeValue());

         Logger.getLogger("RISAWS").info("id=" +
            element.getAttributes().getNamedItem(
            "id").getNodeValue()
             + " default=" + attribute);
    }



    } catch (Exception e) { //It indicates a
       wrong format or validity error
      Logger.getLogger("RISAWS").log(Level.SEVERE
         ,file + " is not valid. ",e.getMessage()
         ); //Log the error encountered in the .
         par file and print it in the RISAWS
    }

  return params;
}

public void setParam(String param, String value)
{
  Logger.getLogger("RISAWS").info("param "+param
     + " value "+value); //Log each parameter and
      its value
  params.put(param, value); //Store each
     parameter and its default value to the
     TreeMap
}

public TreeMap<String,String> getParam()
{
  return params; //Return the parameters saved in
      the TreeMap
}

public String getType(String param)
{
  return _paramType.get(param); //Return the type
      of parameters saved in the HashMap
}

public boolean isMandatory(String param)
{

  String mandatory = mandatoryParams.get(param);
     //Extract the mandatoryParams parameters
  if (mandatory == "true")
  {
```

```java
      return true;
    }
    else
      return false;
  }

}
```

E

TASKS.JAVA

```
package sasTask;


import java.util.Iterator;
import java.util.TreeMap;


public class Tasks {

  private Params _params = null;

  private TreeMap<String, Params> _tasksMap = null;

  private TreeMap<String,String> _levels = null;

  public Tasks()
  {
    _params = new Params(); //Calls the Params
        method of Params.java
    _tasksMap = new TreeMap<String, Params>(); //
        Constructs a new TreeMap containing a string
         and the Params class output
    _levels = new TreeMap<String,String>(); //
        Constructs a new, empty map, sorted
        according to the keys' natural order
  }

  public Params getTasks(String task)
  {
    return _tasksMap.get(task); //Returns the task
        that's saved in the _tasksMap.
  }

  public TreeMap<String,Params> getTasks()
  {
    return _tasksMap; //Returns the _tasksMap
        containing all the tasks
  }

```

```
public void readParams(String taskName)
{
  _params.readSASParam(taskName);
}


public void printInfo()
{
  Iterator<?> myWorkflowIterator = _tasksMap.keySet().iterator(); //Iterate the keys of the _tasksMap TreeMap
  while(myWorkflowIterator.hasNext()) { //For all the elements included in the _tasksMap
    String entry = (String) myWorkflowIterator.next();
    System.out.println("TASKS: "+entry); //Print each element of the _tasksMap
  }
}

public void setTask(String task)
{
  _tasksMap.put(task,_params); //Associate the tasks included in the _tasksMap with the parameters included in the _params
  _params = new Params();
}

public void setLevel(String task,String level)
{
  _levels.put(task, level); //Associate the tasks included in the _tasksMap with a level
}

public void append(Tasks task)
{
  Iterator<?> myWorkflowIterator = task.getTasks().keySet().iterator(); //Iterate the tasks
  while(myWorkflowIterator.hasNext()) { //For all the tasks
    String entry = (String) myWorkflowIterator.next();
    _tasksMap.put(entry, task.getTasks().get(entry)); //Associate each value with an iterator key
  }

  Iterator<String> levelsIt = task._levels.keySet().iterator(); //Iterate the levels
  while(levelsIt.hasNext()) //For all the keys contained in the _levels
```

```
    {
      String entryLevel = (String) levelsIt.next();
      _levels.put(entryLevel,task._levels.get(
         entryLevel)); //Associate each value with
         an iterator key
    }
  }

  public void filter(String level) {

    Iterator<String> myLevelIterator = _levels.
       keySet().iterator();
    while(myLevelIterator.hasNext()) {
      String taskLevel = (String) myLevelIterator.
         next();

      if (!_levels.get(taskLevel).equals(level)) //
         If the iterated string in _levels is not
         equal to a level
        _tasksMap.remove(taskLevel); //Remove this
           particular string
    }

  }
}
```

F

LYTREADER.JAVA

```java
package layoutCreator;

import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.GridBagConstraints;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.TreeMap;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.awt.event.ActionEvent;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import sasTask.Params;
import sasTask.Tasks;

public class LYTReader{

  private static final long serialVersionUID = 1L;

  private JFrame _frame = null;
  private BufferedReader _in;
  private boolean _ok = true;
  private static int i;
  private boolean _pageFlag = false;
  private JButton jButton = null;
  private JButton jButton1 = null;
  public String s = null;
```

```
private List<LayoutPane> _frameList = new
    ArrayList<LayoutPane>();
private List<BuildPanel> _choiceList = new
    ArrayList<BuildPanel>();
private List<BuildPanel> _enableList = new
    ArrayList<BuildPanel>();

private List<BuildParameter> _ParamList = new
    ArrayList<BuildParameter>();
List<String> _result = new ArrayList<String>();
private String inputLine = null;

LayoutPane _layout = null;

private Tasks _task = null;
private Params _params = null;

private String windowName;

public LYTReader()
{
  super();
  initialize();
  i=0;

}

private void initialize() {

}

public  boolean open(String fileName)
{
  //Read the .lyt and .par files
  try {
    _in = new BufferedReader(new FileReader("
        config/"+fileName+".lyt")); //Read the .
        lyt file named "filename"
    _task = new Tasks();
    windowName = fileName; //Where "filename" is
        actually the name of the task the end-user
         chooses

    _task.readParams(fileName);
    _task.setTask(fileName);
    _task.setLevel(fileName, "observation"); //
        Task level "observation" (see Appendix A)
    TreeMap<String,String> paramsHM = null;
    _params = _task.getTasks(fileName);
    paramsHM = _params.getParam(); //Read the .
        par file default parameter values
```

```
      System.out.println("paramsHM "+paramsHM);

      Iterator<String> myTaskIterator = paramsHM.
         keySet().iterator(); //Iterate all default
          parameter values
      while(myTaskIterator.hasNext()) {
        System.out.println("PARAMS "+myTaskIterator
           .next());
      }

      return true;
    } catch (IOException e) {
      System.out.println("ERROR Reading Layout file
         ");
      return false;
    }

  }
  public void parseLine()
  {
    //parse each line of the .lyt file
    boolean status = true;
    try {
      while (status){
        inputLine = _in.readLine();
        if (inputLine.trim().length() != 0) //Omit
           the blank lines
        {
          if (!readAgain(inputLine)) //If there is
             no comment in the line
          {
            breakup(inputLine); //Split the line
               considering the regular expression
               as one word
            status = false;
          }
          else
            status = true;
        }
        else
          status = true;
      }
    } catch (IOException e) {
      e.printStackTrace();
    }
  }


  public void breakup(String inputLine) {
    //Create a regular expression and search for it
       through each line of the .lyt file
    String val = null;
    _result.clear(); //Clear the List containing regular expressions
    String[] lineString = inputLine.trim().split(" "); //Split words by whitespaces
    _result.add(lineString[0]); //Add the first word of the parsed line to the List _result
    System.out.println("Result contents: "+_result);

    String regex = "(\".*\")"; //This is a regular expression recognizing "whatever(characters, whitespaces, numbers)" as one word
    Pattern regexp = Pattern.compile(regex);
    Matcher matcher = regexp.matcher(inputLine);
    matcher.matches();

    if (matcher.find()) {
      for (int j = 0; j < matcher.groupCount(); j++) {
        System.out.print("[" + matcher.group(j) + "]");
        System.out.println();
        val = (matcher.group(j));
        val = (val.substring(1, val.length()-1)); //We trim the " " from the regular expression found
        _result.add(val); //Add the trimmed regular expression to the List _result
      }
    }
    else
    {
      if (lineString.length > 1) //If there is a second word in the line we're parsing
        _result.add(lineString[1]); //then add the second word to the _result

    }
    matcher.reset(); //reset the matcher in order to look for the next regular expression
    System.out.println("Let's see what we've got : "+_result);

  }


  private boolean readAgain(String inputLine) {
    // Read the line again in order to find comments (#) and ignore them
    boolean status = false;
    if (inputLine.trim().length() != 0) //If the
       line is not blank
    {
      //Check for comments
      String patternComment = "(#.*)";
      Pattern pattern = Pattern.compile(
          patternComment);
        Matcher matcher = pattern.matcher(inputLine
            );
        status = matcher.matches();    //Comment
            line found
        if (status) {
          System.out.println("Comment line found -
              ignore : " + matcher.group());
          return true;
        }
    }

    return status;
  }

  public boolean parse()

  {
    boolean status = true;
    parseLine(); //Read the contents of each line
       of the .lyt file, find the regular
       expressions (and extract them), find the
       comments (and ignore them)

    if (_result.contains("Parameter"))
      status = parseParameter(); //If the line
         contains the keyword Parameter then go to
         the parseParameter method
    else if (_result.contains("Cards"))
      status = parsePagedLayout(); //If the line
         contains the keyword Cards then go to the
         parsePagedLayout method
    else if (_result.contains("Frame"))
      status = parseFrameLayout(); //If the line
         contains the keyword Frame then go to the
         parsePagedLayout method
    else if (_result.contains("Enable"))
      status = parseEnableLayout(); //If the line
         contains the keyword Enable then go to the
          parseEnableLayout method
    else if (_result.contains("Choice"))
      status = parseChoiceLayout(); //If the line
         contains the keyword Choice then go to the
          parseChoiceLayout method
    else if (_result.contains("Row"))
```

```
      status = parseRowLayout(); //If the line contains the keyword Row then go to the parseRowLayout method
    else if (_result.contains("Column"))
      status = parseColumnLayout(); //If the line contains the keyword Column then go to the parseColumnLayout method
    else if (_result.contains("end"))
      status = false; //If the line contains the keyword end then break the loop
    else {
      System.out.println("Layout keyword not recognised:" + _result);
      _ok = false;
      return false;
    }
    return status;
  }


  private boolean parseColumnLayout() {
    if(_layout == null) //If the _layout List is null
    {
      _layout = new BuildPanel("FIRST PANEL"); // Create a new panel
    }
    System.out.println("Creating Column Panel");

    BuildPanel panel = new BuildPanel("Column");
      panel.setName("Column");
      String tooltip = "TOOLTIP"+i;
      panel.setToolTipText(tooltip);
      _frameList.add(panel); //Add the panel to the _framelist List


      while(parse()) //While searching for keywords (Parameter, Cards etc) in each line of the .lyt
      {
        if (_ParamList.size() != 0) //If the _ParamList is not empty
          //add a component to the panel, with respect to the GridBagConstraints set right below
        {
          GridBagConstraints gridBagConstraints = new GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = GridBagConstraints.RELATIVE;
        gridBagConstraints.fill = GridBagConstraints.BOTH;
          panel.add(getBuildParameter(), gridBagConstraints);
          System.out.println("NUMBER OF COMPONENTS "+ panel.getComponentCount());
        }

          System.out.println("ITEM ADDED!!!");
      }

      if(_frameList.size()==1) //If the _frameList List contains one item
        //add another item with respect to the GridBagConstraints set right below
      {
      GridBagConstraints gridBagConstraints = new GridBagConstraints();
      gridBagConstraints.gridx = 0;
      gridBagConstraints.gridy = GridBagConstraints.RELATIVE;
      gridBagConstraints.fill = GridBagConstraints.BOTH;
        System.out.println("Adding the frame to the current panel!!! "+i);

        _layout.add((BuildPanel)getFrame(), gridBagConstraints); //Add the frame to the _layout with respect to the gridBagConstraints set right above

        /*
         * At this point, the size of _frameList must be zero. That means that we have finished with
         * one nest.
         */
      }
      else
      {
        System.out.println("Number of frames inside _frameList "+_frameList.size());
        BuildPanel last = (BuildPanel) getFrame();
        BuildPanel prev = (BuildPanel) getFrame();
      GridBagConstraints gridBagConstraints = new GridBagConstraints();
      gridBagConstraints.gridx = GridBagConstraints.RELATIVE;
      gridBagConstraints.gridy = 0;
```

```java
      gridBagConstraints.fill = GridBagConstraints.BOTH;
        prev.add(last,gridBagConstraints); //Add the frame named "last" to the frame named "prev" with respect to the gridBagConstraints set right above
        _frameList.add(prev);

      }

      System.out.println("Column Done!!!!");

      i++;
      return _ok;

    }

  private boolean parseRowLayout() {
    if(_layout == null) //If the _layout List is null
    {
      _layout = new BuildPanel("FIRST PANEL"); // Create a new panel
    }
    System.out.println("Creating Row Panel");

    BuildPanel panel = new BuildPanel("Row");
      panel.setName("Row");
      String tooltip = "TOOLTIP"+i;
      panel.setToolTipText(tooltip);
      _frameList.add(panel); //Add the panel to the _framelist List



      while(parse()) //While searching for keywords (Parameter, Cards etc) in each line of the .lyt
      {
        if (_ParamList.size() != 0) //If the _ParamList is not empty
          //add a components to the panel, with respect to the GridBagConstraints set below
        {
          GridBagConstraints gridBagConstraints = new GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = GridBagConstraints.RELATIVE;
        gridBagConstraints.fill =
           GridBagConstraints.BOTH;
          panel.add(getBuildParameter(),
             gridBagConstraints);
          System.out.println("NUMBER OF COMPONENTS
             "+ panel.getComponentCount());
        }

          System.out.println("ITEM ADDED!!!");
      }

      if(_frameList.size()==1) //If the _frameList
         List contains one item
        //add another item with respect to the
           GridBagConstraints set below
      {
      GridBagConstraints gridBagConstraints = new
         GridBagConstraints();
      gridBagConstraints.gridx = 0;
      gridBagConstraints.gridy = GridBagConstraints
         .RELATIVE;
      gridBagConstraints.fill = GridBagConstraints.
         BOTH;
        System.out.println("Adding the frame to the
            current panel!!! "+i);

        _layout.add((BuildPanel)getFrame(),
           gridBagConstraints);

        /*
         * At this point, the size of _frameList
            must be zero. That means that we have
            finished with
         * one nest.
         */
      }
      else
      {
        System.out.println("Number of frames inside
             _frameList "+_frameList.size());
        LayoutPane last =  getFrame();
        LayoutPane prev =  getFrame();
      GridBagConstraints gridBagConstraints = new
         GridBagConstraints();
      gridBagConstraints.gridx = 0;
      gridBagConstraints.gridy = GridBagConstraints
         .RELATIVE;
      gridBagConstraints.fill = GridBagConstraints.
         BOTH;
        prev.add((BuildPanel) last,
           gridBagConstraints); //Add the frame
```

```
        named "last" to the frame named "prev" with respect to the gridBagConstraints set right above
        _frameList.add(prev);
      }

      System.out.println("Frame Done!!!!");

      i++;
      return _ok;
  }

  private boolean parseChoiceLayout() {
    if(_layout == null) //If the _layout List is null
    {
      _layout = new BuildPanel("FIRST PANEL"); // Create a new panel
    }

    System.out.println("New Choice Panel create!!");


    BuildPanel panel = new BuildPanel(_result.get(1)); //Name the panel after the word situated next to each "Choice" keyword in the .lyt line
    _choiceList.add(panel); //Add the panel to the _choicelist List
    _frameList.add(panel); //Add the panel to the _framelist List

    GridBagConstraints gridBagConstraints = new GridBagConstraints();
    gridBagConstraints.fill = GridBagConstraints.VERTICAL;
    gridBagConstraints.gridy = 0;
    gridBagConstraints.weightx = 1.0;
    gridBagConstraints.gridx = 0;

    //A JComboBox represents a drop down menu
    panel.add(panel.getJComboBox(), gridBagConstraints); //add each JComboBox created, to the current panel
    final BuildComboBox combo = panel.getJComboBox(); //get each JComboBox that's created

    combo.addItemListener(new ItemListener() {
      //create a listener which applies to all the JComboBoxs' created
      //and reports an action every time a choice is clicked from the user
      public void itemStateChanged(ItemEvent evt) {
        System.out.println("ItemChanged "+evt);

        BuildPanel panel = getChoice(combo.getSelectedItem().toString()); //call the getChoice method, get the selected JComboBox's item and return a String containing the item's name


        List<BuildPanel> bpList = panel.getFrameStuff(); //Create a List with the frame components

        for(int i = 0; i<bpList.size(); i++){

          System.out.println("COMPONENT!!!");

          if (panel.getJComboBox().getSelectedItem().toString().equals(bpList.get(i).getName())) //If the name of the JComboBox component equals the name of one of the bpList names
          //which means that one of the JComboBox items is chosen by the user
          {
            System.out.println("binSize....");
            bpList.get(i).setVisible(true); //Set the items of the JComboBox chosen as visible
          }
          else
          {
            System.out.println("imageSize....");
            bpList.get(i).setVisible(false); //Set the items of the JComboBox chosen as hidden
          }
        }
      }
    });

    while(parseChoice())
    {
      System.out.println("Parse Choice Page finished!!!!");
    }
    System.out.println("Page Panel created!!!!");

      if(_frameList.size()==1) //If the _frameList
         List contains one item
        //add another item with respect to the
           GridBagConstraints set below
      {
      GridBagConstraints gridBagConstraints1 = new
         GridBagConstraints();
      gridBagConstraints1.gridx = 0;
      gridBagConstraints1.gridy =
         GridBagConstraints.RELATIVE;
      gridBagConstraints1.fill = GridBagConstraints
         .BOTH;
        System.out.println("Adding the frame to the
            current panel!!! "+i);

        _layout.add((BuildPanel)getFrame(),
           gridBagConstraints1);
        /*
         * At this point, the size of _frameList
            must be zero. That means that we have
            finished with
         * one nest.
         */
      }
      else
      {
        System.out.println("Number of frames inside
            _frameList "+_frameList.size());
        LayoutPane last = getFrame();
        LayoutPane prev = getFrame();
      GridBagConstraints gridBagConstraints2 = new
         GridBagConstraints();
      gridBagConstraints2.gridx = 0;
      gridBagConstraints2.gridy =
         GridBagConstraints.RELATIVE;
      gridBagConstraints2.fill = GridBagConstraints
         .BOTH;
        prev.add((BuildPanel) last,
           gridBagConstraints2); //Add the frame
           named "last" to the frame named "prev"
           with respect to the gridBagConstraints
           set right above
        _frameList.add(prev);

      }
    /*
     * At this point, the size of _frameList must
        be zero. That means that we have finished
        with
     * one nest.
     */
    return _ok;
}

private boolean parseChoice()
{
  boolean status = true;
  parseLine(); //Read the contents of each line of the .lyt file, find the regular expressions, find the comments (and ignore them)

  if (_result.get(0).contains("Page"))
  {
    System.out.println("Adding an Item to the ComboBox "+_result);

    BuildPanel panel = (BuildPanel) getFrame();
    panel.getJComboBox().addItem(_result.get(1)); //Add the second word of the line to the JComboBox and then to the panel
    _frameList.add(panel); //Add the panel to the List _frameList

    BuildPanel frame = new BuildPanel(_result.get(1)); //Create a frame named after the second word of the line that is read
      _frameList.add(frame); //Add the frame to the List _frameList

      while (parse()) {
        //While searching for keywords like Parameter, Cards etc
      if (_ParamList.size() != 0) { //If the List _ParamList is not empty
        //add a frame with respect to the GridBagConstraints listed right below
        GridBagConstraints gridBagConstraints = new GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = GridBagConstraints.RELATIVE;
        gridBagConstraints.fill = GridBagConstraints.BOTH;
        BuildParameter bp = getBuildParameter();
        frame.addLayout(bp);
        frame.add(bp, gridBagConstraints); //add the parameters to the frame with respect to the GridBagConstraints
        System.out.println("NUMBER OF COMPONENTSS "
            + frame.getComponentCount());

        System.out.println("ITEM ADDED!!!");
      }
    }
      System.out.println("Number of frames inside
        _frameList "+_frameList.size());
      LayoutPane last = getFrame();
      LayoutPane prev = getFrame();
    GridBagConstraints gridBagConstraints2 = new
       GridBagConstraints();
    gridBagConstraints2.gridx = 0;
    gridBagConstraints2.gridy =
       GridBagConstraints.RELATIVE;
    gridBagConstraints2.fill = GridBagConstraints
       .BOTH;

    ((BuildPanel)prev).addFrame((BuildPanel) last
       , gridBagConstraints2); //Add the frame
       named "last" to the frame named "prev"
       with respect to the gridBagConstraints set
        right above
    _frameList.add(prev); //Add both frames to
       the List _frameList
    i++;
    status = _ok;
  }
  else if(_result.get(0).contains("end")) //If
     the line contains the word "end", break the
     loop
  {
    System.out.println("Type "+_result.get(0) );
     status = false;

  }
  return status;
}


private boolean parseEnableLayout() {
  if(_layout == null) //If the _layout List is
     null
  {
    _layout = new BuildPanel("FIRST PANEL"); //
       Create a new panel
  }

  boolean status = true;
  System.out.println("Creating Enabled Panel");
  if (_params.getType(_result.get(1)).equals("
      bool")) //If the second word of the _result
      List contains a boolean parameter
  {
    final JCheckBox checkBox = new JCheckBox(
        _result.get(1), true); //Create a
        JCheckBox named after the second word of
        the line
    checkBox.setFocusPainted(false);
    checkBox.setOpaque(true);

    BuildPanel panel = new BuildPanel(_result.get
        (1)); //Name the panel after the word
        situated next to each "Enable" keyword in
        the .lyt line
    panel.setName(_result.get(1));
        String tooltip = "TOOLTIP"+i;
        panel.setToolTipText(tooltip);

    _enableList.add(panel); //Add the panel to
        the List _enableList
    _frameList.add(panel); //Add the panel to the
         List _frameList

    while(parse())
    {
      System.out.println("enable parameter!!!");

      if(_ParamList.size() != 0) //If the
          _ParamList is not empty
          //add a components to the panel, with
              respect to the GridBagConstraints
              set below
      {
        GridBagConstraints gridBagConstraints =
            new GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy =
            GridBagConstraints.RELATIVE;
        gridBagConstraints.fill =
            GridBagConstraints.BOTH;
        BuildParameter bp = getBuildParameter();
        panel.addLayout(bp);
        panel.add(bp,gridBagConstraints); //Add
            the bp to the panel with respect to
            the GridBagConstraints
      }
    }
    System.out.println("NUMBER OF COMPONENTS "+
        panel.getComponentCount());
    //Set the look and feel of the border of each components on the layout

    ComponentTitledBorder componentBorder =
      new ComponentTitledBorder(checkBox, panel, BorderFactory.createEtchedBorder());
    checkBox.addActionListener(new ActionListener(){ //Add a listener to the checkbox
      public void actionPerformed(ActionEvent e){
        boolean enable = checkBox.isSelected();

        System.out.println("Title name "+checkBox.getText());
        BuildPanel panel = getEnable(checkBox.getText());
        List<BuildParameter> bpList = panel.getLayoutStuff();
        for(int i = 0; i<bpList.size(); i++){
          System.out.println("COMPONENT!!!");
          bpList.get(i).setDisable(enable);
        }
      }
    });
    panel.setBorder(componentBorder);

    /*
     * Move the last frame into the previous one
     */

      if(_frameList.size()==1)
      {
      GridBagConstraints gridBagConstraints = new GridBagConstraints();
      gridBagConstraints.gridx = 0;
      gridBagConstraints.gridy = GridBagConstraints.RELATIVE;
      gridBagConstraints.fill = GridBagConstraints.BOTH;
        System.out.println("Adding the frame to the current panel!!! "+i);

        _layout.add((BuildPanel) getFrame(), gridBagConstraints);
        /*
         * At this point, the size of _frameList must be zero. That means that we have finished with
         * one nest.
         */
      }
      else
      {
        System.out.println("Number of frames
           inside _frameList "+_frameList.size())
           ;
        LayoutPane last = getFrame();
        LayoutPane prev = getFrame();
      GridBagConstraints gridBagConstraints = new
          GridBagConstraints();
      gridBagConstraints.gridx = 0; /*This helps
         with the frame width problem*/
      gridBagConstraints.gridy =
         GridBagConstraints.RELATIVE;
      gridBagConstraints.fill =
         GridBagConstraints.BOTH;
        prev.add((BuildPanel) last,
           gridBagConstraints); //Add the frame
           named "last" to the frame named "prev"
            with respect to the
           gridBagConstraints set right above
        _frameList.add(prev);

      }

    System.out.println("Enable Done!!!!");

      i++;
      status = _ok;
  }
  else
  {
    System.out.println("Type "+_result.get(0));
    status = false;
  }
  return status;
}

private boolean parseFrameLayout() {
  if(_layout == null) //If the _layout is empty
  {
    _layout = new BuildPanel("FIRST PANEL"); //
       Add a new panel
  }
  System.out.println("Creating Frame Panel");

  BuildPanel panel = new BuildPanel(_result.get
     (1));

  panel.setName(_result.get(1)); //Get the second
      string of _result and set it as the name of
      the panel
      String tooltip = "TOOLTIP"+i;
      panel.setToolTipText(tooltip);

    _frameList.add(panel); //Add panel to _framelist

    while(parse())
    {
      if (_ParamList.size() != 0) //If _ParamList is empty
      { //Set new gridBagConstraints
        GridBagConstraints gridBagConstraints = new GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = GridBagConstraints.RELATIVE;
    gridBagConstraints.fill = GridBagConstraints.BOTH;
        panel.add(getBuildParameter(), gridBagConstraints); //Add the output of the getBuildParameter method to panel with respect to the gridBagContraints set right above
        System.out.println("NUMBER OF COMPONENTS "+ panel.getComponentCount());
    }

        System.out.println("PARAMETER ADDED!!!");
    }
    System.out.println("Frame END rechead");

    if(_frameList.size()==1) //If _frameList is not empty
    {   //Set new gridBagConstraints
    GridBagConstraints gridBagConstraints = new GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = GridBagConstraints.RELATIVE;
    gridBagConstraints.fill = GridBagConstraints.BOTH;
    System.out.println("Adding the frame to the current panel!!! "+i);

    _layout.add((BuildPanel) getFrame(), gridBagConstraints); //Add frame to _layout with respect to the gridBagConstraints set right above
    /*
       * At this point, the size of _frameList must be zero. That means that we have finished with
```

```
       * one nest.
       */
    }
    else
    {
      System.out.println("Number of frames inside
          _frameList "+_frameList.size());
      LayoutPane last =  getFrame();
      LayoutPane prev =  getFrame();
    GridBagConstraints gridBagConstraints = new
       GridBagConstraints();
    gridBagConstraints.gridx = GridBagConstraints
       .RELATIVE; //This keeps all tabs one next
       to the other
    gridBagConstraints.gridy = GridBagConstraints
       .RELATIVE;
    gridBagConstraints.fill = GridBagConstraints.
       BOTH;
      prev.add((BuildPanel) last,
         gridBagConstraints); //Add last panel to
          previous with respect to the
         gridBagContraints set right above
      _frameList.add(prev);

    }

    System.out.println("Frame Done!!!!");

    i++;
    return _ok;
}

private boolean parsePagedLayout() {
  System.out.println("New Card Panel create!!");

  if(_layout == null) //If _layout is empty
  {
    _layout = new BuildTabPage("FIRST PANEL"); //
       Add a new Tab
  }

  while(parsePage())
  {
    System.out.println("ParsePage false!!!!");
  }

  System.out.println("Page Panel created!!!!");

  return _ok;
}
```

```
private boolean parsePage()
{
  boolean status = true;

  parseLine();

  if (_result.get(0).contains("Page")) //If the first string of _result contains the word "Page"
  {
    System.out.println("Creating Page Panel");

    LayoutPane panel = null;
    if(_pageFlag == true)
    {
      panel = new BuildTabPage(_result.get(1)); //Add a Tab to the panel named as the second string of _result
    }
    else
    {
      panel = new BuildPanel(_result.get(1)); // Add a panel named as the second string of _result
    }

    panel.setName(_result.get(1)); //Set the name of the panel as the second string of _result
      String text = "TOOLTIP"+i;
      panel.setToolTipText(text);

    _frameList.add(panel); //Add panel to _framelist

      while(parse())
      {
        if (_ParamList.size() != 0) //If _ParamList is empty
        { //Set new gridBagConstraints
          GridBagConstraints gridBagConstraints = new GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = GridBagConstraints.RELATIVE;
        gridBagConstraints.fill = GridBagConstraints.BOTH;
          panel.add(getBuildParameter(), gridBagConstraints); //Add the output of the getBuildParameter method to the panel with respect to
```

```
                   the gridBagConstraints set right
                   above
                System.out.println("NUMBER OF
                   COMPONENTS "+ panel.
                   getComponentCount());
              }

                System.out.println("ITEM ADDED!!!");
            }

            if(_frameList.size()==1) //If _frameList is
                 not empty
            {   //Set new gridBagConstraints
            GridBagConstraints gridBagConstraints = new
                 GridBagConstraints();
            gridBagConstraints.gridx = 0;
            gridBagConstraints.gridy =
                GridBagConstraints.RELATIVE;
            gridBagConstraints.fill =
                GridBagConstraints.BOTH;
              System.out.println("Adding the page frame
                  to the current panel!!! "+_layout.
                 getNumberofItems());
              LayoutPane kk = getFrame();
              if(kk instanceof BuildTabPage) //If kk is
                  a Tab
                ( (BuildTabPage) _layout).add((
                   BuildTabPage) kk,gridBagConstraints)
                   ; //Add kk as Tab to _layout with
                   respect to the gridBagConstraints
                   set right above
              else
                _layout.add((BuildPanel) kk,
                   gridBagConstraints); //Add kk as
                   panel to _layout with respect to the
                    gridBagConstraints set right above
              /*
               * At this point, the size of _frameList
                  must be zero. That means that we have
                   finished with
               * one nest.
               */
              System.out.println("Adding the page frame
                  to the current panel!!! "+_layout.
                 getNumberofItems());
            }
            else
            {
              /*
               * Move the last frame into the previous
                  one
```

```
         */
        System.out.println("Number of frames inside _frameList "+_frameList.size());
        LayoutPane last =  getFrame();
        LayoutPane prev =  getFrame();
      GridBagConstraints gridBagConstraints = new GridBagConstraints();
      gridBagConstraints.gridx = 0;
      gridBagConstraints.gridy = GridBagConstraints.RELATIVE;
      gridBagConstraints.fill = GridBagConstraints.BOTH;

        if(last instanceof BuildTabPage) //If last is a Tab
          ((BuildTabPage) prev).add((BuildTabPage) last,gridBagConstraints); //Add last as Tab to prev with respect to the gridBagConstraints set right above
        else
          ((BuildPanel) prev).add((BuildPanel) last,gridBagConstraints); //Add last as panel to prev with respect to the gridBagConstraints set right above
        _frameList.add(prev); //Add prev to _frameList

      }

      System.out.println("Page Done!!!!");

      i++;
      status = _ok;
  }
  else if(_result.get(0).contains("end")) //If the first string of _result is "end"
  {
    System.out.println("Type "+_result.get(0) );
     status = false; //Set boolean status to false

  }
  return status;
}

private boolean parseParameter() {

    BuildParameter bp = new BuildParameter(_params.
        getType(_result.get(1))); //The
        BuildParameter class will get the second
        string of _result and pass it to _params as
        the type of parameter

    bp.getLabel().setText(_result.get(1)); //The
        getLabel() method of BuildParameter class
        will set the second string of _result as
        label for bp
    bp.getJTextField().setText(_params.getParam().
        get(_result.get(1))); //The getJTextField()
        method of BuildParameter class will fetch
        the second string of _result, pass it to the
        getParam() method which will withdraw this
        parameter from the _params and pass it to bp
        as the default parameter value

    _ParamList.add(bp); /Add bp to _ParamList
    System.out.println("SimpleParameter DONE");
    return _ok;

  }

  private BuildParameter getBuildParameter() {

    System.out.println("Parameter List Size:  "+
        _ParamList.size());
    return _ParamList.remove(_ParamList.size()-1);
        //Return the size of _ParamList minus one
  }

  private LayoutPane getFrame()
  {
    return _frameList.remove(_frameList.size()-1);
        //Return the size of _frameList minus one
  }

  private BuildPanel getChoice(String item) { //

    BuildPanel bp = null; //Set bp empty
    for (int n = 0 ; n < _choiceList.size() ; n++)
        //For all the items of _choiceList
    {
      for (int m = 0 ; m < _choiceList.get(n).
          getJComboBox().getItemCount(); m++) //For
          all the JComboBoxes included in
          _choiceList
      {
        if( _choiceList.get(n).getJComboBox().
            getItemAt(m).toString().equals(item)) //
```

```
             If the item no.m equals to the ”item”
             string
          bp = _choiceList.get(n); //Get the no.n
               item of _choiceList and pass it to bp
      }
    }
    return bp;
  }

  private BuildPanel getEnable(String name) {

    BuildPanel bp = null;
    for (int n = 0 ; n < _enableList.size() ; n++)
        //For all the items of _enableList
    {
      if( _enableList.get(n).getName().equals(name)
          ) //If the name of item no.n equals the
          string ”name”
        bp = _enableList.get(n); //Get item no.n
             from _enableList and pass it to bp
    }
    return bp;
  }

  public void show() {

    _frame = new JFrame(); //Create a new JFrame
    _frame.setTitle(windowName); //Set the title as
         the string ”windowName”
    _frame.setLocationRelativeTo(null);

    JFrame.setDefaultLookAndFeelDecorated(true); //
        Set the looks of the JFrame

    BuildPanel buttonPanel = new BuildPanel(null);
        //Add a new frame to JFrame, which will
        contain the Apply and Cancel buttons

    System.out.println(”Add the _layout to the main
         frame”);
    //Set the gridBagConstraints for the main frame
    GridBagConstraints gridBagConstraints = new
        GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 0;
    gridBagConstraints.fill = GridBagConstraints.
        BOTH;

    //Set the gridBagConstraints for the Apply
        button
    GridBagConstraints gridBagConstraintsA = new
        GridBagConstraints();
    gridBagConstraintsA.gridx = 0;
    gridBagConstraintsA.gridy = 1;
    gridBagConstraints.fill = GridBagConstraints.
        BOTH;
    buttonPanel.add(getApplyButton(),
        gridBagConstraintsA);

    //Set the gridBagConstraints for the Cancel
        button
    GridBagConstraints gridBagConstraintsC = new
        GridBagConstraints();
    gridBagConstraintsC.gridx = 1;
    gridBagConstraintsC.gridy = 1;
    gridBagConstraints.fill = GridBagConstraints.
        BOTH;
    buttonPanel.add(getCancelButton(),
        gridBagConstraintsC);

    //Set the gridBagConstraints for the main frame
    GridBagConstraints gridBagConstraintsB = new
        GridBagConstraints();
    gridBagConstraintsB.gridx = 0;
    gridBagConstraintsB.gridy = 1;
    gridBagConstraintsB.fill = GridBagConstraints.
        BOTH;

    //Add a scrollbar to the _layout contained in
        the main frame
    JScrollPane scrollpane = new JScrollPane();
    scrollpane.setViewportView((Component) _layout)
        ;
    _frame.add(scrollpane,BorderLayout.CENTER);
    _frame.add((Component) buttonPanel,BorderLayout
        .SOUTH);
    _frame.pack();
    _frame.setVisible(true);

    //—TEST— Extract the layout objects and "read
        " them
      System.out.println("Number of objects: "+
          _layout.getNumberofItems());
      _layout.printParameters();
    //—TEST— Extract the layout objects and "read
        " them
  }

  private JButton getJButton() {
    if (jButton == null) { //If there is no jButton
         created
      jButton = new JButton(); //Create one
      jButton.setText("Apply"); //Set the text of it as "Apply"
      jButton.setSize(jButton.getSize()); //Automatically resize the jButton depending on the text
      jButton.addActionListener(new java.awt.event.ActionListener() { //Add a listener
        public void actionPerformed(java.awt.event.ActionEvent e) {
          System.out.println("actionPerformed()");
          _frame.dispose(); //Close frame whenever the button is pressed
        }
      });
    }
    return jButton;
  }

  /**
   * This method initializes jButton1
   */
  private JButton getJButton1() {
    if (jButton1 == null) { //If there is no jButton created
      jButton1 = new JButton(); //Create one
      jButton1.setText("Cancel"); //Set the text of it as "Cancel"
      jButton1.setSize(jButton1.getSize()); //Automatically resize the jButton depending on the text
      jButton1.addActionListener(new java.awt.event.ActionListener() { //Add a listener
        public void actionPerformed(java.awt.event.ActionEvent e) {
          System.out.println("actionPerformed()");
          _frame.dispose(); //Close frame whenever the button is pressed
        }
      });

    }
    return jButton1;
  }

  public JButton getApplyButton() {
    return getJButton(); //Return the Apply button
  }

  public JButton getCancelButton() {
    return getJButton1(); //Return the Cancel button
  }

  public void close() { //Close the class
    try {
      _in.close(); //Close the .lyt filereader
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

G

BUILDPANEL CLASS

```
package layoutCreator;

import java.awt.Component;
import java.awt.Container;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import javax.swing.JPanel;
import javax.swing.BorderFactory;
import javax.swing.JComboBox;
import java.util.ArrayList;
import java.util.List;
import layoutCreator.BuildComboBox;
import javax.swing.border.TitledBorder;

public class BuildPanel extends JPanel implements
    LayoutPane {

  private static final long serialVersionUID = 1L;
  private BuildComboBox jComboBox;
  private List<BuildParameter> _addLayout = new
     ArrayList<BuildParameter>();
  private List<BuildPanel> _addFrame = new
     ArrayList<BuildPanel>();
  private String buildPanelName;
  private BuildCheckBox jCheckBox;

  /**
   * This is the default constructor
   */
  public BuildPanel(String panelName) {
    super();
    initialize(panelName);
  }

  /**
   * This method initializes this
   */
  private void initialize(String panelName) {
    this.setLayout(new GridBagLayout()); //Set layout to grid bag layout
    this.setBorder(BorderFactory.createTitledBorder(null, panelName, TitledBorder.LEFT, TitledBorder.TOP, null, null)); //Set the border preferences
    if (panelName != null) //If panelname is not empty
      this.setName(panelName); //Set the name of the panel according to panelname
    buildPanelName = getName(); //Get the name of the panel
    this.pack();
  }


  private void pack() {

  }

  public JPanel getPanel()
  {
    return this; //Return panel
  }

  public void addLayout(BuildParameter bp)
  {
    _addLayout.add(bp); //Add bp to _addLayout
  }

  public List<BuildParameter> getLayoutStuff()
  {
    return _addLayout;
  }

  public BuildComboBox getJComboBox() {
    if(jComboBox == null) { //If jComboBox doesn't exist
      jComboBox = new BuildComboBox(); //Build a new combobox
    }
    return jComboBox;
  }

  public BuildCheckBox getJCheckBox() {
    if(jCheckBox == null) { //If jCheckBox doesn't exist
      jCheckBox = new BuildCheckBox(); //Build a new checkbox
    }
    return jCheckBox;
```

```java

}

public void addFrame(BuildPanel frame, GridBagConstraints gridBagConstraints) {
  frame.setVisible(false); //Set frame invisible
  _addFrame.add(frame); //and add it to the _addFrame
  this.add(frame,gridBagConstraints); //then add frame to panel with respect to gridBagConstraints
  this.pack();
}

public List<BuildPanel> getFrameStuff() {
  return _addFrame; //Return the _addFrame contents
}

public void add(BuildPanel panel, GridBagConstraints gridBagConstraints) {
  System.out.println("IMPLEMENTATION ADD METHOD (BUILDPANEL)!!!! ");

  getPanel().add(panel,gridBagConstraints); //Add panel with respect to gridBagConstraints
}

public int getNumberofItems()
{
  return getPanel().getComponentCount(); //Count the number of panels
}

public void add(BuildParameter panel, GridBagConstraints gridBagConstraints) {
  System.out.println("IMPLEMENTATION ADD METHOD (BUILDPANEL) PARAMETER!!!! ");
  getPanel().add(panel,gridBagConstraints); //Add BuildParameter panel with respect to gridBagConstraints
}

public void printParameters()
{
  System.out.println("Number of parameters: " + this.getComponentCount());
  for (int i = 0 ; i < this.getComponentCount() ; ++i) //For each component
  {
      Component cp  = this.getComponent(i); //Get component no.i
      if (cp instanceof BuildParameter) //If component is a parameter
        ((BuildParameter) cp).getParameter().printContent(); //Fetch the parameter and print the content of it
      else
      {
        System.out.println("Label name: " + ((Container) cp).getName());
        ((LayoutPane) cp).printParameters(); // Print the label of the parameter
        /*JComboBoxes and CheckBoxes are not read. FIX IT*/
      }
    }
  }
}
```

H

BUILDPARAMETER CLASS

```java
package layoutCreator;

import javax.swing.JCheckBox;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import layoutCreator.*;
import java.awt.GridLayout;
import java.util.ArrayList;
import java.util.List;

public class BuildParameter extends JPanel {

  private static final long serialVersionUID = 1L;
  private String _type = null; //Type of parameter
  private JLabel jLabel = null; //The label of each parameter
  private JTextField jTextField = null; //The value of each parameter
  private JCheckBox jCheckBox = null; //Boolean value of each checkbox

  /**
   * This is the default constructor
   */
  public BuildParameter(String type) {
    super();
    _type = type;
    initialize();
  }

  /**
   * This method initializes this
   */
  private void initialize() {
    GridLayout gridLayout = new GridLayout(); // Choose the layout
    gridLayout.setRows(1); //Set the number of rows to 1
    jLabel = new JLabel(); //Create a label for the parameter
    jLabel.setText("JLabel");
    jLabel.setHorizontalTextPosition(SwingConstants.CENTER); //Set the text position
    jLabel.setHorizontalAlignment(SwingConstants.CENTER); //Set the text alignment
    jLabel.getSize(); //Get the label text size
    jLabel.setSize(getSize()); //Set the label size equal to the length of the label text

    this.setLayout(gridLayout); //Set the layout
    this.pack();
    this.add(jLabel, null);
    if (_type.equals("bool")) //If the _type equals to a boolean
      this.add(getJCheckBox(), null); //Create a checkbox
    else
      this.add(getJTextField(), null); //Create a textfield
  }

  private void pack() {
    return;
  }

  /**
   * This method initializes jTextField
   */
  public JTextField getJTextField() {
    if (jTextField == null) { //If the jTextField is null
      jTextField = new JTextField(); //Create a new textfield
      jTextField.getSize(); //Get the textfield text size
      jTextField.setSize(getSize()); //Set the textfield size equal to the text
      jTextField.setHorizontalAlignment(JTextField.LEFT); //Align the textfield
    }
    return jTextField;
  }

  public JCheckBox getJCheckBox() {
    if (jCheckBox == null) { //If the jCheckBox is null
```

```
      jCheckBox = new JCheckBox(); //Create a new checkbox
      jCheckBox.setHorizontalAlignment(JCheckBox.LEFT); //Align the checkbox
      jCheckBox.getSize(); //Get the size of it
      jCheckBox.setSize(getSize()); //Set the size equal to it
    }
    return jCheckBox;
  }

  public JLabel getLabel()
  {
    return this.jLabel; //Return the current label
  }

  public BuildParameter getParameter()
  {
    return this; //Return the parameter
  }

  public void setDisable(boolean b) {
    this.jLabel.setEnabled(b); //Enable the label so it can be visible
    if(getJCheckBox() != null) //If the getJCheckBox is not empty
      getJCheckBox().setEnabled(b); //Enable the checkbox
    if(getJTextField() != null) //If the getJTextField is not empty
      getJTextField().setEnabled(b); //Enable the textfield and make it editable
  }

  public String toString(){
    return _type;
  }

  public void printContent()
  {
    System.out.println("Parameter " + jLabel.getText() + " Value: " + jTextField.getText());
  }

}
```

I

BUILDTAB CLASS

```java
package layoutCreator;

import java.awt.GridBagLayout;
import javax.swing.JPanel;
import javax.swing.JTabbedPane;
import java.awt.GridBagConstraints;

public class BuildTab extends JPanel {

  private static final long serialVersionUID = 1L;
  private JTabbedPane jTabbedPane = null;

  /**
   * This is the default constructor
   */
  public BuildTab(String name) {
    super();
    initialize(name);
  }

  /**
   * This method initializes this
   * @return void
   */
  private void initialize(String name) {
    GridBagConstraints gridBagConstraints = new
        GridBagConstraints(); //Use the
        GridBagConstraints layout class
    gridBagConstraints.fill = GridBagConstraints.
        BOTH;
    gridBagConstraints.gridy = 0;
    gridBagConstraints.weightx = 1.0;
    gridBagConstraints.weighty = 1.0;
    gridBagConstraints.gridx = 0;
    this.setName(name); //Set the name of the tab
    this.pack();
    this.setLayout(new GridBagLayout()); //Set the
        tab layout constraints
    this.add(getJTabbedPane(), gridBagConstraints); //Add tabs to layout with respect to gridBagconstraints
  }

  private void pack() {
    return;
  }

  /**
   * This method initializes jTabbedPane
   * @return javax.swing.JTabbedPane
   */
  public JTabbedPane getJTabbedPane() {
    if (jTabbedPane == null) { //If jTabbedPane is null
      jTabbedPane = new JTabbedPane(); //Add a new jTabbedPane
    }
    return jTabbedPane;
  }

}
```

J

BUILDTABPAGE CLASS

```
package layoutCreator;

import javax.swing.JTabbedPane;
import java.awt.Component;
import java.awt.Container;
import java.awt.GridBagConstraints;

public class BuildTabPage extends JTabbedPane
    implements LayoutPane {

  private static final long serialVersionUID = 1L;
  private String tabPageName;
  /**
   * This is the default constructor
   */
  public BuildTabPage(String name) {
    super();
    initialize(name);
  }

  /**
   * This method initializes this
   */
  private void initialize(String name) {
    this.pack();
    this.setName(name); //Set the tab page name
    tabPageName = getName(); //Get the tabPageName
        string
  }

  private void pack() {
    return;
  }

  public JTabbedPane getTabPane()
  {
    return this; //Return the JTabbedPane
  }
```

```java

  public void add(BuildPanel panel, GridBagConstraints gridBagConstraints) {
    System.out.println("IMPLEMENTATION ADD METHOD (BUILDTABPAGE) !!!! ");

    getTabPane().add(panel,gridBagConstraints); // Add panel to JTabbedPane with respect to gridBagConstraints
  }

  public int getNumberofItems()
  {
    return getTabPane().getComponentCount(); // Count the components number of the JTabbedPane
  }

  public void add(BuildParameter panel, GridBagConstraints gridBagConstraints) {
    System.out.println("IMPLEMENTATION ADD METHOD (BUILDTABPAGE PARAMETER) !!!! ");

    getTabPane().add(panel,gridBagConstraints); // Add parameter panel to JTabbedPane with respect to gridBagConstraints
  }

  public void printParameters()
  {
    System.out.println("Number of parameters: " + this.getComponentCount());
    for (int i = 0 ; i < this.getComponentCount() ; ++i) //For the number of components
    {
      Component cp  = this.getComponent(i); //Get component no.i
      if (cp instanceof BuildParameter) //If cp is a parameter
        ((BuildParameter) cp).getParameter().printContent(); //Print the parameter content
      else
      {
        System.out.println("Tab name: " + ((Container) cp).getName()); //Get the name of cp and use it as the tab name
        ((LayoutPane) cp).printParameters(); /*JComboBoxes and CheckBoxes are not read. FIX IT*/
      }
```

```
        }
    }

}
```

K

BuildComboBox Class

```java
package layoutCreator;

import java.awt.Component;
import java.awt.Container;
import java.awt.GridBagConstraints;
import java.awt.event.ItemListener;
import javax.swing.JComboBox;
import javax.swing.JPanel;


public class BuildComboBox extends JComboBox
   implements LayoutPane {

  private static final long serialVersionUID = 1L;

  public BuildComboBox()
  {
    super();
  }

  public void add(BuildPanel panel,
      GridBagConstraints gridBagConstraints) {

  }

  public void add(BuildParameter panel,
      GridBagConstraints gridBagConstraints) {

  }

  public int getNumberofItems() {
    return this.getItemCount(); //Return the number
         of parameters

  }

  public void setName(String name) { //Set the name
       of each item

  }

  public void setToolTipText(String text) { //The text displays when the cursor lingers over the component

  }


  public void printParameters()
  {
    System.out.println("Number of parameters: " + this.getNumberofItems());
    for (int i = 0 ; i < this.getNumberofItems() ; ++i) //For each parameter
        System.out.println("ComboBox name: " + this.getItemAt(i));

  }

}
```

L

BUILDCHECKBOX CLASS

```java
package layoutCreator;

import java.awt.GridBagConstraints;
import javax.swing.JCheckBox;

public class BuildCheckBox extends JCheckBox implements LayoutPane {

  private static final long serialVersionUID = 1L;

  public BuildCheckBox()
  {
    super();
  }

  public void add(BuildPanel panel, GridBagConstraints gridBagConstraints) { //Add a checkbox

  }

  public void add(BuildParameter panel, GridBagConstraints gridBagConstraints) { //Add the boolean value of the checkbox

  }

  public int getNumberofItems() {
    return this.getComponentCount(); //Return the number of checkboxes in the panel

  }

  public void setName(String name) { //Set the name of the checkbox

  }
  public void setToolTipText(String text) { //The text displays when the cursor lingers over the checkbox

  }

  public int getComponentCount()  {
    return this.getComponentCount(); //Return the number of components

  }

  public void printParameters() {

    System.out.println("Number of CheckBoxes: " + this.getNumberofItems()); //Print the number of checkboxes in the panel
    for (int i = 0 ; i < this.getNumberofItems() ; ++i) //For each checkbox included in the panel
        System.out.println("CheckBox name: " + this.getComponent(i)); //print the name of each checkbox
  }
}
```

M

LAYOUTPANE INTERFACE

```java
package layoutCreator;

import java.awt.GridBagConstraints;

public interface LayoutPane { //This interface contains all the methods commonly used by BuildCheckBox, BuildPanel, BuildComboBox and BuildTabPage classes

  void add(BuildPanel panel, GridBagConstraints gridBagConstraints);

  void add(BuildParameter panel, GridBagConstraints gridBagConstraints);

  int getNumberofItems();

  void setName(String name);

  void setToolTipText(String text);

  int getComponentCount();

  void printParameters();

}
```

N

PASSWINDOWREADER CLASS

```java
package regularExpressions;
import com.cloudgarden.layout.AnchorLayout;

import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;

import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.util.List;
import java.util.logging.FileHandler;
import java.util.logging.Level;
import java.util.logging.Logger;


import javax.swing.ComboBoxModel;
import javax.swing.DefaultComboBoxModel;
import javax.swing.GroupLayout;

import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JProgressBar;
import javax.swing.JTabbedPane;
import javax.swing.JTextField;
import javax.swing.LayoutStyle;

import javax.swing.WindowConstants;
import javax.swing.SwingUtilities;

import layoutCreator.BuildPanel;


```

```
public class PassWindowReader extends javax.swing.
    JFrame {
  /**
   *
   */
  private static final long serialVersionUID = 1L;
  private JPanel ButtonPanel;
  private JPanel jPanel1;
  private JComboBox myComboBox;
  private JTextField jTextField1;
  private JLabel jLabel1;
  private JCheckBox jCheckBox1;
  private JPanel jPanel2;
  private JButton CancelButton;
  private JButton ApplyButton;
  private String labelText;

  private static Object comboBoxtext;
  boolean b = false;
  private static String textField;
  private String checkBoxText;
  private JLabel comboBoxPopupLabel;
  private JTextField comboBoxTextField;
  private JPanel comboBoxPanel;
  private JPanel jPanel4;
  private JPanel jPanel3;
  private JTabbedPane jTabbedPane1;
  private String comboBoxPanelLabel;
  private String comboTextField;
  private String comboBoxLabel;

  /**
  * Auto-generated main method to display this
     JFrame
  */
  public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        PassWindowReader inst = new
            PassWindowReader();
        inst.setLocationRelativeTo(null);
        inst.setVisible(true);
      }
    });
  }


  public PassWindowReader() {
    super();
    initGUI();

  }

  private void initGUI() {
    try {
      GridBagLayout thisLayout = new GridBagLayout
          ();
      thisLayout.rowWeights = new double[] {0.1};
      thisLayout.rowHeights = new int[] {};
      thisLayout.columnWeights = new double[] {};
      thisLayout.columnWidths = new int[] {};
      getContentPane().setLayout(thisLayout);
      setDefaultCloseOperation(WindowConstants.
          DISPOSE_ON_CLOSE);
      {
        ButtonPanel = new JPanel();
        GridBagLayout ButtonPanelLayout = new
            GridBagLayout();
        getContentPane().add(ButtonPanel, new
            GridBagConstraints(1, 1, 1, 1, 0.0, 0.0,
             GridBagConstraints.SOUTH,
            GridBagConstraints.NONE, new Insets(0,
            0, 0, 0), 0, 0));
        Dimension buttonPanelDimension =
            ButtonPanel.getSize();
        ButtonPanel.setSize(buttonPanelDimension);
        ButtonPanelLayout.rowWeights = new double[]
             {0.1, 0.1, 0.1, 0.1};
        ButtonPanelLayout.rowHeights = new int[]
            {7, 7, 7, 7};
        ButtonPanelLayout.columnWeights = new
            double[] {0.1, 0.1, 0.1, 0.1};
        ButtonPanelLayout.columnWidths = new int[]
            {7, 7, 7, 7};
        ButtonPanel.setLayout(ButtonPanelLayout);
        {
          ApplyButton = new JButton();
          GroupLayout ApplyButtonLayout = new
              GroupLayout((JComponent)ApplyButton);
          ButtonPanel.add(ApplyButton, new
              GridBagConstraints(0, 0, 1, 1, 0.0,
              0.0, GridBagConstraints.CENTER,
              GridBagConstraints.NONE, new Insets(0,
               0, 0, 0), 0, 0));
          ApplyButton.setText(”Apply”);
          ApplyButton.setLayout(ApplyButtonLayout);

          //action listener assigned to Apply
              button
          ApplyButton.addActionListener(new java.
              awt.event.ActionListener() {
```

```
              public void actionPerformed(java.awt.
                  event.ActionEvent e) {
                //If the textfield doesn't change,
                    print a blank space else print the
                     new value

                  textField = jTextField1.getText();

                b = jCheckBox1.isSelected();
                String CheckBoxResult = checkBoxText+
                    ": "+b+"\n";//Merge the checkbox
                    label with the checkbox output
                String TextFieldResult = labelText+":
                     "+textField+"\n";//Merge the
                    textfield label with the textfield
                     output

                //If the combotextfield doesn't
                    change, print a blank space else
                    print the new value

                  comboTextField = comboBoxTextField.
                      getText();

                String comboBoxPanelResult =
                    comboBoxLabel+": "+comboTextField+
                    "\n";//Merge the combobox label
                    with the combobox textfield output

                System.out.println("Apply: \n"+
                    TextFieldResult+CheckBoxResult+
                    comboBoxPanelResult);
              //XML logging

                try {
                      FileHandler handler = new
                          FileHandler("RISAlog.xml");
                      Logger logger = Logger.
                          getLogger("java2s.logging");
                      logger.addHandler(handler);
                      logger.log(Level.INFO,
                          TextFieldResult+
                          CheckBoxResult+
                          comboBoxPanelResult);
                    } catch (Exception ex) {
                      ex.printStackTrace();
                    }

              //end of XML logging
              }
            });
            ApplyButtonLayout.setVerticalGroup(
               ApplyButtonLayout.createParallelGroup
               ());
            ApplyButtonLayout.setHorizontalGroup(
               ApplyButtonLayout.createParallelGroup
               ());
          }

          {
            CancelButton = new JButton();
            ButtonPanel.add(CancelButton, new
               GridBagConstraints(3, 0, 1, 1, 0.0,
               0.0, GridBagConstraints.CENTER,
               GridBagConstraints.NONE, new Insets(0,
                0, 0, 0), 0, 0));
            CancelButton.setText("Cancel");

            CancelButton.addActionListener(new java.
               awt.event.ActionListener() {
              public void actionPerformed(java.awt.
                 event.ActionEvent e) {
                System.out.println("Cancel");
                dispose();
              }
            });
          }
        }
        {
          jTabbedPane1 = new JTabbedPane();
          Dimension tabSize = jTabbedPane1.getSize();
          jTabbedPane1.setSize(tabSize);
          getContentPane().add(jTabbedPane1, new
             GridBagConstraints(1, 0, 1, 1, 0.0, 0.0,
              GridBagConstraints.CENTER,
             GridBagConstraints.BOTH, new Insets(0,
             0, 0, 0), 0, 0));
          {
            jPanel1 = new JPanel();
            GridBagLayout jPanel1Layout = new
               GridBagLayout();
            jTabbedPane1.addTab("jPanel1", null,
               jPanel1, null);
            jPanel1Layout.rowWeights = new double[]
               {0.1, 0.1};
            jPanel1Layout.rowHeights = new int[] {7,
               7};
            jPanel1Layout.columnWeights = new double
               [] {0.1};
            jPanel1Layout.columnWidths = new int[]
               {7};
            jPanel1.setLayout(jPanel1Layout);
```

```
  {
    jPanel3 = new JPanel();
    Dimension panel3Size = jPanel3.getSize
       ();
    jPanel3.setSize(panel3Size);
    jPanel1.add(jPanel3, new
       GridBagConstraints(1, 1, 1, 1, 0.0,
       0.0, GridBagConstraints.CENTER,
       GridBagConstraints.BOTH, new Insets
       (0, 0, 0, 0), 0, 0));
    GridBagLayout jPanel3Layout = new
       GridBagLayout();
    jPanel3Layout.columnWidths = new int[]
       {7};
    jPanel3Layout.rowHeights = new int[]
       {7, 7};
    jPanel3Layout.columnWeights = new
       double[] {0.1};
    jPanel3Layout.rowWeights = new double[]
        {0.1, 0.1};
    jPanel3.setLayout(jPanel3Layout);
    {
      jTextField1 = new JTextField();
      jPanel3.add(jTextField1, new
         GridBagConstraints(0, 1, 1, 1,
         0.0, 0.0, GridBagConstraints.
         CENTER, GridBagConstraints.NONE,
         new Insets(0, 0, 0, 0), 0, 0));
      jTextField1.setText("jTextField1");
    }
    {
      jLabel1 = new JLabel();
      jPanel3.add(jLabel1, new
         GridBagConstraints(0, 0, 1, 1,
         0.0, 0.0, GridBagConstraints.
         CENTER, GridBagConstraints.NONE,
         new Insets(0, 0, 0, 0), 0, 0));
      AnchorLayout jLabel1Layout = new
         AnchorLayout();
      jLabel1.setText("A random label");
      jLabel1.setLayout(jLabel1Layout);
      labelText = jLabel1.getText();
    }
  }
  Dimension panel1Size = jPanel1.getSize();
  jPanel1.setSize(panel1Size);
}
{
  jPanel2 = new JPanel();
  jTabbedPane1.addTab("jPanel2", null,
     jPanel2, null);
```

```
        GridBagLayout jPanel2Layout = new
            GridBagLayout();
        jPanel2Layout.rowWeights = new double[]
            {0.1, 0.1, 0.1};
        jPanel2Layout.rowHeights = new int[] {7,
            7, 7};
        jPanel2Layout.columnWeights = new double
            [] {0.1};
        jPanel2Layout.columnWidths = new int[]
            {7};
        Dimension panelSize = jPanel2.getSize();
        jPanel2.setSize(panelSize);
        jPanel2.setLayout(jPanel2Layout);
        {
          comboBoxPanel = new JPanel();
          jPanel2.add(comboBoxPanel, new
              GridBagConstraints(0, 2, 1, 1, 0.0,
              0.0, GridBagConstraints.PAGE_END,
              GridBagConstraints.BOTH, new Insets
              (0, 0, 0, 0), 0, 0));
          GridBagLayout comboBoxPanelLayout = new
               GridBagLayout();
          comboBoxPanelLayout.columnWidths = new
              int[] {7};
          comboBoxPanelLayout.rowHeights = new
              int[] {7};
          comboBoxPanelLayout.columnWeights = new
               double[] {0.1};
          comboBoxPanelLayout.rowWeights = new
              double[] {0.1};
          comboBoxPanel.setLayout(
              comboBoxPanelLayout);
          Dimension comboBoxPanelSize =
              comboBoxPanel.getSize();
          comboBoxPanel.setSize(comboBoxPanelSize
              );
          comboBoxPanel.setVisible(false);
          {
            comboBoxPopupLabel = new JLabel();
            comboBoxPanel.add(comboBoxPopupLabel,
                 new GridBagConstraints(0, 2, 1,
                1, 0.0, 0.0, GridBagConstraints.
                CENTER, GridBagConstraints.BOTH,
                new Insets(0, 0, 0, 0), 0, 0));
            comboBoxPopupLabel.setText("Another
                label");
            comboBoxPopupLabel.setLayout(null);
            comboBoxLabel = comboBoxPopupLabel.
                getText();
          }
          {
```

```
       comboBoxTextField = new JTextField();
       comboBoxPanel.add(comboBoxTextField,
         new GridBagConstraints(0, 3, 1, 1,
         0.0, 0.0, GridBagConstraints.
         CENTER, GridBagConstraints.BOTH,
         new Insets(0, 0, 0, 0), 0, 0));
       comboBoxTextField.setText("
         jTextField2");
      }
     }
     {
      final ComboBoxModel myComboBoxModel =
       new DefaultComboBoxModel(
         new String[] { "Something", "
           Something else" });
      myComboBox = new JComboBox();
      jPanel2.add(myComboBox, new
        GridBagConstraints(0, 0, 1, 1, 0.0,
        0.0, GridBagConstraints.CENTER,
        GridBagConstraints.NONE, new Insets
        (0, 0, 0, 0), 0, 0));
      myComboBox.setModel(myComboBoxModel);
      comboBoxtext = myComboBox.
        getSelectedItem().toString();

      myComboBox.addItemListener(new
        ItemListener() {
       public void itemStateChanged(
         ItemEvent evt) {
        System.out.println("Event: "+evt);

        if (myComboBox.getSelectedItem().
          toString().equals("Something
          else"))//If the name of the
          JComboBox component equals the
          name of one of the bpList names
         //which means that one of the
           JComboBox items is chosen by
           the user
         {
          System.out.println("Visible");
          comboBoxPanel.setVisible(true);
            //Set the items of the
            JComboBox chosen as visible
         }
         else
         {
          System.out.println("Not visible
            ");
```

```
                    comboBoxPanel.setVisible(false);//Set the items of the JComboBox chosen as hidden
                  }

                }
              });
            }
          {
            jCheckBox1 = new JCheckBox();
            AnchorLayout jCheckBox1Layout = new AnchorLayout();
            jPanel2.add(jCheckBox1, new GridBagConstraints(0, 1, 1, 1, 0.0, 0.0, GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(0, 0, 0, 0), 0, 0));
            jCheckBox1.setText("My CheckBox");
            jCheckBox1.setLayout(jCheckBox1Layout);
            checkBoxText = jCheckBox1.getText();

          }
        }
      }
      this.pack();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }


  protected BuildPanel getChoice(String string) {
    // TODO Auto-generated method stub
    return null;
  }
}
```

O

EVSELECT .PAR FILE

```
<FILE>
<CONFIG>

<PARAM id="table" type="table" mandatory="yes">
  <DESCRIPTION> Name of the table to be filtered </
      DESCRIPTION>
</PARAM>

<PARAM id="keepfilteroutput" type="bool" default="
    no">
  <DESCRIPTION> Keep output of filtering process? <
      /DESCRIPTION>
  <CASE>
    <ITEM value="no"> </ITEM>
    <ITEM value="yes">
      <PARAM id="withfilteredset" type="bool"
          default="no">
        <DESCRIPTION> Create a filtered event list
            </DESCRIPTION>
        <CASE>
          <ITEM value="no"> </ITEM>
          <ITEM value="yes">
            <PARAM id="filteredset" type="dataset"
                default="filtered.fits">
              <DESCRIPTION> Name of file for
                  filtered event list </DESCRIPTION>
            </PARAM>
          </ITEM>
        </CASE>
      </PARAM>
    </ITEM>
  </CASE>
</PARAM>

<PARAM id="destruct" type="bool" default="yes">
  <DESCRIPTION> Destructive event selection? </
      DESCRIPTION>
  <CASE>
    <ITEM value="no">
```

```
      <PARAM id="flagcolumn" type="string" default="EVFLAG">
        <DESCRIPTION> Name of integer column for event flagging </DESCRIPTION>
      </PARAM>

      <PARAM id="flagbit" type="int" default="−1">
        <DESCRIPTION> Bit position in flagcolumn column to save select status </DESCRIPTION>
        <CONSTRAINTS>
          flagbit in [−1:31]
        </CONSTRAINTS>
      </PARAM>
    </ITEM>
    <ITEM value="yes"> </ITEM>
  </CASE>
</PARAM>

<PARAM id="filtertype" type="string" default="expression">
  <DESCRIPTION> Type of filtering to use </DESCRIPTION>
  <CASE>
    <ITEM value="dataSubspace">
      <PARAM id="dssblock" type="block" default="">
        <DESCRIPTION> Name of block containing the Data Suspace specification to use for filtering </DESCRIPTION>
      </PARAM>
    </ITEM>
    <ITEM value="expression">
      <PARAM id="expression" type="string" default="true">
        <DESCRIPTION> Filtering expression </DESCRIPTION>
      </PARAM>
    </ITEM>
  </CASE>
</PARAM>

<PARAM id="writedss" type="bool" default="yes">
  <DESCRIPTION> Write data subspace information to the output files </DESCRIPTION>
  <CASE>
    <ITEM value="no"> </ITEM>
    <ITEM value="yes">
      <PARAM id="cleandss" type="bool" default="no">
```

```
          <DESCRIPTION> Remove components from the
              data subspace which select no events </
              DESCRIPTION>
        </PARAM>

        <PARAM id="updateexposure" type="bool"
            default="yes">
          <DESCRIPTION> Update exposure information
              in the output files (XMM specific) </
              DESCRIPTION>
        </PARAM>

        <PARAM id="filterexposure" type="bool"
            default="yes">
          <DESCRIPTION> Filter exposure extensions
              with the GTIs which apply to them </
              DESCRIPTION>
        </PARAM>
      </ITEM>
    </CASE>
</PARAM>

<PARAM id="blockstocopy" type="string" list="yes">
  <DESCRIPTION> Extentions to copy to the extracted
       data sets </DESCRIPTION>
</PARAM>

<PARAM id="attributestocopy" type="string" list="
    yes">
  <DESCRIPTION> Attributes to copy from the input
      table to the output product table </
      DESCRIPTION>
</PARAM>

<PARAM id="energycolumn" type="string" default="PHA
    ">
  <DESCRIPTION> Name of col for energy information
      for spectra and light curves </DESCRIPTION>
</PARAM>

<PARAM id="withzcolumn" type="bool" default="no">
  <DESCRIPTION> Use zcolumn for product
      accumulation </DESCRIPTION>
  <CASE>
    <ITEM value="no"> </ITEM>
    <ITEM value="yes">
      <PARAM id="zcolumn" type="string" default="
          WEIGHT">
        <DESCRIPTION> Column of values to
            accumulate in an image, lightcurve,
            spectrum or histogram </DESCRIPTION>
      </PARAM>

      <PARAM id="withzerrorcolumn" type="bool" default="no">
        <DESCRIPTION> Use zerrorcolumn as the error on zcolumn </DESCRIPTION>
        <CASE>
          <ITEM value="no"> </ITEM>
          <ITEM value="yes">
            <PARAM id="zerrorcolumn" type="string" default="EWEIGHT">
              <DESCRIPTION> Column of the error on the zcolumn value for a lightcurve, spectrum or histogram </DESCRIPTION>
            </PARAM>
          </ITEM>
        </CASE>
      </PARAM>
    </ITEM>
  </CASE>
</PARAM>

<PARAM id="ignorelegallimits" type="bool" default="no">
  <DESCRIPTION> Ignore TLMIN/MAX values when extracting data from columns? </DESCRIPTION>
</PARAM>

<PARAM id="withimageset" type="bool" default="no">
  <DESCRIPTION> Extract an image file </DESCRIPTION>
  <CASE>
    <ITEM value="no"> </ITEM>
    <ITEM value="yes">
      <PARAM id="imageset" type="dataset" default="image.fits">
        <DESCRIPTION> Name of image file to extract </DESCRIPTION>
      </PARAM>

      <PARAM id="xcolumn" type="string" default="RAWX">
        <DESCRIPTION> Name of X coord column for image creation </DESCRIPTION>
      </PARAM>

      <PARAM id="ycolumn" type="string" default="RAWY">
        <DESCRIPTION> Name of Y coord column for image creation </DESCRIPTION>
      </PARAM>

      <PARAM id="imagebinning" type="string" default="imageSize">
        <DESCRIPTION> Use bin sizes or image sizes to determine binning factor </DESCRIPTION>
        <CASE>
          <ITEM value="binSize">
            <PARAM id="ximagebinsize" type="real" default="1" mandatory="yes">
                  <DESCRIPTION> Binning factor for x axis in image creation </DESCRIPTION>
                  <CONSTRAINTS>
                    ximagebinsize in [0:]
                  </CONSTRAINTS>
            </PARAM>

            <PARAM id="yimagebinsize" type="real" default="1" mandatory="yes">
                  <DESCRIPTION> Binning factor for y axis in image creation </DESCRIPTION>
                  <CONSTRAINTS>
                    yimagebinsize in [0:]
                  </CONSTRAINTS>
            </PARAM>
          </ITEM>
          <ITEM value="imageSize">
            <PARAM id="squarepixels" type="bool" default="no">
                  <DESCRIPTION> Force x and y bin size to be the same when imagebinning=imageSize </DESCRIPTION>
            </PARAM>

            <PARAM id="ximagesize" type="int" default="600">
                  <DESCRIPTION> Image size in the x coordinate (used to determine binning factor) </DESCRIPTION>
                  <CONSTRAINTS>
                    ximagesize in [0:]
                  </CONSTRAINTS>
            </PARAM>

            <PARAM id="yimagesize" type="int" default="600">
              <DESCRIPTION> Image size in the y coordinate (used to determine binning factor) </DESCRIPTION>
                  <CONSTRAINTS>
                    yimagesize in [0:]
                  </CONSTRAINTS>
            </PARAM>
          </ITEM>
        </CASE>
      </PARAM>

      <PARAM id="withxranges" type="bool" default="no">
        <DESCRIPTION> Use min/max values for x axis image extraction </DESCRIPTION>
        <CASE>
          <ITEM value="no"> </ITEM>
          <ITEM value="yes">
            <PARAM id="ximagemin" type="real" default="1" mandatory="yes">
                  <DESCRIPTION> Lower limit of x axis for image extraction </DESCRIPTION>
            </PARAM>

            <PARAM id="ximagemax" type="real" default="640" mandatory="yes">
                  <DESCRIPTION> Upper limit of x axis for image extraction </DESCRIPTION>
            </PARAM>
          </ITEM>
        </CASE>
      </PARAM>

      <PARAM id="withyranges" type="bool" default="no">
        <DESCRIPTION> Use min/max values for x axis image extraction </DESCRIPTION>
        <CASE>
          <ITEM value="no"> </ITEM>
          <ITEM value="yes">
            <PARAM id="yimagemin" type="real" default="1" mandatory="yes">
                  <DESCRIPTION> Lower limit of y axis for image extraction </DESCRIPTION>
            </PARAM>

            <PARAM id="yimagemax" type="real" default="640" mandatory="yes">
                  <DESCRIPTION> Upper limit of y axis
                      for image extraction </DESCRIPTION
                      >
            </PARAM>
          </ITEM>
        </CASE>
      </PARAM>

      <PARAM id="withimagedatatype" type="bool"
          default="no">
        <DESCRIPTION> Use imagedatatype to set the
            type of image created </DESCRIPTION>
        <CASE>
          <ITEM value="no"> </ITEM>
          <ITEM value="yes">
            <PARAM id="imagedatatype" type="string"
                 default="Real64">
                  <DESCRIPTION> Data type of the image
                      to be created </DESCRIPTION>
                  <CASE>
                    <ITEM value="Int16"> </ITEM>
                    <ITEM value="Int32"> </ITEM>
                    <ITEM value="Int8"> </ITEM>
                    <ITEM value="Real32"> </ITEM>
                    <ITEM value="Real64"> </ITEM>
                  </CASE>
            </PARAM>
          </ITEM>
        </CASE>
      </PARAM>

      <PARAM id="withcelestialcenter" type="bool"
          default="no">
        <DESCRIPTION> Shift the center of the image
             to the specified ra and dec </
            DESCRIPTION>
        <CASE>
          <ITEM value="no"> </ITEM>
          <ITEM value="yes">
            <PARAM id="raimagecenter" type="real"
                    default="0" mandatory="yes">
                  <DESCRIPTION> Right ascension of the
                      center of the image, in decimal
                      degrees </DESCRIPTION>
            </PARAM>

            <PARAM id="decimagecenter" type="real"
                    default="0" mandatory="yes">
                  <DESCRIPTION> Declination of the
                      center of the image, in decimal
                      degrees </DESCRIPTION>
            </PARAM>
          </ITEM>
        </CASE>
      </PARAM>
    </ITEM>
  </CASE>
</PARAM>

<PARAM id="withspectrumset" type="bool" default="no">
  <DESCRIPTION> Extract a spectrum file </DESCRIPTION>
  <CASE>
    <ITEM value="no"> </ITEM>
    <ITEM value="yes">
      <PARAM id="spectrumset" type="dataset" default="spectrum.fits">
        <DESCRIPTION> Name of spectrum file to extract </DESCRIPTION>
      </PARAM>

      <PARAM id="spectralbinsize" type="int" default="10">
        <DESCRIPTION> Binning factor for spectrum creation </DESCRIPTION>
        <CONSTRAINTS>
          spectralbinsize in [1:]
        </CONSTRAINTS>
      </PARAM>

      <PARAM id="withspecranges" type="bool" default="no">
        <DESCRIPTION> Use min/max values for spectral channels </DESCRIPTION>
        <CASE>
          <ITEM value="no"> </ITEM>
          <ITEM value="yes">
            <PARAM id="specchannelmin" type="real" default="0" mandatory="yes">
              <DESCRIPTION> Minimum channel for spectrum creation </DESCRIPTION>
              <CONSTRAINTS>
                specchannelmin in [0:]
              </CONSTRAINTS>
            </PARAM>

            <PARAM id="specchannelmax" type="real" default="4095" mandatory="yes">
              <DESCRIPTION> Maximum channel for spectrum creation </DESCRIPTION>
              <CONSTRAINTS>
```

```
                specchannelmax in [0:]
              </CONSTRAINTS>
            </PARAM>
          </ITEM>
        </CASE>
      </PARAM>
    </ITEM>
  </CASE>
</PARAM>

<PARAM id="withrateset" type="bool" default="no">
  <DESCRIPTION> Extract a time series </DESCRIPTION>
  <CASE>
    <ITEM value="no"> </ITEM>
    <ITEM value="yes">
      <PARAM id="rateset" type="dataset" default="rate.fits">
        <DESCRIPTION> Name of time series file to extract </DESCRIPTION>
      </PARAM>

      <PARAM id="timecolumn" type="string" default="TIME">
        <DESCRIPTION> Name of col for time information </DESCRIPTION>
      </PARAM>

      <PARAM id="timebinsize" type="real" default="1">
        <DESCRIPTION> Size of time bins for time series files </DESCRIPTION>
        <CONSTRAINTS>
          timebinsize in [0:]
        </CONSTRAINTS>
      </PARAM>

      <PARAM id="withtimeranges" type="bool" default="no">
        <DESCRIPTION> Use min/max values for time series extraction </DESCRIPTION>
        <CASE>
          <ITEM value="no"> </ITEM>
          <ITEM value="yes">
            <PARAM id="timemin" type="time" default="0" mandatory="yes">
              <DESCRIPTION> Lower limit for time series </DESCRIPTION>
            </PARAM>
            <PARAM id="timemax" type="time" default="1000" mandatory="yes">
              <DESCRIPTION> Upper limit for time series </DESCRIPTION>
            </PARAM>
          </ITEM>
        </CASE>
      </PARAM>

      <PARAM id="maketimecolumn" type="bool" default="no">
        <DESCRIPTION> Include a time column in the time series extention </DESCRIPTION>
      </PARAM>

      <PARAM id="makeratecolumn" type="bool" default="no">
        <DESCRIPTION> Produce a lightcurve of rates rather than counts </DESCRIPTION>
      </PARAM>
    </ITEM>
  </CASE>
</PARAM>

<PARAM id="withhistogramset" type="bool" default="no">
  <DESCRIPTION> Extract a general histogram </DESCRIPTION>
  <CASE>
    <ITEM value="no"> </ITEM>
    <ITEM value="yes">
      <PARAM id="histogramset" type="dataset" default="histo.fits">
        <DESCRIPTION> Name of the histogram file to extract </DESCRIPTION>
      </PARAM>

      <PARAM id="histogramcolumn" type="string" default="TIME">
        <DESCRIPTION> Name of col for histogram generation </DESCRIPTION>
      </PARAM>

      <PARAM id="histogrambinsize" type="real" default="1">
        <DESCRIPTION> Size of bins for histogram files </DESCRIPTION>
        <CONSTRAINTS>
          histogrambinsize in [0:]
        </CONSTRAINTS>
      </PARAM>
      <PARAM id="withhistoranges" type="bool"
          default="no">
        <DESCRIPTION> Use min/max values for
           histogram extraction </DESCRIPTION>
        <CASE>
          <ITEM value="no"> </ITEM>
          <ITEM value="yes">
            <PARAM id="histogrammin" type="real"
                default="0" mandatory="yes">
              <DESCRIPTION> Lower limit for
                 histogram </DESCRIPTION>
            </PARAM>

            <PARAM id="histogrammax" type="real"
                default="1000" mandatory="yes">
              <DESCRIPTION> Upper limit for
                 histogram </DESCRIPTION>
            </PARAM>
          </ITEM>
        </CASE>
      </PARAM>
    </ITEM>
  </CASE>
</PARAM>

</CONFIG>
</FILE>
```

P

EVSELECT .LYT FILE

```
Cards
  Page General
    Frame InOut
      Parameter table
      Enable keepfilteroutput
        Enable withfilteredset
          Parameter filteredset
        end
      end
      Enable writedss
        Parameter cleandss
        Enable updateexposure
          Parameter filterexposure
        end
      end
      Parameter blockstocopy
    end
    Frame Filtering
      Choice filtertype
        Page expression
          Parameter expression
        end
        Page dataSubspace
          Parameter dssblock
        end
      end
      Parameter destruct
      Parameter flagcolumn
      Parameter flagbit
      Parameter ignorelegallimits
    end
    Frame MultiUseColumns
      Parameter energycolumn
      Enable withzcolumn
        Parameter zcolumn
        Enable withzerrorcolumn
          Parameter zerrorcolumn
        end
      end
    end
  end
  Page Image
    Enable withimageset
      Parameter imageset
      Frame Columns
        Parameter xcolumn
        Parameter ycolumn
      end
      Frame Ranges
        Enable withxranges
          Parameter ximagemin
          Parameter ximagemax
        end
        Enable withyranges
          Parameter yimagemin
          Parameter yimagemax
        end
      end
      Frame Binning
        Choice imagebinning
            Page imageSize
              Parameter ximagesize
              Parameter yimagesize
              Parameter squarepixels
            end
            Page binSize
              Parameter ximagebinsize
              Parameter yimagebinsize
            end
        end
      end
      Enable withimagedatatype
        Parameter imagedatatype
      end
      Enable withcelestialcenter
        Parameter raimagecenter
        Parameter decimagecenter
      end
    end
  end
  Page Spectrum
    Enable withspectrumset
      Parameter spectrumset
      Parameter spectralbinsize
      Enable withspecranges
        Parameter specchannelmin
        Parameter specchannelmax
      end
    end
  end
```

```
  Page Lightcurve
    Enable withrateset
      Parameter rateset
      Parameter timecolumn
      Parameter timebinsize
      Parameter maketimecolumn
      Parameter makeratecolumn
      Enable withtimeranges
        Parameter timemin
        Parameter timemax
      end
      Enable withzcolumn
        Parameter zcolumn
      end
    end
  end
  Page Histogram
    Enable withhistogramset
      Parameter histogramset
      Parameter histogramcolumn
      Parameter histogrambinsize
      Enable withhistoranges
        Parameter histogrammin
        Parameter histogrammax
      end
    end
  end
end
```