Undergraduate Research Opportunities in OSS

Cornelia Boldyreff, Andrea Capiluppi, Thomas Knowles, and James Munro
Centre for Research in Open Source Software
University of Lincoln,
Lincoln,
LN6 7TS,
United Kingdom
{cboldyreff, acapiluppi}@lincoln.ac.uk

Abstract. Using Open Source Software (OSS) in undergraduate teaching in universities is now commonplace. Students use OSS applications and systems in their courses on programming, operating systems, DBMS, web development to name but a few. Studying OSS projects from both a product and a process view also forms part of the software engineering curriculum at various universities. Many students have taken part in OSS projects as well as developers.
At the University of Lincoln, under the Undergraduate Research Opportunities Scheme (UROS), undergraduate student researchers have the chance to work over the summer embedded within an existing research centre on a UROS project. Here two such projects within the Centre for Research in Open Source Software (CROSS) are described: Collaborative Development for the XO Laptop (CODEX) and Software Modularity in Open Source Software (SoMOSS). The CODEX project focused on creating resources to support students undertaking software application development for the XO laptop, and the SoMOSS project focused on architectural studies of OSS instant messaging software.
Both projects achieved successful research outcomes; more importantly, both student researchers benefited directly from the encouragement and concrete assistance that they received through interaction with the wider OSS research community. Both projects are ongoing and present further research opportunities for students.

1 Introduction

Many staff and students working in schools and universities are active users and contributors to the Open Source Software communities through initiatives such as School Forge[1] and the UK Joint Information Systems Committee (JISC) OSS Watch supporting colleges and universities[2]. While cost is a factor influencing users, the openness of OSS applications is also an important factor especially where students need to study and possibly adapt the source code. This is particularly the case where OSS is used in teaching Computer Science and related subjects. Many students start to learn programming using Java supported by OSS Java IDEs such as BlueJ and Greenfoot[3] progress-

[1] http://www.schoolforge.net
[2] http://www.oss-watch.ac.uk
[3] BlueJ, http://www.bluej.org and Greenfoot, http://www.greenfoot.org

ing to Eclipse[4]. Linux often forms the basis of studies in operating systems along with commercially available operating systems. Many courses on Database Management Systems employ MySQL[5] in conjunction with associated web scripting languages such as PHP, Python, Ruby and Javascript. Students learn about web server technology through studying and using Apache. Courses in Software Engineering in the past often suffered from lack of realistic systems to use in teaching; with the advent of OSS, there is a wealth of large projects available for study by Software Engineering students. In addition, there are many OSS software engineering environments and tools available for student use such as Eclipse with its plug-ins. Many students studying Software Engineering have undertaken projects where their software development has formed the basis of a contribution to an existing OSS project. At the graduate student level, German has reported on his experiences at the University of Victoria [1] and various Masters level degree programmes have been proposed, see the work of the Master on libre (Open Source) software (MoLOS) project[6] and the towards Open Source Software adoption and dissemination (tOSSad) project[7].

In the Department of Computing and Informatics (DCI) at the University of Lincoln, students are exposed to Linux in their first year in the *Operating Systems* module and they also make use of MySQL and various OSS scripting languages in their modules on databases and web technology. In the second year Software Engineering module, students undertake a detailed study of the maintenance practices with a selected OSS project and identify an area where they can contribute a needed change to the software[8]. The second year group projects undertaken by all second year students in DCI are often based on using OSS applications in the development phase. This year some groups are using GitHub, the social code hosting environment[9], to support their project work. Third year student's individual projects have also employed and studied OSS. The SoMOSS project grew out of an earlier third year project [3].

At the University of Lincoln, under the Undergraduate Research Opportunities Scheme (UROS), undergraduate student researchers have the chance to work over the summer embedded within an existing research centre on a UROS project. A key feature of the UROS projects is that they engage the student researcher as a producer rather than a consumer of knowledge. These projects aim to enhance the links between teaching and research in the undergraduate curriculum with the goal of achieving research-informed teaching.

Here two such projects within the Centre for Research in Open Source Software (CROSS) are described: Collaborative Development for the XO Laptop (CODEX) in Section 2 and Software Modularity in Open Source Software (SoMOSS) in Section 3. A number of lessons have been learnt through these projects and these are highlighted in Section 4. Both projects are ongoing and Section 5 presents the conclusions so far and discusses further research opportunities for students.

[4] Eclipse, `http://www.eclipse.org`
[5] MySQL, `http://www.mysql.com`
[6] `http://www.nongnu.org/masterlibre/`
[7] `http://www.tossad.org`
[8] `http://jarboe.lincoln.ac.uk/sewiki`
[9] GitHub, `http://github.com/`

2 CODEX

The primary aim of CODEX is carry out research in support of the development projects found on all degree courses for students in the Department of Computing and Informatics (DCI) so that all students have the possibility of undertaking projects to develop applications suitable for the XO laptop, the principal system at the heart of the One Laptop Per Child (OLPC) project[10], and to produce a tutorial for students on the XO software providing guidance for the development of XO applications as part of the various projects which are related to student's degree course within DCI.

The OLPC project is seeking to engage children globally in this enterprise of "learning to learn" by equipping them with networked laptops. The XO laptop has explicit support for collaboration and sharing of activities through its Sugar user interface and mesh view which focuses on the activities of its networked users making real things directly supporting the evolution of knowledge as a collaborative enterprise. Education is necessarily a collaborative enterprise with a need for both repositories and also active support for the educational processes as learners engage with one another and their teachers. There is scope for our project students to develop applications for the XO and make a significant contribution.

All of the software associated with the OLPC project is built upon Open Source Software (OSS) and it is freely available to anyone in the world. Much of the development effort is undertaken by members of the open-source community entirely as volunteers; and this helps the project to evolve and grow. The free nature of the OLPC project's vision permits anyone to contribute resources to the project; and their contributions are not limited to just the software code, as any type of contribution is gratefully accepted [2].

Here at the University of Lincoln we have identified the OLPC project as a potentially excellent way of getting students involved in collaborative software development within the open-source community and as a way of allowing students to contribute to an ethical and worthwhile cause. We are aiming to determine a suitable environment in which students can collaborate and develop software for the XO laptop. XO software applications are known as 'activities' for the laptop's Sugar interface. Like the XO hardware, Sugar is designed specifically for the children of developing countries.

So in the initial research, the objective has been to survey these and evaluate the current tools being used in these projects as well as to investigate the current application program interfaces (APIs) of the software currently available for the XO. In brief, the CODEX project's work has involved three main phases:

1. investigative phase: background research on OLPC, study of XO software base and its existing applications, survey of existing XO application development tools and current XO software development projects, investigate the feasibility of adapting or further amplifying Eclipse to form the basis for development environment for XO student group projects.
2. development phase: determine a suitable development environment for XO application development by student groups, possibly based on Eclipse or other available

[10] `http://laptop.org/en/`

Open Source Software available from the OLPC project and develop an installable version of this development environment for student group use.
3. trial usage and evaluation phase: develop a range of simple applications using the deliverable of phase 2, i.e. the development environment, and produce a tutorial guide for project students.

This research has followed a classic form of research related to Software Engineering; it has involved domain analysis (the OLPC investigations), study of relevant technology (XO software and current applications and their APIs), study of supporting methods and tools for collaborative software development (existing OSS projects related to XO applications, their use of methods and tools), followed by the development of an appropriate environment for student group projects and its evaluation through trial usage leading to the production of the tutorial guide for future students.

The student acquired skills in researching technical literature as well as related educational material relating to the OLPC project and its applications. Technical skills were required for the research into development environments and the related development work identified above and for the production of the tutorial. All of the research and development outlined here was built on existing work, but the student had the opportunity to become part of a much larger enterprise by contributing back to the open source community as the CODEX deliverables are available as open source and have been hosted on the CODEX blog[11] and on GitHub [12].

Following initial research, the most appropriate solution was deemed to be using the Sugar interface for Sugar development as this would allow students to collaborate. However, there were not sufficient XO laptops available for student use so alternative development environments were investigated.

After further research of the alternative methods of development it became clear that a Linux LiveCD with Sugar would be the most suitable development environment for the student to produce. A LiveCD is a Linux distribution that can be run directly from the CD - without the need to be installed on the computer. A LiveCD has the following advantages:

1. Self-contained environment providing all the required tools and software;
2. No need to install on computer - easy to deploy; and
3. The system can be carried around wherever the student goes.

Some disadvantages were identified as follows:

1. LiveCDs are typically slow to boot;
2. The system is read-only, students need to save their work to USB drives; and
3. When a new version is created, the student will need a new CD.

It was felt that these were not significant; especially when the boot up speed was compared to the standard desktop system employed on the university computers normally used by the students.

[11] http://learninglab.lincoln.ac.uk/blogs/jmunro/
[12] http://github.com/jdmunro

2.1 Sugar Packages

As development of the LiveCD continued it became important to stay up-to-date with the latest versions of the Sugar software. It was discovered that nobody maintained Ubuntu packages of the Sugar suite and so the CODEX student researcher took on the task of producing these packages. This involved learning how to use many new tools and has proved to be invaluable experience. Much positive feedback from Sugar community members was given to the student researcher and this gave him the motivation to stay on top of this task.

The creation of the packages enabled easy integration with the LiveCD, making the task of updating the Sugar software on the CD trivial. To make this process even easier, a small set of Python scripts that automate some of the tedious aspects of remastering LiveCDs was produced.

2.2 The CODEX LiveCD

The CODEX LiveCD is the main deliverable of the project. The CD is a self contained development environment specifically tailored for use by students. The main features of the CD are listed below:

- Good hardware support;
- Integrated Sugar software and tools (see Figure 1);
- Sugar emulator (run activities without XO hardware);
- Useful menu shortcuts;
- Tutorial content and resources; and
- Example Sugar activities.

The CODEX LiveCD is intended to be very easy to use and supports a large range of software, as a result of the solid foundations of Ubuntu Linux [6] under the bonnet. The CD contains everything that students require to get started in developing activities for the XO laptop and information for future development. It can be used on almost all types of PC hardware as long as it is possible to boot directly from CD, a feature present on all modern computers.

3 SoMOSS

The initial aims of the SoMOSS project focused on the results of the student's earlier analysis of modular open source software of the Pidgin instant messaging (IM) client as part of his third year project and his development of modular plug-in for the client Pidgin. The intention was that a fully functional piece of software (plug-in) would then be produced for the client Pidgin, demonstrating how additional pieces of software connect and interact with the modular piece of software; and that this piece of software would then be ported to 1 or 2 other pieces of similar software, analysing the differentiation of development on several different platforms and the interaction technique implied by these.

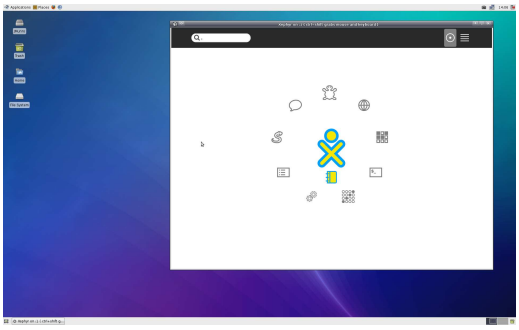

Fig. 1. The LiveCD running a Sugar emulator session

However, as often happens in research, the work changed direction during its course and became instead a series of architectural studies of OSS instant messaging systems. This research was carried out using Doxygen and studied the evolution of the systems' architecture. The principal results of SoMOSS have been reported in [3]. The student's blog records his intermediate working throughout the project[13].

3.1 The Shared Architecture for IM Systems

The first part of the project argued that a "common" architecture could be retrieved from a range of different IM systems. Three IM projects (Ayttm, Miranda, Pidgin) were selected for study, and all of their public releases analysed using the Doxygen suite. During the analysis of the evolution of the studied systems, recurring patterns have been observed in the naming of the *folders* containing source code, and the directions and frequency of connections among them, as follows:

1. **Core –** The first set of folders, as noticed in the evolutionary analysis of the selected systems, contained the core functions of an IM client. Connection to IM networks, handling of IM contacts, drawing of the underlying GUI are all examples of the "core" functions of an IM system.
2. **Protocols –** The second cluster of folders was observed when dealing with a number of supported IM protocols (*e.g.*, Yahoo, Jabber, etc.). A container "protocol" folder, keeping the source code shared by several protocols, was commonly found, while some protocols were further expanded in other subfolders (e.g., libyahoo, libjabber, etc.).
3. **Plugins –** The third observed and recurring component comprised the set of plugins handled by the IM client, ranging from GUI skins managers to connectors to email clients. A "plugin" umbrella folder was often found for the same purposes as the "protocol" above. The main purpose of this component is to hold those functionalities which are not considered as fundamental for an IM client. In terms

[13] `http://learninglab.lincoln.ac.uk/blogs/acapiluppi/`

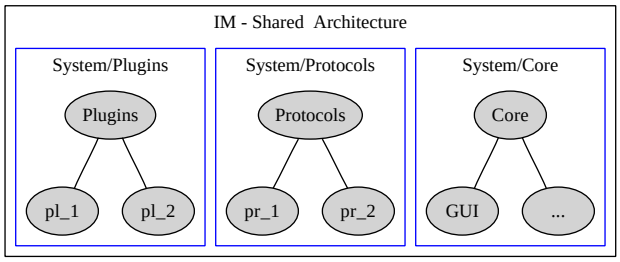

Fig. 2. Shared IM architecture

of connectors, each plugin can be considered as a stand-alone system, typically linked with few couplings to other plugins and other parts of the IM system.

In summary, a common architecture of IM systems has emerged, as visible in Figure 2. The three basic components (core, plugins and protocols) appear as primitives in the early releases of the observed systems, and get better refined and *modularized* during the system's lifetime, meaning that the refined components have few connections to other components. Each component exists as a cluster of several source folders; the hierarchical structure [4] comprises umbrella folders ("core", "plugins", "protocols") containing other folders, typically at the same level of nesting, acting as placeholder for source files of specific modules ("aycryption", "ticker", "yahoo", can be seen as an example module contained in each component).

3.2 Architectural evolution of IM Systems

In the second part of the project, the shared architecture of the three systems was observed in three temporal points, namely the earliest and the latest available releases, and the release in between the first two. The objective was to observe how the architectural components and connectors evolved. We summarize here the results of one of these systems (Pidgin).

The top left part of Figure 3 shows the growth of source folders in Pidgin. Initially (top right of Figure 3), just one folder contained all the source code for the system: even at this point, the Pidgin system already provided the three basic components which appear in each subsequent release. These primitives still have serious dependencies with each other, as visible by the arrows, which depict the amount of *function calls* directed from one component to the other.

The interesting insights we gained from the evolutionary analysis are shown in the two further bottom figure of Figure 3. The dependencies found among components in the first releases were later corrected and ultimately resolved in the latest available release (bottom right of Figure 3), where the "Core" component just relies on itself, and acts as a pure server to the other two). The same component appeared further decomposed into three subsystems, namely the "pidgin" graphical environment, the "libpurple" set of core functions, and the "finch" text-based version of the IM client.

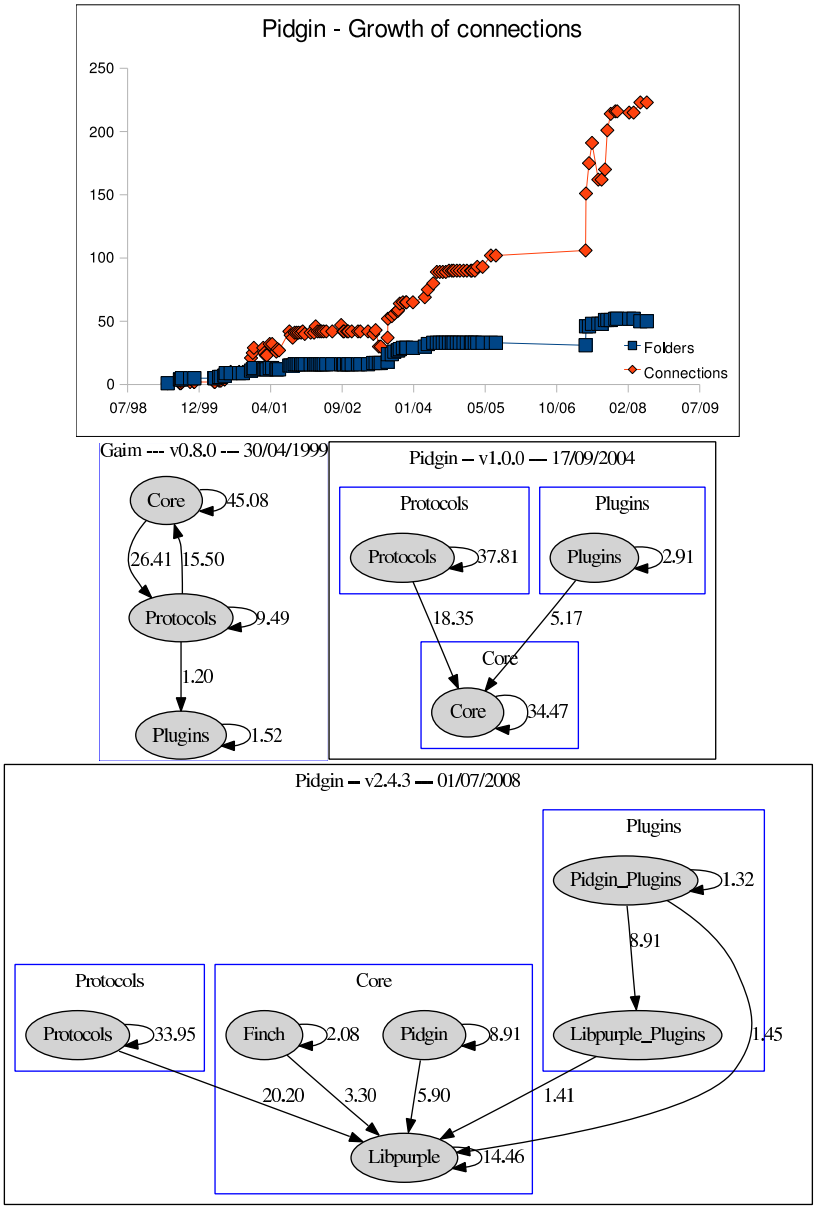

Fig. 3. Results (Pidgin) – Evolution of nodes, connections and architecture

3.3 Reuse Implications

Another conclusion was obtained by this project: from the reuse perspective, the empirical analysis conducted above was particularly relevant and conclusive for the Pid-

gin system: the reuse of its components (or modules) within other systems is greatly simplified by how the connections have been designed and simplified throughout the lifecycle of this application. As reported above, the "core" of this system, in its latest release, shows three modules ("Finch", "Libpurple" and "Pidgin") which are independent from both the protocols and the plugins components: when in need of recreating a new IM client, developers could safely extract the "libpurple" module (responsible for the vast majority of the basic functionalities of an IM system) and reuse it as the basis of a new IM system. In fact, this module acts as a pure server, and does not rely on any other components or modules of the system in which it belongs.

4 Lessons Learnt

The focus of both UROS projects has been on the student as producers so this section concentrates on their perspective.

4.1 The Student Perspective

CODEX The CODEX student undertook the project in the summer following his second year; for him, undertaking a research project was an entirely new experience. It is often not encountered until the final year of a degree when a student undertakes research as part of their dissertation project. He felt initial breadth of the CODEX project was overwhelming; however, his feeling soon subsided when the task was broken down into smaller manageable sections of research with the help of his supervisor. The quality of documentation on larger OSS projects such as The OLPC Wiki [14] and Sugar Labs[15] is commendable and proved extremely useful. The information is often concise, regularly updated and collaboratively produced due to the underlying wiki technology. The support of individuals within the OSS community also provided a valuable resource. Jani Monoses provided patient advice in a series of emails regarding the maintenance of his existing Sugar packages for Ubuntu. Morgan Collet provided encouragement in the production of new up-to-date Sugar packages and later founded the Sugar Team [16] responsible for continuing this task.

The CODEX project received some interest from the Sugar community after creating the Ubuntu packages. The result of this interest became the 'Sugar Team' [5], a group of enthusiasts with the aim of creating high quality, up-to-date Sugar packages for Ubuntu. Initially, the CODEX produced packages were used as a base for the work; however, this base soon evolved as a better and a much more efficient system for producing the packages became apparent. Although the student researcher from CODEX is no longer active in this effort due to time constraints, the hard work of the Sugar Team still continues and he is proud to have been a part of this.

The availability of a blog again proved to be a valuable resource for the CODEX project. The blog provided a tool for self-reflection on the progress of the research

[14] `http://wiki.laptop.org/go/The\_OLPC\_Wiki`
[15] `http://www.sugarlabs.org/go/Main\_Page`
[16] `https://launchpad.net/\~{}sugarteam`

project. It was effective for keeping measure of the progress of the project and also presented an up-to-date narrative of the project's progress for people not directly involved in the project itself.

SoMOSS Research utilising open-source software has opened a myriad of possibilities as a student researcher. Due to the limitations of proprietary instant messaging software in regards to licensing and the closed nature of the source code it became impossible to analyse any of the architecture, quality or interaction of the software. To overcome such issues, open-source software using an unencumbered software licence was used to ensure the flexibility and continuation of our studies.

Due to the open development model of the software (from source, documentation, community and alterations) it became quite simple to not only to review the software but map the modifications of the project, the developers and source code into various diagrams and statistics for analysis. Modification could then be made or suggested without the threat of breaking copyright or licence agreements and freely then be subsequently redistributed back to the community.

During the summer of 2008 the SOMOSS project managed to successfully find common architectures within three similar open-source software systems. This was done through the extraction and analysis of the three pieces of open source software throughout their respective software life-cycles. Whilst utilising free software tools such as Linux, BASH, Doxygen and Perl, it was possible to parse and document the structure of each revision of the software and subsequently map it onto diagrams using free tools such as Dotty and GIMP. Access to the software was possible due to the open and free CVS and FTP servers available to the general public for use. Paper and report writing was also possible utilising LaTeX and OpenOffice, all tools which are freely distributed and support the modification of their software.

To the student who utilised 100% free and open-source software during their project, it became quickly apparent that the maturity and power of these components were underestimated in the academic field. Open-source software helped the project become a success and ultimately our findings were contributed back into the community.

4.2 The Staff Perspective

From a staff perspective having student researchers in a research group over the summer isn't such an unusual occurrence. As both students were working on OSS related research this resulted in additional support for their work from the OSS community; and broadened out their support network beyond their supervisor. The use of a blog, wiki, and forge such as GitHub again are not unusual in research these days, but they are not so common in undergraduate research. As these tools facilitated external comments on the research, they were certainly beneficial and a factor its success. The UROS technical support officer who also acted as an additional advisor on CODEX provided the infrastructure for the blogs based on the OSS Wordpress project as part of his larger work in developing the the Learning Lab, Centre for Educational Research

and Development, University of Lincoln [17]. Students were encouraged to blog as part of their UROS project induction; little staff encouragement was needed to get students on the CROSS projects blogging.

5 Conclusions and Further Work

Both projects were successful in achieving their specific research aims although in both cases these aims developed along with the research. Both have also achieved the wider UROS aims of enhancing the links between teaching and research in the undergraduate curriculum and of informing current undergraduate programmes with existing research. With the CODEX LiveCD and tutorial wiki, future students can explore the Sugar interface and undertake development of applications for the XO laptop. It is now planned to use Sugar in the teaching HCI so that new approaches beyond the desktop can be demonstrated and explored by students. Building on the methods and tools applied in SoMOSS, a further study of OSS games engines is being undertaken by another undergraduate to determine common architectural elements within this domain. The SoMOSS project is ongoing and has become the focus of a postgraduate research project. Both staff and students and the wider community have benefitted as a result of these student research projects in OSS. Both projects were conducted as open projects with each student maintaining a blog reporting on the progress of the research throughout and exposing it to public scrutiny. Conducting the research in such an open way resulted in direct feedback during its course and positively encouraged the student researchers in their endeavours.

Acknowledgement. Both student researchers on the CODEX and SoMOSS projects were supported by the University of Lincoln's Undergraduate Research Opportunities Scheme.

References

1. German, D.M.: Experiences teaching a graduate course in open source software engineering. In: Proceedings of Open Educational Symposium of the 1st International Conference on Open Source Systems, pp. 326–328 (2005)
2. Get Involved. URL `http://wiki.laptop.org/go/The_OLPC_Wiki#Get_Involved`. Checked July 2008
3. Knowles, T., Capiluppi, A.: Extracting shared architectures from evolving floss systems. In: Proceedings of the 1st Workshop on Maintenance and Evolution of FLOSS (MEFLOSS) (2008)
4. Spinellis, D.: Code Reading: The Open Source Perspective. Addison-Wesley Professional (2003)
5. Sugar Team in Launchpad. URL `https://launchpad.net/~sugarteam`. Checked September 2008
6. Ubuntu Desktop Edition. URL `http://www.ubuntu.com/products/whatisubuntu/desktopedition`. Checked September 2008

[17] `http://learninglab.lincoln.ac.uk/`