FundFind

Exploring scholarly funding opportunities

Emanuil Evgeniev Tolev

April 9, 2013

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature .....

Date .....

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature .....

Date .....

Acknowledgements

I would like to thank my supervisor, Professor Reyer Zwiggelaar for his support and unique approach to the process of producing a major project.

Thanks to Dr. Richard Jensen, Dr. Joanne Walker and Professor Simon Cox for their valuable input in the formative stages of the project.

Furthermore, thanks to all Computer Science Department staff who have taught me over the last 4 years for their patience and being such an inspiration with their varied interests and opinions.

I would like to thank Cottage Labs LLP for introducing me to the Open Knowledge movement, as well as teaching me about most of the major technologies used in this project and agreeing to host a FundFind instance.

Thanks to the Open Knowledge Foundation for striking out and encouraging prototype projects, providing ample inspiration for FundFind.

I would like to thank Research Councils UK, a partnership of the seven UK Research Councils, for all the effort they have put into the Gateway to Research API, consolidating data from seven separate institutions and making it available under an Open license.

Abstract

FundFind is a web application which enables scholars, research development officers and post-graduates to share information about funding opportunities. Data about funding opportunities and funding organisations can be submitted, browsed and searched via a mobile-friendly web UI as well as a JSON API. FundFind stores data using an asynchronous approach, which enables the development of small independent modules for automatic funding data harvesting.

The project is a learning excercise, aiming to explore the field of scholarly funding, identifying potential problems with the open sharing of such data. Prototype technical solutions concretise the technical problems. Other types of problems are noted and discussed.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background and Objectives

1.1 The Project

This project is about learning to deal with the field of funding in modern scholarship[1] and exploring how it could be opened up, from the point of view of a software developer. It is set within the context of the Open Knowledge field and seeks to add to this field by exploring how scholarly funding can be made more transparent and easier to find for all interested parties.

The main vehicle of learning and main software output is an open-source web application named "FundFind". It's a simple crowdsourcing application which allows registered users to share information about funding opportunities under open[2] terms. Access to submitted information includes a machine-friendly API (Application Programmable Interface).

One of the project features - getting funding data into the datastore - was developed as a separate technical project due to software licensing issues and due to natural low coupling between the main application and this feature. It is called "Funding Harvest" [**?**].

1.2 Project Aims

- Learn more about the funding sector and identify user requirements within it. Results presented in "Audience and high-level requirements", §1.4 and more concretely in "Requirements", §2.4. The data must be sufficient to enable the creation of a project roadmap for

  a) future research work into the requirements of user groups which were not engaged during this project

  b) concrete development work which would enhance the software built during this project

  The foundations for this are laid in in "Future Work", §6.5.

- Identify how much software it is possible to build by essentially starting and seeing how far it goes in a controlled fashion. The author's own initial lack of domain knowledge about field and technical skill level mean than a fixed feature list is hardly possible. However the progression of features and the final "shape" of the FundFind software is discussed in "Requirements", §2.4.

[1] A note on "scholarship" - the word and its derivatives are used throughout this text to mean all types of academic work, due to "science" being sometimes perceived as exclusionary to the Arts and Humanities fields.

[2] The Open Definition defines "open" well [32].

1.3 Scholarship and Funding

Scholars need money to conduct research. To be precise, research projects need *additional* money (on top of the investigators' salaries) in order to cover material expenses, travel costs, hiring Research Assistants or providing studentships to PhD candidates and other expenses.

In other words, established scholars need to look for funding for aims which may be hard to define, for amounts of money ranging from thousands to millions.

Postgraduate-level scholars (candidates, postgraduates, post-doctoral researchers) are also usually looking for a way to fund the next stage of their scholarly career. (Note: undergraduate-level funding was not considered in this project.)

The quintessential problem is that the information about available funding is published in many different places. Usually each funding organisation publishes its own calls for proposals and announces its own studentships on its own website(s) and mailing list(s).

Keeping up with these sources of information becomes quite difficult - so much so, that it takes a significant portion of research development officers' time. Worse still, this is a significant barrier for those looking into a career in academia - the fragmented information can make it difficult to build a mental picture of what the funding in one's chosen field "looks like". This leads to a feeling of opaqueness instead of transparency and being overwhelmed - instead of having a clear idea of what it takes to succeed in a chosen field. Sometimes it even leads to accusations of cronyism in distributing funding (private conversation as well as [**?**, **?**]).

Furthermore, if this process is that difficult for person working in a specific country, then it can easily be estimated that globalisation of scholarship and scholars' mobility is hindered as well.

These problems seem to stem from the fact that the distribution of scholarly funding seems to have evolved over time from a series of changes to more general money distribution in society. This is actually a well-known problem that the software development field has had to deal with - if multiple stakeholders with distinctive interests try to influence the development of a software system, changing its requirements over time, the results can be disastrous.

There is no easy way to track money - where it gets allocated to, what fields (or even topics) within scholarship get funded more than others. Up until November 2012, there was no publicly accessible *centralised* detailed account of historic funding information in the United Kingdom. The Gateway to Research project being implemented by Research Councils UK finally gives a partial (RCUK-specific) overview of historical data.

However, there is still no mechanism of getting *federated, present, current* data about scholarly funding.

Something which allows searching the (vastly) different, *currently available* streams of funding within academia was thus identified as potentially useful by various people interested in the subject.

1.3.1 The value of Openness in scholarship and elsewhere

The Free Software, Open Source and more recently, Open Knowledge (which Open Access is a part of) movements have all demonstrated that the value of information can be greatly multiplied simply by having more people access it, reuse it and even be creative with it (e.g. visualise or summarise). The Open Knowledge Foundation argues this [35]. Seminal works like "The Cathedral and the Bazaar" [17] have also argued this.

Governments seem to have grasped the benefits as well. The UK Government has mandated that all research funded with public money (which is a lot in the UK - almost everything through the seven Research Councils) should be published as Open Access by 2014 [23]. It has also funded

initiatives such as Open UK governmental data in the form of data.gov.uk [55] and the recent Finch report [2] [57].

The "Open" part of "Open Access" means more than just throwing the data out there in any form and format. The Open Definition [32] has a lot to say about all aspects of Openness. However, principles #1 "Access" and #4 "Absence of Technological Restriction" are relevant, so it can be seen why the current practice of each funder publishing their own HTML pages is not good enough:

- "The work shall be available as a whole [...] "As a whole" prevents the limitation of access by indirect means, for example by only allowing access to a few items of a database at a time."
- "The work shall be available [...] in a convenient and modifiable form. [...] The work must be provided in such a form that there are no technological obstacles [...]. This can be achieved by the provision of the work in an open data format."

In practice, when applied to scholarly output such as publications, these have been observed by the author to mean "PDF-s are not going to cut it", "You must be able to data mine it".

So why not have this for funding data? Currently, separate HTML pages, weekly e-mails and (sometimes) RSS feeds are all that the Research Councils UK do to publish their funding information. RCUK, in particular, are a publicly funded organisation, and it could be argued that this data belongs to the public. In any case, convincing both public and private funders of the benefits of Open funding data is needed.

Private organisations are not far behind and may in fact be leading the way - one of the largest funders of science in the UK and around the world, the Wellcome Trust, has had a progressive and strict Open Access policy since 2006 [10].

1.4 Audience and high-level requirements

The potential audience of this project was discovered to be quite varied. It seems multiple stakeholders stand to benefit from more transparent scholarly funding dissemination.

The requirements or potential uses that these users might want to put FundFind to are closely tied to the benefits that come with opening up funding data.

1.4.1 Professionals Looking For Funding Group

1.4.1.1 Accomplished scholars

- **Who**: Academics - lecturers, professors
- **Aims**: Wish to fund future projects
- **Benefits from opening up funding data**: Easier access and search across funding institutions through tools which aggregate funding data; Transparency and accountability of their funding bodies.
- **Possible requirements from FundFind**: Aforementioned cross-funder search; Sharing opportunities with colleagues.

1.4.1.2 Junior Academics

- **Who**: Postgraduate candidates, postgraduate students, academics in post-doctoral posts (e.g. Research Associates)
- **Aims**: Wish to find suitable positions to move on with their academic career, e.g. a postgraduate with a PhD will be looking for a post-doctoral position
- **Benefits from opening up funding data**: (More) easily find and compare funding sources
- **Possible requirements from FundFind**: Find funding. To quote a postgraduate candidate: "I want to do a taught MSc in Astrophysics but I can't afford to self fund. Where can I go and who will pay me?"

  Share funding information with peers (e.g. if a canididate for a Computer Vision PhD finds a good PhD studentship call for applications in Bioinformatics, they may well want to forward it to a colleague).

1.4.1.3 Research Development Officers

- **Who**: Professionals who find more funding opportunities for scholars
- **Aims**: Wish to record the opportunities they find somewhere and share them with scholars. Also may want to access a list of all opportunities *they* have found over time, so that they can report to their line managers more easily.
- **Benefits from opening up funding data**: Not as many as scholars, unless they are happy to share the opportunities they have found with research officers from other institutions (which they do not always want). However, a system which allows sharing of recorded opportunities with a select subset of users (such as all within the officer's institution) would be welcome as this can allow them to record a funding opportunity once, share it with "their" scholars and include it in reports.
- **Possible requirements from FundFind**: Find funding. Share funding with subset of users. See information submitted per user (themselves).

1.4.2 Funders Group

Those looking to advertise funding opportunities - Funding Stream and Programme managers at various institutions.

The needs of this group are estimated since funders were not engaged as users at this point in FundFind's development due to time constraints.

1.4.2.1 Research Councils in the UK

- **Who**: Main way of allocating *public* scholarly funding in the United Kingdom.
- **Aims**: Wish to attract the best scholars in their own field, but also look to establish new cross-disciplinary projects.
- **Benefits from opening up funding data**: Already committed to opening up historical funding data (e.g. Gateway to Research project [42]), would like to advertise current funding data more widely. The benefits are widening impact and incresing the inter- and intra-field

interconnectedness of scholarship (scholars can learn about interesting funded projects and people that they didn't know about).

- **Possible requirements from FundFind**: Mostly just search and upload of information. In particular: search for submitted items related to their institution. Bulk upload of opportunities. Control over the "funder" profile related to them.

1.4.2.2 Jisc

- **Who**: Used to be known as "Joint Information Systems Committee", but have decided to rebrand [25]. The old name of this public organisation reflects their purpose quite well - helping the adoption of digital technology in research and teaching in the UK.
- **Aims**: Wish to attract talent to work on scholarly infrastructure in the UK, i.e. give their aims and projects more exposure. Scholars rarely choose scholarship infrastructure as their main field.
- **Benefits from opening up funding data**: Have always wanted wider reach, as well as vigorously supported Open Data. Have a lot of funding opportunities, both minor and larger ones. Would like to advertise them better.
- **Possible requirements from FundFind**: The same as RCUK. In addition, however, they might wish to actually use FundFind's data in other projects - in other words, they might be an interested API consumer.

1.4.2.3 Charities, Trusts and Others

- **Who**: Various charitable Foundations, Institutes, Endowments and so on which fund scholarly activities (e.g. the Wellcome Trust [69], Shuttleworth Foundation [48], Knight Foundation [26]).
- **Aims**: Simple - want the best scholars competing for their money (at least the portion dedicated to conventional research).
- **Benefits from opening up funding data**: Need to reach more scholars to build up an ever-improving pool of applicants (in terms of quality). Also, as private institutions, would like to demonstrate greater transparency with their funding dissemination and allocation as part of their efforts to prove they are doing it effectively.
- **Possible requirements from FundFind**: Same as RCUK.

1.4.2.4 Commercial Organisations

- **Who**: All companies which perform R&D activities. Famous UK examples include Rolls-Royce [43] and IBM [18].
- **Aims**: The ones with large enough R&D budgets to get involved with scholars usually invest in whole laboratories and centres. When they do share effort and money with research institutions, they do have to search for the "best scholars" to collaborate with. The smaller ones may have a problem however - they may only need external help with R&D intermittently and thus may not have a "constant" audience of scholars following their every move since they would not be allocating funding all the time.

- **Benefits from opening up funding data**: Greater transparency of funding allocation, potentially greater audience if a federated scholarship funding source like a popular FundFind instance does come into existence at some point.
- **Possible requirements from FundFind**: Same as RCUK.

1.4.3 Analyst group

Those looking to analyse scholarly funding - how it is allocated and/or spent.

A hackday in the middle of March which focussed on the Gateway to Research project provided insight into the usefulness of certain initially planned features and inspired new ideas. The main audience of the hackday was this group. The GtR is an attempt to get the seven UK Research councils to publish historical funding data - who got funded, grant codes, who worked with whom, any related publications [**?**], etcetera via a web interface and a RESTful API. GtR is quite similar to FundFind, except that it relates to *historical*, not current funding information, and of course contains a lot more information than FundFind is currently scoped to hold.

- **Who**: Software developers, journalists and others interested in Open Knowledge and visualising data for the purposes of transparency or advancing the digital economy [35] The author is included in this group. Other interested parties are analysts, bloggers and the general public.
- **Aims**: Want transparency and accountability. May want to visualise how scholarship money is being spent or otherwise help society engage with scholarly spending.
- **Benefits from opening up funding data**: Open Data essential for re-use for their aims. Usually do not have first hand information about what happens behind closed doors when funding is distributed and used by universities. Think that this is strange, since often both funder and fundee are public sector organisations / employees.
- **Possible requirements from FundFind**:

There is another group which belongs here - politicians. They may wish to demonstrate accountability and good decision making. Despite the fact that they are not usually known to readily embrace new technological solutions, they do do it when the advantages (at least from a Public Relations perspective, if no other) become clear. For example, the UK government more or less suddenly embraced Open Access for all publicly funded research [23].

1.4.4 Focusing on certain audience groups

Satisfying the requirements of all user groups is more suitable for a larger, better-resourced project than this one.

This project will focus on the first group of users" meaning "Professionals Looking For Funding Group", §1.4.1. The main reasons are still "time constraints and ease of access to such users". However, the "Analyst group", §1.4.3 was also taken into account. This was partly because the success of a technological project just has to include other developers, especially when it is an open-source project. On the other hand, the author is part of this group and day-to-day interactions create unavoidable bias, especially when they include learning (how to use technologies relevant to FundFind) and feedback on the very project being developed.

As stated previously in the Progress Report, FundFind does have an open machine-ready to communicate with other software - an Application Programmable Interface (API). In terms of concrete people within this group, Cottage Labs LLP [8] still find value in the project and will probably help to take it further after the current development as a piece of course work ends.

In addition to the The Open Knowledge Foundation [35], to whom the project will be presented after it is finished and feedback will be asked for, the Gateway to Research technical team will also be contacted about possibly building a stronger relationship with FundFind. They might be interested in its use of the data provided by their API to make suggestions of interesting projects (to look at) to scholars. Furthermore, they handle *historical* funding data, whereas FundFind handles *current* funding data. Convincing the Research Councils UK to provide the same uniform interface to their *current* funding data as they are to their *historical* data with the GtR project would be a great boon to the Open Knowledge movement.

1.5 Existing works

There is still (as opposed to the start of the project) no single piece or combination of pieces of software which enables involved stakeholders to do what this project would allow them to do upon completion, to the best of the author's knowledge.

Related Open Knowledge projects are still as relevant as at the beginning of the project, in particular the Open Knowledge Foundation ones at [34] and [33]. "Design Details and Rationale", §3.1.1 notes individual projects which informed FundFind's technical design in some way.

The FundFind codebase was initially based entirely on the IDFind project [12], which is available under the MIT license [13]. Cottage Labs LLP and the author started it during the DevXS 2011 conference [50]. IDFind was a good place to start, since the project's context (Open Knowledge, interested parties) is similar, so a similar set up and technologies could be used to accelerate the initial development.

However, IDFind had not done anything like one of the new features which were added during development - harvesting funding data from machine-readable sources. It turned out that an Open Knowledge Foundation project, BibServer, had done something similar. The new Funding Harvest feature is actually technically separate from the rest of FundFind's codebase - it just puts whatever information it finds in FundFind's datastore and the only link between the two codebases is the datastore.

1.6 Licensing

FundFind is licensed under the MIT license, like its parent IDFind.

The reason why FundFind's low coupling with the Funding Harvest feature is stressed above is that BibServer, Funding Harvest's parent, is licensed under the Affero GPL [31]. This requires that Funding Harvest should also be licensed under Affero GPL, as AGPL is a copyleft license. The modular technical design enables FundFind and Funding Harvest to be developed side-by-side, fulfilling user needs without having to change FundFind's permissive open-source license.

Chapter 2

Development Process

2.1 Introduction

2.1.1 Overview

2.1.1.1 The Software Development

An Agile, iterative approach to the development of the software was initially chosen. It turned out that the approach did not fit the project / author / users context. In particular:

1. The initial approach was actually not Agile to begin with (a problem of understanding what Agile actually entails).
2. Agile requires a certain level of understanding of the technology (developer's side) and problem the software is solving ("customers"' or users' side).

The length of the requirements gathering phase and the complexity of developing the most important features was initially underestimated. Analysis of the intermediate outputs and measurement against the initial aims suggested that a different scope was necessary. The new goals required a different approach, but that alone would not provide an answer as to what FundFind should have aimed for while remaining realistic.

Concentrating on and discussing the original goal of the FundFind project with interested parties proved fruitful. The overarching aims of "learning about scholarly funding" and "prototyping a solution for sharing this information" resulted in a leaner, more focussed set of requirements to work towards. Since work on the code had started and a basic web application was running, it was most appropriate to enhance this and peruse the chosen datastore's unique characteristics to add new features while retaining simplicity. This approach in the second phase of the project closely resembled the established Evolutionary Prototyping development process. [**?**]

2.1.2 Project management tools

A project management piece of software suited for Agile approaches was chosen to manage this project and was used successfully throughout: PivotalTracker. The project's progress and a list of features (in priority order) has been publicly available at [53] throughout the project's current lifespan (October 2012 - April 2013).

A private source code version control repository was set up with GitHub [20]. It is only private due to several concerns:

1. The licensing status of the IDFind project which FundFind was forked off, was unclear. In fact, with no license statement anywhere in the repository, the default copyright rules apply and FundFind could not have been legally based on IDFind without explicit written permission from the authors. However, since IDFind's only contributors were Cottage Labs LLP and the author of FundFind, this did not delay FundFind's development. Nevertheless, IDFind's licensing status was not cleared up until April 2013.

2. There were concerns about the quality of FundFind's code and the repository. GitHub is home to enough orphan repositories which are all almost empty or without proper licensing, either of which makes the project useless in terms of its Open Source aspect.

3. It wasn't quite clear whether the code could be released during the official timeline of the final-year Major Project module at the author's institution. One of the main ideas in open-sourcing code is that others can contribute to it, but that would have significantly complicated matters in terms of reviewing the final version of the code. Cottage Labs LLP would like to contribute, but agreed that they are busy enough at the moment to look at the codebase after the 26th April 2013 (by which time a copy of FundFind's codebase would have been handed in).

Furthermore, a private repository holding all of the other outputs of the author's Major Project (Outline Specification, Progress Report, presentation files, this document and so on) was set up purely for back-up purposes.

2.2 Evolution of the Development Process

Initially, Agile practices such as Pair Programming [47], full Scrum [46] and Feature-Driven Development were deemed unsuitable since they are aimed at groups of developers. Behaviour-Driven Development (with Cucumber [3]) was considered, but requires more commitment from the users - they need to learn an English-like domain-specific language which describes how the system is going to behave which didn't sit well with assumption #2 above and turned out to be the right decision as that assumption proved correct.

Thus, initially, with no methodology fitting the precise needs of the project and the Agile Manifesto [6] in mind, a methodology of an Agile *character* was planned, using the Extreme Programming lifecycle as a base [11]. The idea was to use ideas such as Release Planning (plan a set of features), Iteration (develop in small increments) and Stories (described above).

The basic steps which were defined were:

> “
>
> 1. Meet with a variety of potential users from the chosen user group (§1.4.4), trying to pick them so that each one has a different professional perspective on scholarship.
>
> *Emanuil Tolev in FundFind Progress Report [?]* ”

This worked well, with candidate postgraduates, postgraduates, research assistants (post-doctoral), lecturers and developers all agreeing to give a bit of their time. The meetings were not as regular as initially hoped (1 per month with each user) due to users' commitments, author's committments and a particular characteristic of this project.

Looking at requirements presented in §1.4, it is easy to see that the basic searching and sharing of opportunities were two *really* important functions. Sharing is used to different ends by research officers, lecturers, postgraduates and funders, while searching is useful for different reasons to pretty much everybody. However, getting timely feedback from all target users was not considered to be possible - and that is required for iterative development. It was actually considered (and turned out to be - §**??**) difficult enough to produce a prototype which made progress in satisfying the initial requirements of *multiple* groups. Thus, meetings essentially happened at the start at the project (October, November, December) and at the end, in March and April, since most of the time inbetween was spent getting those two important features right. Advanced functionality was planned in March and developed in April and is described below in §2.4.4.

“

> 2. Define *and prioritise* the functionality of the project with each user in the form of the “stories” mentioned above. Record the results using the chosen project management tools (§2.1.2).
>
> *Emanuil Tolev in FundFind Progress Report [?]*

”

This turned out to be a really good idea

“

> 3. Release planning - prepare an initial timeline of which project features are to be released when and denote “release points”. This work can be repeated as required - the order and the features themselves might change on the basis of user feedback over time (this is how Agile projects try to deliver more value by responding to changing requirements). The total size of the work and the size of each feature will most probably remain the same. This piece of work produces the “Project Timeline” and the current version is included in §**??**.
>
> *Emanuil Tolev in FundFind Progress Report [?]*

”

“

> 4. Iterate - every week is a self-contained unit of work consisting of coding, documentation and write-up work. Before starting to code each week, break up that week’s story into tasks so goals can be tracked more effectively and there is something to be accomplished each day.
>
> *Emanuil Tolev in FundFind Progress Report [?]*

”

“

> 5. Repeat items 1 and 2 - meaning regular meetings with previously chosen users - one per user per month, demonstrate the latest features and record the feedback.
>
> *Emanuil Tolev in FundFind Progress Report [?]*

”

Within the coding part of each iteration, Test-Driven Development is employed using both unit and integration tests (the latter via browser automation with Selenium [66]). This will be the evidence (on a basic technical level) for the "correctness" of the software. A basic form of Continuous Integration [28] will also be used - the Jenkins [63] software is currently being evaluated for the job. It basically runs all automated tests continuously, helping to detect regressions in code quality.

This methodology makes for a considerably simpler lifecycle than the Extreme Programming one (a visualisation of which can be accessed at [11]). This is intentional as there are only about 11 (potentially 13) weeks which allow for programming work as can be seen in §**??**.

The length of one iteration will be one week. Considering the tight deliverable deadlines (see §**??**), this will keep work on the project focused and will force the definition of user stories of a manageable size.

Saying "actual features being implemented could change due to the Agile project development methodology" as the Progress Report did does not make the approach Agile, as Agile is not a stick that one uses to prune excessive features at will - it is an approach that the developer sticks to.

2.3 Modifications

In the Progress Report features were planned and *scheduled* in a timeline before development took place, but that is *not* the Agile way at all.

2.3.1 What is Agile?

Agile is about getting work done in an iterative fashion so that there is always working software for the users to use. Estimates in Agile are about effort, complexity and risk. Estimates are also *relative* - when new features are added to the project scope, they are estimated in comparison to previously added ones. (For a commonly used estimation points scale and more elaboration on Agile estimation, see [**?**].) If numeric points are used for the estimation, the project ends up with several hundred / thousand points worth of features.

Since those are prioritised by the customer/users, work can then start. What Agile teams seem to do is to essentially see how quickly they get through work - how many points they go through per week, which is called the "velocity" of the team.

Only then is it possible to estimate how long all of the initially requested features will take to develop - by taking into account their "size" (relative to each other) and the team's abilities. Taking the total project size and dividing it by the weekly velocity gives a rough idea of how many weeks it would take to complete all requested features. Let this be `time_required`.

If `time_available < time_required`, then the scope of the project simply has to be limited, i.e. features have to be dropped, and this only becomes clear once velocity is established (so at least a few weeks after development has started). In other words, however many features there *turns out* to be time for in the development process are *what gets done*, which is why there is so much emphasis on having the users involved - prioritising the functionality themselves.

2.3.2 Is it Agile?

FundFind's Progress Report [**?**] stated that an Agile-like methodology had been chosen - the main concept being user stories which allow capturing user requirements very quickly in an informal way.

Having a list of desirable features is great, having a schedule of which one will be delivered when - not so much. The point of Agile is taking the specifics of the development team and

users' knowledge (and specificity of their desires) into account. A fully pre-defined plan - even one informed by meeting users, such as in this case - disregards all of this and does not conform to Agile's flexibility requirements. (Responding to change better than more conventional approaches is usually one of the first touted benefits of Agile approaches.)

Figure 2.1 presents that initial plan.

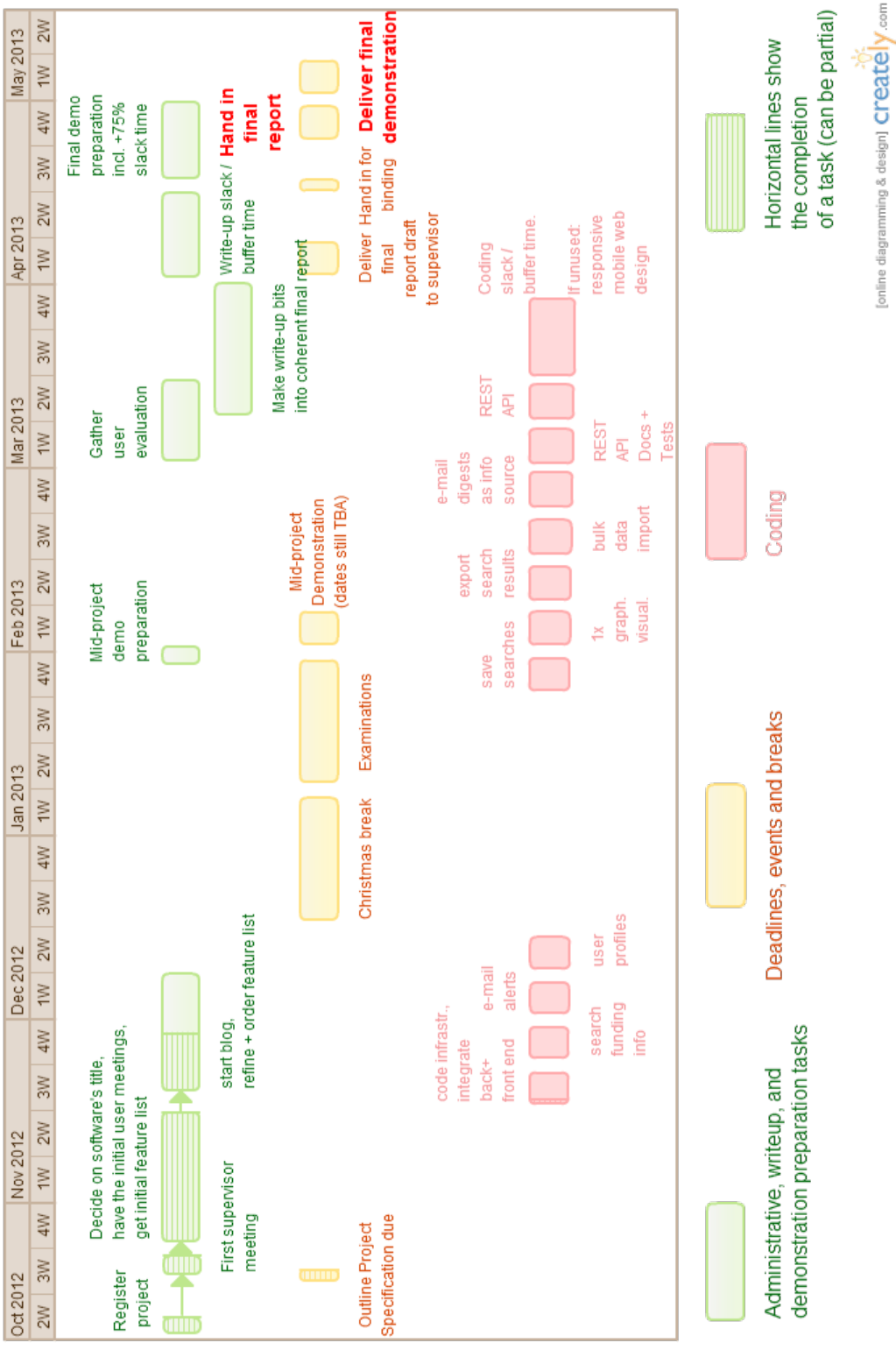

Figure 2.1: FundFind initial timeline

2.3.2.1 Underestimation

The main reasons for choosing Agile were

1. Agile's perceived advantage in pinning down changing requirements.
2. Less formal communication with the users, who are all busy professionals. It was assumed they would not have sufficient time for writing functional specifications, even in collaboration with the author (as can sometimes be assumed in industry).
3. Linked to the user communication reason was a concept which many Agile methodologies used - user stories. These are short sentences (or rarely, paragraphs) which describe a single piece of functionality - a single "thing" that the user wants to do with the system.

However, it turned out that both the complexity of the scholarly funding field and the author's ability to concretise fuzzy requirements, as well as their technical skill had been underestimated (elaborated upon in Chapter **??**). The fact that requirements had to be gathered and concretised is not unusual in Agile, but playing both developer and user was a problem.

Neither did users have enough time to be properly involved throughout the project, nor did the author have enough domain knowledge to understand the requirements and develop quickly enough *in order to get the users involved* in the way in which Agile understands involvement.

This conclusion and the process of arriving at it, ultimately, are the reason why the initial Agile-like development process was deemed unsuitable and why the project's progress is split so starkly into two phases as per Figure **??**.

The initial reasons for devising an Agile-like process (above) proved to (mostly) be steps in the right direction, however:

1. The flexible requirements reason did not work out so well since it turned out the requirements did not change or need to change. Issues with funding data have been present for a while and a system which tries to prototype fixes through Open Data for some of them needs (a) developer(s) who know these issues inside out. Of course, the requirements gathering can still be done in an Agile way, but needs to be mostly completed before development starts. Defining the problem, in this case, had to come before solving it.

   It could be argued that this was only a problem since the developer was also the "initiator" of the project (i.e. had to play user a lot more than usual). If a particular funder, Higher Education Institution or an organisation like the Open Knowledge Foundation wanted a project like this done, they would already know of certain problem characteristics - such as *particular* drawbacks of having so much funding data "closed". Then, more targeted exploration could be done and technical development would also have more precise targets to meet.

   This is, in part, what caused the development process to look a lot more like **evolutionary prototyping** with a lot of requirements gathering between the prototype versions in Figure **??** rather than an Agile iterative approach.

2. The assumption that users won't have time for anything but informal problem specification turned out to be correct. Thus, the "stories" concept proved to play an important role when gathering requirements from the target audience in short meetings, on the go or at hackdays.
3. The project's stories have been publicly accessible at the PivotalTracker management tool continuously throughout the exploration phase of the project which was used to record the

aforementioned informal requirements. The "Icebox" still contains the list of features as prioritised by the various users who have been consulted during the exploration phase.

2.4 Requirements

In this case, the project's requirements (in a suitable share-able form) are actually part of the expected project output, as one of the main aims is to learn about the scholarly funding field and identify needs.

2.4.1 Requirements Gathering

2.4.2 Initial Requirements

The initial requirements were informed by the general needs of the various audience groups ("Audience and high-level requirements", §1.4).

They are listed below alongside changes which were made (kept/dropped/changed/changed priority). Further discussion on the lessons learned and future work enabled by the dropped features is presented in "Future Work", §6.5. Features which went through major changes are discussed below in "New Requirements", §2.4.3.

In **order of importance**:

2.4.2.1 Search For Scholarly Funding Opportunities

- **Why**: It's the implementation of one of the main project goals - making funding information discoverable. Every single audience group needs it in one way or another.
- **Priority rationale**: All users decided it was a top-priority feature, very much the main point of a system like FundFind.
- **Decision**: **Kept as-is**. No information suggested that this had actually become any less important than when the project started, and although the project goals changed when it became clear how much expertise was needed to actually do everything well, this was still an important software feature.

2.4.2.2 Submit Information About A Scholarly Funding Opportunity

- **Why**: In a similar vein to searching for such opportunities, multiple audience groups need it, albeit for different reasons (a funder might want to make their opportunity more discoverable, while a PhD candidate can just share a good opportunity with their peers).
- **Priority rationale**: Core to the crowdsourcing functionality of the project. Actually incentivising users to do this is a different matter entirely. That is part of the point of FundFind though - through providing the capability to share, it gives a reference point to related conversations and feedback, (hopefully) allowing developers to understand why the various users do not want to share funding information.

  Of course, this requires the search functionality to be present to provide great value to the project and is thus still second.
- **Decision**: **Kept as-is**, for the same reasons as the feature above.

2.4.2.3 Submit Information About Funders

- **Why**: At the start of the project there were concerns that the various funding opportunities are vastly different in their purpose, format and thus, the data available about them. In order to alleviate this expected problem, this idea was presented to users and was found to be reasonable.

  In essence, if the funding opportunity data isn't good enough, a catalogue of funders would still be quite valuable. It was presumed to include some information to let the user search for an appropriate funder - like tagging the funders, or describing their interests.

  The idea actually came from a different Open Knowledge-related project called ACat (**A**cademic **Cat**alogue) [**?**] - an attempt to mine all information on all academic publishers, their journals *and articles*. Even though ACat was just a hackday idea and hasn't evolved much since June 2012, it clearly showed the importance and value of having a list as simple as "all UK academic publishers".

- **Priority rationale**: Similar reasons to the feature above for this one to be so high up in the list in terms of crowdsourcing functionality. Its importance is very much tied to submitting information on funding opportunities - the more difficult it is to reconcile data from multiple sources (within the current scope), the more important this becomes.

- **Decision**: **Kept as-is**, for the same reasons as the feature above.

2.4.2.4 Create Profiles

- **Why**: Submissions of data could not be anonymous *to the system* - the submitter has to be authenticated and logged in some way, otherwise the submission forms will likely become the target of spam bots. It also helps with data quality, although admittedly on a very basic level - essentially, if a user is bothered to register, they are less likely to submit random junk into the system.

  This feature could include more than a username and a password however - it could include declaring research interest keywords and affiliations (the latter having been recorded as a separate feature at user meetings - see below).

- **Priority rationale**: Really desirable to enhance the basic functionality described above, but not as vital as actually having said functionality work in the first place.

  Other features which did not fit in the initial timeline at all but were earmarked for inclusion into a "Future work" section also require this (for example, research officers' reports "E-mail Alerts", §6.5.0.3).

- **Decision**: **Kept**. **Not much changed** except "could include more than username and password" became "includes more..." as no particular difficulties arose during the design or implementation of these additional fields.

2.4.2.5 Visualise Data

- **Why**: It was thought that the Analyst audience group "**??**", §**??** would have an interest in this as it might spark other ideas about visualising funding data. Furthermore, scholars themselves might have had use for the result of the analysis - such as knowing how much their field is being funded at the moment.

- **Priority rationale**: Showcased the benefits of opening up funding data, which is one of FundFind's goals.
- **Decision**: **Dropped**. This turned out to be a little ill-defined. It embodied the quintessential software development problem of trying to predict what users want without actually knowing or being able to venture a good guess. The initial reasons as to why a visualisation might be useful are still true, of course, but it turned out developers who had done other Higher Education work understood the benefits of visualising such data quite well. Similarly, scholars could tell how well their field was funded just by looking at the data - the problem being that creating a visualisation that hid data appropriately and thus decreased cognitive load or increased insight was actually quite difficult. Representing the data in some much simpler visual way was not as hard, but also far from useful for people used to interpreting textual and numeric data. These insights were gained mainly during the Gateway to Research hackday ("Analyst group", §1.4.3).

2.4.2.6 E-mail Alerts

- **Why**: Requirements gathering showed users using e-mail alerts / digests to receive the newest calls from funders. Thus, FundFind should probably offer a similar service as it federates such information.
- **Priority rationale**: Simply get users to use FundFind. With it, they would be getting the same information they could elsewhere, so this is just covering for value proposition #2 from "**??**", §**??**.
- **Decision**: **Dropped**. Turned out data - getting data - was more important than the gimmicks of managing data. This is where FundFind's role as a prototype and a limited vehicle of learning came into play.

2.4.2.7 Saving Searches

- **Why**: In a similar fashion to e-mail alerts, this was thought to be important as it allowed for better management of data and reduced repeated search effort.
- **Priority rationale**: Part of value proposition #3 - additional features. Other features such as e-mail alerts actually could be implemented more easily after this one is done - however, users did not think it was such a high priority. It also turned out that searches can already be shared just by copy/pasting the URL of the search page ("Faceted Search", §4.2.1.
- **Decision**: **Dropped**. Similarly to e-mail alerts, it turned out data is the first thing to get into the application and leave enhancements for later.

2.4.2.8 Harvesting Funding Data From E-mail Digests

- **Why**: This is one of the features which actually deal with importing data, fulfilling part of FundFind's core mission of federating funding data.
- **Priority rationale**: The importance of this was downplayed by the author - users did not have much to say on it, since they did not particularly care for the details of how FundFind was going to get its data, as long as it had some and they could use it.

- **Decision**: **Changed**. This is a good idea - however, all the information turned out to be available in machine readable formats (for the funders which were considered). It was also technically quite challenging ("Harvesting Funding Data From E-mail Digests", §4.2.3.1).

2.4.2.9 Tagging Research As "Potentially Of Interest To" Users And Groups

- **Why**: Part of showcasing the benefits of opening up and sharing funding data - enables scholars and research development officers to target the sharing of opportunities. May also enable funder representatives to target their funding calls better, although care has to be taken since the two sides of the "advertisement" may well have different goals.
- **Priority rationale**: This is a nice feature, but builds on profiles, search and submissions working, so just has to be done after them.
- **Decision**: **Kept as-is**.

2.4.2.10 Specify Affiliation And Interests

- **Why**: This is actually the "other end" of tagging research - specifying where the user works (institution, research group, country) and what field the user works in (interests) enables the usage of the tags associated with research data.
- **Priority rationale**: Similarly to tagging research, this feature builds on everything before it, clearly part of value proposition #3.
- **Decision**: **Kept as-is**.

All initial features described below were lower priority or in some way optional in relation to the main aims of the project.

2.4.2.11 Responsive Mobile Web Design

- **Why**: FundFind is all about sharing data, thus it made sense to let people share in as many contexts as possible. The Design Rationale "Design Details and Rationale", §3.1.1 elaborates further on this feature, which is actually a UI design characteristic. It was also of personal technical interest to the author - responsive web user interfaces are everywhere in the Open Knowledge field. Some work better, some worse, yet one thing is clear - learning how to develop them is important. Preliminary research also showed that implementation might not actually be that difficult ("Responsive Mobile Web Design", §4.2.2.1), especially with a more lax testing strategy ("Responsive Mobile Web Design", §5.3.1).
- **Priority rationale**: As technically nice as this is, it was certainly not prioritised highly by users. Therefore, it was given a lower priority than the main functions of the application.
- **Decision**: **Kept as-is**.

2.4.2.12 Bulk Funding Opportunity Data Import

- **Why**: Importing funding information records one-by-one can get quite boring, time-consuming and tiring - all the characteristics of a task that is ready to be automated. This was mainly conceived as a feature for the Analyst audience group ("Analyst group", §1.4.3), specifically

more advanced users who might want to use the software itself and load it up with data they are interested in - not the data all other users have shared on the public FundFind instance. This also includes potential contributors to the code. Another potential target was funder representatives, who may have access to funding opportunity information from their organisation in a structured format and would rather just upload it than copy/paste the records one-by-one.

- **Priority rationale**: This feature has quite a narrow focus - a (very) narrow subset of the audience of the project.
- **Decision**: **Dropped** - it was just really of very limited use. Even backing up the data in the public FundFind instance does not require this feature, as the datastore can be backed up with a simple file copy command ("Datastore", §3.6).

2.4.2.13 Harvesting Historical Funding Information

- **Why**: Historic funding information can be quite valuable since it can grant insight into what was previously funded. If this includes the most recent 1-10 years of data, it could be used to see how fields develop. While hopefully no scholar would choose their field based solely on its history of funding, they could see how to pitch their work most effectively - often, it is difficult to strictly categorise research.

  Something which came up during meetings was the fact that helping gain such insights from historical data might be just as, and sometimes more, helpful to *obtaining* funding than just the sheer ability to *find* it more easily.
- **Priority rationale**: FundFind had to focus on current funding opportunities.
- **Decision**: **Dropped**.

2.4.2.14 RESTful API

- **Why**: Allow integration with other Open Knowledge projects, also vital to being able to consider FundFind an Open Knowledge project. If data is being federated from multiple sources, then it better be exposed in some way. The Design Rationale "Design Details and Rationale", §3.1.1 elaborates further on this feature.
- **Priority rationale**: Targets only the Analyst audience group ("Analyst group", §1.4.3).
- **Decision**: **Kept as-is**.

2.4.2.15 Exporting The Results Of Searches

- **Why**: Simple convenience - being able to save a list of relevant funding opportunities for later perusal.
- **Priority rationale**: Everything else has some function beyond pure convenience and this would not save that much time or effort in any case. Similar to saving searches ("Saving Searches", §2.4.2.7), searches can already be shared via the search page URL ("Faceted Search", §4.2.1) which further reduced the priority of this feature, since the same search should yield quite similar results (based on the assumption that a search worth saving is a pretty specific search). This was also consistently rated as low priority in user meetings.

- **Decision**: **Dropped**. It is hard enough to find *one* highly relevant funding opportunity - finding multiple ones is not something that will happen easily without getting quite a lot of data into FundFind.

2.4.3 New Requirements

A number of new features were inserted into the initial list during development - mostly stemming from new knowledge about related projects and data, or what was technically feasible.

2.4.3.1 Harvest Funding Data From Machine-Readable Sources

- **Why**: It is far easier to process RSS feeds such as the EPSRC Open Calls RSS Feed [14] than it was to process the EPSRC Funding Call E-mail Alerts, as discussed in "Harvesting Funding Data From E-mail Digests", §4.2.3.1.
- **Priority**: Replaced "Harvesting Funding Data From E-mail Digests", §2.4.2.8.

2.4.3.2 Suggesting Relevant Historical Data

- **Why**: As discussed in "Harvesting Historical Funding Information", §2.4.2.13 above, it was discovered that this is actually a good way to achieve the end goal of funding scholarship. This project may have started as being about discovering *current* funding information, but this itself stemmed from the overarching goal getting more funds to more scholars. This only became possible quite late in the development process as Gateway to Research's data was discovered by the author in March and gave a concrete foundation on which to build this feature.
- **Priority**: Lowest, tacked onto the end of the feature list. Very experimental due to potential data issues, lack of discussion with actual scholars (only other developers) and lack of clarity around presenting the information on the user interface. It was also technically quite ill-defined and difficult to achieve.

2.4.4 Final Requirements

1. Search For Scholarly Funding Opportunities
2. Submit Information About A Scholarly Funding Opportunity
3. Submit Information About Funders
4. Create Profiles
5. Harvest Funding Data From Machine-Readable Sources
6. Tagging Research As "Potentially Of Interest To" Users And Groups
7. Specify Affiliation And Interests
8. Responsive Mobile Web Design
9. RESTful API
10. Suggesting Relevant Historical Data

2.5 Design Process

2.6 Techologies used

As "Existing works", §1.5 and "Design Details and Rationale", §3.1.1 point out, Python was chosen as the main programming language of the application. HTML5, CSS and Javascript were used for the user interface. Similarly to the Python choice, this was due to multiple other Open Knowledge projects using them in this manner - so it would be easy for interested developers to read the code and contribute. Previous personal experience with these technologies helped as well, as this is a sole developer project.

Basing the main application on another project that the author had participated in meant that a great deal of bootstrapping time was saved. A lot of knowledge about using the underlying Python frameworks was also reused. This was also partially true when the FundFind data harvester was written as it was again based on another Open Knowledge project, but the author had no previous experience with that code.

On the other hand, trying to create something to the standards enforced by other projects in the same context (Open Knowledge) forced quite a lot of reading that might not have been strictly necessary outside this context.

Python's interactive (REPL - Read, Evaluate, Print, Loop) interpreter was of great help to the development process. Instead of having to write a program in a file and then run it, the developer can type out the rough sequence of commands they want and see the results immediately. Seeing the results of a *previous* command can be very important if the developer does not know or has just forgotten exactly what its output will be or look like. If the interpreter is run in the project's directory, all modules and other existing project bits can be accessed in the same way as in a written script.

Python's philosophy of readability (no semicolons, indentation matters, new line means new statement) and certain programming conventions which have been built up around it meant that the parent codebase was quite easy to understand.

Previous experience combined with the usage of relevant libraries meant that the quirks in developing with HTML and CSS - mainly the way CSS applies to HTML elements - did not delay development. On the other hand, having to learn how to work with the libraries certainly took some extra time, but saved a lot more ("User Interface", §4.1).

The RESTful JSON API that the chosen datastore (elasticsearch) provides also accelerated the process somewhat, since the all of the project's data could be retrieved by pointing a web browser at `http://localhost:9200/fundfind/_search?q=*`.

The command-line **V**i **IM**proved (vim) editor was used in conjunction with multiple graphical console windows (Konsole on KDE, Ubuntu GNU/Linux). This was not because of particular familiarity with vim, but rather because of a desire to gain such familiarity. It is also the only editor available on the testing server which currently hosts FundFind's public instance. Certain common text operations did actually become faster to accomplish in vim than in a conventional WYSIWYG editor over time (deleting and copying bits, lines or blocks of code being the most noticeable). On the other hand, quite a lot of time went into actually learning to use the new editor.

Chapter 3

Design

3.1 Introduction

The technical design of an application is, in essence, the intersection of current "best practice" in the software development field (and the related programming language communities), its target domains and the user needs.

"Technically", FundFind:

- is a web application
- is written in Python
- stores its data in an asynchronous NoSQL data store with built-in search capabilities
- has a mobile-friendly web interface
- exposes all its major features via an API

These characteristics have not changed since the start of the project. The technical decisions that stand behind them proved to be mostly correct and helpful, and are elaborated upon in the next section.

3.1.1 Design Details and Rationale

FundFind

- is a web application

  This was decided right from the start of the project. The way current Open Knowledge Foundation and Cottage Labs projects worked was important since one of the two major target domains is Open Data / Knowledge and Cottage Labs is an interested party. Most OKFN projects are web applications - even the library ones have mostly been used in web applications. Cottage Labs have also focussed on web applications. Such applications have just proven themselves to be far more "co-operative" than traditional desktop applications - easy to access, easy to modify so that they expose their data via an API (interoperability). §3.8.2 details further interoperability decisions - the transport format (JSON) and protocol (HTTP).

- is written in Python

  Similar factors to the interoperability point raised above played a role in deciding the language of the application. A lot of Open Knowledge projects use Python [**?**, **?**, **?**, **?**] - the same goes for Cottage Labs projects [**?**, **?**, **?**, **?**, **?**, **?**, **?**]. The language is also simple and strives encourage and enable readability - "code is read much more often than it is written" [**?**].

- stores its data in an asynchronous NoSQL data store with built-in search capabilities

  FundFind relies on elasticsearch to store its data. Elasticsearch is an indexing server which allows for sophisticated search queries against a body of text and other data [61]. The essential advantages were simplicity, performance, usage by other Cottage Labs projects and certain options it leaves open for further development (see §3.6).

- has a mobile-friendly web interface

  FundFind is an application which tries to enable sharing of information - one of the highest priority features. Owing to the nature of the information it makes sense to try to make it mobile-friendly - scholars do not necessarily have to hear about funding opportunities while sitting at their desks.

  Whether they will also want to share this rather dense information while on the go is another matter entirely. Mobile-friendliness was not noted as having particularly high priority in the Progress Report, and it still does not - it was just easy to implement due to the fact that the libraries used to make the web user interface follow current best software engineering practice. More details are available in §4.1. Essentially, making a good UI by using the libraries properly would have led to a mobile-friendly UI (at the flick of a filename and a few CSS class names).

- exposes all its major features via an API

  The JISC Report "Advantages of APIs" states "the API enables use and re-use; it is a tool by which we can disseminate knowledge" [**?**]. This is the essential advantage of API-s and the reason the concept of an API has become one of the building blocks of the Open Knowledge movement. If FundFind aims to prototype a useful tool which will eventually contribute to Open Data (so taking into account the needs of the "analytical" audience group from §**??**), it needs to eventually have an API. It is simply better to design the project with the API in mind instead of tacking it on later. One example is the HiFi project, which had the following to say on the subject:

  > "We built the API to satisfy these requirements first, then we built our app on top of the API. This turned out to be a great idea. We got to dogfood our API for the entire development process and it made testing a lot simpler."
  > *Kris Jordan in "First we built an API, then we built a CMS" [27]*

3.2 Overall Architecture

All of the entities in the diagram are modules - collections of code related to solving the same problem. The contents of the modules - the details of the technical design of the main application - are discussed below in §3.3 to §3.5.

Full listing of all files with notes is available in "File Listings", Appendix B.

3.2.1 Organising code in Python

Traditionally Object-Oriented Programming talks about "classes" as, essentially, blueprints combining data structures and imperative code (methods) together. These can then be "instantiated" to produce a concrete "object".

This is of course well-supported in Python. However, Python also has an additional convention when it comes to organising code - "modules". The Python documentation says "You may also want to use a handy function that you've written in several programs without copying its definition into each program." [**?**]. While re-use of code is usually achieved by making everything an object in other languages like Java, Python actually allows functions and even just code statements at the top level of a source code file. Thus, instead of having to create a "Util" class with several static methods, Python authors are encouraged to simply define the functions they need at the module level. In other words, modules are a convenient way to bundle together *related* pieces of code - whether that would just be a sequence of statements, function or class definitions. This may seem messy but has proven to be quite efficient by essentially allowing the developer to better express their mental model of how their program should be organised, instead of forcing a particular structure.

An obvious characteristic of Python modules is that they are also implicit namespaces. Thus, the built-in `int()` can easily be distinguished from `random.int()`. A Python program would have to `import random` before it can use `random.int()`.

In this case, `random` happens to be a module that is part of the standard Python distribution. However, exactly the same rules apply to user-defined modules. Thus, the dotted arrows which connect the modules in Figure **??** essentially mean "module X imports module Y and uses something from Y".

Packages are simply a collection of modules. Since a module is a file, packages are just directories (containing a possibly empty `__init__.py` file) - another way to organise related code.

To sum up, packages contain modules. Modules could contain classes (alongside any other code and function definitions).

3.2.2 Changes over time

The modules which form the main web application have not changed much from the parent IDFind project. An overview of IDFind's technical design presented in the Progress Report [**?**] had actually left out the `util` module and mistakenly used the term "class" to refer to certain modules.

This diagram is reproduced in Figure 3.1. Since it was supposed to be a rough overview, another IDFind module was left out - the `tweetlisten` module allows IDFind to respond to Twitter requests in a similar fashion as if the user was using the web user interface. This module was scrapped in FundFind since potential target users (including the author) just could not think of a way in which Twitter integration would be immediately useful. On the other hand, writing any sensible test coverage for an external service would have taken up valuable time which was used for more important features.

This raises another important point - usually, tests can be inherited alongside features from parent codebases. However, IDFind had no automated tests, making for the decision to actually scrap the Twitter integration instead of just leaving it in the FundFind codebase due to FundFind's more comprehensive testing strategy. This is elaborated upon in §5.1.

In terms of further changes, the Harvesters package was added

Finally, the contents of all of thes

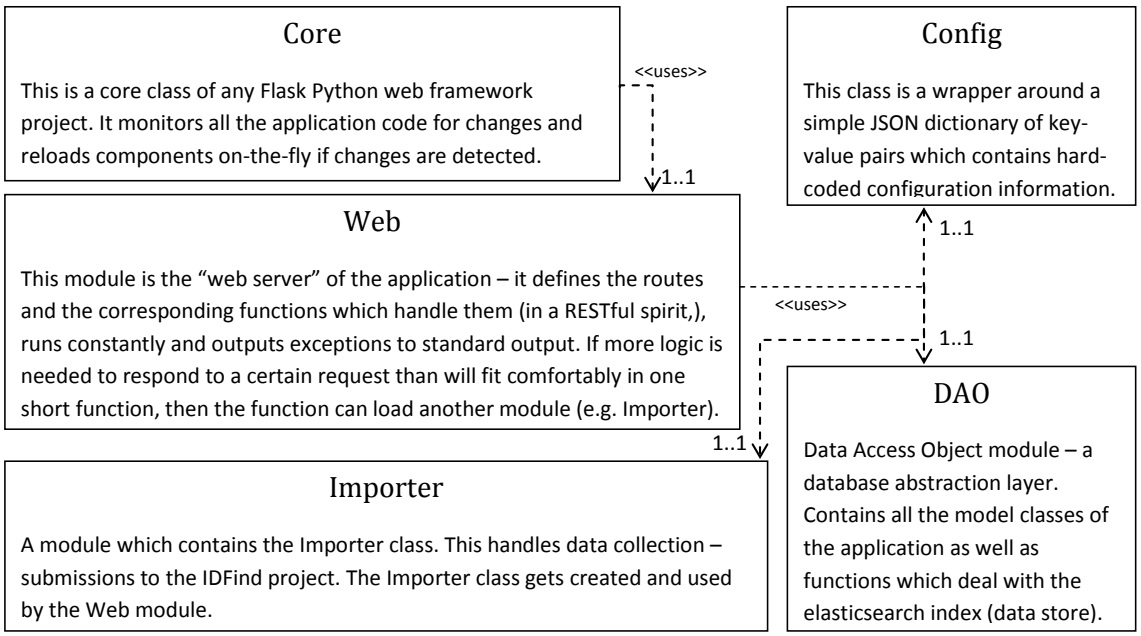

Figure 3.1: Overview of parent project IDFind's technical design presented in the Progress Report [?]

Figure 3.1 was intended only as an overview and had intentionally left out

3.3 fundfind

3.3.1 Core

This is a core class of any Flask Python web framework project. It monitors all application code for changes and reloads components on-the-fly if changes are detected.

3.3.2 DAO

Data Access Object module – a database abstraction layer. Contains all the model classes of the application as well as functions which deal with the elasticsearch index (data store).

3.3.3 Web

This module is the "web server" of the application – it defines the routes and the corresponding functions which handle them (in a RESTful way), runs constantly and outputs exceptions to standard output. If more logic is needed to respond to a certain request than will fit comfortably in one short function, then the function can load another module (e.g. Importer).

3.3.4 Importer

A module which contains the Importer class. This handles data collection – submissions to the IDFind project. The Importer class gets created and used by the Web module.

3.3.5 Config

This class is a wrapper around a simple JSON dictionary of key-value pairs which contains hard-coded configuration information.

3.3.5.1 Default Settings

3.3.6 Util

3.3.7 TweetBot

3.4 Suggestions

3.5 harvesters

3.5.1 HarvesterBase

3.5.2 RssHarvesterBase

3.5.2.1 BbsrcRssHarvester

3.5.2.2 EsrcRssHarvester

3.6 Datastore

Indexing software is usually most easily understood in comparison to more traditional approaches of storing data, like relational databases (MySQL [**?**], Oracle [**?**]). These store data in a highly structured way and the base structural unit is a table (theoretically equivalent to a set, which is where relational theory and such databases come from). The "indexing" part actually means "analysis" in the broadest sense. Given data - any data, images, text, Microsoft Word binary blobs pretending to be text - the software will try to find common features or look for certain characteristics in the data. For example, it will try to discern whether a particular string actually represent a datetime value.

Elasticsearch is actually a NoSQL datastore - which essentially means that it deals with very simple key-value pairs as the basic unit of organising data *instead of* tables, or sets. It also supports slightly more complex, but still very simple structures, such as lists and dictionaries (a.k.a. HashMaps). This means that the storage representation is also quite simple - in this case, all documents stored within an elasticsearch instance can be represented in the JSON data format. This simplicity enables elasticsearch to deal with semistructured data much more easily than a traditional relational database.

A lot of funding data can be considered semistructured - one type of funding call might have a closing date, another one might not. When multiple data sources are considered it becomes even more difficult since different organisations publish different bits of data about their available funding. Since elasticsearch combines its simple storage with a powerful, fast search engine this makes such "holes" in the data a lot less important than usual - the "usefulness" of the data scales with its quality (there is no getting around the fact that less holes is better), but *the software can deal with it* without throwing null pointer exceptions and *the data that is present is still easily discoverable*.

Elasticsearch's RESTful JSON-returning API usually enables rapid prototyping, evaluation and integration with Javascript visualisation libraries such as d3 [60] and BubbleTree [36]. While these

precise features are not implemented in this version of FundFind, the decision to use elasticsearch leaves this option open.

3.7 User Interface

3.7.1 AJAX functionality

3.8 API

3.8.1 Design for Developers

In addition to the functionality exposed through the routes described above, the AJAX functionality (§3.7.1) can also be used by API consumers. For example, the developers of a hypothetical "News for Scholars" newsfeed reader application might want to suggest previously funded bids to its UK readers. They have two options:

1. Re-use the classes within the Suggest module which enable FundFind's "this may be of interest" feature. If the hypothetical application is written in Python, this will just involve copying + editing the code and perhaps reading the Gateway to Research API documentation (for the UK information). If it is written in another language, "re-use" will mean looking at the logic contained within the

3.8.2 Technical design

3.8.2.1 Data format

3.9 Alternative Designs

Chapter 4

Implementation

4.1 User Interface

4.2 Implementing The Main Application

4.2.1 Faceted Search

4.2.2 The Final Features

4.2.2.1 Responsive Mobile Web Design

4.2.3 The Dropped Features

The decision to drop some features came after some design and exploratory technical work had been done.

4.2.3.1 Harvesting Funding Data From E-mail Digests

Chapter 5

Testing

5.1 Overall Approach to Testing

5.2 Automated Testing

5.2.1 Unit Tests

5.2.2 Functional / Integration Tests

5.2.3 User Interface Testing

5.2.4 Stress Testing

5.3 User Testing

5.3.1 Responsive Mobile Web Design

Chapter 6

Evaluation

6.1 Approaching the field of scholarly funding

Somewhere in February, the 4th (out of 6) month of the project, I realised that this domain of knowledge, scholarly funding, was going to be a tougher nut to crack than I had thought.

decided late that the project was going to do requirements gathering as a formalised goal - inspired both by G4HE (Cottage Labs project) and difficulties in understanding the field §6.1. These only became clear once the technical work started in earnest and it turned out it was in fact both a pretty big field, and one that was ripe with problems and conflicting interests.

TODO FIXME EXPAND THIS HERE INTO A SUBSECTION ABOUT WHAT LEARNED ABOUT FIELD. Regardless of personal interest, it takes time to develop holistic understanding of a field like this. One analogy would be another human field of endeavour the author has a personal interest in - saving and re-homing homeless animals. Ripe with conflicting interests with a troubled history of ad-hoc evolution resulting from the intersection of these interests. FIXME TODO EXPAND HERE exactly what are the problems? Politics, money and the sincere wish to save every animal encountered, the good intentions sometimes resulting in bad outcomes such as houses full of tens of animals which cannot be taken care of.

Relating information about the field and its problems to people outside the target audience and even across target user groups turned out to be much more difficult than expected. TODO more here?

6.1.1 Condensing the insights gained

Part of the point of the project was to see how the funding field worked. And this went fine, I now know a lot more about it than when I started. However, sharing this information seems really difficult. It feels like my sample was way too small and that any representation I choose for it will just be laughably bad. In reality, there is clearly a place in the world for a "how does scholarly funding work for beginners, from a beginner" collection of information, but presenting this appropriately seems incredibly difficult.

All the insights gained were attained through interviews. However, it was just my opinion of the content of these meetings that was actually going to make it into the final output. This can feel quite daunting - as a developer, I prefer to publish the data and let those whose work I support actually "deal" with it. However, in this case, I *am* the researcher, this mysteriously skilled person who somehow knows what to do with this data. Others do not want to know about conversations I have had, they want knowledge - even understanding. My understanding.

6.2 Collecting Information From Disparate Sources - Technological Suitability

6.3 User Needs

6.3.1 Identifying Requirements

Far too much thought seems to have been given to the priority of features - at the end it's just a list of functions and they're all desirable. There was not a single feature which was later identified as a "bad" idea. Of course, they can be roughly grouped according to what users seemed to want and this is very important in guiding future development, but in hindsight, there does not seem to be much value in the detailed prioritisation presented in "Initial Requirements", §2.4.2. With rough groups such as "vital", "desirable", "low priority", the prioritisation process could be simplified significantly and there would be relatively few features that would be difficult to categorise.

It was still sensible to categorise new features which came up during development, since the feature list was not really fixed until development had ended.

6.3.2 Meeting Needs

6.3.2.1 Those Looking To Analyse Scholarly Funding

6.3.2.2 Exploring Scholarly Funding

Within the group of people with analytical needs, there was a sub-group of developers which inspired and was part of the reason the project was conceived - Cottage Labs LLP, the Open Knowledge Foundation and the author. It was always the project's intention to learn about scholarly funding specifically so that these people could have a better idea of how to meet the Higher Education sector's needs.

6.4 In Retrospect

6.5 Future Work

6.5.0.3 E-mail Alerts

This is a good feature and it's great that it was identified, especially since the author was not previously aware of the current scholars' workflow.

6.5.0.4 Exporting The Results Of Searches

Nonetheless, research development officers' reports mentioned in "Research Development Officers", §1.4.1.3 could benefit from this feature, so it was added to Future Work, "Future Work", §6.5.

6.6 Learning

Appendices

Appendix A

Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

Appendix B

File Listings

This Appendix provides a listing of all files in the source code repositories of `fundfind`, the main web application, and `fundingharvest`, the additional funding data harvesting project.

Its aim is to aid source code review and deeper understanding of the projects' technical details.

2.1 fundfind

2.2 fundingharvest

Annotated Bibliography

[1] AHRC, "AHRC Funding Opportunities," http://www.ahrc.ac.uk/Funding-Opportunities/Pages/Funding-Opportunities.aspx, accessed November 2012.

The root funding page of AHRC, the Arts and Humanities Research Council.

[2] Alok Jha, "Open access is the future of academic publishing, says Finch report," http://www.guardian.co.uk/science/2012/jun/19/open-access-academic-publishing-finch-report, June 2012, accessed November 2012.

A Guardian article describing the Finch report which looked into Open Access and its benefits for academic publishing and society at large.

[3] Aslak Hellesoy, "Cucumber - Making BDD fun," https://cukes.info/, accessed April 2013.

Cucumber is a system for Behaviour-Driven Development, which was considered as one of the options for development methodologies for this project.

[4] O. Balci, W. S. Gilley, R. J. Adams, E. Tunar, and N. D. Barnette, "The Spiral Model," http://courses.cs.vt.edu/csonline/SE/Lessons/Spiral/index.html, accessed November 2012.

A textbook section-length explanation of the Spiral software development model with a graphic visualisation and a table enumerating the various artefacts of the model. Although this an online educational resource from an academic institution, there is no publication date available for the project itself or the section on the Spiral model.

[5] BBSRC, "BBSRC Research Funding," http://www.bbsrc.ac.uk/funding/funding-index.aspx, accessed November 2012.

The root funding page of BBSRC, the Biotechnology and Biological Sciences Research Council.

[6] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for Agile Software Development," http://agilemanifesto.org/, accessed November 2012.

A succinct set of statements enumerating the core values behind the Agile software development collection of approaches to software development.

[7] Browshot, "Browshot - Powerful web screenshots," https://browshot.com/, accessed November 2012.

Homepage of the Browshot service. This allows a developer to make screenshots of web pages in any screen size, as any device including Personal Computers and mobile devices, in a variety of resolutions.

[8] Cottage Labs LLP, "cottage labs - we don't sell products, we trade our skills," http://cottagelabs.com/, accessed November 2012.

The homepage of a partnership of freelance software developers working together to produce open source software for the benefit of Higher Education and commercial companies around the world. Dedicated to Open Knowledge and Open Scholarship.

[9] Cottage Labs LLP and Open Knowledge Foundation, "FacetView - Pure Javascript Frontend for SOLR and ElasticSearch," http://okfnlabs.org/facetview/, accessed November 2012.

Homepage of the Facetview project, a faceted Javascript browser (front-end) to indexing servers such as elasticsearch.

[10] Craig Brierley, "Wellcome Trust strengthens its open access policy," http://www.wellcome.ac.uk/News/Media-office/Press-releases/2012/WTVM055745.htm, June 2012, accessed November 2012.

A press release by the Wellcome Trust stating it is strengthening its Open Access policy and related punishment for breaking the terms of the policy, as well as linking to a full version of the policy.

[11] Don Wells, "Extreme Programming Project," http://www.extremeprogramming.org/map/project.html, accessed November 2012.

A diagram showcasing the Extreme Programming (Agile software development practice) project lifecycle. Different sections of the diagram provide links to more detailed explanations of the various parts of the project lifecycle.

[12] Emanuil Tolev and Cottage Labs LLP, "ID Find - an identifier identifier," http://test.cottagelabs.com/idfind/, accessed April 2013.

The current home page and public instance of the IDFind project.

[13] ——, "IDFind - Got an ID? Not sure what it is? What you can do with it? Well, use this!" https://github.com/CottageLabs/idfind/, accessed April 2013.

The version control repository page of the IDFind project. It tries to guess what kind of identifier an unknown string is.

[14] Engineering and Physical Sciences Research Council, "EPSRC Open Calls RSS Feed," http://www.epsrc.ac.uk/funding/calls/open/pages/rss2.aspx, accessed April 2013.

A machine-readable RSS feed of current funding opportunities at the EPSRC, one of the UK Research Councils.

[15] ——, "Grants on the Web," http://gow.epsrc.ac.uk/NGBODefault.aspx, accessed April 2013.

This is a service on the web run by EPSRC UK which provides information about research, training and public engagement grants. Essentially, it provides historical information about what was funded and has some nice functionality, such as breakdown by field funded, number of funded proposals within that field and several simple quarterly statistics.

[16] EPSRC, "EPSRC Research Funding," http://www.epsrc.ac.uk/funding/Pages/default.aspx, accessed November 2012.

The root funding page of EPSRC, the Engineering and Physical Sciences Research Council.

[17] Eric S. Raymond, "The Cathedral and the Bazaar," http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/, Sept. 2000, accessed November 2012.

A discussion on why / how JSON maps better to intuitive human understanding of the world than XML does and why this is important for a data representation format.

[18] FAST, "Organisation Details - IBM UK Research Laboratories," http://www.fastuk.org/atcommunity/orgview.php?id=3073, accessed April 2013.

"The Foundation for Assistive Technology was founded in 1998 to tackle the inadequate design of assistive technology products and services." This listing of IBM as a "source of funding" in their R&D section seemed as an appropriate reference to IBM Research UK as a commercial R&D vendor in the context of the project.

[19] Gerrard Consulting, "The W-Model," http://www.gerrardconsulting.com/?q=node/531, accessed November 2012.

Explains the W software development lifecycle model and how it follows on from (or has evolved from) the V-model. This model could also be referred to as the double-V model.

[20] GitHub Inc, "GitHub - Homepage," https://github.com/, accessed November 2012.

Currently popular decentralised version control system, usually used for software source code (but suitable for any artefacts although less suitable for binary formats or large files).

[21] Google Inc., "Google Maps," https://maps.google.co.uk/, accessed November 2012.

A geographical mapping service by a commercial company.

[22] Google Inc. and various authors - an open-source project, "Google Refine, a power tool for working with messy data (formerly Freebase Gridworks)," http://code.google.com/p/google-refine/, accessed November 2012.

A piece of open-source software for handling large semistructured datasets, making en-masse changes to them and cleaning them up. Google has now relinquished control and ownership of the project to the community, it is now the OpenRefine project.

[23] Ian Sample, "Free access to British scientific research within two years," http://www.guardian.co.uk/science/2012/jul/15/free-access-british-scientific-research, July 2012, accessed April 2013.

An article in the electronic edition in the Guardian newspaper detailing the UK Government's policy on Open Access publications in 2014.

[24] Jisc, "Jisc - We are the UK's expert on information and digital technologies for education and research," http://www.jisc.ac.uk/, accessed April 2013.

A UK governmental institution focused on enhancing the usage of digital technology in education and research in the UK.

[25] ——, "Our history : JISC," http://www.jisc.ac.uk/, accessed April 2013.

History page of Jisc. JISC have rebranded as Jisc.

[26] Knight Foundation, "Knight Foundation - Informed & Engaged Communities," http://www.knightfoundation.org/, accessed November 2012.

The homepage of a global charitable foundation which supports (financially and otherwise) ideas that aim to change the world. They are particularly interested in quality journalism, media innovation, engaging the various communities around the world / in society and in fostering the arts.

[27] Kris Jordan, "First we built an API, then we built a CMS," http://www.gethifi.com/blog/first-we-built-an-api-then-we-built-a-cms, Dec. 2010, accessed April 2013.

An article by a HiFi company developer detailing why they built their Application Programming Interface first, right from the start.

[28] Martin Fowler, "Continuous Integration," http://martinfowler.com/articles/continuousIntegration.html, May 2006, accessed November 2012.

An article describing the principle of Continuous Integration in software development and illustrating why a developer or a development team might consider adopting this practice.

[29] NHMRC / Australian Government, "NHMRC research funding datasets 1990 - 2012," http://www.nhmrc.gov.au/grants/research-funding-statistics-and-data/funding-datasets, accessed November 2012.

The NHMRC research funding data for the period 1990 to 2012 covering all new and continuing grants funded and active in that 23 year period.

[30] Open Knowledge Foundation, "BibServer," http://bibserver.org/, accessed April 2013.

Home page of the BibServer project. BibServer is an open-source RESTful bibliographic data server. BibServer makes it easy to create and manage collections of bibliographic records such as reading lists, publication lists and even complete library catalogs.

[31] ——, "BibServer at GitHub," https://github.com/okfn/bibserver, accessed April 2013.

This is the source code of the BibServer [30] project.

[32] ——, "Open Definition v.1.1," http://opendefinition.org/okd/, accessed April 2013.

The Open Definition - defines what "open" is in relation to content and data.

[33] ——, "Open Knowledge Foundation Github Organisation Page," https://github.com/okfn, accessed November 2012.

The homepage of the Open Knowledge Foundation organisation on Github, the currently popular version control system.

[34] ——, "Open Knowledge Labs creates services and tools that unlock the digital commons for everyone." http://okfnlabs.org/, accessed November 2012.

The homepage of a collection of sofware projects which have sprung up to support or resulted from the Open Knowledge Foundation's pursuit of spreading the Openness cause in society.

[35] ——, "Our Vision / We Believe in the Power of Openness," http://okfn.org/about/vision/, accessed November 2012.

Notes on the Open Knowledge Foundation's vision of how openness can be beneficial for society (meaning all of current human civilisation).

[36] ——, "Radial Bubble Tree Visualization," https://github.com/okfn/bubbletree, accessed November 2012.

Homepage of BubbleTree, a higher-level Javascript graphical visualisation library.

[37] ——, "Recline.js - relax with your data," http://okfnlabs.org/recline/, accessed November 2012.

A library for building applications which handle potentially large amounts of the data and intend to allow and focus on more or less advanced data manipulation.

[38] ——, "Where Does My Money Go? - Showing where your taxes get spent - Country & Regional analysis," http://wheredoesmymoneygo.org/bubbletree-map.html, accessed November 2012.

A BubbleTree visualisation of UK public spending.

[39] S. Palmer, "An Introduction to Feature-Driven Development," http://agile.dzone.com/articles/introduction-feature-driven, Nov. 2009, accessed November 2012.

A good introductory overview of what the Feature-Driven software development methodology is about, why it is considered an Agile methodology and when it is applicable (it evolved as an Agile approach for larger projects).

[40] RCUK, "RCUK India - Opportunities provided by UK Research Councils," http://www.rcuk.ac.uk/international/Offices/OfficeinIndia/Funding/Pages/Home.aspx, accessed November 2012.

The root funding page of RCUK / International Office / India.

[41] Research Councils UK, "Welcome to Research Councils UK - Excellence with Impact," http://www.rcuk.ac.uk/Pages/Home.aspx, accessed November 2012.

The home page of Research Councils UK.

[42] Research Councils United Kingdom, "Gateway to Research," http://www.rcuk.ac.uk/research/Pages/gtr.aspx, accessed April 2013.

A description of the Gateway to Research project by its parent organisation, Research Councils UK.

[43] Rolls-Royce plc, "Research - Rolls-Royce," http://www.rolls-royce.com/about/technology/research_programmes/, accessed April 2013.

Rolls-Royce is a commercial company which sports extensive R&D. This page details Research Programmes in a wide variety of (technical and scientific) sectors.

[44] Ron Knott, "Fibonacci Numbers and the Golden Section," http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fibonacci/fib.html, June 2012, accessed November 2012.

An explanation of the Fibonacci number sequence and related mathematical concepts. The Fibonacci sequence is being used as the complexity estimate for each user story in this Agile-managed project.

[45] Scott Ambler + Associates, "Introduction to User Stories," http://www.agilemodeling.com/artifacts/userStory.htm, accessed November 2012.

A explanation of what "stories" or "user stories" are when the term is used in the context of Agile approaches to software development.

[46] Scrum Alliance Inc., "Scrum Is an Innovative Approach to Getting Work Done," http://scrumalliance.org/pages/what_is_scrum, accessed November 2012.

A succinct description of the Scrum framework of Agile software development practices.

[47] J. Shore, "The Art of Agile Development," http://www.jamesshore.com/Agile-Book/pair_programming.html, Feb. 2010, accessed November 2012.

The online edition of a famous work (book) on Agile Development. This section focuses on and provides a good description of Pair Programming, an Agile software development practice.

[48] Shuttleworth Foundation, "Shuttleworth Foundation - Supporting exceptional people to change the world," http://www.shuttleworthfoundation.org/, accessed November 2012.

The homepage of a global charitable foundation which funds individuals with its scholarships and then multiplies the money they invest in projects which aim to bring about social change for the better.

[49] SoftDevTeam, "V-shaped lifecycle model," http://www.softdevteam.com/V-shaped-lifecycle.asp, accessed November 2012.

Explains the V-shaped software development lifecycle model. Also has a nice succinct summary of advantages and disadvantages compared to other models (mostly Waterfall) which seems to fit previous knowledge I have of this model from less reliable sources such as Wikipedia and more reliable ones such as the teaching accompanying my Computer Science degree at Aberystwyth University.

[50] Team DevXS; hello@devxs.org . University of Lincoln, Student as a Producer project., "DevXS Conference," http://devxs.org, accessed April 2013.

DevXS was a national student developer conference running from 11th to 13th November 2011 at the University of Lincoln.

[51] The Document Foundation, "LibreOffice Calc - a Spreadsheet That Meets Any Need," http://www.libreoffice.org/features/calc/, accessed November 2012.

A piece of open-source spreadsheet viewing and editing software, part of the LibreOffice office productivity suite.

[52] The Smart Method Ltd, "The Waterfall Development Methodology," http://www.learnaccessvba.com/application_development/waterfall_method.htm, accessed November 2012.

A brief explanation of the Waterfall software development methodology. Also (briefly) compares Waterfall to Agile approaches.

[53] E. Tolev, "PivotalTracker - FundFind," https://www.pivotaltracker.com/projects/688967, Nov. 2012, accessed November 2012.

Publicly visible project management for the software component of this project. By default, it will show what the author is currently working on and what is in the backlog. Click on "Icebox" (in the top left-hand corner of the web page, next to the "CURRENT" and "BACKLOG" links) to see all recorded user stories and chores for getting the project done. Stories in the Icebox are ordered in descending order in terms of priority, i.e. the more important ones according to aggregated user opinion are at the top.

[54] Twitter Inc., "Bootstrap - Sleek, intuitive, and powerful front-end framework for faster and easier web development." http://twitter.github.com/bootstrap/, accessed November 2012.

The homepage of the Twitter Bootstrap User Interface library.

[55] UK Government - Unknown individual name, "DATA.GOV.UK - OPENING UP GOVERNMENT," http://data.gov.uk/, July 2012, accessed November 2012.

The UK Government Open Data portal containing various datasets released under Open licences.

[56] Various, "What are the pros and cons of XML and JSON?" http://stackoverflow.com/questions/3536893/what-are-the-pros-and-cons-of-xml-and-json, accessed November 2012.

A discussion on why / how JSON maps better to intuitive human understanding of the world than XML does and why this is important for a data representation format.

[57] Various (a UK Government Working Group), chaired by Dame Janet Finch, "Accessibility, sustainability, excellence: how to expand access to research publications," http://www.researchinfonet.org/wp-content/uploads/2012/06/Finch-Group-report-FINAL-VERSION.pdf, June 2012, accessed November 2012.

The Finch report - looked into Open Access and its benefits for academic publishing and society at large, commisioned by the UK Government.

[58] Various Authors, "Search stackoverflow - python import," http://stackoverflow.com/search?q=python+import, accessed April 2013.

A search on programming Questions and Answers website stackoverflow.com showing questions related to importing modules in Python. Too many questions - which is the point. There seems to be some cognitive barrier not allowing people to understand various basic and more advanced concepts of the Python dynamic import system.

[59] Various authors, "Behaviour-Driven Development," http://behaviour-driven.org/, Jan. 2009, accessed November 2012.

A description of Behaviour-Driven Development. This is a Wiki-style web resource, so the year and month of publication are taken from the last edited date and may change by the time the reader accesses this resource.

[60] Various authors - an open-source project, "D3 - Data-Driven Documents," http://d3js.org/, accessed November 2012.

Homepage of D3.js, a lower-level Javascript graphical visualisation library.

[61] ——, "elasticsearch - You know, for Search," http://www.elasticsearch.org/, accessed November 2012.

The homepage of the elasticsearch indexing server.

[62] ——, "Flask - Web development, one drop at a time," http://flask.pocoo.org/, accessed November 2012.

Homepage of the Flask Python web framework project, helps build RESTful web applications in Python.

[63] ——, "Jenkins - An extendable open source continuous integration server," http://jenkins-ci.org/, accessed November 2012.

Homepage of the Jenkins Continuous Integration server, a tool which helps a developer or a team of developers practice the Continuous Integration software development principle.

[64] ——, "jQuery is a new kind of JavaScript Library." http://jquery.com/, accessed November 2012.

The homepage of the jQuery Javascript and User Interface library.

[65] ——, "OpenRefine," https://github.com/OpenRefine/OpenRefine, accessed November 2012.

A piece of open-source software for handling large semistructured datasets, making en-masse changes to them and cleaning them up. Used to be the Google Refine project.

[66] ——, "SeleniumHQ - Web application testing system," http://seleniumhq.org/, accessed November 2012.

The homepage of an open-source software project which enables the automation of web browsers and other testing techniques for the purposes of testing web applications.

[67] Various authors - an open-source project hosting open data, "OpenStreetMap - The Free Wiki World Map," http://www.openstreetmap.org/, accessed November 2012.

An open-source, open data geographical mapping service.

[68] Various authors / Wikipedia, "Faceted search," http://en.wikipedia.org/wiki/Faceted_search, accessed November 2012.

The Wikipedia page on faceted search. Would not make much sense to a reader unfamiliar with the topic, but should do with the example in the report body.

[69] Wellcome Trust, "Wellcome Trust - We are a global charitable foundation dedicated to achieving extraordinary improvements in health by supporting the brightest minds." http://www.wellcome.ac.uk/, accessed November 2012.

The homepage of a global charitable foundation which funds lots of scholarly activities in a variety of areas.