To Re-use is to Re-write: Experiences with Re-using IIR Experiment Software

Mark M Hall
mark.hall@informatik.uni-halle.de
Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg
Halle (Saale), Germany

ABSTRACT

Interactive Information Retrieval experiments have two main requirements. They need to follow a workflow that takes the participant through the individual steps of the experiment and they need to show the user an interface to interact with. Both of these aspects look like they should lend themselves to re-use. This paper analyses the experience of developing and re-using software for both of these aspects across a time period of approximately five years. The main conclusion is that re-use of workflow management software should be possible, but for software for interface creation the question of whether re-use is possible is still open.

CCS CONCEPTS

• **General and reference** → **Evaluation**; • **Software and its engineering** → **Software design tradeoffs**; **Reusability**; **Software evolution**;

KEYWORDS

Interactive Information Retrieval; Re-Use; Software; Evaluation

1 INTRODUCTION

Interactive Information Retrieval (IIR) experiments use a wide range of terminology, research designs, methodologies, resources, and reporting structures. As has been stated before, one of the issues this has led to is that re-use in IIR is, on the face of it, harder and thus less common, a situation that the BIIRRR workshop series seeks to address [1]. While IIR studies can be deployed via a range of devices, delivery via the web is a common scenario and thus creating tools to ensure this process supports as much re-use as possible is a potential starting point.

This paper discusses the experience of creating and re-using two IIR software systems for building IIR experiments across multiple IIR studies.

2 BACKGROUND

This analysis of the issues around re-using IIR web software components is based on the experience of re-using two software systems across three shared tasks (Session TREC, iCHiC, and iSBS) and a number of individual (IIR) studies.

Workshop on Barriers to Interactive IR Resources Re-use at the ACM SIGIR Conference on Human Information Interaction and Retrieval (CHIIR 2019), 14 March 2019, Glasgow, UK

2.1 Software

The two software components that form the focus of this analysis are the *Experiment Support System (ESS)* and the *Python Interactive information Retrieval Evaluation (PyIRE)*.

2.1.1 Experiment Support System. The *ESS* [5] was developed to handle the challenge of introducing and promoting a standardised, yet flexible methodology for a range of IIR evaluation study structures, including generic, standardised measures that can be deployed across studies and then allow for at least partial comparability of the results. Over time, the accumulated studies should also provide a comprehensive data-set that includes both context and process data that may be used by the IR community to test and develop algorithms seated in human cognition and behaviour, and additionally to provide a sufficiently robust, detailed, reliable data-set that may be used to test existing measures and develop new ones. The core aims were to

(1) Provide a systematic way of setting up an experiment or user study that may be intuitively used by students and researchers;
(2) Provide a standard set of evaluation measures to improve comparability;
(3) Ensure that standard and consistent data formats are used to simplify the comparison and aggregation of studies;
(4) Extract a standard procedure for the conduct of IIR studies from past research, so that studies can share a common protocol even if the system, the tasks, and the participant samples are different;
(5) Reduce resource (financial, time, users, ...) commitment in the conduct of such studies.

To achieve this the overarching architecture in Figure 1 was developed, which consists of the following components:

- The **Research Manager** is the primary point of interaction for the researcher setting up an experiment. It is used to specify the workflow of the experiment, the tasks and interfaces to use, and all other measures to acquire. To simplify and standardise both the experiment process and results, the **Research Manager** is primed with a *generic research protocol*, that specifies the basic experiment workflow and into which the researcher only has to add the experiment-specific aspects;
- the **Experiment System** takes the experiment defined by the *Research Manager* and generates the UI screens that the participants interact with. It also ensures that the tasks and interfaces are correctly distributed and rotated between the participants, in accordance with the settings specified in the **Research Manager**. Finally it loads the **Task-specific UI**

Figure 1: Design of the evaluation framework proposed in [5], with the three core and the two study-specific components. In a study not situated in the IIR field, different study-specific components would be used. In the framework, the researcher interacts only with the *Research Manager* and *Data Extractor*, while the participant only ever sees the *Experiment System* and *Task-specific UI*

and records the participants' responses and ensures that they conform to the requirements specified by the researcher. To ensure the flexibility of the system, any web-based system can be used as the **Task-specific UI**;
- the **Data Extractor** takes the participant data gathered by the **Experiment system** and provides them in a format that can be used by analysis packages such as SPSS or R. The data includes not only the participants' responses, but also data on tasks / interfaces used by the participants and the order in which they appeared.

To simplify the setup and further standardise IIR studies, the following two IIR-specific components have been developed. In a study outside the IIR context, these would be replaced with components developed for that context.

- the **Generic IIR Research Protocol** aims to define a standardised and re-usable workflow and set of evaluation measures for IIR evaluation studies;
- the **Task Workbench** provides an extensible and pluggable set of UI components for IIR interfaces, with the aim of simplifying the set-up of IIR evaluation experiments.

The software was written in Python as a web-based application under an OpenSource license. It allows the researcher to define complex experiment workflows, including reponse-driven or data-driven conditional branching, loops, crowdsourcing-style sampling of questions from a data-set, and full latin-square setups. In the case of the data-driven and latin-square functionalities the system also automatically balances participants across the various conditions. Researchers can also import and export individual questions, pages, and complete experiment workflows in order to ease re-use.

2.1.2 PyIRE. The PyIRE system [4] implements what in Figure 1 is referred to as the "Task-specific UI / Task Workbench". It provides a Python-based, standardised API, which allows the researcher to define IIR user-interface (UI) components, their layout on screen, and the data-flows both between the interface and the components and between components directly. To achieve this the PyIRE system uses the architecture shown in 2. To achieve maximum flexibility,

Figure 2: The evaluation workbench consists of the four core modules (Web Frontend, Message Bus, Session, and Logging) into which the IIR components used in the experiment are plugged.

Figure 3: The workbench's main workflow starts with the generation of the initial UI and then waits for the participant to generate a UI event. The event is processed, the affected component's state and UI are updated and the workbench goes back to waiting for the next UI event. A powerful aspect of the workflow is that components, when they receive a message, can generate their own messages.

the system was designed using a message-passing architecture that consists of the following four components:

- **Web Frontend** handles the interface between the participant's browser and the evaluation workbench and is implemented using a combination of client-side and server-side functionality.
- **Message Bus** handles the inter-component communication and forms the core of the system. It is responsible for passing messages from the **Web Frontend** to the IIR components configured to be listening for those messages and also for passing messages directly between the components.
- **Session** handles loading and saving the components' current state for a specific participant, hiding the complexities of web-application state from the individual components.
- **Logging** provides a standardised logging interface that allows the components to easily attach logging information to the UI event generated by the participant.

When the researcher sets up the workbench for their experiment, they can freely configure which components to use, how to lay them out, and which components to connect to which other components. Based on this configuration the **Web Frontend** generates the initial user-interface that is shown to the participants. Then, when the participant interacts with a UI element (fig. 3), the resulting UI event is handled by the **Web Frontend**, which generates a message based on the UI event. This message is passed to the **Message Bus**, which uses the configuration provided by the

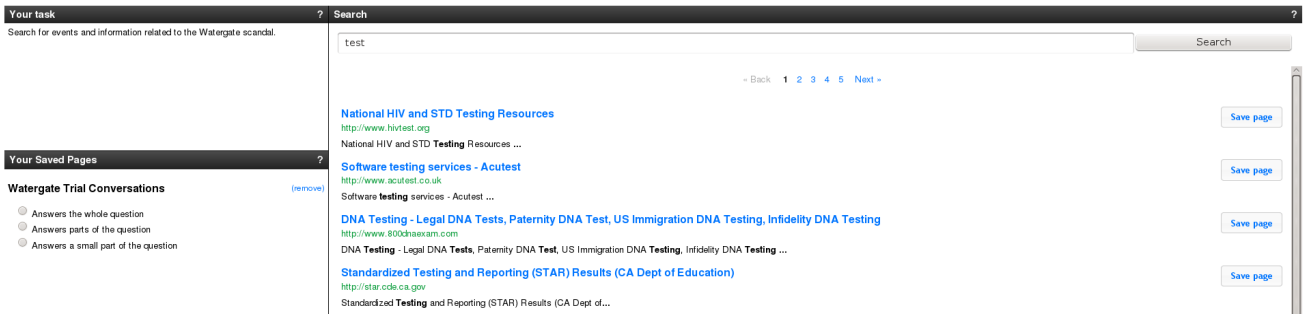

Figure 4: Example Interface built with the PyIRE workbench. The interface here consists of five separate components (task, saved pages, search box, pagination, and search results list, which are joined together via the interface configuration.

```
[SearchResults]
handler = application.components.SearchResults
name = search_results
layout = grid-9 vgrid-expand
connect = search_box:query
```

Figure 5: Example configuration for a *Standard Results List* component, showing how the component's layout (9 grid-cells wide and vertically expanding) and connections to other components (to the "search_box" component via the query message) are specified.

researcher to determine which components to deliver the message to. The components that are listening for that message update their own **Session** state based on the message and then mark themselves as *changed*. After message processing has been completed for all components, the **Web Frontend** then updates the UI for each of the *changed* components.

An example of the configuration used to set-up the experiment is shown in Figure 5 (from the experiment in figure 4), specifying the configuration of the "search_results" component. It specifies that the component should be displayed 9 grid-cells wide (the application layout uses a 12-by-12 cell grid layout) and should expand vertically to use as much space as is available. The component is configured to be connected to the "search_box" component via the "query" message. It is this ability to freely plug components together that, we believe, makes the framework sufficiently flexible to support the wide range of IIR experiments, while remaining simple to set-up and use.

The message-passing architecture should allow arbitrary components to work together. This should allow the researcher to take components from other experiments, for example a novel search result visualisation component, and combine it with other components from their own research, such as a specific search backend.

2.2 Experiments

The two software components were developed and then further re-used in a series of shared evaluation tasks and stand-alone IIR studies.

2.2.1 Session TREC. The Session TREC shared task [2] ran from 2011 - 2014 as part of the Text REtrieval Conference's series of tracks. The aim was to provide participating teams with multi-query search sessions in order to develop and evaluate improved ranking algorithms that took previous queries and results into account. In order to provide participating teams with the necessary multi-query session data, for the 2012 iteration, the decision was made to acquire this session data through a custom IIR experiment.

The initial run (2012) used custom software, from which re-usable aspects were identified. The *ESS* and *PyIRE* software packages were developed in the following year to support both the Session TREC data acquisition and the iCHiC shared task described below.

2.2.2 iCHiC. The CHiC interactive (iCHiC) task was added to the longer-running Cultural Heritage in CLEF lab in 2013 [7]. The interactive task focused on acquiring and analysing interactive information retrieval data-set describing undirected exploration and browsing in a collection of approximately 1.1 million English-language Cultural Heritage items. The task included both an online and an in-lab part. The task UI provided three methods for the participants to explore the collection. On the left there was a category browser, that showed a hierarchical structure into which a sub-set of the items in the collection (approximately 250,000) had been mapped automatically [3]. The second option was to use the search box to type in and run a query. The third method was to click on an item's meta-data, which would run a search for other items with the same meta-data. In all three cases, the items for the selected category, user-provided query, or meta-data query would be shown in the central grid.

As stated above the exploration/browsing interface was built using the PyIRE software.

2.2.3 iSBS. Th interactive Social Book Search task in the CLEF Social Book Search lab ran for three years from 2014 - 2016 [6] and combined ideas from the iCHiC task with research questions from the longer-running Social Book Search (SBS) lab. Users looking for books online are confronted with both professional meta-data and user-generated content. The goal of the Interactive Social Book Search Track was to investigate how users used these two sources of information, when looking for books in a leisure context.

In the first year, the PyIRE workbench was used to construct two UIs, one a traditional faceted search interface and one a novel three-stage interface, based on the Vakkari search stages [8]. In the second year, the three-stage interface was modified, while in the third year only an unchanged three-stage interface was tested, but using a wider range of tasks.

2.2.4 WorldCat. The WorldCat experiment [unpublished] looked at known-item search tasks within a large bibliographic data-set. The PyIRE workbench was used to construct a replica of the WorldCat interface, but used the SBS book data-set to provide a controlled data-set. The *ESS* was used to manage the experiment workflow.

2.2.5 Spatial Language & Jokes Transcription. The Spatial Language and Jokes Transcription experiments only used the *ESS* to handle the experiment workflow aspects. Neither of these experiments was a traditional IIR study, but both re-used major parts of the workflow developed in [5]. However, the experiment-specific UIs were custom built for both of the experiments.

3 EXPERIENCE

The primary take-away message from the experience of develping, re-using, and maintaining the two software packages over the course of five years is that the more generic the software, the easier it is to re-use that component. That is not particularly surprising, as it is in line with the re-use of other software components in IIR. For example, few IIR experiments build a new search backend from scratch for their IIR studies, as the generic search engines that are available, are easily adaptable to the specific data requirements.

3.1 ESS

The experience of re-using and evolving the *ESS* has mostly been a positive experience, with the majority of issues encountered primarily common software development issues, rather than IIR specific issues.

As the *ESS* was re-used throughout the years, the main change was the addition of increasingly complex and powerful features. The initial version was designed to allow the combination of standard survey-style questions with data-driven, task-specific crowdsourcing questions (where the question is wholy or in part driven by a data-set stored in the system). As the complexity of the experiment workflows increased, the *ESS*' functionality was increased, adding latin-square and conditional branching support for the iSBS task. This in some cases required re-writing parts of the *ESS* implementation, but it was always a matter of software evolution, rather than having to make major conceptual or structural changes to the system.

The generic nature of the *ESS* has also enabled re-use in studies that lie outside the IIR context. This was basically easily possible because of the initial decision to host the experiment-specific functionality outside the *ESS* and include it via the use of HTML frames. Thus the map-based and transcription experiment interfaces developed for the *Spatial Language* and Jokes Transcription experiments could easily be integrated into the *ESS* experiment workflow.

However, there have also been some issues with re-use with the *ESS*. These fall into two categories: issues with embedding the task-specific UI into the *ESS* and issues with documentation.

The documentation issues affected the re-use of the *ESS* in two ways. Missing user-focused documentation on how to use the more advanced functionalities (primarily in the area of latin-squares, data-driven questions, conditional branching) meant that re-use by other academics has been limited to those who have easy access to the *ESS* developer in order to get support in how to set up such experiments. In theory these could have been addressed relatively easily, all that was needed were small tutorials to illustrate in which order to execute the individual steps. For example, for a data-driven crowdsourcing experiment, it is necessary to first create the data-set that has all the different items which are sampled, then create the page to display them on, and finally use text markup to embed the data in the page that is displayed to the participant. However, none of this is particularly apparent from the interface itself.

The other documentation issue is related to the documentation of the code itself, which is very patchy. The result of this is that the *ESS* has reached a point where it is highly functional, but essentially cannot be maintained or developed any further, as any change risks breaking existing functionality in unexpected ways.

Both of these issues are primarily caused by the *ESS* being a side-project, where what time was available was focused on improving the functionality and not documentation. While not a particularly novel conclusion, it does re-iterate the point that without adequate documentation, re-use is essentially highly unlikely.

The second major issue that was encountered was with embedding the task-specific UI in the *ESS*. The first is caused by a limitation of the use of frames for embedding the task-specific UI. In order to produce an embedding that mostly hid the fact that the task-specific UI was embedded, the researcher had to manually use a large amount of CSS and some JavaScript to correctly adapt the size of the frame in which the UI is embedded. This created an instant barrier to re-use, as it required some very specialised technical skills.

The other issue with the task UI embedding arose from the need to link the responses in the *ESS* with those in the task UI. This is necessary as the *ESS* and the task UI are completely separate systems, thus no automatic linking is possible. To create a link, the unique ID of the *ESS* participant can be embedded in the URL that loads the task UI. When the task UI loads, the software generating the UI can access this ID and store it together with the other data collected by the task UI. Then, when the data is extracted, the ID can be used to merge together the two survey responses collected by the *ESS* and the data logged by the task UI.

For the 2013 *Session TREC* experiment, a configuration error caused the same static identififer to be sent for all participants. While the error occured due to a mistake made by the researcher when setting up the experiment, the brittleness of the linkage between the two systems and the difficulty with seeing whether the linking ID data was being transferred correctly, allowed the mistake to go unnoticed. As a result, in that year the *Session TREC* data-set consisted only of the session query logs, but without any information on the participants themselves, significantly limiting the value of the data-set as a whole.

Partly this issue is due to the *ESS* trying to be both a system that requires minimal technical skills, but also a system that is very powerful and flexible, allowing the researcher to adapt the system to a large degree. Based on the experience, I would suggest that

future systems of this kind either focus on the pure ease-of-use or the technical flexibility, but not both, and that from the beginning documentation is a core step.

3.2 PyIRE

While re-use of the *ESS* was fundamenally possible, re-use of the *PyIRE* workbench was not as successful. While the *PyIRE* was used across all three shared tasks and in the *WorldCat* experiment, each re-use of the workbench essentially involved a major re-write of the software.

The re-writes revealed the difficulty of developing truly re-usable UI components and also the difficulty in designing an architecture that allows for minimally connected components that at the same time provide the user with a cohesive use experience. As the *PyIRE* system was re-written to support more complex interface structures, more and more data had to be explicitly passed between components, functionality that had to be added to each component, countering the core idea of plug-and-play reusability.

Another issue was that while the architecture decoupled the components, particularly around rendering the architecture there were some highly coupled interactions between the components and the underlying *PyIRE* functionality. These coupling points meant that in some cases, components had to have complex internal structures, simply because they had to handle the case where the functionality was needed to update the component's display and in other cases just to provide service functionality to other components.

The main effect of the re-writes and the coupling issues was that it is very hard to actually replicate the past experiments, as each one requires a very specific version of the *PyIRE* system to run. While the code is available, this means that for each experiment, a new instance of the *PyIRE* server would have to be run, undermining the point of having a workbench that allows implementing multiple experiments in an easy-to-manage environment.

The big question, which I cannot answer, is whether the problem with the *PyIRE* are due to specific mistakes made in how the decoupled architecture was implemented, or generic issues with the architecture itself.

In particular, the question is related to the *ESS* issue with the target user groups. The way the *PyIRE* can be used was designed to support both researchers wishing to develop their own components, but also researchers who lacked the technical skills to build their own and simply wanted to re-use existing components with other tasks or data. Attempting to support both scenarios created significant additional complexity, essentially making the system hard to use for both groups.

4 CONCLUSION

The main conclusion from this analysis is that the core issue for long-term re-use and maintainability of software for IIR experiments is the availability of adequate documentation and in this respect IIR experiment software is no different to any other software.

The second conclusion is that the development of a system that supports researchers in the development, deployment, and re-use of IIR experiment workflows is possible, as evidenced by the *ESS* system. Ideally this might take the form of multiple systems that are targeted at different levels of technical expertise, but which use a standardised format for describing questions and answer options, page structures, and overall experiment workflows. This would then allow importing and exporting these and moving experiments between the different systems.

The third conclusion is that how to achieve the re-usability of software that helps with building IIR interfaces, is still an open question. Considering that there have been different approaches, including the one documented here, none of which have caught on in the slightest, it is also unclear whether there is actually any value in attempting this.

From the experience I would suggest that future work should really focus on defining re-usable formats to define questions, answer options, page structures, and experiment workflows. These could then be moved between systems, enabling re-usability while allowing for flexibility in what systems people want to use. Re-usability of task UIs is an area where I am currently unconvinced that re-usability is worth pursuing.

REFERENCES

[1] T. Bogers, M. Gäde, L. Freund, M. Hall, M. Koolen, V. Petras, and M. Skov. Workshop on barriers to interactive ir resources re-use. In *Proceedings of the 2018 Conference on Human Information Interaction & Retrieval*, CHIIR '18, pages 382–385, New York, NY, USA, 2018. ACM.

[2] B. Carterette, P. Clough, M. Hall, E. Kanoulas, and M. Sanderson. Evaluating retrieval over sessions: The trec session track 2011-2014. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 685–688, New York, NY, USA, 2016. ACM.

[3] M. M. Hall, S. Fernando, P. Clough, A. Soroa, E. Agirre, and M. Stevenson. Evaluating hierarchical organisation structures for exploring digital libraries. 17(4):351–379, 2014.

[4] M. M. Hall, S. Katsaris, and E. Toms. A Pluggable Interactive IR Evaluation Workbench. In *European Workshop on Human-Computer Interaction and Information Retrieval*, pages 35–38, 2013.

[5] M. M. Hall and E. Toms. Building a Common Framework for IIR Evaluation. In *CLEF 2013 - Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, pages 17–28, 2013.

[6] M. Koolen, T. Bogers, M. Gäde, M. M. Hall, I. Hendrickx, J. Kamps, M. Skov, S. Verberne, and D. Walsh. Overview of the CLEF 2016 Social Book Search Lab. 2016.

[7] E. Toms and M. M. Hall. The CHiC Interactive Task (CHiCi) at CLEF2013. http://www.clef-initiative.eu/documents/71612/1713e643-27c3-4d76-9a6f-926cdb1db0f4, 2013.

[8] P. Vakkari. A theory of the task-based information retrieval process: a summary and generalisation of a longitudinal study. 57(1):44–60, 2001.