Troggle: a revised system for cave data management[1]

Philip Sargent and Aaron Curtis
Cambridge University Caving Club
August 2020

philip.sargent@gmail.com

Why cavers need effective data management

Cave exploration is more data-intensive than any other sport. The only way to "win" at this sport is to bring back large quantities of interesting survey, and possibly photos or scientific data. Aside from the data collection requirements of the game itself, setting up a game (an expedition) of cave exploration often involves collection of personal information ranging from dates available to medical information to the desire to purchase an expedition t-shirt.

If an expedition will only happen once, low-tech methods are usually adequate to record information. Any events that need to be recorded can go in a logbook. Survey notes must be turned into finished cave sketches, without undue concern for the future expansion of those sketches.

However, many caving expeditions are recurring, and managing their data is a more challenging task. For example, let us discuss annual expeditions. Every year, for each cave explored, a list of unfinished leads (which will be called "Question Marks" or "QMs" here) must be maintained to record what has and has not been investigated. Each QM must have a unique id, and information stored about it must be easily accessible to future explorers of the same area. Similarly, on the surface, a "prospecting map" or GPS overlay showing which entrances have been investigated needs to be produced and updated at least after every expedition, if not more frequently.

These are only the minimum requirements for systematic cave exploration on an annual expedition. There is no limit to the set of data that would be "nice" to have collected and organized centrally. An expedition might collect descriptions of every cave and every passage within every cave. Digital photos of cave entrances could be useful for re-finding those entrances. Scans of notes and sketches provide good backup references in case a question arises about a finished survey product, and recording who did which survey work when can greatly assist the workflow, for example enabling the production of a list of unfinished work for each expedition member. The expedition might keep an inventory of their equipment or a catalog of their library. Entering the realm of the frivolous, an expedition might store mugshots and biographies of its members, or even useful recipes for locally available food. The more of this information the expedition wishes to keep, the more valuable an effective and user-friendly system of data management.

Troggle in 2020

The Troggle cave exploration data management system was created by Aaron Curtis in 2006. Since then it has been used extensively and diligently to record cave survey data by CUCC in Austria. However many of the original ideas in troggle have never been fully implemented. Some features turned out to be impossible to maintain or not actually as useful as they were expected. So this revised paper omits those which we do not use but which were described in the 2006 paper.

---

1 Minimally revised and updated by Philip Sargent from "Troggle: a novel system for cave exploration information management" by Aaron Curtis, 2006, Cambridge University Caving Club, aaron.curtis@cantab.net

Evolution data management on the CUCC Expeditions to Austria

Since 1976 CUCC has developed methods for handling cave exploration information. Refinements in data management were made necessary by improved quantity and quality of survey; but refinements in data management also helped to drive those improvements. The first CUCC Austria expedition, produced only Grade 1 survey for the most part (ref Cambridge Underground 1977 report). In the 1980s, the use of programmable calculators to calculate survey point position from compass, tape, and clinometer values helped convince expedition members to conduct precise surveys of every cave encountered. Previously, such work required hours of slide rule or log table work. On several expeditions, such processing was completed after the expedition by a FORTRAN program running on shared mainframe time. BASIC programs running on personal computers took over with the release of the BBC Micro and then the Acorn A4. In the 1990s, Olly Betts and Wookey began work on “Survex”, a program in C for the calculation and 3-D visualization of centerlines, with intelligent loop closure processing. Julian Todd's Java program “Tunnel” facilitated the production of attractive, computerized passage sketches from Survex centerline data and scanned hand-drawn notes.

Along with centre-lines and sketches, descriptions of caves were also affected by improvements in data management. In a crucial breakthrough, in 1996 Andrew Waddington introduced the use of the nascent markup language HTML to create an interlinked, navigable system of descriptions. Links in HTML documents could mimic the branched and often circular structure of the caves themselves. For example, the reader could now follow a link out of the main passage into a side passage, and then be linked back into the main passage description at the point where the side passage rejoined the main passage. This elegant use of technology enabled and encouraged expedition members to better document their exploration.

To organize all other data, such as lists of caves and their explorers, expedition members eventually wrote a number of scripts which took spreadsheets (or comma separated value files, .CSV) of information and produced webpages in HTML. Other scripts also used information from Survex data files. Web pages for each cave as well as the indexes which listed all of the caves were generated by one particularly powerful script, `make-indxal4.pl` . The same data was used to generate a prospecting map as a JPEG image. The system of automatically generating webpages from data files reduced the need for repetitive manual HTML coding. Centralized storage of all caves in a large .CSV file with a cave on each row made the storage of new information more straightforward.

Another important element of this system was version control. The entire data structure was stored initially in a Concurrent Version System (CSV) repository, and later migrated to Subversion (SVN). Any edits to the spreadsheets which caused the scripts to fail, breaking the website, could be easily reversed. (Today we use git.)

However, not all types of data could be stored in spreadsheets or survey files. In order a display descriptions on the webpage for an individual cave, the entire description, written in HTML, had to be typed into a spreadsheet cell. A spreadsheet cell makes for an extremely awkward HTML editing environment. To work around this project, descriptions for large caves were written manually as a tree of HTML pages and then the main cave page only contained a link to them.

A less obvious but more deeply rooted problem was the lack of relational information. One table named `folk.csv` stored names of all expedition members, the years in which they were present, and a link to a biography page. This was great for displaying a table of members by expedition year, but what if you wanted to display a list of people who wrote in the logbook about a certain cave in

a certain expedition year? Theoretically, all of the necessary information to produce that list has been recorded in the logbook, but there is no way to access it because there is no connection between the person's name in `folk.csv` and the entries he wrote in the logbook.

The only way that relational information was stored in our csv files was by putting references to other files into spreadsheet cells. For example, there was a column in the main cave spreadsheet, `cavetab2.csv` , which contained the path to the QM list for each cave. The haphazard nature of the development of the “script and spreadsheet” method meant that every cave had an individual system for storing QMs. Without a standard system, it was sometimes unclear how to correctly enter data.

The full history of CUCC cave exploration management can be found at http://expo.survex.com/handbook/website-history.html

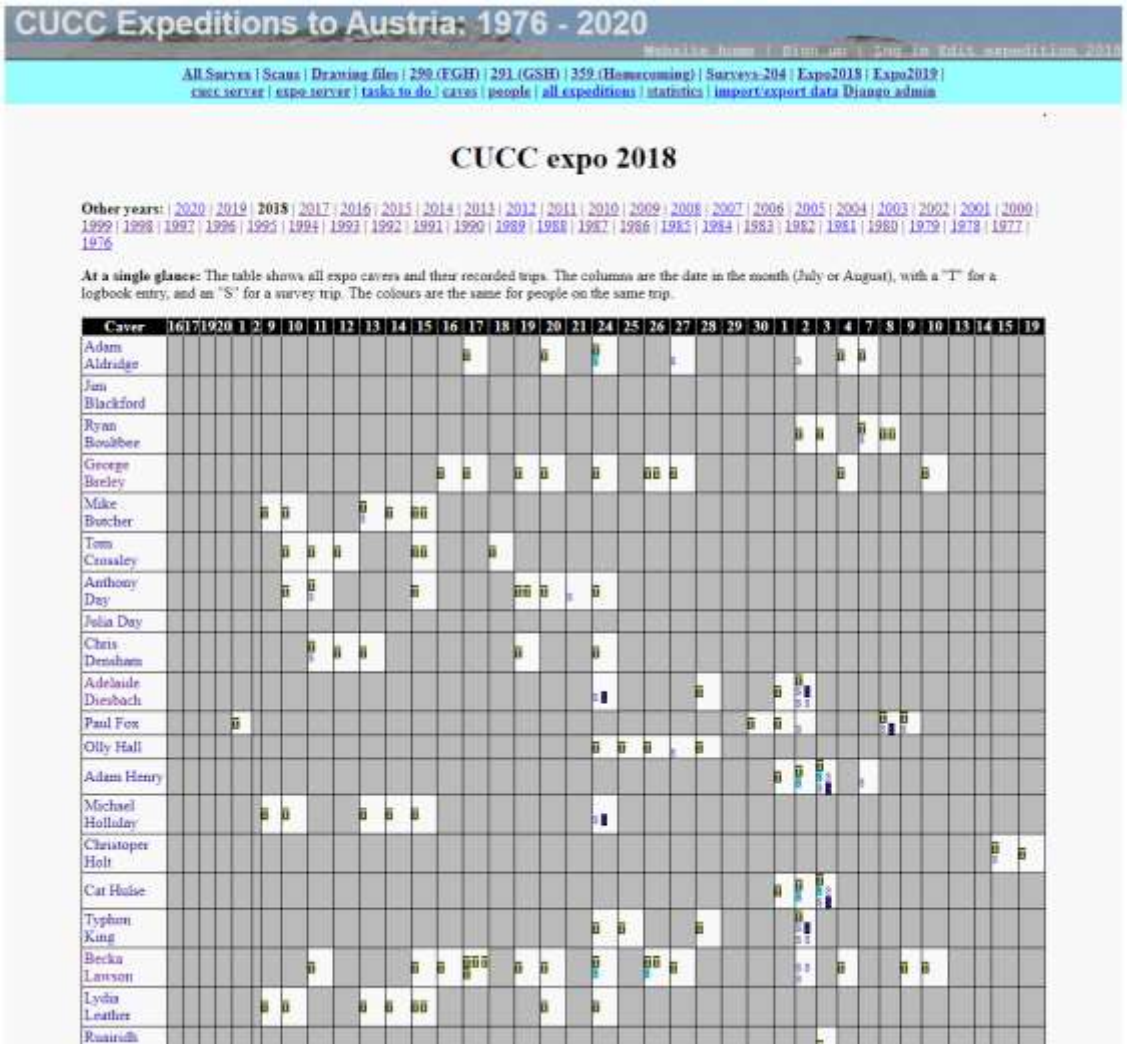


Figure 1 – Expedition page showing participants and trips

What's different about Troggle?

Troggle was indeed another in a long series of paper-based and computerized utilities CUCC has developed to make our expedition run more smoothly. Yet, Troggle was revolutionary: it is a unified system which can manage any type of expedition data, and understands the relationships between the data.

Using a relational database, Troggle represents familiar expedition concepts such as the cave, the person, and the logbook entry in tables. All of the links which exist between these concepts in real life, such as the fact that logbook entries are always written by people and can sometimes refer to

caves; exist in the database as *foreign keys* between models.

Troggle: the input data and server

It was originally intended that all of the data in the Troggle database would be created, viewed, and edited through a web browser. This turned out to be difficult to manage in practice.

Today Troggle imports all the data it needs from files: survex files, logbook files (typed up in HTML), some CSV files for old QMs, drawing files (Tunnel and Therion editable cave drawing files), finished images of surveys, photographs of cave and entrances, XML files of cave descriptions and a thousand-page website of HTML files comprising the Expo Handbook.

Troggle runs on a webserver hosted today at https://expo.survex.com. Internet access is now sufficiently good in Austria that it is easier for everyone to access the main troggle server directly, by phone at topcamp if need be. Critical information such as the Rescue Guide is printed on paper and kept at basecamp and at topcamp.

Troggle can be run without internet on a stand-alone laptop but in practice only the software developers do that as configuring a new machine is somewhat involved and not well documented.

Troggle's database is ephemeral. When Troggle starts up it imports all the data it needs for cross-referencing into a completely empty database and this takes about 4 minutes. The database is only about 5 MB. When Troggle shuts down (which only happens during system maintenance) the database is discarded. There is also about 50 GB of large files (i.e. 50,000 times as much): all the scanned images of logbooks, survey notes, basecamp and cave photograph albums, finished cave survey images etc. Troggle indexes some of these and the rest are simply linked into HTML pages in the expo handbook.

Logbooks and survey trips per day

Date	Logged trips	Surveys
Sat 16 Jun 2018		mash mash2
Sun 17 Jun 2018		hangman hangmansdaughter prangman choked
Tue 19 Jun 2018		snailtrail
Wed 20 Jun 2018		sluggish mudslope
Sun 01 Jul 2018	plateau - Eiskelle - Surface Prospecting	corroborate1
Mon 02 Jul 2018		corroborate2
Mon 09 Jul 2018	Fisch Gesicht - rerigged entrance	
Tue 10 Jul 2018	264 - Balkonhoehle/ 1st rig Tunnocks - 1st rig Fisch Gesicht- pushing to liquid luck	
Wed 11 Jul 2018	plateau - Prospecting past Fisch Gesicht, beyond the ski pole line plateau - Prospecting visiting known holes near Organhöhle	2012-dd-08
Thu 12 Jul 2018	Tunnocksschacht 258 - String Theory to Procrastination Rig plateau - Prospecting Kleine Wildkogel	
Fri 13 Jul 2018	Homecoming - CUCC2018 DM07 (Homecoming Hole) - First push plateau - Surface recce for Futility series back door Homecoming - 2nd push in Heimkommen höhle (Homecoming) plateau - Steinbrucken Tarp Topo Fisch Gesicht - pushing phreatic flies and surveying liquid luck	blitzen_to_liquidluck
Sat 14 Jul 2018	plateau -Surface walk Stoger Weg 115 (Schnellzughöhle) - actually cucc-ps01-2018 Fisch Gesicht -Pushing Urinal and fettling rope rigging	
Sun 15 Jul 2018	258 - Tunnock's Rig Fisch Gesicht - Pushed to Toto and surveyed free attic frys and Urinal Fisch Gesicht - Found Ulysses	rootblock freeattic_flys
Mon 16 Jul 2018	Balkon - Cathedral Kazam - Wild Honeycomb shaft - The Hangman	
Tue 17 Jul 2018	Balkon - Wild Honeycomb shaft - The Hangman - Hangman's Daughter - **Tunnock's Connection** Balkon - Rigging Guide: Balkon Honeycomb Shaft -> Hangmans Daughter	

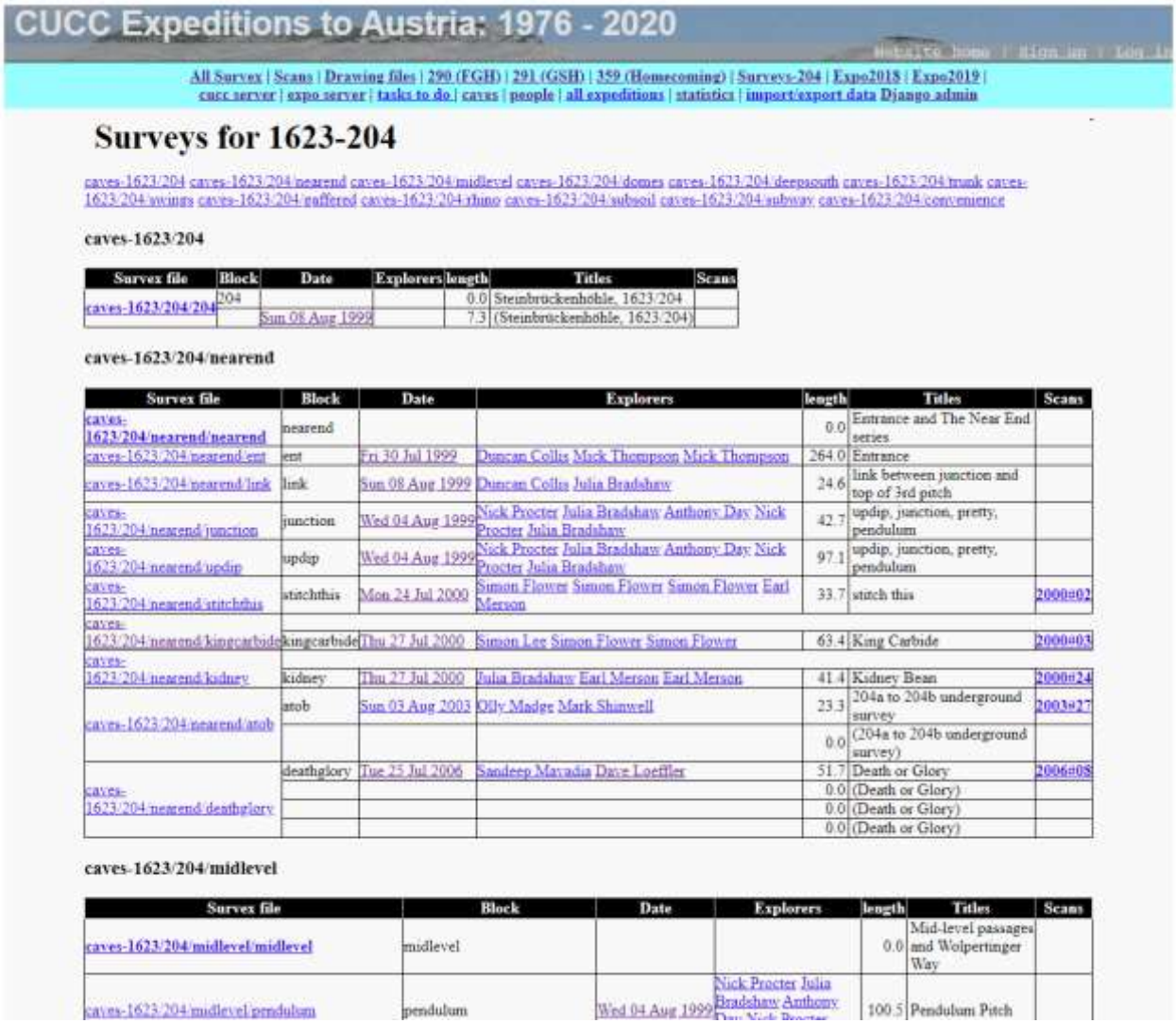
Figure 2: The lower part of the Expedition page: calendar of trips and survex files

During expo

On the expedition itself, each trip is written up in handwriting in the paper logbook. During and after expo, this logbook is scanned and the image PDF is archived. All the trips are typed up into an HTML file which is imported into Troggle so that the people on trips can be cross-referenced with surveys done.

An expedition member will typically spend four or five days at top camp caving before she hikes back down to base camp. When she arrives in base camp, she writes up the trips in the paper logbook and ideally she also types up the trips: who was on each trip, where the trip went, dates, which going leads (QMs) are no longer going, which new going leads exist, and whether the trip did any surveying.

She either downloads the new DistoX data or types up new survey data (tables of survey stations and compass/clino. measurements) into survex files where she documents the QMs. She uses the expo laptop at basecamp and then she, or an expo nerd, uploads the data to the correct place in the version control system on the expo server. Every couple of days Troggle is triggered to do a new import of all the data.



CUCC Expeditions to Austria: 1976 - 2020

All Survex | Scans | Drawing files | 290 (EGH) | 291 (GSH) | 359 (Homecoming) | Surveys-204 | Expo2018 | Expo2019 | cucc server | expo server | tasks to do | caves | people | all expeditions | statistics | import/export data Django admin

Surveys for 1623-204

caves-1623/204 caves-1623/204/nearend caves-1623/204/midlevel caves-1623/204/domes caves-1623/204/deepsouth caves-1623/204/trunk caves-1623/204/swings caves-1623/204/suffered caves-1623/204/rhino caves-1623/204/subsoil caves-1623/204/subway caves-1623/204/convenience

caves-1623/204

Survex file	Block	Date	Explorers	length	Titles	Scans
caves-1623/204/204	204			0.0	Steinbrückenhöhle, 1623/204	
		Sun 08 Aug 1999		7.3	(Steinbrückenhöhle, 1623/204)	

caves-1623/204/nearend

Survex file	Block	Date	Explorers	length	Titles	Scans
caves-1623/204/nearend/nearend	nearend			0.0	Entrance and The Near End series	
caves-1623/204/nearend/ent	ent	Fri 30 Jul 1999	Duncan Collis Mick Thompson Mick Thompson	264.0	Entrance	
caves-1623/204/nearend/link	link	Sun 08 Aug 1999	Duncan Collis Julia Bradshaw	24.6	link between junction and top of 3rd pitch	
caves-1623/204/nearend/junction	junction	Wed 04 Aug 1999	Nick Procter Julia Bradshaw Anthony Day Nick Procter Julia Bradshaw	42.7	updip, junction, pretty, pendulum	
caves-1623/204/nearend/updip	updip	Wed 04 Aug 1999	Nick Procter Julia Bradshaw Anthony Day Nick Procter Julia Bradshaw	97.1	updip, junction, pretty, pendulum	
caves-1623/204/nearend/stitchthis	stitchthis	Mon 24 Jul 2000	Simon Flower Simon Flower Simon Flower Earl Merson	33.7	stitch this	2000#02
caves-1623/204/nearend/kingcarbide	kingcarbide	Thu 27 Jul 2000	Simon Lee Simon Flower Simon Flower	63.4	King Carbide	2000#03
caves-1623/204/nearend/kidney	kidney	Thu 27 Jul 2000	Julia Bradshaw Earl Merson Earl Merson	41.4	Kidney Bean	2000#24
caves-1623/204/nearend/atob	atob	Sun 03 Aug 2003	Olly Madge Mark Shanwell	23.3	204a to 204b underground survey	2003#27
				0.0	(204a to 204b underground survey)	
caves-1623/204/nearend/deathglory	deathglory	Tue 25 Jul 2006	Sandeep Mavadia Dave Loeffler	51.7	Death or Glory	2006#08
				0.0	(Death or Glory)	
				0.0	(Death or Glory)	
				0.0	(Death or Glory)	

caves-1623/204/midlevel

Survex file	Block	Date	Explorers	length	Titles	Scans
caves-1623/204/midlevel/midlevel	midlevel			0.0	Mid-level passages and Wolpertinger Way	
caves-1623/204/midlevel/pendulum	pendulum	Wed 04 Aug 1999	Nick Procter Julia Bradshaw Anthony	100.5	Pendulum Pitch	

Figure 3: The cave survey (survex) data for Cave 161 showing each survex block

The Expo Handbook describes in detail the procedure for managing the survey notes. The underground notes – made in pencil on waterproof paper – are photographed with a phone or scanned using the basecamp flatbed scanner. These images are copied to a directory on the expo

laptop called a “virtual wallet”. The actual pages are stored in a plastic enveloped wallet in a lever-arch file at basecamp.

The names of the wallets are of the form “2015#15” and these are entered into Troggle in the *ref field of the survex files. The last column ‘Scans’ in Figure 3 below gives the wallet number for the original data which is parsed and displayed in the body of the table. Each trip participant is identified and each survex block (a *begin/*end section within a survex files) is identified. We also keep records of the survey instruments used for each trip and when they were calibrated.

Troggle: under the hood

What makes all of this magic possible? Troggle is programmed in python and built using the Django framework[2]. Django provides the structure of routing URL web queries to the right bits of code to compute the pages and maps python objects to relational database tables. Nearly half the code in Troggle is parsers to import the data from the files and the rest is cross-referencing and constructing the webpages. We use Apache as the webserver for directly providing the simple image files and photographs and Django works with Apache to deliver the rest.

Django represents all of the data models as tables in a relational database and automatically creates SQL commands to control that database. Currently, Troggle is uses MariaDB on the server but Django can use almost any database – or none.

When a user requests a URL, Django checks the incoming URL against a list of python regular expressions. Troggle's flexibility with URLs comes from the fact that we have written very loosely matching regular expressions. Once Django finds a regular expression which returns a match for the URL, it runs a python function (a “view”) which takes the URL, draws relevant information from the database, and uses it to fill in a template written in the Django template language, producing a HTML page for display user.

Database access is facilitated by the Django Object Relational Mapper (ORM. Each object type, such as the “Cave”, “Person”, or “Logbook Entry”, is accessible as a Python object with useful methods that allow searching, editing, and saving of the database from python. Here are some lookup scenarios, ranging from the simple to complex, with the necessary python code.

Get a list of all people who have been on the expedition and store it as `result`

```
result=People.objects.all()
```

Get a list of all people who were `on` the 2004 expedition and store it as `result`

```
result=Expedition.get(year=2004).people_set.all()
```

Get the text of the logbook entry with “intrepid voyage” in the title and store it as as `result`

```
result=LogbookEntry.objects.get(title__contains=”intrepid
voyage”).text
```

Of course, the API is not accessible through a web browser, and Troggle users should never need to query the database directly using the commands in the example above. However, this internal API is what drives the views and templates and is therefore used whenever a page is loaded.

In 2020 we have an experimental program to see whether we can take out the ORM and replace it with straightforward python code which stores the objects directly on disc as JSON files. This is because the learning curve for the Django ORM is more than most of our potential system maintainers can cope with. Since the database is now ephemeral this is a realistic design

[2] https://www.djangoproject.com/ “The web framework for perfectionists with deadlines.”

possibility.

In 2020 we are also beginning to provide API equivalents via `http://` so that new and different special-purpose front-ends can be written for phones and web browsers.

Tailoring Troggle to your needs

Three aspects of Troggle's design make it highly customizable.

First, all paths and URLs are relative to variables in a settings file.

Second, because Troggle takes advantage of Django's template inheritance, one HTML file and one CSS file control the appearance and style of the entire website.

Third, Troggle's modular structure allows the addition of alternate or new components.

The locations of files and other aspects of in system setup are set as variables for using the file `localsettings.py`. This file also contains all of the information necessary to access the database Troggle will use. Changing the variable `DATABASE_ENGINE` allows the use of different database systems; Postgresql, MySql, Sqlite3, and other databases are supported.

Block name	**Purpose**
title	The title of the page, to be displayed at the top of the browser
head	Holds elements that need to be inside the HTML <head> tag, such as metadata and scripts
logininfo	This is used to display “welcome,” “log in,” or “sign up” links. Should not be replaced by “extending” templates.
nav	Used for child templates, such as the “virtual survey binder,” that include a lefthand menu (no longer used)
contentheader	The header for the actual information being displayed
content	The actual information being displayed
footer	Content at the footer of the page. Should not be replaced by “extending” templates. (no longer used)

Table 1: Template blocks specified in base.html , which provide the framework for inserting content into Troggle's web interface.

To customize the appearance of the Troggle website as presented to users, it is generally sufficient to modify the file `base.html` , and the Cascading Style Sheet which it references. In Django terms, all of the other templates “extend” `base.html` . The other templates only serve to replace “blocks” in `base.html` with content. The location of `base.html` and all other template files is specified in `localsettings.py`, in the variable `TEMPLATE_DIRS`, which is a tuple of strings that are absolute paths to the directory or directories where templates are stored. The stylesheet used

by `base.html` is stored under the CSS directory of the path specified in `MEDIA_ROOT` and must be served at the URL specified in `MEDIA_URL` .

Custom modules are easy to add to Troggle. For example, CUCC's troggle installation has a folder called `Parsers` which contains scripts for importing data from our data files into the database. If your expedition has a similar data structure to import from, these parsers can be replaced with your own. One of the parsers, survex.py, imports from Survex files (this was entirely rewritten in 2020 to use the latest Survex format). To use survey data in other formats, such as Compass or CMAPS, a new parser would need to be written.

Unfinished capabilities

For more detailed descriptions of how Troggle works and what it now does and does not do see the online documentation in:
http://expo.survex.com/handbook/troggle/trogindex.html

Troggle was originally meant to be used for individual expedition membership registration for every caver who would have their own login account. This was intended to be used for expedition administration and travel planning. This turned out to be unmanageable.

The planned online-prompting for progressing through the actions necessary to turn the scanned notes in the survey wallet into finished cave surveys (see the 2006 version of this report) was partly done outside Troggle by Martin Green using a separate python script, but this is still somewhat incomplete.

The `folk.csv` file still exists as do the legacy `qm.csv` files for QMs which predate the process of including QMs in survex files. There are three separate places where caver names are parsed, in the standalone preparatory `make-folklist.py` script and two places within Troggle so some names fail to resolve correctly or reliably.

Troggle documentation has been somewhat inadequate for bringing new programmers and maintainers up to speed though this has recently improved. Nevertheless the maintenance workload is heavy and nearly more than we can cope with. If it had not been for the COVID-enforced idleness freeing up programming time it is possible that Troggle may have failed entirely in 2020.

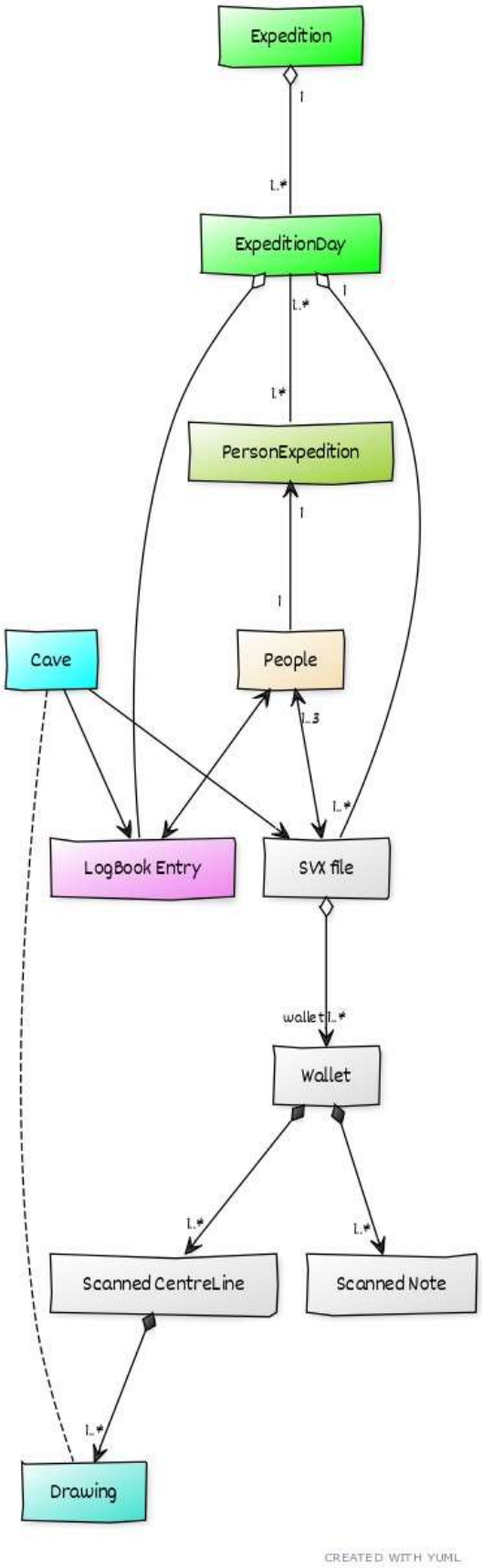


Figure 1: A UML diagram of the main Troggle Classes

The biggest change in 2020 is possibly that Troggle no longer stores every single survey station

position and survey leg data. It stores summary data when parsing the survex files but no one had ever got around to writing code to use the 76,000 survey point data so the huge import job was pointless. As a result of this and other cleanup the import process now takes 4 minutes instead of 4-5 hours.

Changes in style of use

Troggle began by imposing good management protocols and naming schemes. The file import was assumed to be only happening once, and then the database itself would be the maintained and central source of truth. It didn't happen like that.

The file import parsers were sketchily done as it was assumed that the data would be cleaned up and edited within Troggle. This meant that the parsers were not forgiving or well documented. As the habit of use developed of regular and frequent file importing this became a problem in that only very few people knew how to do it, and any failure due to typos in input data crashed the entire system with an assert() failure. This was not very friendly for a beginner caver surveyor. The online pages for editing survex data and logbook entries became unreliable through lack of maintenance and so file uploading become the only way of entering data.

Similarly the allowed naming scheme for files was very controlled but cavers did not read the documentation and called the files whatever they liked. So for many years those files were silently passed over by the parsers which ignored them and yet provided no error messages.

There was little explicit consistency checking between survey data, scan files and logbook data because it was assumed that these would be produced in sync by the same software at the same time. That did not happen, and so inconsistencies accumulated.

In 2020 we have moved to an entirely different style of operation: rather than enforcing compatibility and consistency and forbidding anything that isn't (with no error messages), Troggle now accepts as much as it can and provides tools and reports for managing and fixing the inevitable inconsistencies.

With the benefit of hindsight we realise that the data was never going to be entirely consistent with new untrained people entering data every year. So Troggle provides an increasing number of tools to help us find the hidden inconsistencies on old data as well as to match up the surveys, scans and trip reports in current data.

We are still working on reviewing the older data from the 80s and 90s as the wallet system only really began to be used in 1999. Early cave explorers are not yet fully credited with their discoveries in the summary reports.

Acknowledgements

Over three-quarters of this report is exactly as Aaron Curtis wrote it in 2006. In recent years Paul Fox, Sam Wenham and Wookey have maintained the Troggle code and coddled the server through many migrations and upgrades to later versions of Django (a new version of Django is released every 8 months) and across several machines.

Troggle began as a joint effort between Martin Green, Julian Todd, and Aaron Curtis in 2006. For years, there had been discussion regarding a new paradigm for data management on the expedition, and it was Martin who introduced the idea of a Django project. Julian Todd's web programming and scraping experience was invaluable for many of the views and the logbook parsing. Logbook formats for each year of the expedition are different, but thanks to Becka

Lawson's hard work cleaning up and standardizing past logbook files and Julian's year by year parser coding, we now have over a decade of trips in the database. I would also like to thank Vic Brown and Andrew Waddington for their helpful accounts of the data management techniques they used on early CUCC expeditions, going back all the way to 1976.