Coursework Submission

This is an evolving document, this version 06/12/18 09:54:16.

Proposal

We drop "submit" from 2019/20 academic session. We replace with multiple alternatives depending on per-course teaching/submission and administrative requirements. We strongly advocate one default submission mechanism.

Reasoning

The "submit" programme currently used by the majority of our courses was written over 20 years ago, but crucially has only been barely maintained since that point. While it does meet quite a lot of our needs it is not a robust code base. For example, in the last year two security issues have been identified that could have been trivially exploited to give any student full access to all submissions. The command is only available on the DICE command line – making it harder for external students on courses and DLAS students. The command is used for submission in online examinations and it is very hard to see any alternative (external system) being as usable or robust in that context. OTOH for the purpose of online examinations its functionality in this context could be entirely replaced by a very short Python script for example. Some argument can be made that use during semester will help familiarity during the exam – this is true but the usage is simple enough, students already have to use an unfamiliar command to get the exam paper itself, also they can attend the mocks so I don't think this argument really holds any weight.

Risks

I think we risk ending up with too many different submission solutions – in some cases there is need for specialism but in many there is not. Hence why we should prefer one. The "submit" programme is very easy to use and has pleasant back end features for scripting of work collation – these are less likely to be available.

Alternatives

The following are all possible alternatives:

- Learn assignment tool
- Learn test tool (online quizzing)
- Noteable (Jupyter Notebooks in the Cloud service)
- Moodle Coderunner (accessed via Learn)
- GitHub Classroom
- Questionmark Perception (online exams)
- Turnitin (originality checking and online marking)

- WebPA (peer assessment of group work)
- Microsoft Sharepoint
- Course blogs via the Academic Blogging Service.
- Make a local alternative – not that hard in practice, all submission is really doing is copying stuff from A to B, with some kind of timed verification for the student, some kind of central logging and sufficiently fine grained authenticated access to the submissions.

We will expand on pros/cons below.

Comments

We are currently looking from feedback from students and academics regarding this proposal.

Requirements

Below is an initial list of requirements for a submission system by Paul Anderson. They are specific to his needs and courses but nonetheless we think largely apply across the board. More suggestions from other academics, including course specific requirements, welcome. Requirements from the student side also welcome – these however might be as simple as: intuitive, robust and reliable.

- **Anyone with an EASE account should be able to submit any work.**

  Restricting submissions doesn't seem necessary, involves extra work, and is a potential source of problems. The converse (having unregistered students submit work) is unlikely, and not a problem anyway. (as long as they are authenticated/identifiable).

  In particular, it should not be restricted to DICE - some students (eg. distance) may never have used their DICE accounts, and some (auditing Phd students) may not actually have them by the time the submission is due. Similarly, it shouldn't be restricted to students on some explicit course list (although it is good to warn students if they are submitting work to a course for which they are not registered).

- **Submission should be possible from anywhere.**

  In particular, it should not require the use of a (Unix) command line (although this would be a nice option). Some students have never used the Unix command line, and are not required to. It should not require the installation of any extra software on any of the common platforms. A web form is probably the most obvious approach.

- **Submissions should be accepted in a format determined by the individual assignments.**

(eg. a single PDF file, or a ZIP file of source code). It should be possible to verify that the required files are present (or not) and report this to the student (even if the files are inside an archive). Platform-specific formats such as MSWord or tar should be avoided unless there is an explicit reason.

- **It should be possible to put a per-assignment size limit on the submissions.**

I have seen 10-100Gb submission attempts (unnecessarily large, uncompressed images)

- **Submissions should be possible at any time.**

Students should be able to submit multiple copies of their work at any time (even after the deadline). This allows retrospective decisions to be made about extensions and special circumstances. It also allows students to "try out" the system early, and to resubmit up to the deadline. The students should be able to see all of their submissions and see which one is to be marked.

- **There should be a persistent and stable URL for the submission of each item of work.**

In particular, this should not involve manually navigating some internal system via menus / buttons / links. This allows the URL to be unambiguously published on web pages and assignment handouts.

- **A receipt or check on the submitted files is sometimes useful.**

This might be a simple checksum of the file, or a digital signature of the submission. This is useful, for example, to verify that code being demonstrated is the same as the code which was submitted.

- **Staff should be able to download all of the submissions using a scriptable command line.**

(A web interface may be desirable too). By default, this should probably download only the most recent submissions before the deadline (taking any extensions into account). But it should be possible to access other versions too.

- **Any configuration for the system (deadlines, required file, size limits, etc) should be possible by supplying a simple text file in a standard format. A web interface for editing this might be an optional extra.**

- **It is useful to remind students about "Good Scholarly Practice" when they submit their work.**

  Making them click a button to "sign" that this is their own work is something which some people might like (but probably shouldn't be compulsory).

- **Some stats are a nice extra.**
  Percentage of students submitted. Graph of submission times …

- **Some people are using Git for submission.**
  This implies a somewhat different architecture/approach, but I think this is worth some more thought. In its raw form, it doesn't meet all of the above requirements, but it does allow staff to see the development of the work over time, which is very useful. It also introduces students to the concept of version control, and to an important practical skill. Maybe there is some way of using this as the core of a system, which is exposed to varying degrees ... (?)

Exploring alternatives

Tool	Advantages	Disadvantages
Learn assignment tool	• Supports multiple submissions of any file type • Can submit any time • Supports on screen annotation for markers (for certain file types) • Supports delegated grading • Supports anonymous grading • Receipt on submission (accessed via Learn + emailed) • Instructor can download all submissions via web browser (not command line) • Various reports available • Moving to the Cloud Summer 2019	• Restricted to students enrolled on the course • No verification of file type • No size limit on submission • No double blind marking • Currently accessible only via web browser (although REST APIs are coming – apparently)
Learn test tool	• Suitable for quick 'knowledge checks'	• Restricted to students enrolled on the course
Noteable	• Service is keen to adopt users and is very responsive to requests • Supports multiple languages • Instructor sets type of environment on launch • Nbgrader extension now added	• Restricted to students enrolled on the course

	• Successfully trialled in Inf1A and Inf1-cg • Nora Vazbyte looking to develop plugin to run automated tests on code	
Moodle Coderunner	• Open-source • Runs program code submitted by students in many languages	• Restricted to students enrolled on the course • Currently pilot project run by Maths (ie not centrally supported)
GitHub Classroom	• To be tested (piloted by GeoSciences) • Students become familiar with an environment they will most likely use after graduation • Instructions can be written in markdown • Can clone all student repositories with one click	
Questionmark Perception		• Doesn't support non-standard characters • Time consuming to author
ExamOnline	• To be tested	
Turnitin	• Originality check of text • (future) originality check of code (MOSS) • Supports on screen annotation • Receipt on submission (accessed via Learn + emailed) • Offline marking supported (only via iPad)	• Restricted to students enrolled on the course • Size limit of 40MG on submission • No verification of file type • No double blind marking
WebPA	• Addresses common concerns amongst students relating to group work • Instructor can 'tweak' the variables to affect the multiplier applied to group work	• Restricted to students enrolled on the course • Robust, but would benefit from some tlc

Microsoft Sharepoint	• To be tested • Can integrate with Teams for classroom activities / submissions • Customise workflows for all stakeholders (eg students / markers / ITO)	
Course blogs	• Encourages active reflection and improves writing skills • Students become familiar with an environment they will most likely use after graduation (Wordpress) • Student can easily transfer blog on graduation as a record of previous work	• Restricted to students enrolled on the course
Make a local alternative	• Can be customised to meet specific requirements of the school	• Resource implications for continuing support