

PeerPigeon: A Web Application to Support Generalised Peer Review

David E. Millard, Patrick Sinclair, David Newman

Learning Societies Lab
School of Electronics and Computer Science
University of Southampton, UK

Abstract: Peer Review (also known as Peer Assessment) is an important technique in learning, but can be difficult to support through e-learning due to the complexity and variety of peer review processes. In this paper we present PeerPigeon, a Web 2.0 style application that supports generalised Peer Review by using a canonical model of Peer Review based on a Peer Review Pattern consisting of Peer Review Cycles, each defined in terms of Peer Review Transforms. We also demonstrate how PeerPigeon makes use of a Domain Specific Language based on Ruby to define these plans, and thus cope with the irreducible complexity of the flow of documents around a peer network.

Introduction

Peer Review, sometimes called Peer Assessment or Peer Evaluation, is an important technique for educators where students produce feedback (or grades) for each others' work. Peer Review activities can be formative or summative, and vary greatly in their complexity: from simple cases where students temporarily swap work with a partner, to sophisticated networks between Peers involving many different artifacts (Millard et al. 2007).

This diversity is a challenge to any system attempting to support Peer Review, as such a system must somehow model the flow of documents around the Peers and provide a *Peer Review Engine* that processes that model and takes appropriate actions (such as prompting participants to submit a document at the right time).

In this paper we present PeerPigeon, a Web Application that contains a Peer Review Engine that can drive any Peer Review structure. To create PeerPigeon we have had to create a canonical model of Peer Review based on *Peer Review Cycles* (the visible stages of peer review) and *Peer Review Transforms* (the invisible rules that dictate how documents move between peers within each stage).

It is our hope that the PeerPigeon Web Application will not only be used to support Peer Review, but that the PeerPigeon model of cycles and transforms will inform the work of others looking to create similar systems.

Background

Peer Review “is assessment of students by other students, both formative reviews to provide feedback and summative grading.” (Bostock, 2001). Whilst there is a long history of formative peer review in universities, particularly in English and the Arts, in the last ten years increasingly innovative approaches, often involving group production and / or review of learning outputs, have been adopted in a wide range of disciplines (Wheater et al. 2005, Gehringer, 2000). Gehringer records that the first use of computer software to support peer review was at the University of Portsmouth in 1995, before describing a web-based system developed at North Carolina State University. His paper also presents some considered strategies for ‘reviewer mapping’ and grading.

Since the early 1990s, several computer systems have been developed for performing peer assessment exercises. An early project was MUCH (Many Using and Creating Hypermedia) (Rushton et al. 1993). Other systems include Peers (Ngu et al. 1995), OASYS (Bhalerao and Ward, 2001) and Self and Peer Assessment Resource Kit (Freeman and McKenzie, 2002). Peer Grader (Gehringer 2000) is a web-based peer assessment system that allows students to

grade the assignments of other students. Several assignment styles are supported, such as reviewing research papers, research lecture material on the web, annotate lecture notes and so on. The system allows author-reviewer mapping strategies to be generated automatically or manually by the lecturer. Students are able to resubmit their work once they have received feedback from their peers.

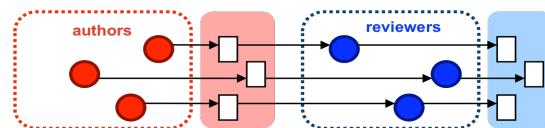
Computer Supported Collaborative/Cooperative Learning (CSCL) collaboration scripts (Dillenbourg, 2002) have inspired the design of peer assessment systems. OPAS (Online Peer Assessment System) uses a collaboration script to describe the different stages of collaborative peer assessment. Fast (Flexible Assignment System) (Topcuoglu 2006) allows different collaborative learning exercises defined by collaboration scripts, including peer assessment, to be planned and executed through a web-based interface. More recently, researchers at the OUNL have modeled an IMS-LD example of peer assessment (Miao et al. 2007).

The advantage of these later approaches is the reuse of existing standards (CSCL, IMS QTI and LD), but there is also an overhead in using more general definition languages and less specific software engines to drive them. With PeerPigeon we wanted to create a system that would specifically support Peer Review, and with a minimum requirement for middleware (i.e. avoiding the need for a complex learning design engine).

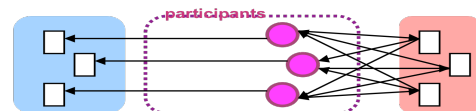
Case Studies of Peer Review Patterns

Peer Review Activities can take many forms. Figure 1 shows six case studies of Peer Review, that demonstrate that in a Peer Review activity there a number of independent factors, including the number of authors, the number of artifacts that those authors create, the number of reviewers, and the number of reviews that those reviewers return.

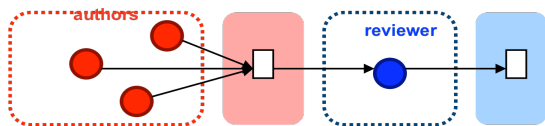
Simple - The simplest form of peer review is where authors and reviewers are paired together (Mangelsdorf 1992).



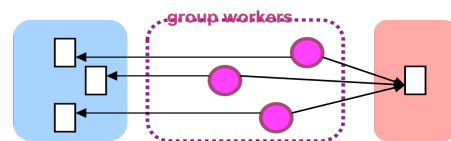
Round Robin - Where participants are grouped, and each participant reviews the work of every other participant in their group (Bostock, 2001).



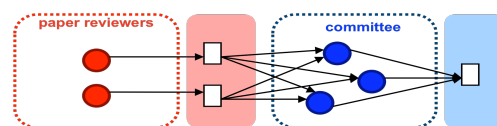
Group Activity - Where a group of authors work together to produce an artefact, and then that artefact is reviewed by a third party (from the authors own experience of an MSc supervised work session).



Group Review - Where a group of authors work together to produce an artefact, and then individually review the efforts of their group (Gregory et al. 2003)



Committee Review - Where a group of reviewers act together and look at several different artefacts in order to produce one review. In the research community we are familiar with this as the conference committee stage of peer review.



Multiplicity - Where multiple authors create multiple artefacts, which are then independently reviewed by multiple reviewers. For example, where students give a paper and presentation and are assessed by their classmates on both (Wheater et al, 2005).

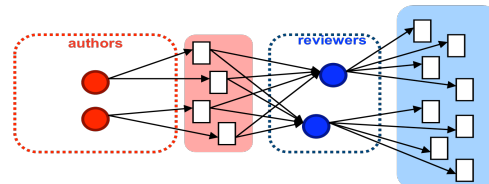


Figure 1: Examples of different peer review shapes
(authors and outputs are red, reviewers and reviews are blue)

A Generalised Model of Peer Review

To build a system that is capable of supporting different patterns of peer review it is necessary to create a generalised model of peer review, which is capable of encoding the peer review process in such a way that a peer review engine could then process it. We call this model a **Peer Review Pattern**; in PeerPigeon each Peer Review Pattern is made of a set of **Peer Review Cycles**, and each Cycle is defined with a set of **Peer Review Transforms**.

Peer Review Cycles

In previous work (Millard et al. 2007) we suggested that a canonical model of peer assessment could be constructed using *Peer Review Cycles* as a building block, where each cycle is a simple combination of three actions: Generate, Submit and Distribute (shown in Figure 2).

The complexity of Peer Review is then accounted for in three ways:

- The cycle can be started in any one of its three states. For example, to begin an activity the student may be asked to *Generate* an artifact, to *Submit* an existing artifact, or the tutor may provide it, in which case the first task is to *Distribute* it.
- The cycles can be interleaved, and can occur in parallel as well as in sequence.
- Each stage within the process may involve 1...n participants (authors/tutors/reviewers), producing 1...m resources (artifacts/ reviews/ marks).

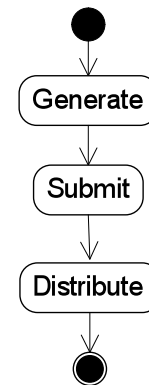


Figure 2: The basic review cycle

For example the Simple Case (from Figure 1) can be described as two iterations of the cycle. The author *Generates* an Artifact which is then *Submitted* and *Distributed* to the reviewer. The reviewer then *Generates* a review which they then *Submit* and is *Distributed* back to the artifact's author).

While Peer Review Cycles are good at capturing the stages of Peer Review (and thus describing the basic structure of the Peer Review Pattern) they do not describe the most complex part of the process, which concerns *how* artifacts are distributed, and what their relationship is to the next stage of generated artifacts.

Peer Review Transforms

The detail in a peer review process lies in the connections between Peer Review Cycles. Given two Peer Review Cycles we can express this connection in terms of a transform from the distributed artifacts in the first cycle to the generated artifacts in the second.

For example in the Simple Case this transform is straightforward:

Every authored artifact is transformed by one reviewer into a review.

The transform contains information about the number of input artifacts, how they are sent to the transforming participants, and what the expected outputs are. Figure 3 shows the transforms for each cycle in a typical academic conference. Some of the cycles have incomplete transforms, for example in Cycle 1 (Author) there is no input artifact, and in Cycle 6 (Final Paper) there is no output artifact. Cycles 4 (Decision Feedback) and 5 (Review Feedback) happen in parallel, but this is modeled in the cycles not in the transforms themselves.

Cycle	Transform		
	Input	Action / Participant	Output
1. Author	-	Authors each write	a Paper
2. Review	Each Paper	is transformed by a Reviewer	into a Review
3. Decision	Each Set of Reviews	is transformed by the Committee	into one Decision
4. Decision Feedback	Each Decision	is given to the <i>appropriate</i> Author	-
5. Review Feedback	Each Set of Reviews	is transformed by the <i>appropriate</i> Author	into a Revised Paper
6. Final Paper	Each Revised Paper	is given to the Committee	-

Figure 3: Cycles and Transforms for a Typical Academic Conference



Figure 4: PeerPigeon HomePage (left) and Participations Page (right)

There can also be informational dependencies between transforms, for example in Cycle 5 (Review Feedback) Reviews are returned to an Author, however these reviews must be matched to the *appropriate* Author, meaning the person whose work resulted in the reviews being generated. This relies on two relationships, matching reviews to papers (Cycle 2), and papers to authors (Cycle 1). Thus Cycle 5 depends on relationships created in Cycles 1 and 2.

PeerPigeon Web Application

PeerPigeon is a Web 2.0 style Web Application that allows registered users to create Peer Review Activities and circulate them to others to join. All users can create peer review activities (we make no distinction between teachers and students), but participants register for a review within a particular *role*, which affects their part in the review.

Name and Description

Schedule of dates and times for each cycle

List of people involved, one for each role in the review

Choice of upload formats (value, text or document)

Peer Pigeon
REST Services for Resource Submission and Distribution to Peer Review Groups

Home My Reviews My Participations

Home > My Reviews > Review: ELEC1052 > Edit

Editing review

Details

Name:

Review Pattern: simple

Description:

Schedule

Cycle Name	Start	End	Tolerance
Paper	01:06 22/04/2008	01:08 22/04/2008	01:09 22/04/2008
Review	01:08 22/04/2008	01:10 22/04/2008	01:11 22/04/2008
Summarize	01:08 22/04/2008	01:10 22/04/2008	01:11 22/04/2008
Revise	01:10 22/04/2008	01:12 22/04/2008	01:13 22/04/2008
Judge	01:12 22/04/2008	01:14 22/04/2008	01:15 22/04/2008

Participants

Copy/paste your participants' logins or advertise the URL for each role:
(Use a space separated list, e.g. "one two three four")

- students [how do I advertise the url?]
- committee_members [how do I advertise the url?]
- tutors [how do I advertise the url?]

Advanced participation options
Note: Update any changes to this page first

Upload Formats

Cycle Name	Accepted Formats
Paper	<input type="checkbox"/> Value <input checked="" type="checkbox"/> Text <input checked="" type="checkbox"/> Document
Review	<input type="checkbox"/> Value <input checked="" type="checkbox"/> Text <input checked="" type="checkbox"/> Document
Summarize	<input type="checkbox"/> Value <input checked="" type="checkbox"/> Text <input checked="" type="checkbox"/> Document
Revise	<input type="checkbox"/> Value <input checked="" type="checkbox"/> Text <input checked="" type="checkbox"/> Document
Judge	<input checked="" type="checkbox"/> Value <input checked="" type="checkbox"/> Text <input type="checkbox"/> Document

Logged in as: bob (drn05r@ecs.soton.ac.uk)
[Change Password](#) | [Change Email](#)

My Reviews: 1
ELEC1052
[Create a new review](#)

My Participations: 3
COMP1001
Multimedia Conference
ELEC1052

[Help](#)
[FAQs](#)
[Review Patterns](#)
[API](#)
[Blog](#)

Powered by:

Icons: Silk icon set 1.3

Figure 5: PeerPigeon Edit Review Page

Figure 4 shows the PeerPigeon HomePage and the Participations Page that lists all the reviews in which a user is currently enlisted. Pending actions are announced by email, but are also listed here (the screenshot shows that the user is involved in three reviews, one of which is waiting for him to submit a review).

A typical flow of activity for PeerPigeon might be:

1. A Teacher creates a Peer Review Activity based on an example Peer Review Pattern (such as Round Robin)
2. Each Role in the Activity (author, reviewer, committee, etc.) has its own URL which can be advertised
3. The Teacher specifies an activity start date from which cycle start and end times are automatically generated
4. Participants enroll for specific roles using the advertised URLs
5. At the allotted time the Peer Review Starts, for each cycle:
 - a. Based on the cycle's transforms participants with roles in this cycle are given an input artifact
 - b. The participants are prompted to take action (e.g. "Please review this paper and submit by April 29th")
 - c. Each participant uploads their output artifact to PeerPigeon (these become input artifacts in the next cycle)

```

review do
  roles :students, :tutors

  cycle :paper do |c|
    c.description 'Write Paper'
    c.deadline '0 days','2 days','3 days'
    c.formats :file

    c.distribution :students do |student, students, group|
      c.transform nil, student, "paper_#{student}"
    end
  end

  cycle :review do |c|
    c.description 'Write Review'
    c.deadline '2 days','4 days','5 days'
    c.formats :textarea, :file

    c.distribution :students do |student, students, group|
      c.transform "paper_#{students.wrap(student, +1)}",
        student,
        "review_#{students.wrap(student,+1)}"
    end
  end
end

cycle :receive do |c|
  c.description 'Receive Reviews'
  c.deadline '5 days','5 days','5 days'
  c.formats :label

  c.distribution :students do |student, students, group|
    c.transform "review_#{student}", student, nil
  end
end
end
end

```

```

# c.transform nil, 1, :paper_1
# c.transform nil, 2, :paper_2
# c.transform nil, 3, :paper_3

```

```

# c.transform :paper_2, 1, :review_2
# c.transform :paper_3, 2, :review_3
# c.transform :paper_1, 3, :review_1

```

```

# c.transform :review_1, 1, nil
# c.transform :review_2, 2, nil
# c.transform :review_3, 3, nil

```

Figure 6: PeerPigeon DSL for a *Simple Review Pattern* (the first pattern in Figure 1)

At any stage the Teacher can log in and see the state of the review, and download any artifacts that have been uploaded for a given cycle. Figure 5 shows the screen for editing a review activity (stage 1 in the list above). The creator's choice of pattern dictates the number of cycles and roles. The whole activity is given a name and a description; each cycle is automatically given three dates, the times of submission, distribution and closure (explained further below); the creator can add PeerPigeon users to each role (or take note of the URL and ask them to add themselves); at the bottom of the page the creator can specify the acceptable formats for each cycle, which are either a *Value* (a text field, e.g. to take a mark), a *Text Item* (a text area, e.g. to take a short comment such as a brief review), or a *Document* (an uploaded file, e.g. to take a full document such as a paper or item of coursework).

Peer Review Scripts

The major challenge in building a generalised system of Peer Review such as PeerPigeon is in specifying the Peer Review Pattern. The Pattern has to define all the cycles and transforms in such a way that it can be run automatically. We have found that in the general case the transforms are *irreducibly complex*, as there is a potentially infinite way to choose how to allocate an incoming artifact to the transforming participant; for this reason we turned away from static definitions of the Review Pattern (for example an XML document) and instead implemented the Pattern as a Domain Specific Language.

A Domain Specific Language (DSL) is a programming language designed for a specific set of tasks, it gives you a clean syntax that can be used to describe the specific steps and information, and provides an agile alternative to static descriptions (which require you to create a schema, a parser and a behavioral system to respond to parser events). PeerPigeon is written using Ruby on Rails, and this meant that it was natural for us to define the PeerPigeon

DSL using Ruby as a base scripting language. This means that the complexity of the Peer Review Plans can be dealt with on a *plan by plan* basis, using the full power of a scripting language where necessary.

Figure 6 shows an example Peer Review Plan expressed as a PeerPigeon DSL script. The review is for the Simple Review (the first pattern in Figure 1). There are three cycles: in the first the students submit their papers, in the second their papers are passed to their peers who generate reviews, in the third these reviews are passed back to the original authors.

Within each cycle the Pattern needs to tell PeerPigeon to run through a number of transforms, the exact number of transforms will vary according to the number of participants and so we cannot just list them, instead we use Ruby code to generate them. The code in the boxes shows what would be generated if we had three students in each review cycle.

Each cycle also defines a brief description, the format of the submission (either a text area, an uploaded document or a simple text field) and three deadlines. The deadlines are expressed in minutes, hours, days or weeks after the beginning of the review (which is chosen when a review with this pattern is created). The first deadline is the *time of distribution* (when the items to be transformed will be sent out), the second is the *time of submission* (when the transformed items are expected to be returned), the third is the *time of closure* (when the cycle ends, and no more submissions are permitted). In our simple example (chosen for its brevity, rather than its completeness) if a participant fails to return an item then the review cycle will continue, but that item will effectively be empty. In the simple case this is probably acceptable, however in some reviews where there are multiple stages an empty item will propagate through the remainder of the review process, and could prevent other people from participating properly. Because we have chosen the DSL approach we can optionally include exception handling in the script to cope with this; for example, by checking if the item is null and reallocating a non-null item in its place, or even by halting the review completely.

Using a DSL gives us a great deal of power and means that we can create extremely complex transforms relatively easily that would be very cumbersome to define using a declarative approach (for example, IMS-LD).

Conclusions and Future Work

In this paper we have presented PeerPigeon, a Web Application that deals with generalised Peer Review. We have shown how a canonical model of Peer Review can be constructed using Peer Review Cycles and Peer Review Transforms, and how these can be encoded using a Domain Specific Language (in PeerPigeon this is based on Ruby) to enable the generalised system to deal with transforms which are irreducibly complex, and which would be very cumbersome to define using the more traditional methods of a schema, parser and event engine.

We are currently finalizing the PeerPigeon implementation and entering a beta phase, where we will begin to use the system with real students and teachers. We are looking at the possibility of implementing further Web 2.0 features, such as a PeerPigeon Widget, and at extending the functionality of the PeerPigeon REST interface (which currently allows you to read the state of a given review, but not to create or alter one).

At present PeerPigeon includes a number of pre-written Peer Review Patterns, these can be easily extended by the PeerPigeon administrator (they are just script files in a directory on the web server) but cannot be extended or altered by users. Giving users the ability to add new plans is problematic, as allowing any user to insert arbitrary script into the system would represent a serious security hazard, and in any case the DSL is non-trivial to write and should be tested properly before being deployed. A solution would be to create a graphical authoring front end to the DSL system, which would allow users access to most (but not all) of the power of the scripting language.

In building PeerPigeon we wanted to create a useable Web 2.0 style application to support Peer Review, but also to demonstrate how a canonical model of Peer Review can be used to define even the most complex Peer Assessment processes. It is our belief that atomic and focused tools like PeerPigeon are a good fit for today's Digital Natives, and can be easily appropriated by both students and teachers into their existing ways of thinking and working. We hope that this will then allow them to create and manage new valuable Peer Review activities with the minimum of time and effort.

References

- Bhalerao, A. and Ward, A (2001) "Towards Electronically Assisted Peer Assessment: A Case Study", *ALT-J volume 9 no 1* (2001), pp. 26-37
- Bostock S. (2001), "Student peer assessment", *Higher Education Academy Article*, 16 Mar 2001
- Dillenbourg, P. (2002), "Over-scripting CSCL: The risks of blending collaborative learning with instructional design.", *Three worlds of CSCL. Can we support CSCL*, P.A. Kirschner, eds., 2002, pp.61-91.
- Freeman, M. and J. McKenzie (2002). "Implementing and evaluating SPARK, a confidential web-based template for self and peer assessment of student teamwork: benefits of evaluating across different subjects". *British Journal of Educational Technology* 33(5): 553-572.
- Gehring, E.F (2000). "Strategies and Mechanisms for Electronic Peer Review". *Frontiers in Education Conference FIE'2000 30 Annual, Kansas City, MO*, October 18–21, 2000, IEEE Education/Computer Society, 1, F1B/2–F1B/7
- Gregory, A., Yeomans, E. and Powell, J. (2003) "Peer Assessment and Enhancing Students' Learning", in *Learning and Teaching for Business: Case Studies in Successful Innovation* (eds. Kaye, R. and Hawkrigde, D), London, Kogan Page, 2003
- Mangelsdorf K. (1992) "Peer reviews in the ESL composition classroom: What do the students think?", *ELT Journal* 1992 46(3):274-284; doi:10.1093/elt/46.3.274, 1992
- Miao Y., Tattersall C., Schoonenboom J., Stevanov K. and Aleksieva-Petrova A. (2007), Using open technical e-learning standards and service-orientation to support new forms of e-assessment, *TENCompetence OPen Workshop*, Manchester, 2007.
- Miao, Y., & Koper, R. (2007). "An Efficient and Flexible Technical Approach to Develop and Deliver Online Peer Assessment". In Proceedings of the 7th Computer Supported Collaborative Learning (CSCL 2007) conference 'Mice, Minds, and Society' (pp. 502-511). July, 16-21, 2007, New Jersey, USA. ISSN 1819-0146.
- Millard, D., Fill, K., Gilbert, L., Howard, Y., Sinclair, P., Senbanjo, D. and Wills, G. (2007) "Towards a Canonical View of Peer Assessment". In: International Conference on Advanced Learning Technologies (ICALT) 2007, July 18-20, Niigata, Japan.
- Ngu, A. H. H., J. Shepherd, et al. "Engineering the 'Peers' system: the development of a computer-assisted approach to peer assessment." *Research and Development in Higher Education* 18: 582-587. 1995
- Rushton, C., Ramsey, P. and Rada, R. (1993), "Peer assessment in a collaborative hypermedia environment: a case study", *Journal of Computer-Based Instruction*, vol 20(3), 75. 1993
- Topcuoglu, H. (2006), "FAST - Flexible Assignment System", Proceedings of E-Learn 2006, Honolulu, Hawaii .
- Wheater C.P, Langan A.M. and Dunleavy P.J., (2005) "Students assessing student: case studies on peer assessment", *Planet*, 15, 2005, pp. 13-15.