Hot Crap!

Eddie Kohler, UCLA

HotCRP, a web-based conference submission and review package designed for flexibility and ease of use during reviewing, was initially written in summer 2006 as the submission system for HotNets V, which I co-chaired with Greg Minshall. It currently consists of about 20,000 lines of PHP plus 500 lines of Javascript, 1200 lines of CSS, and some others, and has been used for two SIGCOMMs, two USENIX Annual Technical Conferences, NSDI, ISCA, SOSP, and others. This paper discusses its design principles (and how they differ from other systems), my experiences with ongoing development and with anonymity, some interesting bugs, and general thoughts on conference review.

Heartfelt thanks to the generous Dirk Grunwald, author of HotCRP's ancestor CRP. Without CRP, HotCRP would not exist (although since essentially all CRP code has been replaced, any bugs are mine).

Design principles Two principles guided HotCRP's UI: *reduce modes* and *prefer search*. These principles make systems sense as well as UI sense and any review package would be improved by following them.

First, reduce modes. The terminology comes from graphical user interfaces. A UI "mode" constrains the actions a user may perform; in the most restrictive modes, such as those induced by error dialog boxes, a user might have only one option ("Click OK to continue"). A *modeless* UI, in contrast, avoids constraining the user: all options are always available. Modeless UIs give users more freedom to design their own workflows, and even early Macintosh user interface guidelines recommended minimizing modes (specifically, modal dialog boxes). For instance, consider how browsers have evolved to report DNS errors and the like. Previously, a modal dialog reported the error; now, error information is displayed modelessly in the browser window itself.

Much of my dissatisfaction with other conference review systems arose from violations of this principle. In CRP, for example, a paper's "PC member view" for a paper doesn't normally display reviews, while the "reviewer view" doesn't display abstracts. Similar issues beset START (www.softconf.com), Linklings's RM 3.2 (www.linklings.com), and EasyChair (www.easychair.org). For instance, EasyChair users switch between "roles," such as "PC member," "reviewer," or "author," getting very different views of paper information in each case. I've even served on committees on which the easiest way to use the conference system was to edit URLs. HotCRP, in contrast, aims to display all accessible paper information on a single "paper view" page. This page shows each viewer exactly the information the viewer is allowed to see, taking conflicts of interest, author state, and PC status into account.

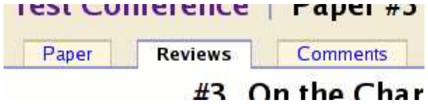


Figure 1: Maybe a blinking arrow would help.

Modelessness improves code structure too. The more ways there are to view a paper, the more code must be checked or updated when system functionality changes, and the more potential channels there are for information leaks or bugs. Inevitably some views will be forgotten. HotCRP aims to implement all paper views and paper viewing policies once each. This makes development harder in the short term—precisely expressing a policy is surprisingly difficult—but hopefully more consistent in the long term. In the end, HotCRP has three pages per paper, for paper viewing and editing, review viewing and editing, and comment viewing and editing. Where CRP ships with 119 user-visible pages, HotCRP currently ships with 26, and HotCRP probably has more features. Dynamic HTML streamlines the UI while keeping all information immediately accessible; for instance, on review pages, abstracts are collapsed by default. A single combined page might be an improvement—some users don't notice the tabs that navigate among pages (fig. 1)—but the combination seems too complex.

HotCRP reduces modes even to the extent of refusing to support certain conference policies. CRP, among other systems, requires reviewers to "finalize" their reviews. A review in "finalized" mode cannot be changed. This supposedly ensures review independence, since a reviewer cannot see other reviews until her own review is frozen. This is nasty. Fixing a single typo in a review requires interactions with the chair. Reviewers cannot learn from each other. Although groupthink is worth avoiding, finalization prevents more benign interactions, such as sharpening arguments when reviewers disagree. Every CRP conference I'd been involved with had specifically *encouraged* groupthink by unfinalizing all reviews after the PC meeting, the intention being for outliers to explain changes of heart inspired by PC discussion! Finally, the whole policy seems unnecessary: once a review is written, how many busy PC members will actually rewrite the review? Those members that *would* rewrite their review for the wrong reasons could probably subvert the process in some other way. HotCRP hides other reviews until a reviewer enters her own review, but allows arbitrary edits thereafter. PC members should be trusted by default.

The second UI principle is to prefer search. Many PC member and chair operations reduce to searching the current paper collection. In HotCRP all paper lists are formed by a common search library. Search is smart: entering a paper number takes you directly to that paper; keywords like "au:," "tag:," and "review:" select specific fields. This is great for users, although care is required to ensure search does not expose inappropriate information.

A secondary design principle was to implement mechanisms rather than policies. This idea can be abused [4], but it works well for user interfaces. For contrast, consider the Continue system, which "implements Oscar Nierstrasz's 'Identify the Champion' pattern for program committees" [6]. This imposes reviewing scales of A–D for championability, a form of overall rating, and X–Z for expertise; Continue further color-codes the scores, putting "the most visually striking colors . . . where there is most need for discussion due to the greatest variance of opinion" [6]. These scales are useful and concise—the programming language community has largely settled on them—but still just one policy among many. HotCRP can implement this and other policies, and the advantages of color coding are achieved in more general ways. The best example is the paper tagging system, which lets PC members and chairs create and manage arbitrary paper sets and orders.

Other conference review packages are based on their own principles, some of which lead to interesting alternatives. For instance, START's author interface is designed to simplify submission, with the goal of attracting new submitters: "[Registration procedures create] an unnecessary 'barrier of entry' to the conference. As a 'regular participant' in conferences, you may find it hard to believe that this kind of thing makes a difference. However, after five years of experience with START, we have discovered that it does make a small difference in the number of submissions received. Simply put, if a potential author can click directly to your submission page, he/she is more likely to submit to your conference." [9] Whether or not systems conferences need more submissions, START's one-page, directly-accessible submission form is cleaner and simpler than HotCRP's process, which includes an email validation step. The START "passcode" system for submission updates is particularly attractive: each paper is assigned a random passcode, and anyone who knows the passcode can edit the paper or view reviews. Unfortunately I have not found START's review process as simple, clean, or principle-based.

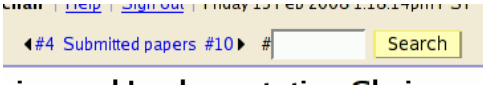


Figure 2: Quick links between papers.

Probably the most interesting alternate review package is Andrei Voronkov's EasyChair (www.easychair.org), which is popular and usually lightning fast. One of its apparent development principles is reducing the information available on any one page. The resulting UI often looks clean and affords fewer opportunities for bugs, but makes it difficult to navigate from paper to paper. "Notes" let reviewers enter private thoughts into a per-paper scratch area, useful while preparing a review. The "events" subsystem is particularly nice. All reviews and comments are sorted and displayed by date, facilitating review conversations. PC members configure a set of papers called the "watchlist" (by default the PC member's review assignment); they receive email when reviews or comments for the watchlist change, and the system exposes a web page listing watchlist events in reverse chronological order. HotCRP would benefit from these features. While EasyChair would benefit from, for example, HotCRP's powerful search facility and consistently clean visuals, the "reduce modes" principle, which argues that most or all paper information should be accessible on every paper page, directly conflicts with EasyChair's minimal UI design.

Ongoing development Minimal user interfaces, such as EasyChair's, can be relatively simple to implement, but user expectations also stay simple—a perhaps unexpected advantage. In contrast, HotCRP's more advanced features often inspire improvement requests. For example, links on paper pages point to the previous and next papers in a list (fig. 2), letting users quickly page through a list in order. This is a mechanism, not a policy: *any* search list can be a source of quick links. For simplicity, each user initially had a single current list, but this meant searching for another paper in a separate tab or window lost the current list state. Now many lists are kept per session, and links between papers must explicitly include the intended list.

Surprisingly many users have asked to limit HotCRP's functionality or strengthen its treatment of conflicts of interest. At an architecture conference, there were complaints that PC members could read papers they were not assigned to review. (What does program committee membership mean if not "can read papers submitted to conference"?) Chairs themselves are not always comfortable with their ability to view or modify reviews for any paper, including conflicts. (HotCRP reports conflicts and hides information by default, but chairs and admins can explicitly override their conflicts.) There have been requests for doling out administrative privilege paper by paper. Some

PC members have even complained that they were able to modify *their own* reviews after submitting them. There's a fundamental difference here in approaches to managing conflicts of interest. There's no way to constrain chairs, reviewers, or authors to behave strictly correctly. Most constraints seem to limit useful behavior, such as review editing, collecting additional reviews from interested PC members, or system administration, and worse, locking down the process seems to make PC members and authors more suspicious, poisoning the atmosphere. The most serious misbehavior occurs openly in reviews and the program committee meeting anyway: PC members try to bully the group into swallowing their opinions, take joy in killing work based on secondary complaints, skim, farm out reviews without reading the papers, write reviews with malice, and use their debating skills skew the program. I think these problems should be addressed not through constraints, but through openness: when misbehavior is exposed to the PC, it might be correctable in the long term. (Several other WOWCS submissions think along these lines.) So far HotCRP is willing to discourage misbehavior only as far as flexibility can be preserved.

HotCRP does not support CRP's paper grading phase, where all PC members enter "grade" score for each paper based mostly on its reviews. I didn't think this was worth the burden—in my experience, PCs generally skipped the step—and eliminating review "finalization" captures some of its benefits. However, Dirk Grunwald, CRP's author, considers paper grading one of CRP's more important features: "[a]s a community, we kept saying that ISCA (and MICRO etc.) were the journal equivalent places to do research, but our review standards were significantly lower than that of journals. Inserting a rebuttal mechanism was the first step towards improving that process. The others, were the 'grading' phase and the emphasis on pre-meeting deliberation. . . ." [5] Perhaps a paper ranking [3] would be a good replacement for the paper grading phase; reviewers might rank the papers they reviewed or, as in CRP, every PC member might rank all submitted papers.

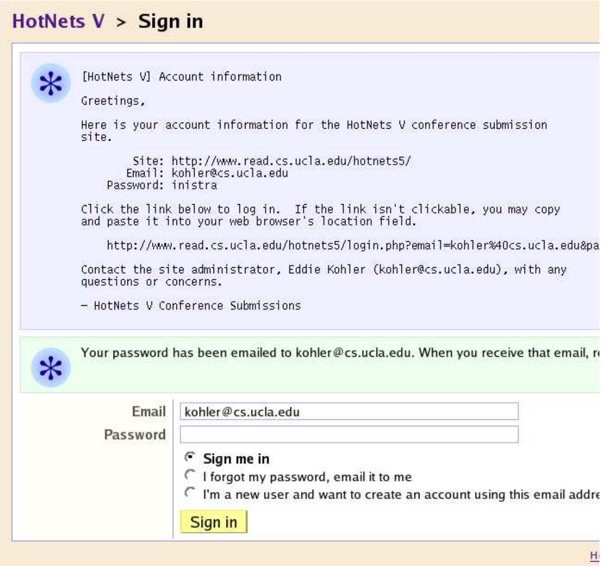
HotCRP also does not support CRP's reviews of reviewers, where paper authors can rank reviews on how helpful they were. So far no conference has been interested, although several WOWCS submissions suggest a similar feature. Allowing *reviewers* to review other reviews might be the right tweak on this feature.

Web development is fantastic, but (a cheap metaphor for systems research?) the sheer number of technologies involved makes it difficult to achieve the feeling of permanence and solidity of a good Unix tool. There's too many chances to go wrong. For instance, which header should be used to make a page uncacheable: "Cache-Control: no-cache," an "Expires:" with a date in the past, or both? The answer is "Expires": "Cache-Control" forces browsers to reload the page when the Back button is pressed, losing values entered in forms, such as review preferences for a hundred papers. (Sorry, Andrew Myers.) Another implementation day was lost figuring out how to fit non-ASCII characters into mail subject lines. And don't talk to me about databases.

CRP was hosted on Sourceforge. Dirk Grunwald "found that the 'open source' model under sourceforge was problematic (because other people broke things & I needed to fix them)." [5] The main benefit of open source for HotCRP has been the prior availability of CRP, a huge benefit. I've gotten few useful patches, but many more useful problem reports and feature requests.

Anonymity HotCRP's most underappreciated feature is, I think, its support for optional anonymity. This trivially solves the issue of double- vs. single-blind review: authors fearful of bias, misunderstanding, or long-term consequences can submit anonymously; most authors will choose not to. Selective anonymity was the excuse for building HotCRP in the first place. Greg Minshall and I were, as chairs, skeptical of the benefits of double- and single-blind review. In other program committees I had seen some pretty low-quality reviews—a handful of sentences, really nothing more than a score. Procrastination had led me to turn in suboptimal reviews of my own. I felt that openness would discourage drive-by reviewing. However, since imposing openness on a whole PC would be unfair, the system should allow, but not require, reviewers to sign their reviews. Any system lets a reviewer put their name into the body of their review, but an explicit "anonymity" checkbox would encourage reviewers to consider their choice. For consistency, submitters should also be able to choose whether to submit their papers anonymously. This would also let us run an experiment on the consequences of anonymity. Since existing review packages enforced double- or single-blind review for all submissions, to support selective anonymity we'd have to do something ourselves.

Just over one-fourth of the 114 HotNets submissions were anonymous—fewer than we expected. Only 6.9% of anonymous submissions were accepted, compared to 24.7% of non-anonymous submissions. This is also reflected in the overall merit scores, where anonymous submissions did substantially worse. Rejected anonymous submissions came from top worldwide industrial research labs and top US universities, among other places. Although it is possible that our reviewers were biased against anonymous papers, I believe the anonymous submissions were simply lower quality. For instance, some anonymous submissions from well-known authors seemed constructed at the last minute. Submitters seem less willing to put their names on a less-than-top-quality submission. This does not argue for universal anonymity.



HotNets V > Sign in

```
[HotNets V] Account information

Greetings,

Here is your account information for the HotNets V conference submission
site.

        Site: http://www.read.cs.ucla.edu/hotnets5/
       Email: kohler@cs.ucla.edu
    Password: inistra

Click the link below to log in.  If the link isn't clickable, you may copy
and paste it into your web browser's location field.

    http://www.read.cs.ucla.edu/hotnets5/login.php?email=kohler%40cs.ucla.edu&pa

Contact the site administrator, Eddie Kohler (kohler@cs.ucla.edu), with any
questions or concerns.

- HotNets V Conference Submissions
```

Your password has been emailed to kohler@cs.ucla.edu. When you receive that email, r

Email: kohler@cs.ucla.edu
Password:

● Sign me in
○ I forgot my password, email it to me
○ I'm a new user and want to create an account using this email addre

Figure 3: Painful.

Although both Greg Minshall and I had planned to sign our reviews, in the end, only Greg and one of our external reviewers did so. I was not sure that my own reviews were sufficiently careful, high quality, and genteel. This seems justified since one of them was recently criticized as emotionally immature [8]. I have signed reviews for other conferences since, especially USENIX (which seems to have a nicely down-to-earth submitter population), and the result has been useful engagement with the authors. However, as I am exposed secondhand to misplaced author anger ("Did you write one of the reviews that killed my paper?") and more-or-less petty academic squabbles, I feel more reluctant to sign reviews, not less. There should be positive incentives to sign reviews.

Information exposure HotCRP has had a humbling number of bugs that inappropriately exposed paper information. The most interesting was the first. In "email debug" mode, the system does not send email to users; rather, it displays the constructed email in the response HTML page. This is usually perfectly safe—most emails are of the form "Dear author, this mail confirms that you just changed your paper." However, like many web systems, an "I forgot my password" login option will send the named user an email containing their password. In email debug mode, any user could use this option to find out any other user's password—for instance, mine (fig. 3). A user reported this bug immediately before HotNets V's reviews were released. My appreciation for the report was tempered: the user (actually, one of their students) had used my account in the meantime to download several of the accepted papers. For me, this bug demonstrates in microcosm all the claims I and others have made about the difficulty of secure programming. More security-minded programmers wouldn't have perpetrated this bug, but they might not have written the system in the first place: the mindset required to write new features seriously differs from the mindset required to enumerate how combinations of features can be abused. I wish information flow control systems were usable in practice. A flexible information flow control layer would have prevented all of HotCRP's information exposure bugs. It's not clear, however, that any extant information flow control system could handle HotCRP's policies; see fig. 4. Later information flow exposure bugs could have been prevented by reducing the number of system modes—for instance, one bug affected fully anonymous conferences (HotNets was optionally anonymous), another affected rebuttals (HotNets didn't have a rebuttal period), and another affected the textual format for downloaded reviews (the web format was fine). At this point I think all serious information flow exposure bugs are fixed, but if you find one, I will not pay you $2.56.

```
return (($this->privChair && $forceShow)
  || ($prow->timeSubmitted > 0
      && (($prow->conflictType >= CONFLICT_AUTHOR
           && $conf->timeAuthorViewReviews() && $rrowSubmitted
           && (!$this->reviewsOutstanding || !$this->isReviewer))
          || ($this->privChair && $prow->conflictType == 0)
          || ($this->isPC
              && $prow->conflictType == 0 && $rrowSubmitted
              && ($conf->timePCViewAllReviews()
                  || defval($prow, "myReviewSubmitted") > 0))
          || (defval($prow, "myReviewType") > 0
              && $prow->conflictType == 0 && $rrowSubmitted
              && defval($prow, "myReviewSubmitted") > 0
              && ($this->isPC
                  || $conf->settings["extrev_view"] >= 1))
          || (defval($prow, "myReviewType") == REVIEW_SECONDARY
              && $prow->conflictType == 0 && $rrowSubmitted
              && $prow->myReviewNeedsSubmit === null)
          || ($rrow && $rrow->paperId == $prow->paperId
              && $rrow->contactId == $this->contactId))));
```

Figure 4: Can this user view a paper's reviews?

Review form design I close with more general thoughts on the review process, in several sections.

No commonly reviewed quality is as subjective as excitement. The difference between "exciting but flawed" and "flawed but flawed" lies mostly in the authors' talent for writing and spin. It is not clear whether this talent should be prized. "Exciting and not flawed" papers, though rare, are generally liked and need no special help. "Novelty" scores are similarly subjective; I'm not sure it captures anything useful. Although perceptions of reviewer expertise are extremely variable, the score is still useful enough to keep.

Review scores should have five options at most. For overall merit, the four-value scale suggested by "Identify the Champion" [7] works well.

The best way to keep reviewing viable in the face of increasing submission load is to reduce the burden on reviewers. This means reducing the number of fields in the review form (do you really need "1–3 sentences on why the paper should be accepted" and "1–3 sentences on why the paper should be rejected"?) and introducing a community standard that bad papers get brief reviews. Although this would make it more difficult for authors to improve, perhaps publishing all reviews for accepted papers would help those who wanted to be helped. It is not the community's job to help the others, such as those who appear never to have read a research paper.

A minimal but complete form, worth considering: Overall merit (4 options), Reviewer expertise, Comments to author, Comments to PC. The next fields to add would be Reasons to accept and Reasons to reject.

Reviewing goals Reviews should grapple with a paper, and reviews that go out of their way to be nice are often superficial. Tone is a secondary concern. Your job as a reviewer is to explain your opinion and your vote to the authors. Sugarcoating does no one any favors. The skill of how to learn from a review is more important than the skill of how to write a review. (So why the recent boomlet of reviewing advice and recommendations?)

It is sometimes necessary and appropriate to punish an author's paper *n* for a flaw of its predecessors.

I like reading papers that contain tricks or ideas I could use in my own systems. Authors should describe not why their system is good, but why its ideas are worth reusing. This perspective can be criticized as incremental—"big advances" are not immediately usable—but progress is incremental. Conferences that lose track of usefulness quickly become academic, in the pejorative sense.

The purpose of a program committee meeting is for the people who read the paper to generate consensus in the people who didn't read the paper. If the people who didn't read the paper are expected to stay quiet, then there should be no program committee meeting.

Conferences in general Conference review produces random results for medium-quality papers. There's not much difference in quality between the lowest-ranked accepted papers and the highest-ranked rejected papers. There is no way to control this: the best papers published in a conference are much better than the second tier, but while the first tier is too small to stand on its own, the second tier is too large to treat uniformly. Additional reviews do not greatly affect the randomness of the result. Consider the anecdote of "[o]ne particularly controversial submission" that "received nine reviews before being accepted" [1]. Swapping this for a comparable rejected paper might not change either program quality or the total amount of worldwide author happiness (it is impossible to know without understanding the controversy). Additional reviews should be used, if at all, to improve the reviews given to the authors, not to improve the set of accepted papers. If program chairs believe reviewers are wrong, let them accept papers without program committee consensus (a power too little used).

Many parts of the conference review cycle, including shepherding and rebuttals, don't greatly change the quality of the program. Rebuttals help authors blow off steam, and shepherding is occasionally useful, but a model where the shepherd is available to answer the authors' questions matches reality better than a model where the shepherd "approves" the final version.

A 14-page conference paper is too long. The limit encourages running on at the mouth, particularly in the initial motivational sections—papers that go for five pages without describing a technical idea are no longer uncommon. In many of the sciences authors must describe complex ideas in short papers; we should set the same goal. Conference papers should not be longer. Even 14-page papers cannot be described in 25 minutes; longer ideas demand journal presentation. I would like to review extended abstracts, but the extended abstract must contain fully described technical ideas as well as motivation.

The proliferation of conferences makes journals increasingly attractive.

Some of the profit from running a conference, if any, could be used to hire a professional editor. *;login:* is better edited than any conference, which is too bad.

Sheridan, ACM's print shop, pointlessly enforces early deadlines on accepted authors. ACM's conference styles are ugly and inconsistent (unlike its journal styles) but Sheridan pointlessly enforces them too.

Future work The following ideas from other WOWCS submissions are attractive enough for me to want to implement them: ranking rather than (or in addition to) rating, public distribution of (anonymous) reviews, public distribution of all submissions (a great idea), publication of paper rankings, and reviews of reviewers. Specific thoughts on how these features should work would be welcome. The following ideas are not as attractive, because either they would constrain flexibility or they disagree with my taste: constraining reviewers' use of certain scores, constraining author feedback, and privately passing submissions and reviews from conference to conference (public distribution is better). The effectiveness of peer pressure in improving process quality may have been exaggerated [2]; peer pressure can lower standards as easily as raise them, and arguably it has.

Conclusion My favorite HotCRP compliment came from Adnan Darwiche, who said something like "I like your system, it—moves *with* you." That's what I wanted.

Thanks to Dirk Grunwald, Greg Minshall, Anja Feldmann, Bernhard Ager, Jeff Chase, Frans Kaashoek, Jim Larus, Matthew Frank, Stefan Savage, Geoff Voelker, Jon Crowcroft, Akos Ledeczi, and anyone who has used HotCRP.

References

[1] T. Anderson. Towards a model of computer systems research. In *Proc. Workshop on Organizing Workshops, Conferences, and Symposia for Computer Systems*, San Francisco, Apr. 2008.

[2] J. Crowcroft, S. Keshav, and N. McKeown. Scaling Internet research publication processes to Internet scale. In *Proc. Workshop on Organizing Workshops, Conferences, and Symposia for Computer Systems*, San Francisco, Apr. 2008.

[3] J. R. Douceur. Paper rating vs. paper ranking. In *Proc. Workshop on Organizing Workshops, Conferences, and Symposia for Computer Systems*, San Francisco, Apr. 2008.

[4] D. R. Engler, M. F. Kaashoek, and J. O'Toole. Exokernel: An operating system architecture for application-level resource management. In *Proc. 15th ACM Symposium on Operating Systems Principles*, pages 251–266, Copper Mountain Resort, Colorado, Dec. 1995.

[5] D. Grunwald. Personal communication, 7 Feb. 2008.

[6] S. Krishnamurthi. The Continue server (or, how I administered PADL 2002 and 2003). In *Proc. 2003 Symposium on the Practical Aspects of Declarative Languages*, New Orleans, Jan. 2003.

[7] O. Nierstrasz. Identify the champion: An organisational pattern language for programme committees. In N. Harrison, B. Foote, and H. Rohnert, editors, *Pattern Languages of Program Design 4*, pages 539–556. Addison Wesley, 2000. URL http://www.iam.unibe.ch/~oscar/Champion/.

[8] T. Roscoe. Writing reviews for systems conferences, Mar. 2007. URL http://people.inf.ethz.ch/troscoe/pubs/review-writing.pdf.

[9] Softconf.com. Softconf.com – software for conferences – author view. URL http://www.softconf.com/index.php?option=com_content&task=view&id=17&It%emid=46.