Using Intelligent Task Routing and Contribution Review to Help Communities Build Artifacts of Lasting Value

Dan Cosley, Dan Frankowski, Loren Terveen, John Riedl
CommunityLab*
University of Minnesota
Minneapolis, MN 55455
{cosley, dfrankow, terveen, riedl}@cs.umn.edu

ABSTRACT
Many online communities are emerging that, like Wikipedia, bring people together to build *community-maintained artifacts of lasting value* (CALVs). Motivating people to contribute is a key problem because the quantity and quality of contributions ultimately determine a CALV's value. We pose two related research questions: 1) How does *intelligent task routing*—matching people with work—affect the quantity of contributions? 2) How does reviewing contributions before accepting them affect the quality of contributions? A field experiment with 197 contributors shows that simple, intelligent task routing algorithms have large effects. We also model the effect of reviewing contributions on the value of CALVs. The model predicts, and experimental data shows, that value grows more slowly with review before acceptance. It also predicts, surprisingly, that a CALV will reach the same final value whether contributions are reviewed before or after they are made available to the community.

Author Keywords
online communities, contribution models, intelligent task routing, member-maintained, Wikipedia, editorial review

ACM Classification Keywords
H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces—Collaborative computing

INTRODUCTION
Wikipedia members have collaboratively produced over one million encyclopedia articles in dozens of languages since 2001. Distributed Proofreaders users have produced thousands of books for Project Gutenberg [13]. ESP Game players have contributed millions of labels to an image database [18]. 35,000 RateYourMusic.com users are building themselves a music database, while freedb.org's online music service receives thousands of weekly CD submissions. The common theme: groups of volunteer contributors building community-maintained artifacts of lasting value (CALVs).

The content of CALVs is meant to be persistent and have value to the entire community. This raises many issues, from "who is the community?" (are trolls part of a Usenet group?) to "what is valuable?" (comp.lang.perl split into subgroups partly because of frustration with novice questions). In this paper, we focus on two fundamental, related problems that communities building CALVs must solve: motivating people to contribute and ensuring that contributions are valuable.

Research Question 1: How does intelligent task routing affect contributions to a CALV? Thorn and Connolly analyzed the problem of encouraging contributions using discretionary databases [16], an abstract model that applies reasonably well to CALVs. A key problem is that discretionary databases are public goods [6]. That is, everyone can consume the information without using it up for others. It is rational for individuals to consume information but not to produce it because contributing has costs. Some people contribute despite the cost [4], but the community as a whole suffers because all would be better off if all contributed.

Reducing costs can encourage contributions. Wikipedia allows almost anyone to edit almost anything, making it easy to find work to do. Distributed Proofreaders uses mentors to teach new members. The ESP Game makes contributing fun. freedb.org uses music software to semi-automatically submit information. RateYourMusic.com piggybacks on the value people gain by maintaining their music collections.

We explore a computational approach to reducing contribution costs, *intelligent task routing*, that leverages social psychology theory to match people with appropriate tasks. The collective effort model [8] suggests a number of factors that affect people's motivation to contribute to groups. Online communities that know something about their members may be able to match people with tasks based on attributes of the tasks that map to factors in the collective effort model. We develop several general and simple task routing algorithms that most communities could implement by choosing items:

* CommunityLab is a collaborative project of the University of Minnesota, University of Michigan, and Carnegie Mellon University. http://www.communitylab.org/

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CHI 2006, April 22-27, 2006, Montréal, Québec, Canada.
Copyright 2006 ACM 1-59593-178-3/06/0004...$5.00.

- that a given user will probably like
- that a given user has experience with (i.e., has rated)
- that need work (like Wikipedia's Community Portal)
- randomly (like Slashdot meta-moderation)

It turns out people are much more likely to contribute when asked to edit items they have rated.

Research Question 2: How does reviewing contributions before accepting them affect the value of CALVs? Just contributing is not enough. Because Wikipedia allows anyone to contribute, and because these contributions become visible right away, it has become a lightning rod for the problem of ensuring contributions are valuable.[1] Reviewing contributions is a natural strategy, but designing effective review systems is hard because reviewing behavior is complex. For example, the timing of contributions affects whether they receive adequate review in Slashdot [9]. Viégas et al. use visualizations of editing behavior in Wikipedia to understand how its review mechanisms improves quality by helping members to repair vandalism and to negotiate disagreements over content [17]. In our work we focus on how the structure of review mechanisms affects contribution behavior. We studied whether review is needed at all in prior work. It is, but in our domain, peers were as effective as experts [3].

Here we focus on another important structural question: should contributions be reviewed before being added to the CALV, or can they be added first and reviewed later? Both approaches can succeed. Distributed Proofreaders and RateYourMusic require contributions to be reviewed before they are added to the CALV. In Wikipedia, contributions are immediately available and are reviewed by their consumers. Both approaches can also fail. The ChefMoz.org restaurant directory is impeded by its slow contribution review process, while a 2005 *LA Times* experiment with wiki-based editorials ended, overrun by vandals, in just two days.

We develop simple mathematical models of contributors' behavior to explore the effect of reviewing contributions before and after inclusion. The models predict that review before inclusion hurts the CALV's short-term value because of review overhead. Surprisingly, the models predict no gain in long-term value for review before inclusion. We compare the models to data from MovieLens, where some contributors used a review before inclusion system while others used review after inclusion. It turns out that the model is correct that making contributions available immediately wins in the short term, and it fits the data fairly well. However, we do not have enough data to evaluate the long-run prediction that both systems will achieve the same level of value.

EXPERIMENTAL DESIGN AND OVERVIEW

We address our research questions using field experiments with the MovieLens recommender system, a web site with thousands of active users per month. MovieLens contains about 8,800 movies that members can rate and receive recommendations for. It keeps a modest amount of information about movies, including directors and actors, movie genres, languages, release dates, and video availability. Members can use this information to filter recommendations.

[1]e.g., http://slashdot.org/article.pl?sid=05/08/05/2012229.

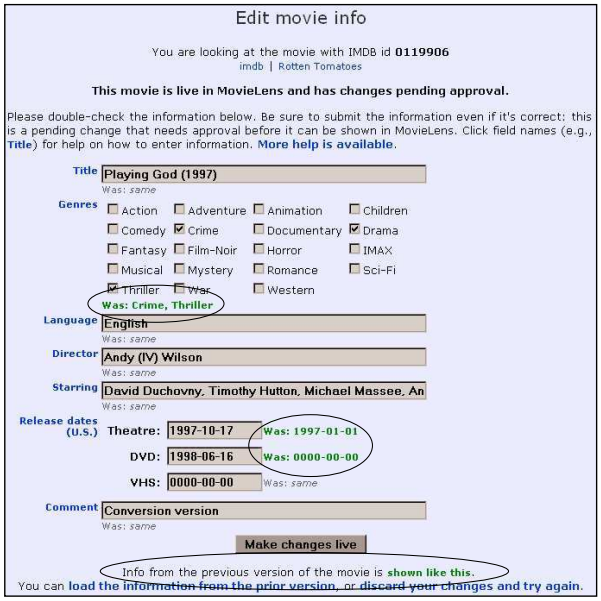

Figure 1. The interface for editing movie information. The previous editor's changes are highlighted.

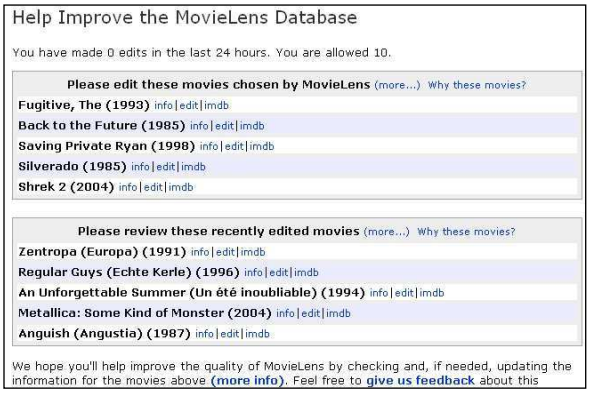

Figure 2. The front page interface, showing the five visible movies on the chosen and recently edited lists.

Its movie information is incomplete. For most of its life the MovieLens database has been maintained by a single movie guru. When the guru is busy, the database suffers. Sometimes he does not add actors and directors, movies released on DVD are not always updated as "new DVDs," and so on. About 1/3 of the fields in the database are blank. This has a direct impact on the value of MovieLens, for example, when searches fail to return relevant movies.

To improve the database, we created an interface that allows MovieLens members to edit information for movies (Figure 1). The interface highlights fields changed by the last editor of a movie and displays text appropriate for a subject's

Strategy	Description	Theoretical justification	Practical justification
HighPred	Choose movies a subject is predicted to like.	People often find recommended items valuable; personal value increases motivation.	Almost any community that models its members can compute some notion of relevance.
RareRated	Choose movies the subject has rated but few others have.	Editing is easier if you know the movie; easier tasks increase motivation. Further, rareness implies a special ability to contribute; knowing that contributions matter increases motivation.	Also easy to implement, and targets effort toward items that might not otherwise be corrected.
NeedsWork	Choose movies with the lowest *Nfields* value.	More missing information allows more valuable contributions; knowing that contributions matter increases motivation.	Chooses tasks that provide the maximum potential to increase value. Similar to Wikipedia's Community Portal.
Random	Choose movies randomly.	Baseline algorithm.	Easy and provides wide coverage. Used by Slashdot meta-moderation.

Table 1. Four algorithms for matching people with tasks.

experimental group. We publicized the ability to edit movie information by asking people to edit selected movies to edit on the MovieLens main page (Figure 2). Prior to this, only the guru had the power to edit movies in the database.

The main page displays a list of chosen movies and a list of recent movies. The chosen list contains movies selected by one of four intelligent task routing algorithms. The recent list, similar to Wikipedia's recent changes, contains the most recently edited movies. Each list showed five visible movies and a link to 15 more. Members were also able to edit information for any movie they saw while using MovieLens.

Subjects and conditions

MovieLens members who logged in during the experiment were randomly assigned one of four task routing algorithms shown in Table 1. They were also randomly assigned to one of two contribution review systems: ***Wiki-Like***, where contributions were made visible immediately, or ***Pre-Review***, where contributions needed to be reviewed by another ***Pre-Review*** subject before becoming visible.

We placed few restrictions on editing. Members who had been with MovieLens for at least one month were eligible. We limited subjects to 10 edits day to make the analysis less sensitive to huge outliers. We debated this—why not let users do what they want?—but it also has reasonable practical justifications: involving more users increases community robustness, while industrious subjects are discouraged from writing scripts to automate contributions using data from other websites (which at least one person did!). Finally, ***Wiki-Like*** subjects were not allowed to edit movies pending review by a ***Pre-Review*** subject, and ***Pre-Review*** subjects were not allowed to review their own edits.

Metrics

We used three metrics. A crucial aspect of whether a community succeeds in building a CALV is how much value the community creates. A coarse metric, *Nedits*, counts the number of edits people make. A finer-grained metric, *Nfields*, counts the number of non-blank fields for a movie. *Nfields* is not very precise because it cannot detect when bad information is corrected. We chose it because many communities should be able to develop similar syntax-based metrics. More nuanced value metrics are possible, such as asking members to flag high- and low-value items or using read and edit wear [7] to estimate value.

Another indicator of a community's success is how many members participate in the CALV's upkeep; communities that spread work among many members are more inclusive and robust. This leads to the third metric, *Neditors*, the number of people who edited at least one movie.

We supplement these metrics with results from a survey we conducted after the experiment concluded. 119 people, most of whom used the editing features, responded to the survey.

Overview of editing activity

We collected behavioral data for 53 days in summer 2005. A total of 2,723 subjects saw the contribution interface, with 437 subjects editing at least once. They performed a total of 2,755 edits. Of these editors, the mode was one edit, though two subjects edited well over 100 times—quite a feat considering the 10 edit per day limit. Editing activity appears to follow a power law distribution, a pattern we have seen in many aspects of MovieLens, from movie popularity to the number of logins, ratings, and forum posts per user.

RQ1: DOES TASK ROUTING AFFECT CONTRIBUTIONS?

We turn now to our first research question: *How does intelligent task routing affect contributions to a CALV?* A community might want to match members with tasks for a number of reasons. For example, it might be useful if people who review recent changes in Wikipedia know the topic they are reviewing. User modeling research shows how to build profiles of user interests (e.g., [14, 15]) and expertise (e.g., [11, 12]) that can help match people with topics. Another reason to match people with tasks is to reduce their workload. Wikipedia's recent changes list is long because people make dozens of changes per minute. A personalized view that shows only pages the viewer is likely to work on might increase viewers' ability to contribute. A community might also prefer to focus on tasks that seem most needed. Wikipedia could highlight large recent changes rather than small ones or solicit people to expand short articles rather than review recent changes.

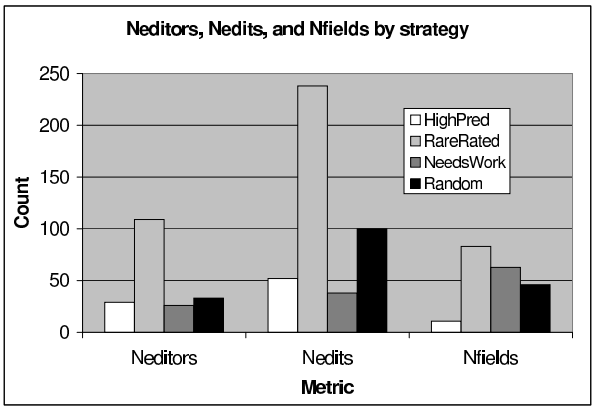

Figure 3. Counts of unique editors, movies edited, and fields edited for each strategy. *RareRated* does best on all, while *NeedsWork* has the highest *Nfields* per edit.

Karau and Williams' collective effort model [8] calls out factors that influence people's motivation to contribute to groups. These factors include how much the person values the task, how much effort the task requires, and how much they believe their contribution matters to the group. Communities that model members' behavior can try to operationalize these factors and use them to stimulate contributions.

We developed four algorithms to match people with movies: ***HighPred***, ***RareRated***, ***NeedsWork***, and ***Random***. Table 1 gives a brief description for each algorithm along with practical justifications and reasons why they might motivate contributions based on the collective effort model. We chose simple, single-strategy algorithms because they are easy to understand and easy to implement. Further, they represent algorithms used by real communities. Wikipedia's Community Portal is based on a ***NeedsWork***-like algorithm, while Slashdot assigns meta-moderation using ***Random***.

When a subject logs in, the algorithm ranks all movies, removes movies chosen for any subject in the last four hours (to prevent movies from being picked repeatedly), and populates the subject's chosen list with the top 20 movies.

Results

To concentrate on the algorithms' effect on behavior, here we limit the analysis to the five movies visible on subjects' chosen lists. For this experiment we collected data for 24 days in August 2005. In these data, 197 of 1,982 subjects edited at least one movie chosen for them. Figure 3 shows the performance of each algorithm on *Neditors*, *Nedits*, and *Nfields*. We use ***Random*** as our baseline.

RareRated outperforms all algorithms, including ***Random*** both on *Neditors* ($\chi^2(3, N = 1,982) > 114, p < 0.01$) and *Nedits* ($\chi^2(3, N = 44,352) > 203, p < 0.01$). The difference on *Neditors* was striking: over 22% of ***RareRated*** subjects edited at least one movie, compared to about 6% for the other groups.

NeedsWork is interesting. It does worst on *Neditors*—a surprise because survey respondents claimed they preferred to edit movies that need more work than less (57 agreed, 50 neutral, 12 disagreed). But because ***NeedsWork*** selects movies that have the most missing information, its per-edit improvement (ratio of *Nfields* to *Nedits*) is much higher than for the other strategies (1.65 versus ***HighPred***'s 0.21, ***RareRated***'s 0.36, and ***Random***'s 0.45).

	High Pred	***Rare Rated***	***Needs Work***	***Random***
Avg. prediction	4.33	3.10	2.57	2.85
Avg. # ratings	2,585	191	212	1,432
Avg. *Nfields*	5.65	5.76	3.10	5.53
Total showings	10,506	11,785	10,767	11,494
User-movie pairs	4,606	3,647	8,564	11,261
Distinct movies	1,598	2,903	770	6,363

Table 2. Statistics on the behavior of the four algorithms.

Finally, ***HighPred*** is dominated by ***RareRated*** on all metrics and by ***Random*** on *Nedits* and *Nfields*. Its per-item improvement is especially low.

Algorithm characteristics

We computed statistics about the movies each algorithm chose in order to better understand these differences. Table 2 shows that the four algorithms had distinct patterns of selecting movies. The first three rows tell us that the algorithms accomplished their goals: ***HighPred*** chose movies with high predictions, ***RareRated*** chose movies with relatively few ratings, and ***NeedsWork*** chose movies missing the most information. Otherwise, the algorithms chose movies of roughly the same quality and predicted rating. ***HighPred*** chose relatively popular movies, while ***RareRated*** and ***NeedsWork*** both choose relatively obscure movies.

NeedsWork chose many fewer distinct movies than the other algorithms because it is not personalized—it always chose the movies that needed the most work. Similarly, the user-movie pairs row from Table 2 shows that ***HighPred*** and ***RareRated*** tended to show the same movie to the same user multiple times. In particular, several subjects in the ***RareRated*** group complained they wanted to see movies in their chosen lists that they had not already edited. Designs that choose different movies each time they ask a user to contribute might perform better.

A closer look at RareRated

RareRated was the most effective strategy for convincing people to contribute. It has a number of aspects that might increase people's motivation to participate according to the collective effort model. First, people are more likely to know about movies they have seen; editing known movies is easier and easier tasks increase motivation. Second, people are more apt to like movies they have seen; personal value increases motivation. Third, being one of a few people who has seen a given movie might induce people to feel their contribution is harder to provide and thus matters more; knowing that contributions matter increases motivation (e.g., for discussions and ratings [2, 10]).

I prefer to edit...	Agree	Neutral	Disagree
movies I like.	69	36	14
movies I have rated.	87	24	8
movies with few ratings.	43	65	11

Table 3. Survey responses to factors the collective effort model predicts may affect motivation. Having rated an item matters most, rareness matters least ($\chi^2(4, N = 357) > 37, p < 0.01$).

We used our post-experiment survey and a logistic regression model to help tease apart these factors. The survey asked subjects people questions corresponding to the three factors above. Table 3 shows people were most likely to agree that having rated a movie matters and least likely to agree that rareness matters.

We also built a logistic regression model to predict whether a movie was edited using four attributes: whether the subject had rated the movie, the subject's predicted (or actual) rating for the movie, log of the movie's popularity, and the movie's *Nfields* score. The first three correspond to the three reasons ***RareRated*** might increase motivation, while the last seemed important to include. We used only movies shown to the ***Random*** group to create an unbiased sample. The model has some predictive power ($\chi^2(3, N = 11,568) = 44, p < 0.01$). Having rated the movie and the movie's *Nfields* score are useful predictors ($p < 0.01$); liking the movie and movie popularity were not useful. Taken together, these results suggest lead us to hypothesize that a *RatedNeedsWork* algorithm would be a good alternative to explore in future research.

Discussion

These results show that intelligent task routing has large effects. ***NeedsWork*** is intuitively appealing from the community's point of view, maximizing the value the CALV gets per edit. However, it fails to consider individual motivation. ***RareRated*** did well because it both personalizes the movies shown and, importantly, chooses movies people have seen. The poor performance of ***HighPred*** suggests that information retrieval-style relevance by itself is not enough.

Having rated an item was so important that any task routing algorithm should consider members' experience with items when possible. A natural question is, given this, what should a task routing algorithm consider next? We explored this question by building a second logistic regression model to predict editing of movies shown to the ***RareRated*** group. Again, the model has some predictive power ($\chi^2(3, N = 12,020) = 77, p < 0.01$). *Nfields* and the person's rating are useful predictors ($p < 0.01$), while popularity and average ratings were not useful. Since *Nfields* matters, developing useful measures of item quality is a logical next step in improving task routing.

RQ2: HOW DOES REVIEW TIMING AFFECT CALVS?

We now turn from the problem of motivating individuals to contribute to our second research question: *How does reviewing contributions before accepting them affect the value of CALVs?* A common tactic for ensuring quality is to review contributions, either by other members, as Slashdot does [9], or by experts, as many high-quality conferences do. There are many design questions when building a system for reviewing contributions, including whether review is needed at all, who can be a reviewer, and whether contributions need to be reviewed before the community can use them.

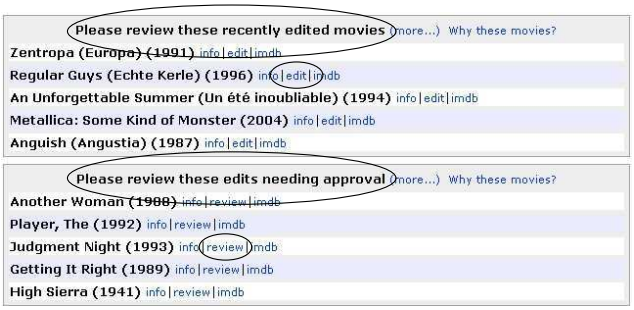

Figure 4. The recent list for the *Wiki-Like* (top) and *Pre-Review* (bottom) conditions. Differences are highlighted.

We have explored the first two design questions in the context of adding movies to MovieLens [3]. We found that at least some review is required both to prevent anti-social behavior and to encourage people to contribute. We also found that peers were about as good at reviewing as experts, at least in MovieLens. Further, people were just as willing to contribute whether peers or experts provided review, supporting the goal of using peer-based review systems as a mechanism for building scalable, robust, and valuable CALVs.

Here, we concentrate on the timing of review. As we saw in the introduction, both including contributions immediately and reviewing contributions before accepting them can succeed. We tried both approaches during the field experiment, randomly assigning members to one of two conditions. In the ***Wiki-Like*** condition, contributions went directly into the MovieLens database. In the ***Pre-Review*** condition, contributions went to a queue of pending edits and only went live after being reviewed (and possibly edited) by a second member. Subjects saw a recent list that contains the items most recently edited by others in their group. The recent list displayed five movies with a link to another 15. Figure 4 shows how the interface differed for the two groups.

Below we present a model of how review timing affects CALV quality, then use our experimental data to test the model's predictions. We combine modeling with a field experiment to build support for our main result: accepting contributions immediately is a win in the short term and will do just as well in the long run.

A MODEL OF CONTRIBUTIONS AND VALUE

Survey respondents believe ***Wiki-Like*** systems would increase value more quickly while ***Pre-Review*** systems would result in higher long-term value. We present a model that suggests they are half right. The model predicts how review timing will affect the quality of CALVs. We make a number of simplifying assumptions in order to get at the heart of the effect of review timing, following Axelrod: "If the goal is to deepen our understanding of some fundamental process,

then simplicity of the assumptions is important, and realistic representation of all the details of a particular setting is not." [1]. Many of these assumptions can be lifted by adding complexity to the model, but we believe the high-level predictions of the simple model apply to many CALVs.

In the model, a CALV consists of a number of *items*, each of which has some *value*. Value might simply be the presence of an item: a MovieLens movie, a Wikipedia page, or a response to a Usenet post. Value might account for quality, perhaps by asking people to flag low-quality items. Value might also include frequency of use, by counting page views in Wikipedia or by weighing fields in MovieLens based on their use in searches. We assume an item's value ranges from 0 to some maximum, that the value of a given item does not change unless someone contributes to it[2], and that the number of items remains constant over the modeling period.

The CALV as a whole then has a value, V, ranging from 0 to V_{max}, the sum of the maximum value of all items. Let V_t be the value of a CALV at time t. We assume that time proceeds in discrete periods. During a time period, the community can increase the value of items, e.g. by correcting information for a movie or improving a Wikipedia page. We model the amount of value the community creates in a given time period as G_t, or "good work". The community also sometimes destroys value (e.g., through trolls, vandals, spammers, and well-intentioned mistakes), which we model in the aggregate as B_t, or "bad work". Good and bad work can overlap during the same time period. We can state a basic model of how a CALV's value evolves:

$$V_{t+1} = V_t + G_t - B_t \tag{1}$$

The value of the CALV grows when $G_t > B_t$, remains the same when $G_t = B_t$, and falls when $G_t < B_t$.

Modeling *Wiki-Like*

G_t and B_t are not constant; otherwise, the equation above suggests that CALVs can grow without limit. Many factors influence how much value is created and destroyed in a given time period, including the CALV's current state and the motivation and abilities of community members. We assume all of these factors are fixed except for V_t. The intuition is that as V_t grows, members will find it harder to locate useful work and easier to damage already-existing value. Much of the work the community would do, then, is expended in finding work rather than doing it.

Let γ be the amount of good effort the community as a whole is willing to expend in a given time period to improve the CALV, and let β be the similar amount of bad effort it would expend to harm the CALV. Remembering that $0 \leq V_t \leq V_{max}$, we let $P_t = V_t/V_{max}$ be the proportion of its potential value a CALV possesses at time t. We extend equation 1 to incorporate the task of finding work, by multiplying γ by the probability that a given item needs work and β by the probability that a given item is already correct.

[2]This is a bad assumption when information decays rapidly. For instance, a database of gas prices at local stations must be constantly updated or its value drops rapidly.

	Good Check	Bad Check
Good Edit	+1	0
Bad Edit	0	-1

Table 4. Contingency table for contribution checking, under the assumption that bad people are as malicious as possible. Good checking of good edits creates value, while bad checking of bad edits destroys it.

	Good Check	Bad Check
Good Edit	$\frac{1}{2}\frac{H_t^2}{W_t} = \frac{40^2}{100} = 16$	$\frac{1}{2}\frac{H_t C_t}{W_t} = \frac{40 \cdot 10}{100} = 4$
Bad Edit	$\frac{1}{2}\frac{C_t H_t}{W_t} = \frac{10 \cdot 40}{100} = 4$	$\frac{1}{2}\frac{C_t^2}{W_t} = \frac{10^2}{100} = 1$

Table 5. A concrete example of computing the number of edits in each cell of the contingency table where $H_t = 40$, $C_t = 10$, and $W_t = 50$.

That is, $G_t = (1 - P_t)\gamma$ and $B_t = P_t\beta$:

$$V_{t+1} = V_t + (1 - P_t)\gamma - P_t\beta \tag{2}$$

Equation 2 has the convenient property that as long as $0 < \beta, \gamma < V_{max}$, it ensures that $0 <= V_t <= V_{max}$. Further, at some point G_t will equal B_t, and the CALV will reach a value equilibrium V_{lim} at a time t where $(1 - P_t)\gamma = P_t\beta$. A little algebra shows this equation is satisfied when $P_t = \gamma/(\gamma + \beta)$, allowing us to compute the equilibrium:

$$V_{lim} = \frac{\gamma}{(\gamma + \beta)} V_{max} \tag{3}$$

In other words, a ***Wiki-Like*** system reaches a value equilibrium below its potential that is determined only by the proportion of good to bad effort members are willing to expend.

Modeling *Pre-Review*

We now turn to modelling the ***Pre-Review*** system, where a second person must approve contributions before they are added to the CALV. Here, value is divided into two parts: working on items, and approving (or rejecting) work. We return to our fundamental model from Equation 1, $V_{t+1} = V_t + G_t - B_t$. Now, the formulae for G_t and B_t must account for the fact that some of the community's contributions are allocated to editing items and some to checking those edits.[3]

One of four things can happen to a contribution, depending on whether the editor and the checker are good or bad. Table 4 shows a payoff matrix that models "bad" people that maliciously destroy content, one of the most common threats seen in wikis.[4] If both the edit and check are good, value is created. If both are bad, value is removed. If one is good and one bad, value is unchanged: either a bad edit is appropriately rejected or a good one is incorrectly rejected.

We now compute how the value of the CALV changes during a given time. To do this we must first know how much total work is done. Let $H_t = (1 - P_t)\gamma$, $C_t = P_t\beta$, and $W_t = H_t + C_t$ be the amount of good, bad, and total work

[3]The work of checking in ***Wiki-Like*** systems is represented implicitly in β and γ: some of the effort of γ at time $t+1$ will repair errors introduced through the effort in β at time t.

[4]But not the only possible matrix. Bad actors might sometimes create value, e.g., a spammer might approve good contributions in order to increase the community's value as a market.

done during time t. We assume people evenly divide work between editing and checking. The formulae for all four squares in the contingency table have the same structure. For instance, the probability of a Bad Edit is C_t/W_t, and the (assumed independent) probability of a Good Check is H_t/W_t, so the probability of a Bad Edit followed by a Good Check is C_tH_t/W_t^2. To compute the number of times a Bad Edit followed by a Good Check occurs, we multiply this probability by the total number of edits performed: $W_t/2$ since half the work goes into edits. The product is $C_tH_t/2W_t$. Table 5 presents all four formulae along with a numeric example.

Now we are ready to apply equation 1 and the payoff matrix from Table 4. In this case G_t is just the number of edits that wind up in the Good-Good quadrant, while B_t is the number of edits that wind up in the Bad-Bad quadrant. The equation for how much gets done per time period is:

$$V_{t+1} = V_t + \frac{1}{2}\frac{H_t^2}{W_t} - \frac{1}{2}\frac{C_t^2}{W_t} \quad (4)$$

Again, we can figure out the equilibrium for the value of the CALV, V_{lim}, which happens when:

$$\begin{aligned} \frac{1}{2}\frac{H_t^2}{W_t} &= \frac{1}{2}\frac{C_t^2}{W_t} \\ H_t &= C_t \\ (1-P_t)\gamma &= P_t\beta \end{aligned}$$

We saw this equation when finding the ***Wiki-Like*** equilibrium; the equilibrium for a ***Pre-Review*** system is:

$$V_{lim} = \frac{\gamma}{(\gamma+\beta)}V_{max} \quad (5)$$

which is surprisingly the same as Equation 3! That is, if V_{max}, γ, and β are the same, checking before accepting contributions does not improve the eventual value equilibrium the CALV reaches, compared to accepting them right away.

Rate of Convergence

Instead, checking imposes a cost. We can compare how quickly ***Pre-Review*** and ***Wiki-Like*** approach the value equilibrium. The *rate of convergence* for a sequence that converges linearly to ξ is usually defined as μ, where:

$$\mu = \lim_{k\to\inf} \frac{|x_{k+1}-\xi|}{|x_k-\xi|} \quad (6)$$

That is, if the order of convergence is linear, the rate of convergence μ is the ratio between the error at step $k+1$ and the error at step k [5]. μ must be between 0 and 1, and the closer to 0, the faster the convergence. The convergence rate for both systems is given by:

$$\mu = 1 - x\frac{\gamma+\beta}{V_{max}} \quad (7)$$

where $x = 0.5$ for ***Pre-Review*** and $x = 1$ for ***Wiki-Like***.[5] In other words, both systems reach the same final value, but the ***Wiki-Like*** system gets there faster.

[5]We derive these rates in the appendix.

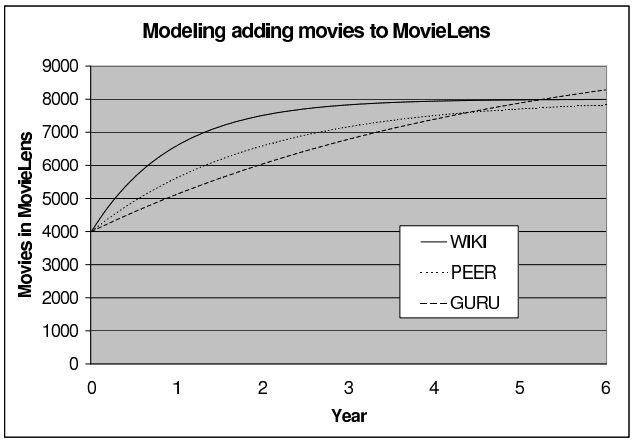

Figure 5. Modeling the effect of different systems for adding movies to MovieLens., with $V_0 = 4,000$ and $V_{max} = 10,000$. For the PEER and WIKI systems, $\gamma = 160$ and $\beta = 40$, while for the GURU system, $\gamma = 40$ and $\beta = 0$. WIKI is faster than the others. PEER never catches up, but GURU does—after 5 years.

A concrete, yet fictional example

We first apply the model to a slightly fictionalized version of adding movies to MovieLens. Our movie guru took over the MovieLens database about 6 years ago, when it had about 4,000 movies. Based on his criteria for adding movies to MovieLens, there are about $10,000$ movies eligible to include. (This number grows by a few hundred each year, but we will follow the model and assume that it is fixed at $10,000$). Based on his behavior, we can estimate that $\gamma = 40$ and $\beta = 0$ if only the guru adds movies. We will discuss how to estimate these parameters later, when we evaluate the model against the experimental data.

We can use the model to look at how the database might have evolved if we had allowed more people to participate. Based on data from [3], we estimate the community's $\gamma = 160$ and $\beta = 40$ for the task of adding new movies to MovieLens.[6] With these estimates, we can compare what might have happened had we allowed the community to add movies. Figure 5 shows that the value of the database would have grown much faster than it did, and further, it would have grown even faster if contributions were made visible right away. The guru would eventually create a more valuable database—in five years.

A summary of assumptions and predictions

The model relies on a number of assumptions about how editors and checkers interact through the CALV, which we collect here for convenience.

1) **Discrete Time Steps.** One time step completes before the next begins. Good and bad activities within each time step may overlap.
2) **Cumulative Value.** The value of the CALV is the sum of the value of individual items.

[6]The $\beta = 40$ consists of mistakes made when adding movies, such as forgetting to add any information besides the title or adding information for the wrong movie.

	Pre-Review	*Wiki-Like*	p-value
Subjects	1,353	1,370	-
Neditors	239	198	$p < 0.10$
Nedits	1,534 (873+661)	1,220	$p < 0.05$
DB *Nedits*	661	1,220	$p < 0.01$
DB *Nfields*	770	1,017	$p < 0.50$

Table 6. Overall editing performance of two groups. *Pre-Review* had 1,534 total edits, of which 661 were checks. DB *Nedits* and DB *Nfields* are the changes that actually occurred in the database. P-values are from t-tests except for *Neditors*, where we used a chi-squared test.

3) **Steady Item Value.** The value of an individual item only changes when someone changes it.
4) **Value Ceiling.** There is a fixed maximum attainable value for the CALV. (The number of items is also fixed.)
5) **Stable Motivation.** The total amount of available effort is stable while the value of the CALV changes.
6) **Random Search For Work.** Users randomly encounter items to work on.
7) **Effort Is Equal.** The effort to fix an item, to check whether an item needs fixing, and to check whether a fix is correct is about the same.
8) **Edits Equal Checks.** In ***Pre-Review***, as many checks happen as edits.
9) **No User Roles.** Anyone can edit or check in ***Pre-Review***.

These are simplifications. For instance, people may give up on a CALV with low value or one that grows too slowly, violating **Stable Motivation**. Rapidly growing CALVs like Wikipedia violate **Value Ceiling**. **Cumulative Value** fails to model the fact that some items are more popular than others. Many review systems violate **No User Roles** by having relatively few editors who are allowed to check contributions.

We nevertheless believe these are reasonable assumptions for understanding the broad effects of review timing. The model makes several interesting predictions. The most surprising is that the final quality will be exactly the same for the ***Wiki-Like*** and ***Pre-Review*** methods of making changes to a CALV. Proponents of both models, from people discussing Wikipedia on Slashdot to several of this paper's authors, argue that their favorite model is better. Finding they should yield the same quality in the long run was a surprise. A second interesting prediction is that ***Wiki-Like*** will converge much faster to this long-run quality. The reason is simple: ***Wiki-Like*** does not waste as much effort checking.

RESULTS FOR REVIEW TIMING

We now examine what happened during the experiment, then use that data to evaluate the model. In 53 days, 437 of 2,723 subjects edited at least one movie. Table 6 shows subjects' behavior under the two systems. ***Pre-Review*** outperformed ***Wiki-Like*** on total *Nedits* and *Neditors*. However, ***Wiki-Like*** subjects made more edits that appeared in the database.

These differences happen because of checking in the ***Pre-Review*** group. Of 1,534 edits, 871 were initial edits while 661 were checks. This left 212 edits pending approval at the end of the experiment—wasted work because these contributions were not available to the community. Survey responses confirm that people prefer editing to checking (38 preferred to edit, 17 to check, 64 did not care).

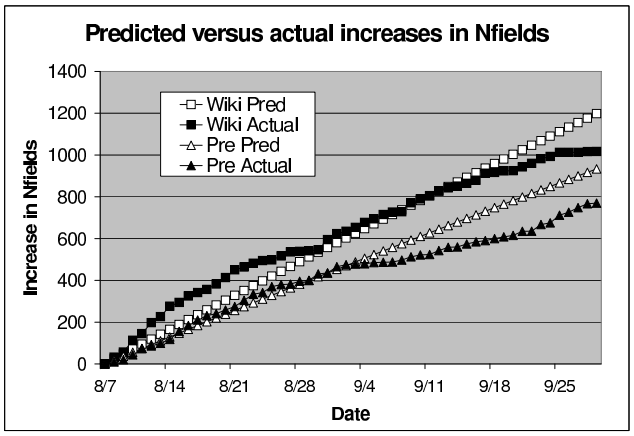

Figure 6. Comparing the model's predictions to actual editing behavior, using estimates of γ and β for both systems based on the half of the editing data. As the model predicts, *Wiki-Like* outperforms *Pre-Review*.

Pre-Review had a higher *Nfields* per movie changed in the database than ***Wiki-Like*** (1.16 vs. 0.83). The average increase in *Nfields* for both groups on the initial edit of a movie was almost identical, so the increase happened when ***Pre-Review*** reviewers made changes that improved the original contribution. Note that the model does not account for this increase; by using the payoff matrix in Table 4, it assumes that reviewers can only approve or reject changes without adding value of their own.

The model meets reality

We now compare the model to the experimental data. The model makes three high-level predictions about the ***Wiki-Like*** and ***Pre-Review*** systems. 1) ***Wiki-Like*** outperforms ***Pre-Review*** in the short run. 2) The amount of value created tapers over time. 3) They reach the same long run equilibrium. We will first fit the model to our data, then evaluate how well it fits and whether its predictions are accurate.

To fit the model, we need to estimate V_0, V_{max}, γ, and β. We use the *Nfields* metric. 8,770 items with a maximum per-item value of 8 yields $V_{max} = 70,160$. The sum of *Nfields* when the experiment began was $V_0 = 47,280$. To estimate γ and β, we use observed behavior for the first half of the experiment. In 27 days, the ***Wiki-Like*** group increased *Nfields* by 654, so we know $G_t - B_t = 654$. The ratio of good to bad edits was $G_t/B_t \approx 60$. This lets us estimate $B_t \approx 11$ and $G_t \approx 660$. Since $P_0 = V_0/V_{max} \approx 0.67$, $G_t = (1 - P_0)\gamma$, and $B_t = P_0\beta$, that gives us $\gamma \approx 2,000$ and $\beta \approx 16$ for the ***Wiki-Like*** group. Similar calculations for the ***Pre-Review*** group give $\gamma \approx 3,240$ and $\beta \approx 54$.

Based on those estimates, Figure 6 compares the model's predictions to actual behavior. The model's first two high-level predictions are supported: ***Wiki-Like*** outperforms ***Pre-Review*** in the short run, and the amount of value created

tapers off over time. The predictions fit reasonably well, overestimating somewhat for both groups. The quality of the predictions also depends on how much data is available for estimates. Had we only collected data for one or two weeks, our estimates of γ would be 30 to 40 percent higher because contributions taper off faster than the model predicts. Its predictions over the experimental period are nearly linear because $\gamma + \beta$ is small relative to the maximum value of repository. Finally, we do not have enough data to evaluate the model's third prediction that ***Wiki-Like*** and ***Pre-Review*** will converge to the same value equilibrium in the long run.

Note that the model missed on ***Pre-Review*** in two ways. First, our payoff matrix from Table 4 was incorrect. Checking added about 40% more value to an initial edit because reviewers were allowed to improve contributions they checked. This means 1.4 would have been a more accurate value in the Good-Good quadrant. Second, the model did not account for wasted work because of the **Edits Equal Checks** assumption. In this setting, about 1/4 of edits were never checked. These two effects roughly balanced each other here. This is not likely to be true in general, and a more complex model that accounts for these effects might be more directly useful in designing systems.

DISCUSSION

These results suggest that ***Wiki-Like*** systems create value faster than ***Pre-Review*** ones. More people contributed under the ***Pre-Review*** system overall, but since people prefer editing to checking, a backlog of wasted work builds up. The model also suggests that the ***Pre-Review*** group will do about the same as the ***Wiki-Like*** group in the long term. Our estimates of γ and β put V_{lim} near V_{max} for both systems because β is small in MovieLens. This prediction would be easier to test in a system with more bad contributions.

At a high level, the model accurately reflects the relative behavior of the ***Pre-Review*** and ***Wiki-Like*** systems. However, contributions taper off faster than predicted. As a new feature, the contribution interface might have been used more heavily at first than it would be in the long-term, violating the **Stable Motivation** assumption. This would explain the overestimation and the rapid taper. It might also be that $(1-P)$ and P are not the right probabilities of finding useful work to do. Finally, it may be that counting all fields equally made our value function too simple. Members rarely search for films available on VHS; perhaps they would not notice or care enough to fix errors in a movie's VHS release date.

Pre-Review systems may increase people's willingness to contribute (increase γ) or deter people from damaging the system (decrease β) compared to ***Wiki-Like***. Here, the ***Pre-Review*** group had more editors and total contributions, while prior work showed that review before acceptance reduced antisocial behavior compared to a system with no review [3]. Designers might use the model to reason about trade-offs between short-term speed and long-term quality. Fielding a ***Wiki-Like*** system until contributions taper off and then switching to a higher-equilibrium ***Pre-Review*** system may let designers have it both ways. Changing review policy in an established community requires caution, however: imagine the outcry from its members if Wikipedia were to switch to a ***Pre-Review*** system.

Although ***Wiki-Like*** systems accumulate value more rapidly, they also allow more bad content into the CALV that is eventually corrected by other members. This is often held against Wikipedia: even though there is much good content, members may not know which content to trust, thus limiting Wikipedia's usefulness as a reference—and perhaps reducing their motivation to contribute. Recent pages on fake pop stars and articles modified by marketers show that this concern is real.[7] On the other hand, bad content is often quickly removed, with obvious vandalism disappearing in minutes [17]. Incorrect content may take longer to fix.

Extending the model to consider the amount of bad content seen by members might be useful. The model could be extended in a number of other ways as well. In the experiment, we noted that reviewers can add value and that work can be wasted. The model assumed these effects away, but incorporating them would not be hard. More complete (but complicated) models that reduce the number of necessary assumptions may be more useful to designers. Enhancing the model's ability to account for the cost of finding work is a natural next step. Systems that help people find work, perhaps using intelligent task routing, will be able to redirect effort from finding useful work to doing it. This will improve their ability to create value, and the model should account for that as well. The model is an interesting starting point for thinking about designing systems that encourage contribution, but it is by no means the last word.

CONCLUSION

This work takes a number of steps toward improving system design for community-maintained artifacts of lasting value.

- Systems that solicit contributions from members can use intelligent task routing to increase contributions. Even simple algorithms have large effects.
- Algorithms based only on the community's needs are less likely to interest members than algorithms that consider a person's knowledge and ability.
- The model suggests CALVs tend to reach a quality equilibrium that depends only on the proportion of good to bad contributions from community members, not on whether contributions are reviewed before acceptance.
- Both the model and empirical results show that review before acceptance slows the accumulation of value.
- Designers can use the model of CALV value evolution to help reason about design alternatives once they collect a moderate amount of behavioral data.

We are excited by the potential of CALVs to increase the value and scope of community on the web. As tools for

[7] http://www.smh.com.au/news/icon/wikipedia-worries/2005/08/23/1124562860192.html

finding and communicating with others became standardized and accessible, discussion-based online communities became widespread. We predict a similar rise of groups that build these lasting, valuable community-specific resources. Understanding how to build the tools that will help these groups survive and thrive is an important next step for the CHI and CSCW communities.

ACKNOWLEDGEMENTS

This work was supported by grants 0325837, 0324851, and 0534420 from the National Science Foundation.

APPENDIX: DERIVING CONVERGENCE RATES

Here we derive the rates of convergence for the ***Wiki-Like*** and ***Pre-Review*** systems. A useful equality is:

$$\frac{\gamma}{V_{lim}} = \frac{\gamma}{\frac{\gamma}{\gamma+\beta}V_{max}} = \frac{\gamma+\beta}{V_{max}} \tag{8}$$

We derive the convergence for ***Wiki-Like*** to V_{lim}:

$$\begin{aligned}
\mu &= \frac{V_{t+1}-V_{lim}}{V_t - V_{lim}} \\
&= \frac{V_t - V_{lim} + (1-P_t)\gamma - P_t\beta}{V_t - V_{lim}} \qquad (9)\\
&= \frac{V_t - V_{lim}}{V_t - V_{lim}} + \frac{\gamma - P_t\gamma - P_t\beta}{V_t - V_{lim}} \\
&= 1 + \frac{\gamma - (\gamma+\beta)P_t}{V_t - V_{lim}} \\
&= 1 - \frac{\frac{V_t}{V_{max}}(\gamma+\beta) - \gamma}{V_t - V_{lim}} \\
&= 1 - \frac{V_t\frac{\gamma}{V_{lim}} - \gamma}{V_t - V_{lim}} \text{ (by equation 8)} \\
&= 1 - \frac{(\frac{V_t}{V_{lim}} - 1)\gamma}{V_t - V_{lim}} \\
&= 1 - \frac{\frac{V_t - V_{lim}}{V_{lim}}\gamma}{V_t - V_{lim}} \\
&= 1 - \frac{\gamma}{V_{lim}} \\
&= 1 - \frac{\gamma+\beta}{V_{max}} \text{ (by equation 8)} \qquad (10)
\end{aligned}$$

Pre-Review also converges to V_{lim}:

$$\begin{aligned}
\mu &= \frac{V_{t+1}-V_{lim}}{V_t - V_{lim}} \\
&= \frac{V_t + \frac{1}{2}\frac{H_t^2}{W_t} - \frac{1}{2}\frac{C_t^2}{W_t} - V_{lim}}{V_t - V_{lim}} \\
&= \frac{V_t - V_{lim} + \frac{1}{2}\frac{(H_t - C_t)(H_t + C_t)}{W_t}}{V_t - V_{lim}} \\
&= \frac{V_t - V_{lim} + \frac{1}{2}(1-P_t)\gamma - P_t\beta}{V_t - V_{lim}}
\end{aligned}$$

which is in nearly the same form as equation 9, except for the $1/2$, and the derivation proceeds from there.

REFERENCES

1. R. Axelrod. *The Complexity of Cooperation*. Princeton University Press, 1997.
2. G. Beenen et al. Using social psychology to motivate contributions to online communities. In *Proc. CSCW2004*, 2004.
3. D. Cosley et al. How oversight improves member-maintained communities. In *Proc. CHI2005*, pages 11–20, 2005.
4. R. M. Dawes and R. H. Thaler. Anomalies: Cooperation. *The Journal of Economic Perspectives*, 2(3):187–197, 1988.
5. W. Gautschi. *Numerical analysis: an introduction*. Birkhauser Boston Inc., Cambridge, MA, USA, 1997.
6. R. Hardin. *Collective Action*. Johns Hopkins, 1982.
7. W. C. Hill, J. D. Hollan, D. Wroblewski, and T. McCandless. Edit wear and read wear. In *Proc. CHI1992*, pages 3–9, 1992.
8. S. J. Karau and K. D. Williams. Social loafing: A meta-analytic review and theoretical integration. *Journal of Personality and Social Psychology*, 65(4):681–706, 1993.
9. C. Lampe and P. Resnick. Slash(dot) and burn: distributed moderation in a large online conversation space. In *Proc. CHI2004*, pages 543–550, 2004.
10. P. J. Ludford, D. Cosley, D. Frankowski, and L. Terveen. Think different: increasing online community participation using uniqueness and group dissimilarity. In *Proc. CHI2004*, pages 631–638, 2004.
11. D. W. McDonald and M. S. Ackerman. Expertise recommender: a flexible recommendation system and architecture. In *Proc. CSCW*, pages 231–240, 2000.
12. A. Mockus and J. D. Herbsleb. Expertise browser: a quantitative approach to identifying expertise. In *Proc. ICSE2002*, pages 503–512, 2002.
13. G. B. Newby and C. Franks. Distributed proofreading. In *Proc. JCDL2003*, pages 361–363, 2003.
14. M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Mach. Learn.*, 27(3):313–331, 1997.
15. P. Resnick et al. Grouplens: an open architecture for collaborative filtering of netnews. In *Proc. CSCW1994*, pages 175–186, Chapel Hill, NC, 1994.
16. B. K. Thorn and T. Connolly. Discretionary data bases: A theory and some experimental findings. *Communication Research*, 14:512–528, 1987.
17. F. B. Viégas, M. Wattenberg, and K. Dave. Studying cooperation and conflict between authors with history flow visualizations. In *Proc. CHI2004*, pages 575–582, 2004.
18. L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proc. CHI*, pages 319–326, 2004.