Metavines: A Product Line Architecture for Word-of-Mouth Social Software

Bharath Kumar M. Srikanta J. Bedathur

IISc-CSA-TR-2004-11
http://archive.csa.iisc.ernet.in/TR/2004/11/

Computer Science and Automation
Indian Institute of Science, India

October 2004

Metavines: A Product Line Architecture for Word-of-Mouth Social Software

Bharath Kumar M.* Srikanta J. Bedathur†

Abstract

A social network is the pattern of friendship, advice, communication or support which exists among the members of a social system. Several researchers have studied the flow of information in a social network through the word-of-mouth phenomenon, and have shown that it is pervasive in many aspects. We extend this phenomenon to virtual societies formed over the Internet, assisting the natural social information filtering process. We present Metavines, a product line architecture, which facilitates rapid development of many flavors of such social software. The underlying model of Metavines allows, (i) speedy dissemination of ranked, relevant information, (ii) assessment of the social neighborhood in terms of past interactions, and (iii) evolution of social networks by bringing people of similar interest together. In order to study the efficacy of this approach, we have developed and deployed a URL dissemination software called Webvines, based on the Metavines architecture. We also discuss our initial experiences with Webvines.

1 Introduction

Social network researchers have offered clear evidence of the extent to which knowledge diffusion occurs via social relations [11]. Research has shown that in organizations, engineers and scientists are roughly five times more likely to turn to a person for information than seek through databases or other repositories [4]. Over the past years, these social phenomena have spanned the Internet, breaking conventional geographical boundaries of society. The line of distinction between the traditional and the virtual social network has blurred, both in terms of making and maintaining contacts [14].

*Author for TR correspondence. mbk@csa.iisc.ernet.in

†srikanta@dsl.serc.iisc.ernet.in.

1.1 The Word-of-Mouth Phenomenon

Perhaps the most unique feature of a social network is the propagation of ideas, information and rumors through the *Word-of-Mouth* phenomenon. This could lead to the receipt of valuable information, recommended and filtered through the social neighborhood, although not originally sought for.

The word-of-mouth phenomenon is best exemplified through well studied scenarios,

Stock Holdings. In [13], Shiller writes that:

> "A fundamental observation about human society is that people who communicate regularly with one another think similarly. There is at any place and in any time a *Zeitgeist*, a spirit of the times. Word-of-mouth transmission of ideas appears to be an important contributor to day-to-day or hour-to-hour stock market fluctuations..."

Further, in [7], Hong et. al., show empirical evidence for the hypothesis that investors spread information and ideas about stocks to one another directly, through word-of-mouth communication.

Job Contacts. We can find cookbooks [2, 8], which suggest a job seeker or employer to use "the network" to find options. A job seeker specifies his qualifications and capabilities to his circle of friends, who then pass on the same to their contacts and so on. Thus, an option "hidden away" somewhere becomes accessible. Taking the network route makes the process more dependable and appropriate in finding a match.

These examples exhibit the following characteristics:

1. Each node associates a perceived *credibility*, which could vary depending on the context, with every other node in its immediate social neighborhood.

2. The perceived relevance or importance of information is dependent on the credibility of the recommender(s).

3. Individual experience with the information, alters the credibility of the recommender(s).

4. Information is passed on only if it is perceived that the neighbor has an interest in it.

Such network recommendations essentially lead to *social information filtering*. People recommend information only if they perceive it as important, relevant or useful to others in their social vicinity. Information received is perceived as valuable depending upon the receiver's past experience with recommender(s). Social information filtering could result in serendipitous discovery supplementing existing forms of information retrieval.

1.2 Automating word-of-mouth

One of the early efforts in this direction has been Ringo [12], an album recommendation system that recommends new music albums based on ratings given by 'similar' users in the system. Web based systems like Epinions [1], collect opinions (recommendations or critiques) from users on various products to rank them. Such software, which are gaining in popularity, have been termed as "social software" reflecting the views, opinions, trends and prejudices of the society.

Software for stock alerts, job profiles, knowledge dissemination, media recommendations etc. are various possible flavors of recommender systems that can exploit word-of-mouth on an underlying social network. The credibility of the recommender(s) can be maintained by continuously tracking the relevance of information to the user. By obtaining succinct feedback about the recipients' experience, the recommenders can focus their efforts towards their target audience. Although the nature of information, the way it is produced, and consumed is different in each of these applications, the underlying principles tend to remain the same.

Word-of-mouth based recommender systems naturally fit into a distributed setting, where people, the users of the system, reside on the 'edge' of the Internet. Such systems also result in cost reduction, scalability and freedom from centralized servers.

In this paper, we present *Metavines*, a product line architecture [5] that facilitates rapid development of distributed applications that allow for social information filtering and information diffusion over virtual societies. Metavines derives its name from **Meta**-grape**vines**, representing its ability to instantiate several forms of word-of-mouth social software.

Organization

In section 2, we describe the underlying model used by envisioned word-of-mouth social software. Section 3 introduces the architectural concepts behind Metavines showcasing its reusable aspects. Next, section 4, describes a product built over Metavines, *Webvine*, which is a fully functional and

deployed application, with a small and growing user base, used to disseminate useful and interesting web links over the social network. We also discuss some early experiences from the deployment of the Webvine system over a small social network of computer science researchers. In section 5, we touch upon related work, and compare and contrast our research with them. Finally, we conclude in section 6.

2 Modeling Word-of-Mouth Social Software

Certain domain forces strongly influence the model of word-of-mouth social software. An analysis of these forces and their implications on the model are discussed in this section.

Distributed Nature. Recommender systems have been around for many years. It is important to investigate if social information filtering, based on word-of-mouth software can succeed over other recommendation techniques like collaborative filtering. Lueg and Muller argue in [10], that a 'rationalization' of the social process by machines, taken by most approaches to collaborative filtering, do not capture the social embedding of a cooperative process that involves mutual commitments, being under obligation, being responsible, etc. Overlaying an information dissemination system on a social network in a distributed setup would reflect the natural process better, rather than collect user behavior at a central server so as to mine logs and make suggestions.

Scope of Automation. Accurately determining relevance and usefulness of information to a user automatically would be impossible, since people's interests change dynamically over time and context. Word-of-mouth social software, should only assist the natural phenomenon of selection and rejection, something that has been trained by evolution, rather than completely automating it. The goal of such software is to minimize the effort needed in dissemination of seemingly useful information, than to do away with human cognitive ability. The Metavines' models do not take any unilateral decisions for humans, but instead are designed to assist them in the process.

Strength of Ties. While studying the diffusion of knowledge in societies, social network theorists have focused much of their attention on structural properties of networks, especially tie strength at the dyadic level [6]. Ties are characterized as strong or weak depending on closeness or interaction frequency. In [9], Levin et al., make four observations:

1. Overall, stronger ties–more so than weaker ones–lead to the receipt of useful knowledge.
2. The link between strong ties and receipt of useful knowledge is mediated by benevolence-based trust and competence-based trust.
3. After controlling for competence and benevolence-based trust, it is weaker ties–more so than stronger ones–that lead to the receipt of useful knowledge.
4. Competence-based trust is more important to the receipt of useful knowledge when that knowledge is tacit (i.e. not written or codified) than when explicit.

Since strong social ties are benevolent towards a person's needs and interests, and their competence is well understood, believable useful information is obtained from them. Strong social ties, tend to relate increasingly similar people. At the same time, a weak tie is beneficial because it provides information from socially distant regions of a network, which will be relatively unknown to the person. However, information from a weak tie needs to be trusted for it to be useful. This trust could be achieved either through verifiability of the information itself, or through reassurance provided by a trusted person (possibly a strong tie).

Some forms of electronically disseminated information, like a URL link to an Internet resource, or an article, tend to be explicitly verifiable. So, trust is not of paramount importance here. Others like stock alerts and music, may require stronger ties, since stronger ties reflect trust and personal taste better.

Thus, a model that needs to address all forms of information dissemination over the network, will have to allow for:

- speedy dissemination of information in a distributed setting,
- receipt of useful information from socially distant regions (allow for many weak ties),
- a good measure of credibility on the information source,
- seconding/ reassurance of information originally obtained from a weak tie, by a more trusted tie.

Clearly, the kind of information being disseminated has a bearing on the social contacts made and maintained, and the importance given to trust.

With a large in-degree, the inflow of information onto a person could be potentially high, and could result in information overload. This calls for a ranking measure which will recommend seemingly more relevant, credible or trusted information.

We now present the model underlying the core processes of the Metavines system.

2.1 Foundations

A virtual society is modeled as a graph G consisting of a finite nonempty set V of nodes, and a collection E, of labeled, directed and weighted edges between them. Each node in the graph represents a person involved in the social network, and edges represent the social ties of the person. The directed edge, $v \rightarrow w$, defines a *publisher-subscriber* relationship between the two nodes, v and w. We use the term *agent* to indicate the software component that interfaces the person with the network.

Every unit of information, called *metavine*, travels along the direction of an edge in this network. Any node in the network could either create a new metavine or forward a metavine to its subscribers. No metavine can be altered once it is released into the network – i.e., forwarders cannot modify the contents of the metavine. Any subscriber can receive the same metavine through different paths, although all originate from the same node, the author of the metavine. Every subscriber attaches a *value* to a metavine, as a measure of its perceived usefulness. This value is communicated to the publisher(s) as *feedback* for the metavine.

Every subscriber(publisher) maintains a *credibility* (*satisfaction*) measure for each of his publishers(subscribers). These measures are dynamic in nature, and change based on value/feedback associated with every metavine that is transferred.

Further, the model operates under the following invariants[1]:

- Each $v \in V$ has a globally unique identity in the graph G.
- Every metavine is identified through a globally unique identifier.
- Metavines travel only between nodes that are connected in the graph G, and only in the direction of the connection. Hence, no metavine is received from an unsolicited source.
- No metavine traverses the same edge twice.

[1] A practical implementation can enforce all these invariants only for a limited window in time, due to resource constraints.

- Multiple copies of the metavine arriving through different paths are consolidated.
- The agent never forwards a metavine automatically.
- All the measures in the model, $value$, $feedback$, $credibility$ and $satisfaction$, are nonnegative real values.

2.2 Metavine Contexts

The value attached to a metavine by a person is highly context-dependent. A *metavine context* is any brief term, possibly from an ontology, that throws some light on the metavine itself. Einstein's words on the Theory of Relativity are more highly regarded than his views on theology. The worth or relevance of his message, depends on the context in which it was spoken. As another example, an employee of a company may show greater interest in an article on "Philosophy of Bill Waterson" during the late hours of the day, while during the working hours a technically relevant information is preferred. While Alice may prefer a music recommendation from Bob in alternative music, his opinion on classical music may not be highly regarded. A person owing to his good network contacts may forward better information, than his original postings.

The model uses the notion of metavine contexts to fine tune its recommendations and the ranking, thereby making suggestions at a better level of granularity. Associated with every metavine is a set of contexts to which it belongs. Thus, a metavine on a new music album could have its metavine context as {*Alternative, Morning, Original*} to indicate that it can be classified as alternative music, posted/ received in the morning, and is an original suggestion from the publisher. A metavine context can be derived from an ontology as well, e.g.: {*Eastern.Indian.Classical.Carnatic.Vocal, Evening, Forwarded*}.

A generic model cannot provide an exhaustive set of all contexts. Provision of the contexts, and deducing the metavine context of every individual metavine, is largely application specific, and is best handled at that level. The model just observes a metavine context as a set of contexts that apply to a metavine. However, the model requires to know this set to be able to manage the publisher credibilities and subscriber satisfaction more accurately. The general/detailed specification of metavine context is a usability issue, and a good trade-off to ease of use and accuracy is hoped for, while designing the application.

2.3 Publisher Credibility Management

Publisher credibility management deals with the maintenance of publisher credibility, as perceived by his subscriber, over a window of past metavine postings, along different metavine contexts.

A person receives metavines from many publishers. To formulate a ranking system that tries to rate the relevance of a metavine to the person, it is necessary to maintain a past history of postings by every publisher and the individual experience of the person with each of them. Posting behavior may change over time. A publisher who starts with irrelevant postings, may indeed turn valuable to a person over time when their interests may change and coincide. Maintaining a simple single credibility value would accumulate older history experience and would make newer behavioral changes hard to infer. Hence, the model stores posting and experience history only for a limited period into the past.

In order to strike a balance between the resources (e.g. memory) consumed by maintenance of history and its accuracy, the concept of a histogram is imposed and the history is organized into a sequence of fixed size *buckets*. When the history is full, the older buckets are emptied, to make way for newer, more recent buckets. Each bucket is of size $bucketSize$ and stores the cumulative sum of values the person associated with those $bucketSize$ postings. E.g.: If the bucket size is 4, and for four metavine postings, the person associated the values 0.5, 1, 1.5, 0 with each of them, then the cumulative sum for that bucket (the $bucketValue$) is 2. This experience history is maintained separately for every metavine context the publisher has a posting in. The credibility along every context is computed by the equation,

$$credibility = \frac{\sum_{i=1}^{bucketCount} bucketValue_i}{bucketCount * bucketSize} \tag{1}$$

The effective credibility of a publisher for a metavine, with metavine context MC, is given by the equation,

$$effectiveCredibility = \frac{\sum_{i \in MC} credibility_i * weight_i}{\sum_{i \in MC} weight_i} \tag{2}$$

where $weight_i (\geq 1 e.g.$ is the weight associated with every context. Since some contexts may deserve greater importance than others, every context defines the weightage it deserves. E.g.: Due to a large deviation between relevance in metavines that are forwarded or originally posted, these contexts (Forwarded, Original) are given a higher weightage (say 2), while the time of day contexts (like Morning, Evening) are given a lesser weightage (say 1).

When a new metavine is received, with metavine context MC, from a publisher, and gets an associated *value* (its relevance to the person), the publisher's credibility history for every $context \in MC$ of the publisher is updated. This update may involve a simple addition to the latest bucket, or a deletion of the oldest bucket, followed by a new bucket created with this *value* inserted.

2.4 Rank Computation

An incoming metavine is ranked among other 'competing' metavines based on the effective credibility of the publisher(s). The rank of a metavine is simply the sum of the effective credibilities of the publishers who recommend it.

Since a metavine will not be recommended by all publishers at the same time, the ranking is best described in an incremental fashion. Whenever a publisher recommends a metavine, its rank is updated by using the equation

$$rank = rank' + effectiveCredibility_{publisher} \tag{3}$$

where *publisher* is the publisher who recommended this metavine.

The rank computation reflects the heuristic, "Information recommended by many people tends to be more important".

2.5 Subscriber satisfaction

Subscriber satisfaction deals with the maintenance of subscriber satisfaction, as perceived by his publisher, over a window of past metavine postings, along different metavine contexts.

Since the value associated with every metavine is sent across to all the publishers of a metavine as feedback, the publishers maintain a measure of how 'happy' their subscribers are with their postings along each metavine context. This measure is managed in the same way as publisher credibilities.

2.6 Target recommendations

When a publisher is about to send a metavine with metavine context MC, the model computes the list of subscribers that are 'satisfied' (based on a threshold value[2]) with postings from this person for the context MC. These subscribers are recommended as targets for this metavine. However, the publisher is free to include more targets that appear relevant to him. This procedure also results in 'voluntary' spam control since a publisher will not post metavines to persons who do not care for it.

[2]All threshold values are configurable and empirically stabilized

2.7 Publisher recommendations

Metavine ranking, along with target recommendations, supplement social information filtering. The ease of forwarding results in information reaching farther than it would normally would without such assistance. In our experiments, we have found that it is not surprising to see information travel three to four hops on the average, even in a very small social network. This has often resulted in information reaching subscribers who are unknown to the original author.

The model keeps track of all such remote authors, and the value associated with their postings. The credibility history of remote authors is maintained, although not as elaborately as described in the subsection 2.3. Specifically, a consolidated history is used across all metavine contexts to save on the resource utilization. When the credibility of a remote author is high for a good support value (a good number of postings have been seen from the unknown publisher and liked), this person is recommended as a publisher. The subscriber can then choose to add the recommended person to his publisher list, and thereby get metavines directly. This results in a 'short circuiting' of the social network, helps the individual's network to grow, and allows for the identification of more people of similar interest.

3 The Metavines Product Line Architecture

Analysis of several possible word-of-mouth social software like stock alerts dissemination, job contact network, music recommendations on the social network, web link dissemination, dissemination of technical knowledge in an organization, etc. reveals certain commonalities and variabilities. In summary,

Commonality: Word-of-mouth social software share the same core processes, information dissemination model, data structures for social network management and a network transport mechanism.

Variability: Different flavors of such applications vary in terms of semantics of rendering, representation, value system and contextual summarization for the information.

Therefore, a single software module will not suffice for all of these applications. This commonality-variation interplay suggests that a Product Line Architecture (PLA) is best suited for such application families.

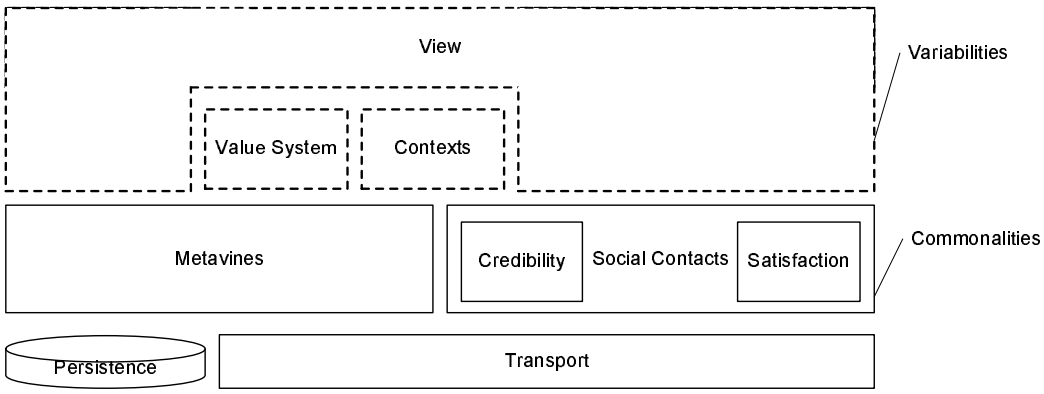

Figure 1: Metavines Product Line Architecture

Figure 3 depicts the common and variable components in the high level architecture of Metavines. The functionality provided by the components are as follows:

Transport: The Transport component provides the network communication functionality, along with the basic identity management and authentication mechanism. This component can be implemented as a wrapper over Instant Messaging (as in our implementation) or could use any other network communication protocol like smtp, http etc.

Persistence: The Persistence component provides serialization services for the data model in the Metavines architecture. Our current implementation persists objects onto the local disk. To make the system more device independent, this component can be re-implemented to use a network store, possibly on an Internet location.

Metavines: The Metavines component stores and organizes incoming, and outgoing metavines according to their ranks. An individual metavine itself is defined using an XML format. Applications can store specialized content as the metavine body.

Social Contacts: The Social Contacts component consists of the user's publishers, subscribers and recommended authors, and their credibilities and satisfaction indices.

Contexts: The Contexts component defines the set of all Metavine Contexts used by the application. Contexts is used by Metavines to associate a Metavine context with every incoming and outgoing metavine; and by Social Contacts to maintain a history of contextual credibilities and satisfaction.

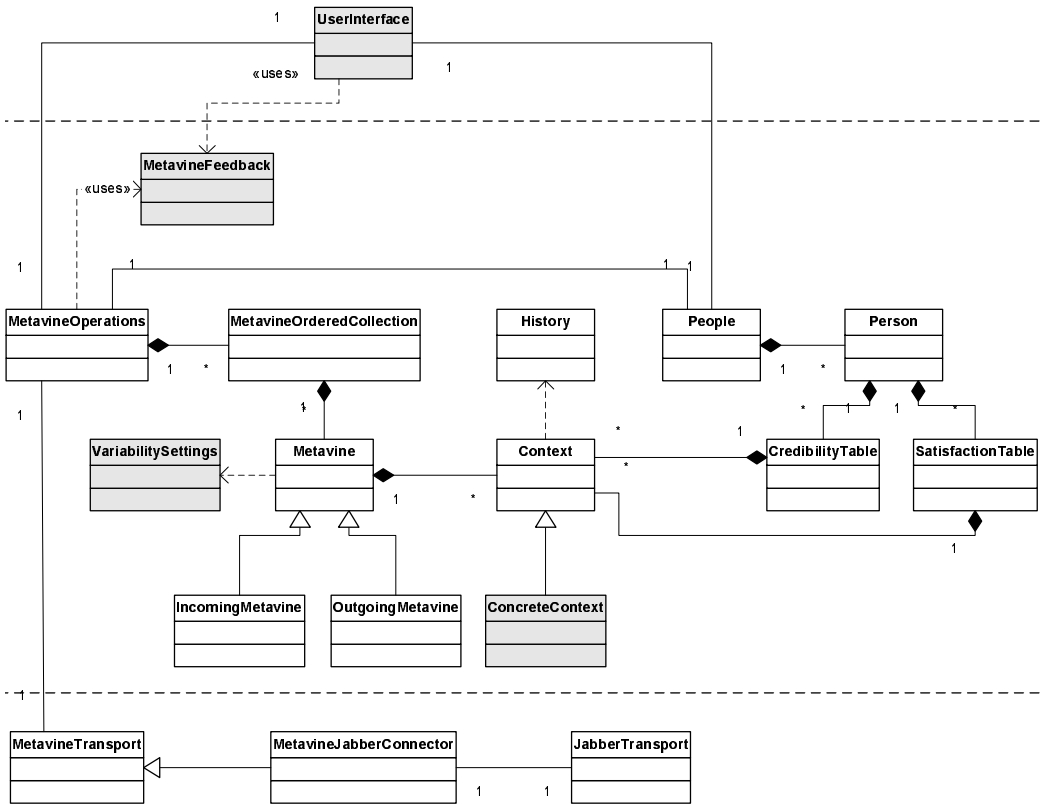

Figure 2: Class Diagram of Metavines PLA

Value System: The Value System defines the important and critical mapping between application specific events and associated values for metavines. Since explicit solicitation of values for each metavine could hamper the usability of the system, it is recommended that values are deduced through User Interface events.

View: The View provides the User Interface to the whole system, and also tracks events on the interface to facilitate the estimation of the value of metavines. The View is the most configurable part of the entire system, and is largely driven by the application semantics and user preferences.

A newly envisaged application has to implement just the variable components, viz., Value system, Contexts and the View. The core components, Metavines, Social Contacts, Persistence and Transport can be reused. Our implementation of the Metavines architecture is based on the Java^{TM} programming language and environment.

Figure 3 portrays a simplified class diagram of the Metavines PLA, showing the important classes of the system. The classes shaded in gray are the

variabilities of the system. At this class diagram view, the component Value System is implemented by the class *MetavineFeedback*; New contexts are specified by extending *Context* like *ConcreteContext*; and the View is implemented by *UserInterface*. Further, another class *VariabilitySettings* should be modified whenever a new context class is added as a child of Context.

A new application developer will create his own product based on the Metavines PLA in the following fashion:

UserInterface: The UserInterface essentially implements three interfaces, *MetavineCollectionView*, *MetavineView*, and *PeopleWatcher*. Every time the metavine collection is modified, due to the arrival of a new Metavine, a rank is updated, or a Metavine is deleted, the interface's methods are called, as a callback. MetavineView is notified if any changes in it contents are observed. PeopleWatcher is used to notify updates on the person's social network, either when a new person is added or removed, or when the credibility or satisfaction ratings change.

Feedback: Ascertaining the valuation of a metavine could be implicit or explicit. While an explicit method is always accurate, it hampers the usability of the system. Hence, where possible, GUI events are used as indicators to user interest. While using an implicit valuation mechanism, applications use the *MetavineFeedback* class to specify mappings between user events and associated increments to a metavine's value. This class implements a simple method called,
"double getEventValue(String eventName)" for this purpose.

Contexts: *Context* is an abstract class which uses the class *History* to store the history of past valuations of metavines. It also defines an abstract method called *"boolean isApplicable(Metavine mvine)"*. Any concrete implementation of *Context* (the application defined contexts) should implement this method. The method is used to determine applicability of a context for a given metavine.

Although defining a new context is quite easy, the system still needs to be notified about the newly available contexts. The *Metavine* class depends on the class *VariabilitySettings* to know about all the available application specific contexts. *VariabilitySettings* is more like a registry that keeps a record of all the contexts available to the system. So, while adding a new context, this class has to be modified as well. *VariabilitySettings* creates a pool of all the contexts defined, and provides a method *"Vector getApplicableContexts(Metavine mvine)"* which returns a vector of applicable contexts for a particular metavine.

As as example to explain the specification of these variabilities, we consider a networked music media server, and a word-of-mouth based link dissemination system for streamed music from that server. The social software could have an integrated music player as well, which is used to play music from the networked server. This system would implement the variabilities in the Metavines PLA as follows:

Context: Preference in music is known to have a strong correlation to time of the day. Music is organized in an ontology based on its genre. Thus, the contexts can be in the dimensions of time of day, consisting of {*Morning, Afternoon, Evening, Night*} and genre, consisting of all defined genres on the music server.

Classes specifying the Context are written in the following fashion:

```
class Afternoon extends Context {
   boolean isApplicable(Metavine mvine) {
       Calendar cal = Calendar.getInstance();
       cal.setTime(mvine.arrivalTime);
       if (cal.get(cal.HOUR_OF_DAY) >= 12 &&
            cal.get(cal.HOUR_OF_DAY) < 17)
          return true;
       else
          return false;
    }
}

class Genre extends Context {
   String genreName;
   Genre(String name) {
       genreName = name;
   }

   boolean isApplicable(Metavine mvine) {
        if (mvine.content instanceof Musicvine) {
            Musicvine mv = (Musicvine)mvine.content;
            if (mv.genre.equals(genreName)) {
              return true;
            }
        }
        return false;
   }
}
```

Next, VariabilitySettings is modified to reflect the above mentioned classes:

```
class VariabilitySettings {
   Vector contexts;
   VariabilitySettings() {
      contexts = new Vector();
      contexts.add(new Afternoon());

      contexts.add(new Genre("Western Classical"));
      contexts.add(new Genre("Indian Classical"));
      /* ... */
   }

   Vector getApplicableContexts(Metavine mvine) {
      Vector retVal = new Vector();
      for (int i = 0; i < contexts.size(); i++) {
         Context context = (Context)contexts.get(i);
         if (context.isApplicable(mvine)) {
            retVal.add(context.clone());
         }
      }
      return retVal;
   }
}
```

Value System: User interest can be captured by the events, purging, listening, uninterrupted complete play, archival and recommendation. The value system provides a mapping from these user events on the interface to real values. E.g.: (Purging = 0.0, Listening = 0.2, Recommending = 0.4, Complete Listening = 0.4, Archive = 0.4). The value associated with a metavine (musicvine in this example) is the aggregation of individual values for each event observed. The code snippet for making these changes in MetavineFeedback is as follows:

```
class MetavineFeedback {
   double getValue(String eventName) {
      if (eventName.equalsIgnoreCase("Purging")
         return 0.0;
      if (eventName.equalsIgnoreCase("Listening")
         return 0.2;
      if (eventName.equalsIgnoreCase("Recommending")
         return 0.4;
```

```
        if (eventName.equalsIgnoreCase("Completed")
           return 0.4;
        if (eventName.equalsIgnoreCase("Archive")
           return 0.4;
        return 0.0;
     }
  }
```

View: The view could display the ranked music recommendations as a playlist to the user. Thus over time, the user notices the most appealing recommendations on top.

With minimum changes to just 2-3 classes, adapts the Metavines architecture to different applications. By using the Metavines product line architecture, word-of-mouth social software can reuse its information ranking system, the social network credibility management, network transport and persistence. The user interface forms the only bulk component the social software needs to implement.

4 Webvines: A Case-study

Nowadays, most of the information is published over the World Wide Web, and avid surfers (such as graduate students) tend to exchange a lot of information that they find over the Web through informal discussions with their peers. Some of the interesting websites and articles on the web are recommended to their friends, who in turn could pass on this information further. In this section, we describe **Webvines**, an application that was built using Metavines architecture for automating this word-of-mouth exchange of links among WWW surfers. A snapshot of the Webvines system in use by one of the authors is shown in figure 4.

Webvines is based on the Metavines core. An individual webvine, the unit of information passed through the social network, comprises of a triple, the subject of the posting, the WWW URL, and a brief textual summary. Webvine is transferred embedded inside a metavine. Incoming webvines are ranked and presence in their order of relevance. The Webvines view is integrated with popular browsers, which are used to display WWW links. The Webvines system allows for archival of interesting links for later viewing. The interface also reflects the value as perceived by the subscribers, based on feedback, for every webvine posted from the user. Apart from these features, Webvines provides for two neighborhood assessment visuals to pictorially represent publishers' credibilities and subscribers' satisfaction. The

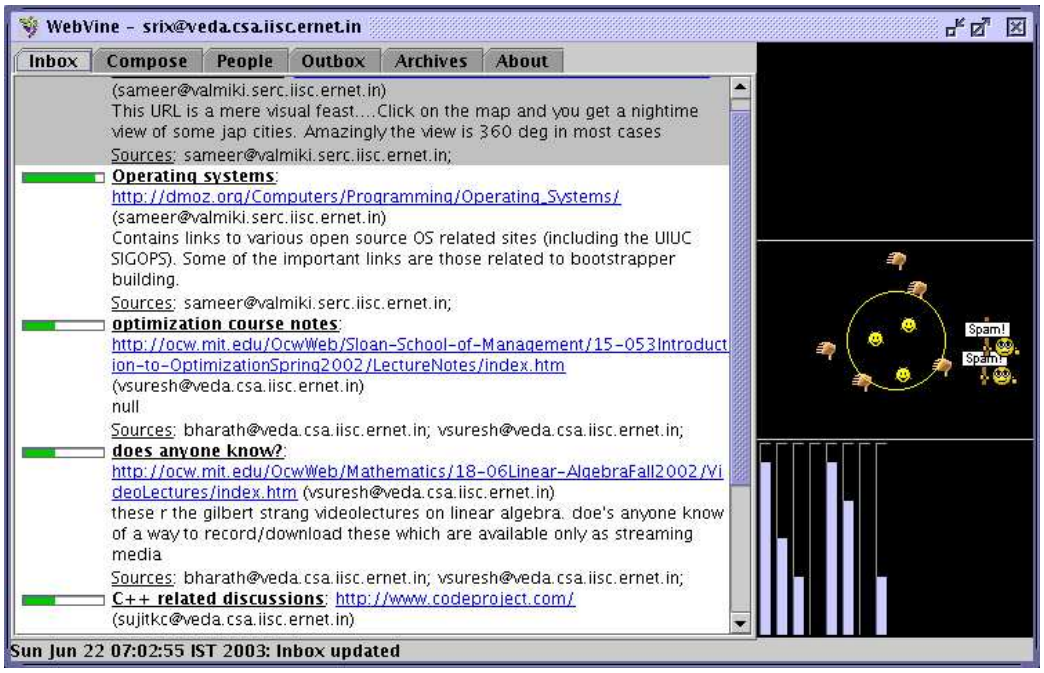

Figure 3: Snapshot of WebVines

visual representation towards the bottom right of the snapshot corresponds to the publisher credibilities as perceived by the user. The visual above the publisher ratings depicts subscriber satisfaction. Subscribers with high satisfaction levels show happy icons, those getting increasingly dissatisfied show thumbs down, and those who are unhappy by the user's postings show the "spam" signboard. The payoff for a person to post good links is largely visual. Bad recommendations and postings will result in most subscribers being 'unhappy'. The top right visual of the snapshot (empty in the figure), is used to recommend remote authors.

4.1 Interplay with Metavines

1. Webvines sends the WWW link information as payload embedded in a metavine.

2. Webvines specifies simple contexts based on time of day and forwarder/author role of the publisher.

3. Three events, {Read URL, Recommend (to subscribers), Archive} are used to infer the value of a webvine implicitly. Each of these events are given a weightage of 0.5.

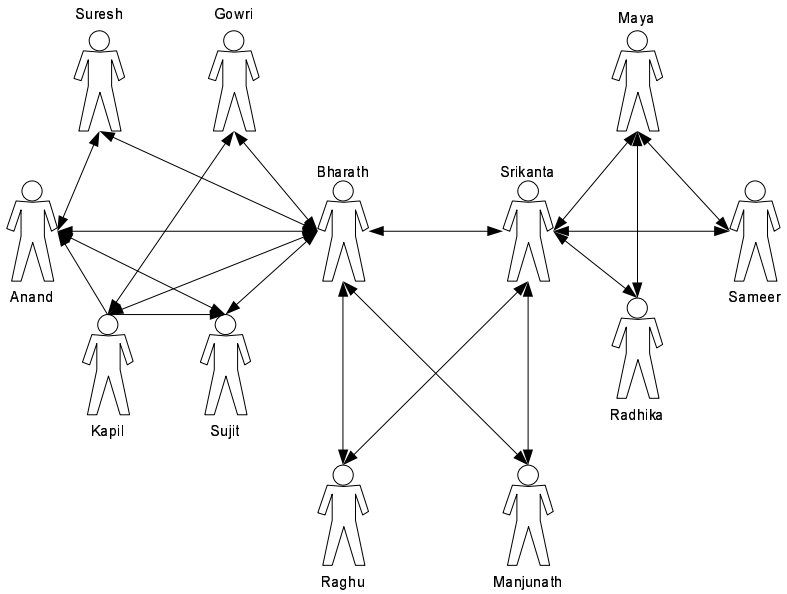

Figure 4: The Webvine Social Network

4.2 Deployment Studies

Webvines was first deployed and tested in a small social network, largely comprising of graduate students of computer science. The initial network, which is growing as we write this paper, is depicted in the figure 4.2. Most of the observations we report in this network are qualitative in nature.

The network had two clusters, one small and one big, the cluster on the left hand side with students interested in compiler design, and the cluster on the right with students interested in databases. Clearly, Bharath and Srikanta acted as hubs during the experiment. We could make the following observations by watching the posting behavior on this network.

- As expected, postings of compiler related interest did not make rounds in the rest of the network. General links were found to be more popular in this network of varied interests.

- Very soon, the network saw divisions in roles. Some students turned avid composers of webvines, while the hubs found it to be their responsibility to recommend news to the other side of the 'world'. The hubs were seen to have differing credibilities among their subscribers for their original and forwarded webvines.

- With growing interest among users, authors from one corner of the network were recommended to subscribers at the other side, thus short

circuiting the need for the hub.

- It was observed that people who knew that they were not hubs and were part of a cluster, did not recommend any webvines at all. Recommendation was seen as a task meant only for hubs. This resulted in some webvines not getting the due credit they may have deserved otherwise.

- There was general curiosity in the network, whenever a posting was received from an remote author. The hubs (through which these postings got forwarded) were enquired about the remote publishers.

5 Related Work

There has been an extensive body of research in the area of social software, especially in recommendation systems. One of the earliest recommender system is Ringo [12]. In this system, users with similar tastes in music are identified by imposing a similarity measure and are used to recommend a new album to the user. When a user first joins the system, an initial interest profile is constructed through a survey-like questionnaire and is subsequently tuned based on later recommendations by the user. Thus, Ringo used the "word-of-mouth" to provide recommendations to its users about music albums.

In essence, Ringo forms the "social clique", through its similarity measure, *without* utilizing the cognitive abilities or situatedness of the user in the social network. This makes the recommendations to be less contextual and come with no measure of credibility for the user. On the other hand, Metavines takes a totally different approach of delegating the decision making to the user and allowing for a distributed, continuously adaptive network of agents closely reflecting the dynamics in the society.

Another work that has close similarities to Metavines is [3], on the management trust in virtual communities. They suggest a model in which an agent (not the person) builds up a measure of trust with the agents it is interacting with, using contextual trustworthiness measures, similar to the contextual credibility management of Metavines. However, they limit the applicability of their scheme to reputational information while Metavines tries to address a much larger notion of information flow in the social network. Further, their focus is not on defining a architecture, which could be used for instantiating many different forms of such social software. Metavines addresses this very need and provides an architecture based on strong social

principles and utilizing the human cognition to enable dissemination of better quality information.

6 Conclusion

In this paper, we presented the Metavines product line architecture, to the best of our knowledge, the first of its kind, for a family of social software based on the word-of-mouth phenomenon. This architecture facilitates rapid development of social information filtering applications. We also presented Webvines, a member of the Metavines product line. This software has been deployed and observed within our university campus. Our initial experience with the Webvines system indicates that our approach to social information filtering is effective, provides a way to enable evolution of social networks to bring people of similar interests together and thereby receipt of relevant information. The Webvines system can be downloaded and tried at the following URL http://purana.csa.iisc.ernet.in/~mbk/metavines/.

References

[1] Epinions.com. http://www.epinions.com.

[2] Network intelligence gathering. http://www.collegegrad.com/jobsearch/8-0.shtml.

[3] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Hawaii Int. Conference on System Sciences 33*, jan 2000.

[4] T. Allen. *Managing the Flow of Technology.* MIT Press, Cambridge, MA, 1977.

[5] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns.* Addison Wesley, 2001.

[6] M. S. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6), 1973.

[7] H. Hong, J. D. Kubik, and J. C. Stein. Thy neighbor's portfolio: Word-of-mouth effects in the holdings and trades of money managers. http://www.stanford.edu/ hghong/ThyNeighbor.pdf, April 2003.

[8] B. D. Krueger. *College Grad Job Hunter.* Adams Media Corporation, March 2003.

[9] D. Z. Levin, R. Cross, and L. C. Abrams. The strength of weak ties you can trust: The mediating role of trust in effective knowledge transfer. In *Best Papers Proceedings of the Academy of Management*, 2002.

[10] C. Lueg and M. Muller. Cooperative systems: The right direction? In *Second International Conference on the Design of Cooperative Systems*, pages 315–329, 1996.

[11] E. Rogers. *Diffusion of Innovations.* Free Press, New York, 1995.

[12] Upendra Shardanand and Patti Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

[13] R. J. Shiller. *Irrational Exuberance.* Broadway Books, New York, 2000.

[14] B. Wellman and M. Gulia. Net surfers don't ride alone: Virtual communities as communities. In P. Kollock and M. Smith, editors, *Communities in Cyberspace: Perspectives on New Forms of Social Organization.* University of California Press, 1997.