CoFM: A Web-based Collaborative Feature Modeling System for Internetware Requirements' Gathering and Continual Evolution

Li Yi, Wei Zhang, Haiyan Zhao, Zhi Jin, Hong Mei
Institute of Software, School of EECS, Peking University Beijing, China
Key Laboratory of High Confidence Software Technology, Ministry of Education of China
{yili07, zhangw, zhhy, zhijin}@sei.pku.edu.cn, meih@pku.edu.cn

ABSTRACT

Internetware is a paradigm of open, decentralized and continually evolvable software systems running on the Internet. In the development of Internetware, the enormous amount of its stakeholders brings challenges to the gathering of common and essential requirements among these stakeholders and continual evolution of the requirements. In this paper, we present a web-based collaborative feature modeling system (CoFM) developed as a platform for gathering, organizing, evaluating, and negotiating Internetware requirements. The basic idea is to express and organize requirements in terms of user-perceivable *features* of desired Internetware application, and to allow stakeholders to propose, evaluate and negotiate these features collaboratively, in a shared *feature model* of the application. During the collaboration, the application provider can discover the common and important features that need to be implemented at present, and the special but valuable features that might be provided in the future. Moreover, the provider can track the up-to-moment evolution of the features, which enables the provider to quickly respond to the changes in the Internetware requirements.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specification –, *tools*

General Terms

Design, Experimentation, Human Factors

Keywords

Feature Model, Internetware, Requirements Elicitation, Collaboration.

1. INTRODUCTION

Internetware is a paradigm of open, decentralized and continually evolvable software systems running on the Internet [15]. The increasing interest of Internetware is witnessed by the emergence of Web services, Service Oriented Computing (SOC), Service Oriented Architecture (SOA) and cloud computing. Particularly, the Software as a Service (SaaS) approach in cloud computing has achieved great commercial success in recent years. For example, Salesforce.com maintains a Customer Relationship Management (CRM) application on the Internet and about 70,000 subscribed clients are using this application now.

A significant characteristic of Internetware is that the amount of its potential stakeholders is usually enormous, due to the openness and global availability of the Internet. This characteristic brings challenges to the development of Internetware. Specifically, in the requirements engineering for Internetware, there are needs of approaches and tools that: 1) support for gathering, organizing, evaluating, and negotiating requirements from large number of stakeholders; 2) synthesize the requirements to help Internetware provider identify the most common and essential requirements among the stakeholders, to enable the provided Internetware satisfy most stakeholders; 3) allow stakeholders to track the up-to-moment changes of the requirements, to make Internetware evolve continually and correctly. We believe that the emergence of such approaches and tools will have positive influence on the requirements engineering for the Internetware, and eventually, improve the dependability of the Internetware applications.

In this paper, we present a web-based collaborative feature modeling system (CoFM) developed as a platform for gathering, organizing, evaluating, and negotiating Internetware requirements. The basic idea is to express and organize requirements in terms of user-perceivable *features* of the desired Internetware application, and to allow stakeholders to propose, evaluate and negotiate these features collaboratively, in a shared *feature model* of the application. Specifically, the CoFM system includes the following characteristics:

- Utilize *feature models* to explicitly model the relationships between the gathered features. The relationships may have great influence on decisions in the later stages of the Internetware development. For example, when making the decision of whether to implement a specific feature, stakeholders should aware that all features *required by* this feature must be implemented before it.

- Incorporate a collaboration mechanism to allow multiple stakeholders to concurrently create new features as well as comment and vote on existing features. All the changes on the feature model immediately become visible to all stakeholders.
- Present the statistics of the features for requirements analysis. Particularly, the votes on the features help Internetware application providers identify the common features and the special ones, as a basis for feature prioritization.

Besides, we conduct a case study of gathering and negotiating the requirements of an *Online Recruiting Management* system in the CoFM system, and observations show that the CoFM provides an effective way for capturing multiple stakeholders' requirements in a collaborative manner.

The reminder of this paper is organized as follows. Section 2 gives some preliminaries on traditional feature models. Section 3 introduces the method for collaborative feature modeling. Section 4 presents the CoFM system, and the case study is presented in Section 5. Related work is discussed in Section 6. Finally, Section 7 concludes this paper with a brief summary and future work.

2. PRELIMINARIES: THE FEATURE MODEL

In this section, we introduce a meta-model of traditional feature models. They serve as bases of the extended feature models constructed in CoFM..

Figure 1 shows a meta-model of traditional feature models. Generally, a feature model consists of a set of features and relationships between them.

The concept of *feature* can be understood in two aspects: intension and extension [16]. In *intension*, a feature denotes a cohesive set of individual requirements. In *extension*, a feature describes a software characteristic having sufficient user/customer value.

There are two types of relationships, namely *refinements* and *constraints*. The *refinement* relationships organize features with different levels of abstraction or granularities into a hierarchical structure. In this hierarchical structure, each feature is either *mandatory* or *optional* (i.e. the *optionality* of a feature). If a feature is selected, its mandatory children features must be selected as well, while its optional children can either be removed or selected.

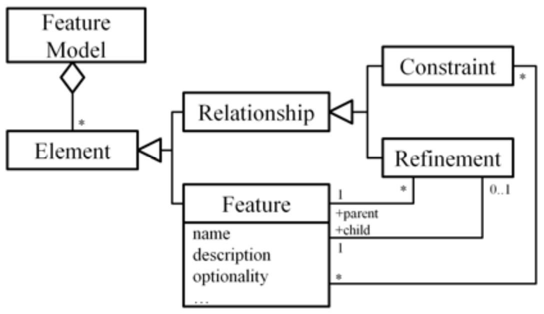


Figure 1. A meta-model of feature models

The *constraint* relationships describe dependencies between features. There are two basic kinds of constraints: *requires* and *excludes* [10][11]. Given two features *A* and *B, A requires B* means that if *A* is selected, *B* must be selected as well. *A excludes B* means that at most one of them can be selected at the same time. More types of constraints can be found in [16].

3. THE CoFM METHOD

The CoFM method proposes an approach to constructing a kind of extended feature models (EFMs) collaboratively. In this section, we give an overview of the CoFM method. We first introduce the meta-model of the EFMs, and then present the process of constructing the EFMs.

3.1 Extended Feature Models (EFMs)

Figure 2 shows the meta-model of EFMs. An EFM is composed of a set of *vote-able elements*. The top-level elements are *features* and *relationships*. Each feature relates to a set of second-level vote-able elements, that is, a feature has one vote-able *optionality* attribute, and may have one or more vote-able *names* and *descriptions*. The result of vote on each element is recorded as its *supporters* and *opponents;* both are a set of *stakeholders.*

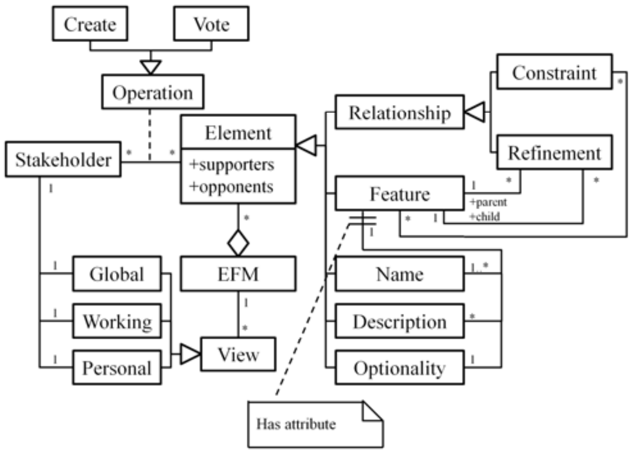


Figure 2. The meta-model of EFMs

In CoFM, stakeholders construct EFMs with two kinds of basic modeling operations, namely the *creating* operation and *voting* operation. The function of creating operation is to add new elements to a feature model. The function of voting operation is to allow stakeholders to express their opinions on existing elements: a *yes* vote means that the voter thinks an element is valid for the EFM, while a *no* vote shows the opposition.

We do not provide explicit *deleting* and *modifying* operations, in order to prevent stakeholders from editing and overwriting others' work directly. Therefore different opinions can be expressed in a feature model equally. However, the *deleting* operation can be simulated through the combination of the *creating* and *voting* operations, that is, an element will be deleted *if and only if* it has no supporters (i.e. all votes on the element are *no*). In addition, the *modifying* operation is the combination of *deleting* and *creating* operations, therefore it can also be simulated in CoFM.

To present an EFM for stakeholders, we define three types of EFM *views*, namely the *global view, working view* and *personal* *view*. In an EFM's construction, each stakeholder has a global view, a working view and a personal view. The global view is composed of all undeleted elements in an EFM, that is, the whole picture of the EFM. The working view is composed of elements that have not been voted *no* by this stakeholder, that is, elements on which the stakeholder vote *yes*, and elements created by others but the stakeholder has not shown any opinions on. The personal view is composed of elements supported (voted *yes*) by the stakeholder, that is, the stakeholder's personal perspective of the EFM. These views are automatically generated and updated for each stakeholder during the construction of an EFM.

3.2 The Process

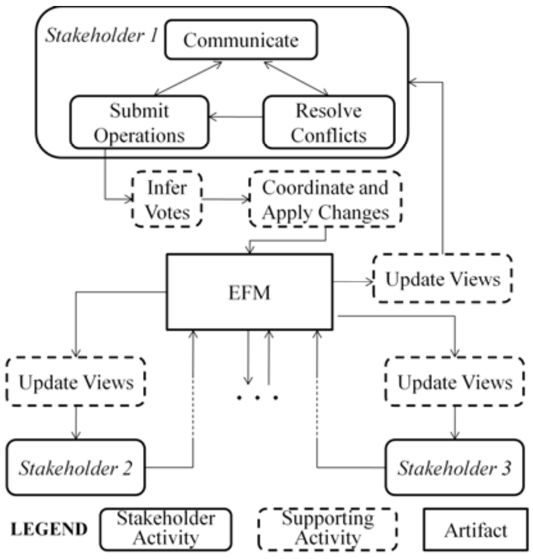


Figure 3. The process of CoFM

Figure 3 shows a typical process of collaborative construction of an EFM. In the beginning of a stakeholder's work, the system will initialize the global view, working view and personal view for the stakeholder (*Update Views*). In a stakeholder's work, s/he mainly engages three kinds of activities, including communicating with others, resolving conflicts and submitting modeling operations. Particularly, when s/he submits a modeling operation, the operation will be sent to the central EFM immediately. A valid operation will be applied and broadcast to all stakeholders (including the originator of the operation), which in turn, causes views of all stakeholders be updated.

There is no stopping rule in the process to determine whether the construction has completed. However, there is a particular way in CoFM to determine the *minimal necessary work* of each stakeholder, that is, the minimal necessary work of a stakeholder is to resolve all conflicts presented to the stakeholder. (The conflicts are often caused by modeling operations from multiple stakeholders.) Therefore, if any stakeholder hasn't finished his/her minimal necessary work, the construction must be unfinished.

4. THE CoFM SYSTEM

In this section, we present our CoFM system – the implementation of the CoFM method. We first provide an overview of key functions of the system, and then describe these functions in details.

4.1 An Overview of the System

Figure 4 gives a simplified version of the system's architecture. The CoFM system is designed with a typical Client/Server architecture. The client follows the Model-View-Controller design pattern. The views include *feature browser*, *feature editor*, *constraints browser* and other UI elements. The main model is a local work copy of the being constructed *feature model*; besides, there are various *data-views* (i.e. data subsets) for the *feature model*, such as the *name set* (a set contains existing feature names in the feature model). The controller consists of many *commands* (handlers of user operations), the *event dispatcher* and the *connector* (handles client-server data exchange). The server contains components for *protocol handling*, *data filtering*, *request handling* and *database access*.

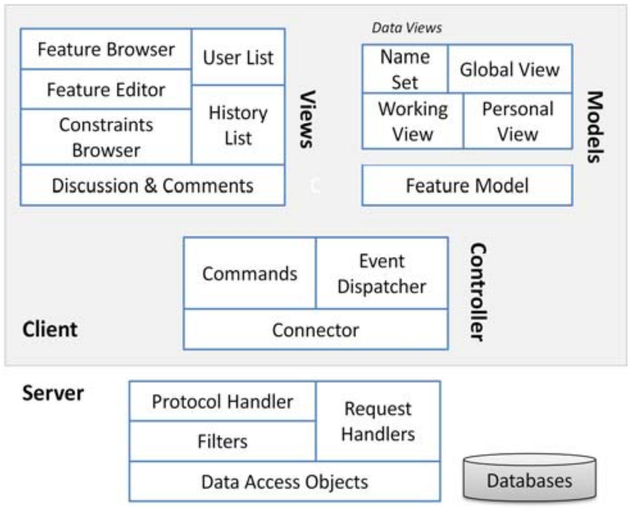


Figure 4. The architecture of the CoFM system

The CoFM system is a kind of CSCW (Computer-Supported Cooperative Work) systems. Researchers have identified several central concepts of CSCW systems [5][6][8][13], and these concepts have shaped the design of CSCW systems over the years, despite tremendous changes of technologies. The most important concepts are *production*, *awareness*, and *coordination*, thus dividing key functions of CSCW systems into three categories.

Production refers to objects produced and shared in a CSCW system, and operations applied on these objects [13]. In CoFM, it refers to the functions for constructing, viewing and checking shared feature models. (See Section 4.2.)

Awareness means that CSCW systems should provide necessary information about activities of other people in the shared workspace, which provides the context for individual's work [5]. For example, a collaborative writing tool should display the position of coauthors' edit cursor, to help authors identify their work context, i.e. their editing boundaries. The support for awareness in CoFM is described in Section 4.3.

Coordination relates to mediating and meshing individual work in a shared workspace [6]. In the context of collaborative work, it is highly possible that multiple stakeholders are working on the same set of elements at the same time, thus their changes of the elements need to be coordinated. The coordination mechanism in CoFM is explained in Section 4.4.

4.2 Modeling Support

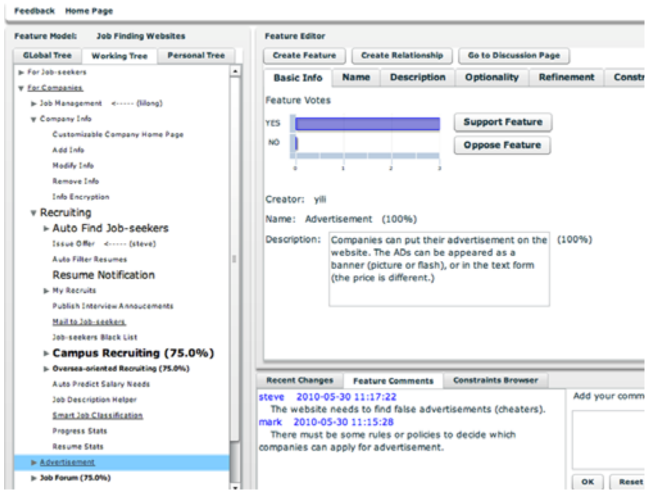
The key functions of modeling support in the CoFM system fall into three categories: basic modeling functions, feature attribute customizing functions, and model checking functions.

4.2.1 Basic modeling functions

Figure 5 is a screenshot of the modeling page in the CoFM system. The modeling page consists of three regions: the feature browser (left), the feature editor (right-top), and additional information viewers (right-bottom). The feature browser displays the EFM as feature trees, in global view, working view and personal view, respectively. The feature editor shows details of a specific feature. Additional information such as constraints list, comments for current feature and recent changes can be viewed in the right-bottom panels.

Stakeholders can create new elements in the modeling page. The creation in CoFM is more general than in other feature modeling approaches such as [9] and [10]. For example, in CoFM, a stakeholder can create several names for the same feature, while in other approaches the stakeholder can only assign exactly one name to a feature. However, no duplicate feature names are allowed within or between features, because feature names are considered as identifiers of features. For the purpose of reminding stakeholders of duplicate feature names, existing feature names are shown when stakeholders are typing the name of a new feature. (See Figure 6.)

For each feature, in the feature editor, we generate the basic information about the feature by combining the most supported values of each attribute (Figure 7), that is, the most supported name and description of the feature. In addition, stakeholders can vote on the current feature in the *basic info* tab. To create, vote on, or view the details of each attribute, stakeholders can switch to the specific tabs.

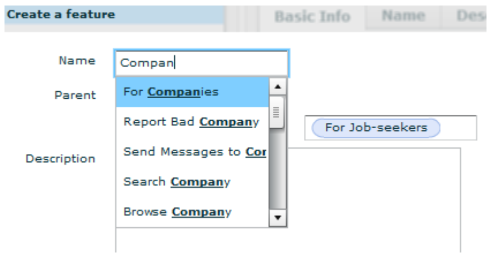


▸ For Job-seekers
▾ For Companies
▸ Job Management <------ (lilong)
▾ Company Info
Customizable Company Home Page
Add Info
Modify Info
Remove Info
Info Encryption
▾ Recruiting
▸ Auto Find Job-seekers
Issue Offer <------ (steve)
Auto Filter Resumes
Resume Notification
▸ My Recruits
Publish Interview Annoucements
Mail to Job-seekers
Job-seekers Black List
▸ Campus Recruiting (75.0%)
▸ Overseas-oriented Recruiting (75.0%)
Auto Predict Salary Needs
Job Description Helper
Smart Job Classification
Progress Stats
Resume Stats
▸ Advertisement
▸ Job Forum (75.0%)

Creator: yili
Name: Advertisement (100%)
Description: Companies can put their advertisement on the website. The ADs can be appeared as a banner (picture or flash), or in the text form (the price is different.) (100%)

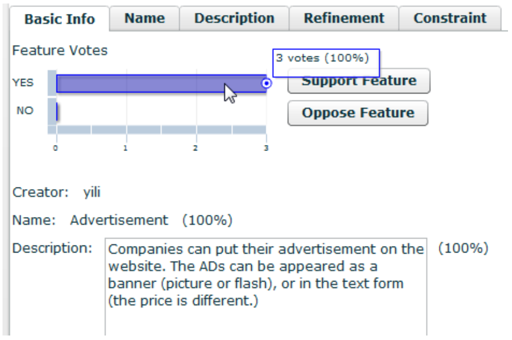
steve 2010-05-30 11:17:22
The website needs to find false advertisements (cheaters).
mark 2010-05-30 11:15:28
There must be some rules or policies to decide which companies can apply for advertisement.

Figure 5. A screenshot of CoFM's modeling page



Name: Compan
For Companies
Report Bad Company
Send Messages to Cor
Search Company
Browse Company
For Job-seekers

Figure 6. An example of creating a feature



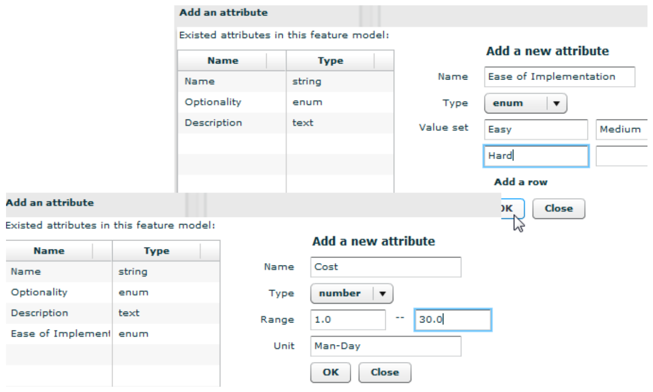
Creator: yili
Name: Advertisement (100%)
Description: Companies can put their advertisement on the website. The ADs can be appeared as a banner (picture or flash), or in the text form (the price is different.) (100%)

Figure 7. An example of feature editor

4.2.2 Feature attribute customizing functions

In the use of the early version of CoFM, some stakeholders have complained about the expressive power of feature models. For example, they often want to express the ease of implementation and the importance of the features; however, they cannot model these attributes explicitly but can only write them in features' descriptions. This makes further collaboration on these attributes (e.g. vote on the importance) impractical.

Our solution to such problem is to incorporate a mechanism called *feature attribute customization*. In an EFM, each feature has default attributes as before, i.e. *name*, *description* and *optionality*. Besides, stakeholders can add new attributes to all features in the EFM, and other EFMs are not affected. The attributes must belong to one of the predefined types: single-line *string*, multi-lines *text*, *enumeration*, and *number*. (We may define more types in the future.) If the type of new attribute is enumeration, the



Existed attributes in this feature model:

Name	Type
Name	string
Optionality	enum
Description	text

Name: Ease of Implementation
Type: enum
Value set: Easy, Medium, Hard

Existed attributes in this feature model:

Name	Type
Name	string
Optionality	enum
Description	text
Ease of Implement	enum

Name: Cost
Type: number
Range: 1.0 -- 30.0
Unit: Man-Day

Figure 8. An example of feature attribute customization

creator must enumerate valid values. If the type is number, then the unit and the valid range of its values must be defined as well. An illustrative example can be found in Figure 8. After new attributes are created, stakeholders can add new values of the attributes or vote on existing values.

4.2.3 *Model checking functions*

Several model checking methods have been adopted in feature modeling approaches to detect errors and defects in feature models [17]. However, model checking in CoFM is different in two ways:

First, model checking is performed only in working views, not the whole EFM. The reason is that EFMs are designed to tolerate divergence of different stakeholders, thus conflicts in EFMs (e.g. conflicting constraints created by two stakeholders) are acceptable. By contrast, a stakeholder's working view is his/her individual workspace, thus no errors or defects are tolerable, and the working view needs to be checked. In traditional feature modeling approaches, it is always the whole feature model that is to be checked.

Second, there are more types of errors/defects in CoFM than in traditional feature modeling approaches, as shown below.

- *Unnamed Feature (UF)*: the stakeholder votes *no* on all existing names of a feature, but doesn't create a new name. This leads to an unnamed feature in his/her working view. (According to the definition of working view, elements voted *no* by a stakeholder are excluded from his/her working view.)
- *Non-positioned Feature (NPF)*: the stakeholder votes *no* on all existing positions of a feature without creating a new one. (A feature must either be positioned as a root feature, or as a child of another feature.)
- *Conflicting Refinements (CR)*: multiple refinements existing in a working view are considered conflicting if they involve the same feature as the child but different features as the parent.
- *Conflicting Constraints (CC)*: multiple constraints existing in a working/personal view are considered conflicting if there are two involved features *A* and *B*, where both *A excludes B* and *A requires B* can be deduced from these constraints.

In traditional feature models, only the last situation (conflicting constraints) is considered as errors. We incorporate existing model checking methods for the conflicting constraints. The other types (UF, NPF and CR) can be detected without much difficulty.

These errors in a working view actually define the minimal amount of work that a stakeholder must do to obtain a usable feature model that conforms to traditional feature models. For example, a newly joined stakeholder only needs to resolve conflicting refinements and conflicting constraints to transform his/her working view into a usable feature model, which has a traditional hierarchical structure. In addition, there is a bonus: aliases and descriptions of features contributed by different stakeholders can help him deepen the understanding of the current domain.

4.3 Awareness Support

Awareness support is essential for improving the usability of CSCW systems [8]. It is becoming more and more apparent that being able to stay aware of others plays an important role in the fluidity and naturalness of collaboration. According to Gutwin and Greenberg's framework of awareness [8], the CoFM system provides six kinds of awareness information, as Table 1 shows.

Table 1. An overview of awareness information in CoFM

Awareness Information	**Purpose (According to [8])**	**Display the information**	
		Where	**How**
Others' edit location	Location (Where are they working?) Presence (Who is participating?)	Feature Browser	Show others' names next to being edited features.
Controversy	Artifact (What is the state of the objects?)	Feature Browser	Bold the controversial features.
Error	Artifact	Feature Browser	Use different colors for different types of errors.
My creation	Authorship (Who did that?)	Feature Browser	Underline the features created by current user.
Recently changed features	Action (What are they doing?)	Feature Browser	Use larger font for recently changed features.
Change History	Action History (What has been done?)	Change History Browser	List the details of changes made on the EFM.

Others' edit location tells a user who is working on the feature model currently and which feature the others are editing. The information is conveyed by displaying people's names next to the being edited features. For example, in Figure 9, the feature *Job Management* is being edited by the user *lilong*.

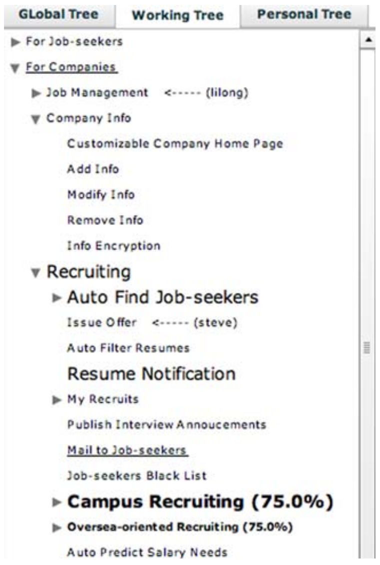
Controversy information helps users be aware of controversial features (controversy on a feature or its attributes), and if the controversy is about a feature, then the support rate of the feature's validity is shown as a number. For example, in Figure 9, the support rate of validity of the feature *Campus Recruiting* is 75 percent.

Error information is shown by coloring, and different types of errors are displayed in different colors.

My creation (features created by current user) is underlined in the feature browser. Besides, creator of each feature is shown in a feature editor panel. (Clicking a feature in feature browser will open the feature in the feature editor.)

Recently changed features have larger font size than other features, in the feature browser. We only treat features with *unread recent changes for current user* as recently changed ones, in other words, different users have different recently changed features. This information can help users follow the recent progress of the feature model, especially for the large scale feature models.

Change history is different from recently changed features. This information shows all changes that have been made on the feature model, ordered by time. By selecting a specific change, the corresponding feature will be shown in the feature editor.



```
GLobal Tree | Working Tree | Personal Tree
▶ For Job-seekers
▼ For Companies
    ▶ Job Management   <----- (lilong)
    ▼ Company Info
        Customizable Company Home Page
        Add Info
        Modify Info
        Remove Info
        Info Encryption
    ▼ Recruiting
        ▶ Auto Find Job-seekers
        Issue Offer   <----- (steve)
        Auto Filter Resumes
        Resume Notification
        ▶ My Recruits
        Publish Interview Annoucements
        Mail to Job-seekers
        Job-seekers Black List
        ▶ Campus Recruiting (75.0%)
        ▶ Oversea-oriented Recruiting (75.0%)
        Auto Predict Salary Needs
```

Figure 9. An example of displaying awareness information

During the use of our system, we find that the most frequently used UI region is the feature browser, so we display most of the awareness information in the feature browser. Furthermore, we use different UI elements for different types of information, which enables multiple awareness information of the same feature to be correctly displayed, by the combination of corresponding UI elements. For example, in Figure 9, the feature *Campus Recruiting* is controversial (bolded) and contains unread recent changes (large font size).

4.4 Coordination Support

In collaborative work, when multiple stakeholders are working on the same set of elements simultaneously, their changes of the elements may need to be coordinated. In CoFM, we incorporate a *centralized broadcasting strategy* to coordinate modeling operations submitted by multiple stakeholders. The basic idea of the strategy is that each operation performed by each stakeholder will be sent to the central server where the shared EFM is stored. The server will attempt to execute these operations on the EFM in the order of the reception of the operations. If an operation is valid at the time of its execution, it will be executed; if it is not valid and cannot be transformed into other valid operations, it fails. Executed operations will be broadcasted to all stakeholders, including its originator, and the local copy of the EFM for each stakeholder is updated. Figure 10 gives a sketch of the strategy.

Table 2. Pre-condition and transformation of operations

Operation	**Pre-condition**	**How to transform an invalid operation**
Create Feature *X*	The name *X* has not been used in current EFM.	(If *X* have been used by feature *F*) Vote *yes* on *F*; Vote *yes* on *X*.
Create Name *X*	The name *X* has not been used in current EFM.	Vote *yes* on *X*.
Create a value *V* for other attributes (e.g. description)	The value *V* has not existed in current feature.	Vote *yes* on *V*.
Create Relationship *R*	*R* has not existed in current EFM.	Vote *yes* on *R*.
Vote on element *E*	*E* must exist in current EFM.	*(Cannot be transformed.)*

Table 2 describes the pre-condition of an operation and how to transform an invalid operation into valid ones (if possible), such transformation will preserve the semantics of the original operation. For example, the operation *Create Feature(X)* is invalid if the name *X* has been used by another feature *F*. According to Table 2, the creation will be transformed into two voting operations, and the semantics of original operation (*create a feature named X*) is the same with the voting operations (*support the existence of the feature named X*).

```
The Client
SEND (o: Operation)
    Send operation o to the server.
RECEIVE (o: Operation)
    Update local copy of EFM by operation o.

The Server
MAIN_PROCESS (loop)
    p ← The first operation in the Operation_Queue.
    if (p is valid)
        Execute p on the EFM.
        Broadcast p.
    else if (p can be transformed)
        q ← transformed(p)
        Execute q on the EFM.
        Broadcast q.
    else
        Inform the invalidity of p to its originator.
RECEIVE (o: Operation)
    Put o at the end of Operation_Queue.
```

Figure 10. The sketch of centralized broadcasting strategy

5. CASE STUDY

In order to explore the use, strengths and weaknesses of CoFM, we conduct a case study with four participants. The four participants use CoFM to propose, discuss and evaluate the requirements of an *Online Recruiting Management* system. This online system is a SaaS application that allows multiple registered companies (clients) to perform online recruiting activities like job publishing, job applying, interview arranging, and offer issuing. Each participant stands for a distinct client.

The participants spend about three hours on the collaborative work, and finally they proposed 113 features. There's no rigid process for their work, however, we have observed that the collaborative work can be roughly divided into two phases; we call them the *brainstorming phase* and the *evaluation phase*.

The work starts at the brainstorming phase. In this phase, the participants are focusing on proposing requirements as much as possible (i.e. creating new features). The way they prefer is to read an existing feature and try to refine it (create child features for it) or create a related feature. An important observation is that because CoFM displays others' edit location, the participants always attempt to choose a different feature from others to read, thus leads to parallel creation at different parts of the feature model. As a result, a large number of initial features are collected over a short period of time (about an hour).

As the feature model grows larger, the participants start to focus on evaluating the proposed features. In this evaluation phase, participants browse the whole feature model for many times, and often jump to recent changes immediately if any change happens. Specifically, their main activities include the following ones:

- Remove redundant features. If two features have different names but the same semantics, then one of them is redundant. This occurs mainly because the terminologies used by participants are different, for example, the feature *Find Jobs* and the feature *Search for Jobs*. To remove redundant features, the participants prefer to post a topic in the discussion page, such as "Redundant features of *Search for Jobs*", and ask others to vote *no* on redundant features to remove them from the feature model.
- Improve unclear features. Some participants are reluctant to carefully describe a feature when they create it, so others cannot understand the feature very well. The situation gets worse when the feature has an improper or vague name. As we observed, participants often post a comment on unclear features to ask for improvement, and some of them even vote *no* on these features until their creators improve the description and/or name.
- Find missing features. The participants try to find missing features when they are browsing the whole feature model.
- Adjust and create relationships. The participants adjust refinements and create constraints when most of the features can be well understood.
- Confirm requirements. For each participant, s/he browses each feature and vote *yes* on it if it is a desired feature, no matter whether the feature is proposed by her/him or others.

Figure 11 shows the statistics of features in every 20 minutes. The features are divided into three categories: changed, unchanged, and newly created, in last 20 minutes. A feature is treated as *changed feature* if values of its attributes are created or deleted, or relationships involving it are created or deleted, or new comments are posted on it. Besides, the total number of features at the time is also shown in Figure 10. We can clearly see that in the first 60 minutes (the brainstorming phase), most of the work is to create new features. After that, there are less creations and more changes of existing features (the evaluation phase), and the total number of features is decreased due to the removal of unclear and redundant features.

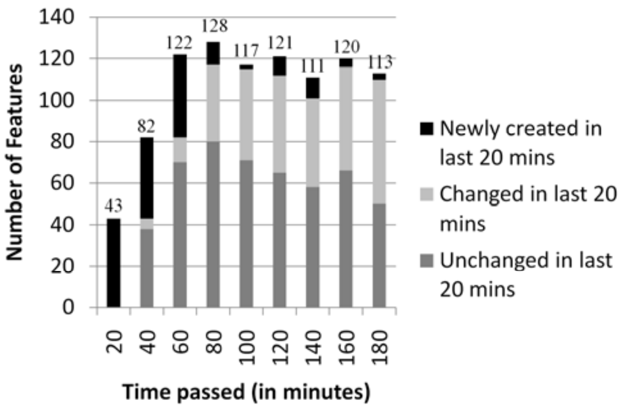


Figure 11. The change of features during the collaboration

Table 3 shows the number of confirmed features for each participant. We can see that a large part of confirmed features is proposed by others, which means by collaborating in CoFM, the participants' work load can be reduced; in fact, the participants have reported that, "Although I have to understand others' work, it is still easier to review an existing feature than to think of a new one by myself, therefore my efficiency of work is improved."

Table 3. Number of confirmed features for each participant

Participant	Number of confirmed features		
	Total	Created by this participant	Created by others
P1	91	21 (23.1%)	70 (76.9%)
P2	87	37 (42.5%)	50 (57.5%)
P3	94	34 (36.2%)	60 (63.8%)
P4	104	21 (20.2%)	83 (79.8%)

Table 4 shows the proportion of common and special features in the final result. 83 of the 113 features are common (confirmed by all participants), and these features are esstential. Besides, 29 features are confirmed by two or three participants, so these features are probably also have high user/customer value. There is only 1 unique feature (confirmed only by one participant), so if the cost of the features is acceptable, the SaaS application provider can deliver a specialized version for this client.

Table 4. Statistics about the proposed features

Number of proposed features				
Total	**Common (Confirmed by four)**	**Confirmed by three**	**Confirmed by two**	**Unique**
113	83 (73.5%)	24 (21.2%)	5 (4.4%)	1 (0.9%)

An important observation is that the awareness information is crucial to the collaborative work. The participants have reported that *others' editing location* is the most useful information for their work, because they can find which part of the feature model is idle and needs attention, thus they can try their best to work in parallel. Besides, the awareness of *recent changes* is also important, because it helps participants follow others' work and know which part of the feature model is the current interest, especially for feature models of large scale.

In summary, the case study shows that use of features and feature models in Internetware requirements elicitation is feasible. With adequate awareness support and proper collaboration mechanism, the efficiency of the requirements gathering and reviewing is improved.

6. RELATED WORK

The concept of feature models was first introduced in the FODA method [8]. The idea of feature models is to encapsulate requirements into a set of features and dependencies among features, and then to reuse these encapsulated requirements by selecting a subset of features from a feature model, while maintaining dependencies among features. In FODA [8], FORM [10], FeatuRSEB [7], and FOPLE [11], feature models are constructed through the commonality and variability analysis to a set of applications in a domain. The construction task is performed by domain analysts, with intensive collaboration with other stakeholders such as users, domain experts and requirements analysts. However, none of these approaches has explicitly incorporated mechanisms or techniques to support collaboration among stakeholders in the construction of feature models.

In the research of requirements engineering, several methods have been proposed to support collaboration among stakeholders. The EasyWinWin system [1] provides an environment for stakeholders to propose requirements, and then vote and review these requirements collaboratively. Decker et al. [4] proposed a Wiki-based approach to asynchronous collaborative requirements engineering. This approach utilize collaboration mechanisms provide by the wikis to facilitate collaboration among stakeholders, and more particularly, to facilitate stakeholders' collaborative editing on a set of use cases contained in a set of web pages. However, although these tools allow different or even conflicting opinions to be expressed, they can only tolerate temporary divergences; all divergences must be eliminated in the end of work. By contrast, feature models used in CoFM can tolerate permanent divergences; the divergences are treated as indicators of domain variability. Furthermore, in these tools, conflicts between stakeholders need to be manually identified, while in feature models, conflicts are well-defined and can be automatically detected.

Noor et al. [13] proposed a collaborative product line planning (CoPLP) approach, in which the requirements of desired products are proposed and discussed in terms of features However, CoPLP did not explicitly utilize feature models to express and organize the features, therefore there is a lack of explicit modeling of constraints between the features. By contrast, CoFM uses an extended feature model, therefore not only the features of a product can be collaboratively collected and reviewed, but also the relationships between these features can be collaboratively created and discussed.

In addition, Laurent and Cleland-Huang [12] have pointed out that feature request is a common form of requirements elicitation in real world's open source projects. In these projects, stakeholders propose desired features by posting topics in the project forum. However, a major problem is that the requested features are often lack organization; in other words, the topics in the forum are often lack management and bring chaos to further refinement of the features. For example, discussions about the same feature are scattered over several topics, therefore it is difficult for developers to understand the full details of the feature. Castro-Herrera et al. [2] utilized the data mining technologies to help cluster the scattered topics. In CoFM, feature models are incorporated to organize the proposed features, and can guide the feature request towards a clear and structured process.

7. CONCLUSIONS AND FUTURE WORK

This paper presents a web-based collaborative feature modeling system (CoFM) developed as a platform for gathering, organizing, evaluating, and negotiating Internetware requirements. In this system, requirements are expressed in terms of user-perceivable features. The feature model is utilized to explicitly model the relationships between the gathered features. Stakeholders can propose, evaluate and negotiate these features collaboratively, in a shared feature model of the application. A collaboration mechanism is designed to allow concurrent modification on the feature model by multiple stakeholders. Statistics of the feature models are also be presented to stakeholders, to help them make further decisions in the development of the Internetware applications.

In the future, we want to first improve the usability of CoFM. Then we plan to conduct more case studies with larger scale and more people. In addition, new techniques and mechanisms will be incorporated into the CoFM to enhance its capability. For example, more types of statistics for users, an algorithm for calculating confidence or trustworthiness of users' operations, and mechanisms for helping users find constraints between features.

8. REFERENCES

[1] Boehm, B., Grunbacher, P., Briggs, R. O. 2001. Developing groupware for requirements negotiation: Lessons learned. *IEEE Software*. 18, 3, 2001, 46-55.

[2] Castro-Herrera, C., Cleland-Huang, J., Mobasher, B. Enhancing stakeholder profiles to improve recommendations in online requirements elicitation. In *Proceedings of Intl. Conf. on Requirements Engineering*. RE '09. 37-46.

[3] Conklin, J., Begeman, M. gIBIS: A hypertext tool for exploratory policy discussion. *ACM Trans. On Information Systems (TOIS)*. 6, 1988, 303-331.

[4] Decker, B. Ras, E., Bech, J., Jaubert, P. and Rieth, M. Wiki-Based Stakeholder Participation in Requirements Engineering. *IEEE Software*. 24, 2, 2007, 28-35.

[5] Dourish, P., Bellotti, V. Awareness and coordination in shared workspaces. In *Proc. of the 1992 ACM Conf. on CSCW*. 107-114.

[6] Ellis, C. A., Gibbs, S. J. Rein, G. Groupware: Some Issues and Experiences. *Communications of ACM*, 34, 1991, 39-58.

[7] Griss, M. L., Favaro, J., d'Alessandro, M. Integrating Feature Modeling with the RSEB. In *Proc. Of the Fifth Intl. Conf. on Software Reuse.* ICSR '98. 76-85.

[8] Gutwin, C., Greenberg, S. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work.* 11, 3, 2002, 411-446.

[9] Kang, K. C., Cohen, C., Hess, J., Nowak, W., Peterson, S. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990.

[10] Kang, K. C., Kim, S., Lee, J., Kim, K., Kim, G. J., Shin, E. FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. *Annals of Software Engineering.* 5, 1998, 143-168.

[11] Kang, K. C., Lee, J., Donohoe, P. Feature-Oriented Product Line Engineering. *IEEE Software*. 19, 4, 2002, 58-65. DOI= 10.1109/MS.2002.1020288.

[12] Laurent, P., Cleland-Huang, J. Lessons Learned from Open Source Projects for Facilitating Online Requirements Processes. In *Proc. Of Working Conf. on RE: Foundations for Software Quality.* REFSQ '09.

[13] Laurillau, Y., Nigay, L. Clover Architecture for Groupware. In *Proc. of the 2002 ACM Conf. on CSCW*. 236-245.

[14] Noor, M. A., Rabiser, R., Grunbacher, P. Agile product line planning: A collaborative approach and a case study. J. Syst. Software. 2007. DOI=10.1016/j.jss.2007.10.028

[15] Tsai, W., Jin, Z., Bai, X. Internetware Computing: Issues and Perspective. In *Proceedings of the First Asia-Pacific Symposiumn on Internetware* (Internetware '09). 2009.

[16] Zhang, W., Mei, H., Zhao, H. Y. A feature-oriented approach to modeling requirements dependencies. In *Proc. of the 13th IEEE Intl. Conf. on Requirments Engineering.* RE '05. 273-282

[17] Zhang, W., Zhao, H., and Mei, H. A Propositional Logic-Based Method for Verification of Feature Models. In *Proceedings of Sixth International Conference on Formal Engineering Methods* (ICFEM'04). Springer Berlin/Heidelberg, LNCS 3308, 2004, 115-130.